aws

Guida per l'utente per Aurora

Amazon Aurora

Copyright © 2026 Amazon Web Services, Inc. and/or its affiliates. All rights reserved.

Amazon Aurora: Guida per l'utente per Aurora

Copyright © 2026 Amazon Web Services, Inc. and/or its affiliates. All rights reserved.

I marchi e il trade dress di Amazon non possono essere utilizzati in relazione ad alcun prodotto o servizio che non sia di Amazon, in alcun modo che possa causare confusione tra i clienti, né in alcun modo che possa denigrare o screditare Amazon. Tutti gli altri marchi non di proprietà di Amazon sono di proprietà dei rispettivi proprietari, che possono o meno essere affiliati, collegati o sponsorizzati da Amazon.

Table of Contents

Che cos'è Aurora? 1
Modello di responsabilità condivisa di Amazon RDS 2
Funzionamento di Amazon Aurora con Amazon RDS 2
Cluster di database Aurora 4
Versione di Aurora 6
Motori di database supportati 6
Come specificare una versione Aurora 7
Controllo delle versioni di Aurora 7
Aggiornamento dei cluster di database Aurora 13
Supporto per le versioni di Aurora 16
Regioni e zone di disponibilità 18
AWSRegioni 19
Zone di disponibilità 29
Fuso orario locale per i cluster DB 30
Funzionalità supportate da Aurora per regione e motore 37
Convenzioni tabella 38
Distribuzioni blu/verdi 38
Configurazioni dei cluster Aurora 39
Flussi di attività di database 39
Esportazione dei dati del cluster in Amazon S3 50
Esportazione dei dati snapshot in Simple Storage Service (Amazon S3) 52
Database globali di Aurora 53
Autenticazione del database IAM 65
Autenticazione Kerberos 66
Aurora machine learning 79
Approfondimenti sulle prestazioni 92
Integrazioni Zero-ETL 105
Server proxy per RDS 112
Integrazione di Secrets Manager 125
Aurora Serverless v2 125
Aurora Serverless v1 132
API dati in RDS 137
Applicazione di patch senza tempi di inattività (ZDP) 148
Aurora PostgreSQL Limitless Database 149

Funzionalità native del motore ... 149
Connessioni endpoint ... 150
Tipi di endpoint di Aurora ... 151
Visualizzazione degli endpoint ... 152
Endpoint ed elevata disponibilità ... 153
Endpoint del cluster ... 154
Endpoint di lettura ... 155
Endpoint dell'istanza ... 156
Endpoint personalizzati ... 157
Classi di istanze database ... 177
Tipi di classi di istanza database ... 177
Motori di database supportati ... 181
Determinazione del supporto delle classi di istanza database nelle Regioni AWS ... 197
Specifiche dell'hardware ... 202
Storage ... 214
Panoramica dello storage di Aurora ... 215
Contenuto del volume del cluster ... 215
Configurazione dell'archiviazione dei cluster Aurora ... 215
Ridimensionamento dello storage ... 216
Fatturazione dei dati ... 218
Affidabilità ... 218
Riparazione automatica dello storage ... 219
Cache delle pagine superstite ... 219
Ripristino in seguito a riavvii non pianificati ... 220
Sicurezza di Aurora ... 221
Utilizzo di SSL con i cluster di database di Aurora ... 222
Elevata disponibilità ... 222
Elevata disponibilità di dati Aurora ... 223
Architetture ad alta disponibilità per istanze database Aurora ... 223
Elevata disponibilità nelle regioni AWS con database globali Aurora ... 224
Tolleranza ai guasti ... 225
Elevata disponibilità con Amazon RDS Proxy ... 227
Replica ... 227
Repliche di Aurora ... 227
Aurora MySQL ... 230
Aurora PostgreSQL ... 230

Fatturazione delle istanze database per Aurora 231
Istanze di database on demand 234
Istanze database riservate 236
Configurazione dell'ambiente 252
Registrati per un Account AWS 252
Crea un utente con accesso amministrativo 253
Concessione dell'accesso programmatico 254
Determinazione dei requisiti 256
Fornire l'accesso al cluster DB 258
Nozioni di base 261
Creazione e connessione a un cluster di database Aurora MySQL 261
Prerequisiti 263
Fase 1: Creare un' EC2istanza 263
Fase 2: creazione di un cluster di database Aurora MySQL 269
(Facoltativo) Crea VPC, EC2 istanza e cluster Aurora MySQL utilizzando CloudFormation .. 274
Fase 3: connessione a un cluster di database Aurora MySQL 276
Passaggio 4: Eliminare l' EC2 istanza e il cluster DB 279
(Facoltativo) Elimina l' EC2 istanza e il cluster DB creati con CloudFormation 280
(Facoltativo) Connessione del cluster database a una funzione Lambda 281
Creazione e connessione di un cluster di database Aurora PostgreSQL 281
Prerequisiti 283
Fase 1: Creare un' EC2 istanza 283
Fase 2: creazione di un cluster di database Aurora PostgreSQL 289
(Facoltativo) Crea VPC, EC2 istanza e cluster Aurora PostgreSQL utilizzando CloudFormation 294
Fase 3: connessione a un cluster di database Aurora PostgreSQL 296
Passaggio 4: Eliminare l' EC2istanza e il cluster DB 299
(Facoltativo) Elimina l' EC2 istanza e il cluster DB creati con CloudFormation 300
(Facoltativo) Connessione del cluster database a una funzione Lambda 300
Tutorial: creazione di un server Web e di un cluster database Amazon Aurora 302
Avvio di un'istanza EC2 per la connessione al cluster di database 303
Creazione di un cluster DB 309
Installazione di un server Web 320
Tutorial e codice di esempio 333
Tutorial in questa guida 333
Tutorial in altre guide AWS 334

Tutorial e codice di esempio in GitHub ... 335
AWS Ricettario del database ... 336
AWS seminario - PostgreSQL ... 337
AWS seminario - MySQL ... 338
Lavorare con AWS SDKs ... 339
Accesso programmatico ad Amazon Aurora ... 341
Console-to-Code ... 342
Configurazione di un cluster di database Aurora ... 343
Creazione di un cluster di database ... 344
Prerequisiti ... 345
Creazione di un cluster di database ... 352
Impostazioni disponibili ... 363
Impostazioni che non si applicano ad Aurora per i cluster di database ... 384
Impostazioni che non si applicano alle istanze database Aurora ... 385
Creazione di risorse con AWS CloudFormation ... 388
Aurora e modelli CloudFormation ... 388
Ulteriori informazioni su CloudFormation ... 388
Connessione a un cluster database ... 389
Connessione ai cluster di database Aurora con i driver AWS ... 390
Connessione ad Aurora MySQL ... 392
Connessione ad Aurora PostgreSQL ... 398
Risoluzione dei problemi relativi alle connessioni ... 401
Gruppi di parametri ... 402
Panoramica dei gruppi di parametri ... 402
Gruppi di parametri di cluster database ... 406
Gruppi di parametri database ... 425
Confronto di gruppi di parametri database ... 442
Specifica dei parametri del database ... 443
Migrazione di dati a un cluster database ... 449
Aurora MySQL ... 449
Aurora PostgreSQL ... 449
Creazione di una cache ElastiCache da Amazon RDS ... 450
Panoramica della creazione di una cache ElastiCache con le impostazioni del cluster di database Aurora ... 450
Creazione di una cache ElastiCache con le impostazioni di un cluster di database Aurora esistente ... 451

Migrazione automatica dei database EC2 455
Panoramica 455
Prerequisiti 456
Limitazioni 457
Creazione di risorse IAM 458
Configurazione della migrazione dei dati 466
Gestione delle migrazioni 468
Monitoraggio 470
Tutorial: creazione di un cluster di database MySQL con un gruppo di parametri personalizzati 472
Prerequisiti 473
Creazione di un gruppo parametri personalizzato 473
Modifica del valore del parametro 474
Creazione di un cluster di database 474
Gestione di un cluster DB Aurora 476
Avvio e arresto di un cluster 477
Panoramica dell'avvio e dell'arresto di un cluster. 477
Restrizioni 478
Arresto di un cluster di database 479
Mentre il cluster di database è in fase di arresto 480
Avvio di un cluster di database 480
Connessione di un'istanza EC2 482
Panoramica 483
Connessione di un'istanza EC2 488
Visualizzazione delle risorse di calcolo connesse 491
Connessione a un'istanza database che esegue un motore DB specifico 492
Connessione di una funzione Lambda 493
Panoramica 495
Connessione di una funzione Lambda 506
Visualizzazione delle risorse di calcolo connesse 508
Modifica di un cluster database Aurora 510
Modifica del cluster di database tramite la console, la CLI e l'API 510
Modifica di un'istanza database in un cluster database 513
Modifica della password dell’utente master 515
Impostazioni disponibili 518
Impostazioni che non si applicano ai cluster di database Aurora 555

Impostazioni che non si applicano alle istanze database Aurora 556
Aggiunta di repliche di Aurora 558
Auto Scaling con repliche Aurora 565
Aggiunta di una policy di dimensionamento automatico 571
Modifica di una policy di dimensionamento automatico 584
Eliminazione di una policy di dimensionamento automatico 587
Gestione delle prestazioni e del dimensionamento 590
Dimensionamento dello storage 590
Dimensionamento delle istanze 597
Dimensionamento della lettura 597
Gestione delle connessioni 598
Gestione dei piani di esecuzione delle query 598
Clonazione di un volume per un cluster database Aurora 599
Panoramica sulla clonazione Aurora 599
Limitazioni della clonazione Aurora 600
Come funziona la clonazione Aurora 601
Creazione di un clone Aurora 605
Clonazione tra VPC 616
Clonazione tra più account 635
Integrazione con i servizi AWS 653
Aurora MySQL 653
Aurora PostgreSQL 653
Manutenzione di un cluster database Aurora 655
Panoramica degli aggiornamenti di manutenzione dei cluster di database 655
Visualizzazione della manutenzione in sospeso 657
Scelta della frequenza degli aggiornamenti di manutenzione di Aurora MySQL 659
Finestra di manutenzione 660
Applicazione di aggiornamenti 666
Aggiornamenti automatici delle versioni secondarie per i cluster di database Aurora 668
Aggiornamenti del sistema operativo 673
AWS Organizations implementazione degli aggiornamenti 682
Riavvio di un cluster o di un'istanza Aurora DB 686
Riavvio di un'istanza database in un cluster Aurora 687
Riavvio di un cluster Aurora con disponibilità di lettura 688
Riavvio di un cluster Aurora senza disponibilità di lettura 690
Verifica del tempo di attività per i cluster e le istanze Aurora 691

Esempi di operazioni di riavvio Aurora 694
Failover di un cluster di database Aurora 711
Eliminazione di cluster e istanze di Aurora 713
Eliminazione di un cluster database Aurora 713
Protezione dall'eliminazione per i cluster Aurora 721
Eliminazione di un cluster Aurora arrestato 722
Eliminazione di cluster Aurora MySQL che sono repliche di lettura 722
Lo snapshot finale durante l'eliminazione di un cluster 722
Eliminazione di un'istanza database da un cluster database Aurora 723
Applicazione di tag alle risorse Aurora e RDS 726
Perché utilizzare i tag RDS? 727
In che modo funzionano i tag RDS 728
Best practice 731
Copia di tag in snapshot di cluster database 732
Etichettatura dei backup automatici 733
Aggiunta ed eliminazione di tag in Amazon RDS 733
Tutorial: utilizza i tag per specificare i cluster DB Aurora da arrestare 738
ARN in Amazon RDS 742
Costruzione di un ARN 742
Recupero di un ARN esistente 750
Aggiornamenti di Aurora 754
Identificare la versione Amazon Aurora 755
Estensione del supporto di RDS 756
Panoramica del supporto esteso di RDS 757
Costi del supporto esteso di RDS 758
Evitare i costi del supporto esteso di RDS 759
Versioni con supporto esteso di RDS 759
Responsabilità con il supporto esteso per RDS 760
Responsabilità di Amazon Aurora 760
Le tue responsabilità 760
Creazione di un cluster di database Aurora o un cluster globale 761
Comportamento del supporto esteso di RDS 761
Considerazioni per il supporto esteso di RDS 762
Creazione di un cluster di database Aurora o un cluster globale con il supporto esteso di RDS 763
Visualizzazione della registrazione del supporto esteso per RDS 764

Visualizzazione delle date di supporto 767
Ripristino di un cluster di database Aurora o un cluster globale 769
Comportamento del supporto esteso di RDS 770
Considerazioni per il supporto esteso di RDS 770
Ripristino di un cluster di database Aurora o un cluster globale con il supporto esteso di RDS 771
Utilizzo di Deployments Blue/Green per gli aggiornamenti del database 773
Panoramica delle implementazioni blu/verde 774
Disponibilità di regioni e versioni 775
Vantaggi 775
Flusso di lavoro 775
Autorizzazione di accesso 782
Considerazioni e limitazioni 783
Best practice 792
Creazione di un'implementazione blu/verde 797
Preparazione di una implementazione blu/verde 797
Specifica delle modifiche 800
Creazione di un'implementazione blu/verde 800
Impostazioni disponibili 803
Visualizzazione di un'implementazione blu/verde 805
Switchover di un'implementazione blu/verde 809
Timeout dello switchover 810
Guardrail dello switchover 810
Azioni dello switchover 811
Best practice per lo switchover 812
CloudWatch Verifica delle metriche prima del passaggio al digitale 814
Monitoraggio del ritardo della replica prima dello switchover 814
Passaggio blue/green da una distribuzione all'altra 815
Dopo lo switchover 817
Eliminazione di un'implementazione blu/verde 819
Backup e ripristino di un cluster DB Aurora 823
Panoramica di backup e ripristino 824
Backup 824
Backup window (Finestra di backup) 825
Ripristino dei dati 828
Clonazione dei database 829

Backtrack 829
Mantenimento dei backup automatici 830
Periodo di conservazione 830
Costi di retention 831
Come impedire l’eliminazione dei backup automatici 831
Limitazioni 831
Visualizzazione dei backup automatici mantenuti 832
Eliminazione dei backup automatici mantenuti 832
Storage di backup 835
Storage di backup automatici 835
Archiviazione delle istantanee 836
Metriche CloudWatch per lo storage dei backup 836
Calcolo dell'utilizzo dello storage dei backup 837
Domande frequenti 838
Creazione di uno snapshot del cluster database 841
Determinare se lo snapshot è disponibile 843
Ripristino da uno snapshot cluster database 844
Gruppi di parametri 845
Gruppi di sicurezza 845
Considerazioni Aurora 846
Ripristino da uno snapshot 846
Copia di uno snapshot del cluster di database 849
Copia di uno snapshot di cluster di database tramite Console di gestione AWS 849
Limitazioni 851
Considerazioni 852
Copia di uno snapshot cluster di database non crittografata 855
Copia di uno snapshot di cluster di database crittografato 857
Copia di uno snapshot cluster database tra account 861
Condivisione di uno snapshot cluster database 865
Condivisione di uno snapshot 866
Condivisione di snapshot pubblici 869
Condivisione di snapshot crittografati 871
Interruzione della condivisione degli snapshot 875
Esportazione dei dati del cluster database in Amazon S3 877
Considerazioni 879
Configurazione dell'accesso a un bucket S3 881

Creazione di attività di esportazione del cluster di database ... 885
Monitoraggio delle esportazioni del cluster database ... 889
Annullamento di un'esportazione del cluster database ... 891
Risoluzione dei problemi ... 892
Esportazione dei dati dello snapshot del cluster di database in Amazon S3 ... 895
Considerazioni ... 897
Configurazione dell'accesso a un bucket S3 ... 910
Creazione di attività di esportazione di snapshot ... 916
Monitoraggio delle esportazioni di snapshot ... 920
Annullamento di un'esportazione di snapshot ... 922
Prestazioni di esportazione in Aurora MySQL ... 924
Risoluzione dei problemi ... 924
Point-in-time recupero ... 927
Ripristino point-in-time di un cluster di database ... 928
Ripristino point-in-time da un backup automatico mantenuto ... 931
Ripristino point-in-time (PITR) mediante AWS Backup ... 934
Eliminazione di una snapshot del cluster di database ... 940
Eliminazione di una snapshot del cluster di database ... 940
Tutorial: Ripristino di un cluster di database da uno snapshot ... 942
Ripristino di un cluster di database mediante la console ... 942
Ripristino di un cluster di database mediante la AWS CLI ... 947
Monitoraggio dei parametri in un cluster di database Aurora ... 954
Piano di monitoraggio ... 954
Baseline delle prestazioni ... 955
Linee guida per le prestazioni ... 955
Strumenti di monitoraggio ... 957
Strumenti di monitoraggio automatici ... 957
Strumenti di monitoraggio manuali ... 959
Visualizzazione dello stato del cluster ... 961
Visualizzazione di un cluster di database ... 962
Visualizzazione dello stato del cluster del DB ... 968
Visualizzazione dello stato dell’istanza database di in un cluster Aurora ... 972
Suggerimenti di Amazon Aurora ... 980
Visualizzazione dei suggerimenti ... 982
Applicazione delle raccomandazioni ... 990
Ignorare una raccomandazione ... 996

Modificare le raccomandazione ignorate in attive 998
Guida di riferimento sui suggerimenti 999
Visualizzazione delle metriche nella console Amazon RDS 1022
Visualizzazione della dashboard di Approfondimenti sulle prestazioni 1023
Scelta della nuova visualizzazione di monitoraggio nella scheda Monitoraggio 1024
Scelta della nuova visualizzazione di monitoraggio dalla pagina Approfondimenti sulle prestazioni 1026
Creazione di una dashboard personalizzata 1028
Scelta della dashboard preconfigurata 1031
Monitoraggio di Aurora con CloudWatch 1033
Panoramica di Amazon Aurora e Amazon CloudWatch 1034
Visualizza i parametri CloudWatch 1036
Esportazione dei parametri di Approfondimenti sulle prestazioni in Amazon CloudWatch ... 1041
Creazione di allarmi CloudWatch 1047
Monitoraggio con Database Insights 1049
Prezzi 1049
Supporto di classe di istanza, Regione e motore 1049
Attivazione della modalità avanzata 1053
Attivazione della modalità standard 1058
Monitoraggio delle query lente 1063
Considerazioni 1065
Monitoraggio del carico DB con Performance Insights 1066
Panoramica su Performance Insights 1067
Attivazione e disattivazione di Performance Insights 1078
Schema delle prestazioni per Aurora MySQL 1085
Policy di Performance Insights 1091
Analisi delle metriche utilizzando il pannello di controllo Performance Insights 1106
Visualizzazione dei consigli proattivi di Approfondimenti sulle prestazioni 1140
Recupero dei parametri con l'API Performance Insights 1143
Registrazione di log delle chiamate di Performance Insights mediante AWS CloudTrail 1168
Endpoint VPC (AWS PrivateLink) 1170
Analisi delle prestazioni con Guru for RDS DevOps 1174
Vantaggi di DevOps Guru for RDS 1175
Come funziona DevOps Guru for RDS 1176
Configurazione di Guru per RDS DevOps 1177
Monitoraggio del sistema operativo con il monitoraggio avanzato 1186

Panoramica sul monitoraggio avanzato ... 1186
Configurare e abilitare Monitoraggio avanzato ... 1188
Visualizzazione delle metriche nella console RDS ... 1195
Visualizzazione dell'utilizzo dei parametri del sistema operativo CloudWatch Logs ... 1198
Riferimento per i parametri di Aurora ... 1199
CloudWatch metriche per Aurora ... 1199
CloudWatch dimensioni per ... 1240
Disponibilità dei parametri Aurora nella console Amazon RDS ... 1241
Metriche CloudWatch per Performance Insights ... 1245
Metriche contatore per Performance Insights ... 1248
Statistiche SQL per Performance Insights ... 1281
Metriche del sistema operativo nel monitoraggio avanzato ... 1290
Monitoraggio di eventi, registri e flussi di attività di database ... 1299
Visualizzazione di registri, eventi e flussi nella console Amazon RDS ... 1300
Monitoraggio di eventi Aurora ... 1305
Panoramica degli eventi per Aurora ... 1305
Visualizzazione di eventi Amazon RDS ... 1307
Utilizzo della notifica degli eventi di Amazon RDS ... 1311
Creazione di una regola che si attiva su un evento Amazon Aurora ... 1337
Categorie di eventi e messaggi di evento di Amazon RDS per Aurora ... 1341
Monitoraggio dei registri di Aurora ... 1379
Visualizzazione ed elenco dei file di log del database ... 1379
Download di un file di log di database ... 1381
Controllo di un file di log di database ... 1382
Pubblicazione in CloudWatch Logs ... 1384
Lettura dei contenuti del file di log con REST ... 1387
File di log del database MySQL ... 1388
File di log del database PostgreSQL ... 1398
Monitoraggio delle chiamate API di Aurora in CloudTrail ... 1409
Integrazione di CloudTrail con Amazon Aurora ... 1409
Voci del file di log Amazon Aurora ... 1410
Monitoraggio di Aurora tramite i flussi di attività del database ... 1415
Panoramica ... 1415
Prerequisiti di rete Aurora MySQL ... 1419
Avvio di un flusso di attività di database ... 1421
Recupero dello stato del flusso di attività ... 1425

Arresto di un flusso di attività di database 1426
Monitoraggio dei flussi di attività 1428
Esempi di policy IAM per flussi di attività 1466
Monitoraggio delle minacce con la protezione RDS GuardDuty 1469
Utilizzo di Aurora MySQL 1471
Panoramica di Aurora MySQL 1472
Miglioramenti alle prestazioni di Amazon Aurora MySQL 1472
Aurora MySQL e dati spaziali 1473
Aurora MySQL versione 3 compatibile con MySQL 8.0 1474
Aurora MySQL versione 2 compatibile con MySQL 5.7 1509
Sicurezza con Aurora MySQL 1512
Privilegi dell'utente master con Aurora MySQL. 1513
Connessioni TLS 1514
Aggiornamento delle applicazioni per i nuovi certificati TLS 1523
Determinazione se un'applicazione si connette al cluster databse Aurora MySQL utilizzando SSL 1524
Determinare se un client richiede la verifica del certificato per la connessione 1524
Aggiornare l'archivio di trust delle applicazioni 1526
Codice Java di esempio per stabilire connessioni TLS 1526
Utilizzo dell'autenticazione Kerberos per Aurora MySQL 1528
Panoramica dell'autenticazione Kerberos per Aurora MySQL 1529
Limitazioni 1530
Configurazione dell'autenticazione Kerberos per Aurora MySQL 1531
Connessione ad Aurora MySQL con l'autenticazione Kerberos 1542
Gestione di un cluster di database in un dominio 1546
Migrazione di dati ad Aurora MySQL 1548
Migrazione da un database MySQL esterno a Aurora MySQL 1553
Migrazione da un'istanza database MySQL a Aurora MySQL 1581
Gestione di Aurora MySQL 1608
Gestione delle prestazioni e del dimensionamento per Amazon Aurora MySQL 1608
Backtrack di un cluster database 1620
Test di Amazon Aurora MySQL mediante query Fault Injection 1642
Alterazione delle tabelle in Amazon Aurora mediante Fast DDL 1647
Visualizzazione dello stato del volume per un cluster di database Aurora 1654
Ottimizzazione di Aurora MySQL 1656
Concetti essenziali per la regolazione di Aurora MySQL 1656

Regolazione di Aurora MySQL con eventi di attesa 1660
Regolazione di Aurora MySQL con stati del thread 1718
Ottimizzazione di Aurora MySQL con approfondimenti proattivi di Amazon DevOps Guru .. 1726
Query parallela per Aurora MySQL 1732
Panoramica delle query in parallelo 1732
Creazione di un cluster abilitato per le query in parallelo 1736
Attivazione e disattivazione della query parallela 1741
Ottimizzazione della funzionalità di query in parallelo 1744
Verifica dell'utilizzo della funzione di query in parallelo 1749
Monitoraggio della funzionalità di query in parallelo 1753
Costrutti SQL per query in parallelo 1761
Audit avanzato con Aurora MySQL 1783
Abilitazione dell'audit avanzato 1783
Visualizzazione dei log di audit 1787
Dettagli dei log di audit 1787
Replica con Aurora MySQL 1789
Repliche di Aurora 1789
Opzioni di replica 1791
Prestazioni di replica 1792
Filtri di replica 1792
Monitoraggio della replica 1800
Replica in più regioni 1801
Utilizzo della replica del log binario (binlog) 1817
Replica basata su GTID 1862
Inoltro di scrittura locale 1869
Abilitazione dell'inoltro di scrittura locale 1870
Verifica se l'inoltro di scrittura è abilitato un cluster database 1872
Compatibilità delle applicazioni e di SQL con l'inoltro di scrittura 1873
Livelli di isolamento per l'inoltro di scrittura 1875
Consistenza di lettura 1875
Esecuzione di istruzioni a più parti con inoltro scrittura 1879
Transazioni con inoltro di scrittura 1880
Parametri di configurazione per l'inoltro di scrittura 1880
Parametri per l'inoltro di scrittura 1882
Identificazione delle transazioni e delle query inoltrate 1886
Integrazione di Aurora MySQL con i servizi AWS 1889

Autorizzazione di Aurora MySQL per accedere ai servizi AWS 1889
Caricamento di dati da file di testo in Amazon S3 1909
Salvataggio dei dati dei file di testo in Amazon S3 1923
Chiamare una funzione Lambda da Aurora MySQL 1935
Pubblicazione di log Aurora MySQL in CloudWatch Logs 1947
Modalità di laboratorio per Aurora MySQL 1953
Caratteristiche della modalità di laboratorio per Aurora 1953
Best practice con Aurora MySQL 1955
Determinazione dell'istanza database a cui si è connessi 1956
Best practice per le prestazioni e il dimensionamento 1956
Best practice per l’elevata disponibilità 1966
Suggerimenti per le caratteristiche MySQL 1968
Valutazione dell’utilizzo dell’istanza database per Aurora MySQL con le metriche di Amazon CloudWatch 1976
Risoluzione dei problemi delle prestazioni di Aurora MySQL 1978
Opzioni di monitoraggio AWS 1978
Cause più comuni dei problemi di prestazioni del database 1979
Risoluzione dei problemi di carico di lavoro 1980
Registrazione di log per Aurora MySQL 2011
Risoluzione dei problemi di connessione al database 2014
Risoluzione dei problemi delle prestazioni delle query 2031
Riferimento Aurora MySQL 2036
Parametri di configurazione 2036
Variabili di stato globali 2112
Eventi di attesa 2130
Stati del thread 2136
Livelli di isolamento 2141
Suggerimenti 2148
Informazioni di riferimento sulle stored procedure 2152
Tabelle information_schema 2200
Aggiornamenti di Aurora MySQL 2209
Verifica dei numeri di versione 2210
Versioni beta e con supporto a lungo termine 2212
Preparazione per la fine del ciclo di vita di Aurora MySQL versione 2 2214
Preparazione per la fine del ciclo di vita di Aurora MySQL versione 1 2220
Aggiornamento dei cluster database Amazon Aurora MySQL 2223

Aggiornamenti e correzioni per il motore del database per Amazon Aurora MySQL 2360
Utilizzo di Aurora PostgreSQL 2361
L'ambiente di anteprima del database 2363
Tipi di classi di istanza database supportate 2363
Funzionalità non supportate nell’ambiente di anteprima 2364
Creazione di un nuovo cluster di database nell’ambiente di anteprima 2365
PostgreSQL versione 17 nell’ambiente di anteprima del database 2366
Sicurezza con Aurora PostgreSQL 2367
Informazioni su ruoli e autorizzazioni di PostgreSQL 2368
Sicurezza dei dati Aurora PostgreSQL con SSL/TLS 2385
Mascheratura dinamica Aurora PostgreSQL 2398
Aggiornamento delle applicazioni per i nuovi certificati SSL/TLS 2434
Determinare se un'applicazione si connette al cluster DB Aurora PostgreSQL utilizzando SSL 2435
Determinare se un client richiede la verifica del certificato per la connessione 2436
Aggiornare l'archivio di trust delle applicazioni 2437
Utilizzo delle connessioni SSL/TLS per diversi tipi di applicazioni 2437
Utilizzo di Autenticazione Kerberos 2439
Disponibilità di regioni e versioni 2440
Panoramica dell'autenticazione Kerberos 2440
Configurazione 2441
Gestione di un cluster di database Aurora PostgreSQL in un dominio Active Directory 2456
Connessione con Autenticazione Kerberos 2458
Utilizzo dei gruppi di sicurezza AD per il controllo degli accessi di Aurora PostgreSQL 2461
Migrazione di dati ad Aurora PostgreSQL 2473
Migrazione di un'istanza database RDS for PostgreSQL tramite uno snapshot 2475
Migrazione di un'istanza database RDS for PostgreSQL tramite una replica di lettura Aurora 2482
Ottimizzazione delle prestazioni delle query in Aurora PostgreSQL 2496
Prestazioni delle query migliorate con Letture ottimizzate per Aurora 2497
Ottimizzazione delle sottoquery correlate in Aurora PostgreSQL 2504
Miglioramento delle prestazioni delle query utilizzando join adattivo 2511
Utilizzo delle tabelle non registrate in Aurora PostgreSQL 2513
Creazione di tabelle non registrate 2513
Gestione delle tabelle non registrate durante la migrazione 2514
Conversione di tabelle non registrate in tabelle registrate 2514

Tabelle non registrate e replica logica ... 2515
Funzionamento di PostgreSQL Autovacuum ... 2515
Allocazione di memoria per il vacuum ... 2516
Riduzione della probabilità che si verifichi il wraparound dell'ID delle transazioni ... 2518
Determinare se le tabelle nel database devono essere sottoposte a vacuum ... 2519
Determinare quali tabelle sono attualmente idonee per l'Autovacuum ... 2520
Determinare se l'Auotvacuum è attualmente in esecuzione e per quanto tempo ... 2522
Esecuzione di un congelamento manuale del vacuum ... 2523
Indicizzare di nuovo una tabella quando l'autovacuum è in esecuzione ... 2525
Gestione di autovacuum con indici di grandi dimensioni ... 2527
Altri parametri che influenzano l'autovacuum ... 2530
Impostazione dei parametri di autovacuum a livello tabella ... 2531
Registrazione delle attività di autovacuum e vacuum ... 2532
Comprensione del comportamento della funzionalità di autovacuum con database non validi ... 2533
Identificazione dei blocchi dei processi di vacuum ... 2535
Gestione della contesa TOAST OID ... 2560
Comprendere le operazioni TOAST ... 2561
Identificazione delle sfide prestazionali ... 2561
Raccomandazioni ... 2563
Monitoraggio ... 2564
Utilizzo di Babelfish per Aurora PostgreSQL ... 2569
Limitazioni di Babelfish ... 2571
Comprendere l'architettura e la configurazione di Babelfish ... 2572
Creazione di un cluster database Babelfish per Aurora PostgreSQL ... 2618
Migrazione di un database SQL Server a Babelfish ... 2629
Autenticazione del database con Babelfish per Aurora PostgreSQL ... 2639
Connessione a un cluster database Babelfish ... 2658
Utilizzo di Babelfish ... 2671
Risoluzione dei problemi relativi a Babelfish ... 2755
Disattivazione di Babelfish ... 2757
Gestione degli aggiornamenti della versione di Babelfish ... 2758
Informazioni di riferimento su Babelfish ... 2781
Prestazioni e dimensionamento per Aurora PostgreSQL ... 2839
Dimensionamento delle istanze database Aurora PostgreSQL ... 2840
Numero massimo connessioni ... 2841

Limiti di storage temporaneo 2842
Huge Pages per Aurora PostgreSQL 2846
Test di Amazon Aurora PostgreSQL mediante query Fault Injection 2847
Visualizzazione dello stato del volume per un cluster di database Aurora 2853
Specificare il disco RAM per stats_temp_directory 2854
Gestione dei file temporanei con PostgreSQL 2855
Sintonizzazione degli eventi di attesa per Aurora PostgreSQL 2861
Concetti essenziali per l'ottimizzazione di Aurora PostgreSQL 2863
Eventi di attesa Aurora PostgreSQL 2868
Client:ClientRead 2870
Client:ClientWrite 2874
CPU 2876
IO: BufFileRead e IO: BufFileWrite 2882
IO:DataFileRead 2891
IO:XactSync 2902
IPC:DamRecordTxAck 2905
Eventi di attesa IPC:parallel 2906
IPC: ProcArrayGroupUpdate 2912
Lock:advisory 2915
Lock:extend 2918
Lock:Relation 2921
Lock:transactionid 2927
Lock:tuple 2930
LWLock:buffer_content (BufferContent) 2935
LWLock:buffer_mapping 2939
LWLock:BufferIO (IPC:BufferIO) 2942
LWLock:lock_manager 2944
LWLock:MultiXact 2949
LWLock:pg_stat_statements 2953
Timeout: PG Sleep 2957
Ottimizzazione di DevOps 2958
Il database ha una connessione di transazione inattiva da molto tempo 2958
Best practice con Aurora PostgreSQL 2962
Evitare il rallentamento delle prestazioni, il riavvio automatico e il failover per le istanze database Aurora PostgreSQL 2962
Diagnosi delle dimensioni della tabella e dell'indice 2963

Migliore gestione della memoria in Aurora PostgreSQL 2967
Failover rapido 2969
Ripristino rapido dopo il failover 2981
Gestione dell’abbandono della connessione 2988
Gestione delle connessioni interrotte in PostgreSQL 2997
Ottimizzazione dei parametri di memoria per Aurora PostgreSQL 3000
Analizza l'utilizzo delle risorse con i parametri CloudWatch 3009
Utilizzo della replica logica per un aggiornamento a una versione principale 3013
Risoluzione dei problemi di archiviazione in Aurora PostgreSQL 3022
Replica con Aurora PostgreSQL 3023
Repliche di Aurora 3024
Miglioramento della disponibilità di lettura delle repliche Aurora 3025
Monitoraggio della replica 3027
Panoramica della replica logica 3028
Configurazione della replica logica 3029
Disattivazione della replica logica 3031
Monitoraggio della cache di scrittura e degli slot logici 3032
Esempio: Utilizzo della replica logica 3034
Esempio: replica logica con Aurora PostgreSQL e AWS DMS 3036
Configurazione dell'autenticazione IAM per le connessioni di replica logica 3039
Inoltro di scrittura locale in Aurora PostgreSQL 3042
Limitazioni e considerazioni sull’inoltro di scrittura locale in Aurora PostgreSQL 3043
Configurazione di Aurora PostgreSQL per l’inoltro di scrittura locale 3044
Utilizzo dell’inoltro di scrittura locale per Aurora PostgreSQL 3047
Monitoraggio dell’inoltro di scrittura locale in Aurora PostgreSQL 3051
Utilizzo di Aurora PostgreSQL come Knowledge Base per Amazon Bedrock 3057
Prerequisiti 3058
Preparazione di Aurora PostgreSQL per l’utilizzo come Knowledge Base 3059
Creazione di una Knowledge Base nella console Bedrock 3062
Creazione rapida di una Knowledge Base per Amazon Bedrock di Aurora PostgreSQL 3063
Integrazione di Aurora PostgreSQL con i servizi AWS 3066
Importazione di dati da Amazon S3 in Aurora PostgreSQL 3067
Esportazione di dati PostgreSQL in Amazon S3 3087
Richiamare una funzione Lambda da Aurora PostgreSQL 3105
Pubblicazione di log Aurora PostgreSQL in CloudWatch Logs 3121

Monitoraggio dei piani di esecuzione delle query e della memoria di picco per Aurora PostgreSQL ... 3133
Accesso ai piani di esecuzione delle query e alla memoria di picco utilizzando le funzioni Aurora ... 3134
Documentazione di riferimento dei parametri per i piani di esecuzione delle query Aurora PostgreSQL ... 3135
Gestione dei piani di esecuzione delle query per Aurora PostgreSQL ... 3139
Panoramica della gestione del piano di query per Aurora PostgreSQL ... 3140
Best practice per la gestione del piano di query Aurora PostgreSQL ... 3148
Gestione del piano di query ... 3151
Acquisizione dei piani di esecuzione Aurora PostgreSQL ... 3153
Utilizzo dei piani gestiti per Aurora PostgreSQL ... 3156
Esame dei piani di query Aurora PostgreSQL nella vista dba_plans ... 3161
Miglioramento dei piani ... 3162
Eliminazione dei piani ... 3166
Esportazione e importazione dei piani gestiti ... 3168
Documentazione di riferimento dei parametri ... 3169
Informazioni di riferimento sulle funzioni ... 3177
Riferimento per la visualizzazione apg_plan_mgmt.dba_plans ... 3188
Funzionalità avanzate della gestione del piano di query ... 3193
Utilizzo di estensioni e wrapper di dati esterni ... 3208
Utilizzo del supporto delle estensioni delegate di Amazon Aurora per PostgreSQL ... 3209
Gestione di oggetti di grandi dimensioni in maniera più efficiente con il modulo lo ... 3223
Gestione dei dati spaziali con PostGIS ... 3227
Gestione delle partizioni con l'estensione pg_partman ... 3237
Pianificazione della manutenzione con l'estensione pg_cron ... 3243
Utilizzo di pgAudit per registrare l'attività del database ... 3253
Utilizzo di pglogical per sincronizzare i dati ... 3266
Wrapper di dati esterni supportati per Amazon Aurora PostgreSQL ... 3280
Utilizzo di Trusted Language Extensions per PostgreSQL ... 3296
Terminologia ... 3297
Requisiti per l'utilizzo di Trusted Language Extensions ... 3298
Impostazione di Trusted Language Extensions ... 3301
Panoramica di Trusted Language Extensions ... 3305
Creazione di estensioni TLE ... 3307
Eliminazione delle estensioni TLE da un database ... 3312

Disinstallazione di Trusted Language Extensions ... 3313
Utilizzo di hook PostgreSQL con le estensioni TLE ... 3314
Riferimento sulle funzioni per Trusted Language Extensions ... 3320
Riferimento sugli hook per Trusted Language Extensions ... 3334
Riferimento Aurora PostgreSQL ... 3338
Fascicolazioni Aurora PostgreSQL per EBCDIC e altre migrazioni del mainframe ... 3338
Regole di confronto supportate in Aurora PostgreSQL ... 3340
Informazioni di riferimento sulle funzioni Aurora PostgreSQL ... 3341
Aurora PostgreSQL parametri ... 3399
Eventi di attesa Aurora PostgreSQL ... 3462
Aggiornamenti di Aurora PostgreSQL ... 3492
Identificazione delle versioni di Amazon Aurora PostgreSQL ... 3492
versioni di Aurora PostgreSQL ... 3494
Versioni delle estensioni per Aurora PostgreSQL ... 3495
Aggiornamento dei cluster database Amazon Aurora PostgreSQL ... 3495
Uso di versioni con supporto a lungo termine (Long-Term Support, LTS) ... 3524
Utilizzo di Aurora PostgreSQL Limitless Database ... 3527
Architettura di Limitless Database ... 3528
Termini chiave ... 3530
Tipo di tabella ... 3532
Fatturazione ... 3532
Nozioni di base su Limitless Database ... 3533
Requisiti e considerazioni su Aurora PostgreSQL Limitless Database ... 3534
Requisiti ... 3534
Considerazioni ... 3535
Funzionalità non supportate ... 3537
Prerequisiti per l'utilizzo di Limitless Database ... 3538
Abilitazione delle operazioni dei gruppi di shard del database ... 3538
Creazione di un cluster di database che utilizza Limitless Database ... 3539
Correlazione della capacità massima con router e shard ... 3539
Creazione di un cluster di database ... 3542
Utilizzo di gruppi di shard di database ... 3552
Connessione al cluster di database Aurora PostgreSQL Limitless DB ... 3553
Individuazione del numero di router e di shard in un gruppo di shard di database ... 3554
Descrizione dei gruppi di shard di database ... 3554
Riavvio di un gruppo di shard di database ... 3555

Modifica della capacità di un gruppo di shard del database ... 3556
Suddivisione di uno shard ... 3558
Aggiunta di un router ... 3564
Eliminazione di un gruppo di shard del database ... 3568
Aggiunta di un gruppo di shard di database a un cluster di database Limitless Database esistente ... 3569
Creazione di tabelle di Limitless Database ... 3575
Creazione di tabelle Limitless utilizzando variabili ... 3576
Conversione di tabelle standard in tabelle Limitless ... 3581
Schemi di esempio ... 3585
Caricamento dei dati in Limitless Database ... 3587
Utilizzo del comando COPY con Limitless Database ... 3589
Utilizzo dell'utility di caricamento dei dati di Limitless Database ... 3591
Esecuzione di query su Limitless Database ... 3620
Query a shard singolo ... 3621
Query distribuite ... 3631
Tracciamento distribuito delle query ... 3632
Deadlocks distribuiti ... 3640
Gestione di Limitless Database ... 3643
Considerazioni sulle dimensioni di database e tabelle ... 3643
Recupero dello spazio di archiviazione tramite il processo di vacuum ... 3644
Monitoraggio di Limitless Database ... 3648
Monitoraggio di Limitless Database con Amazon CloudWatch ... 3649
Monitoraggio di Limitless Database con Database Insights ... 3654
Monitoraggio di Limitless Database con CloudWatch Logs ... 3667
Monitoraggio di Limitless Database con Monitoraggio avanzato ... 3668
Monitoraggio di Limitless Database con Approfondimenti sulle prestazioni ... 3669
Monitoraggio di Limitless Database con la protezione RDS GuardDuty ... 3677
Funzioni e visualizzazioni per Limitless Database ... 3678
Eventi di attesa per Limitless Database ... 3704
Backup e ripristino di Limitless Database ... 3717
Backup di un cluster di database che utilizza Aurora PostgreSQL Limitless Database ... 3717
Ripristino di un cluster di database che utilizza Aurora PostgreSQL Limitless Database ... 3719
Le utilità di backup e ripristino di PostgreSQL non sono supportate ... 3722
Aggiornamento di Limitless Database ... 3723

Aggiornamento di cluster di database che utilizzano Aurora PostgreSQL Limitless Database ... 3723
Riferimento a Limitless Database ... 3725
Comandi DDL ... 3725
Informazioni e limitazioni DDL ... 3734
Comandi DML ... 3773
Limitazioni e informazioni DML ... 3776
Variables ... 3780
Parametri del cluster di database ... 3781
Utilizzo di Database globale Aurora ... 3782
Panoramica di Database globale Aurora ... 3782
Vantaggi di Database globale Amazon Aurora ... 3784
Disponibilità di regioni e versioni ... 3785
Limitazioni di Database globale Aurora ... 3785
Nozioni di base sul Database globale Aurora ... 3787
Requisiti di configurazione ... 3788
Creazione di un database globale ... 3790
Aggiunta di un cluster secondario ... 3804
Creazione di un cluster headless secondario ... 3809
Creazione di un database globale da uno snapshot ... 3812
Gestione di un database globale Aurora ... 3813
Modifica di un database globale Aurora ... 3814
Modifica dei parametri del database globale ... 3816
Rimozione di un cluster da un database globale Aurora ... 3816
Eliminazione di un database globale Aurora ... 3820
Applicazione di tag per Database globale Aurora ... 3822
Connessione a un Database globale Aurora ... 3827
Scelta dell’endpoint che soddisfa le esigenze dell’applicazione ... 3827
Visualizzazione degli endpoint di un database globale ... 3829
Considerazioni sull’utilizzo degli endpoint di scrittura globali ... 3832
Utilizzo dell'inoltro di scrittura in un database globale Aurora ... 3833
Utilizzo dell'inoltro di scrittura in Aurora MySQL ... 3833
Utilizzo dell'inoltro di scrittura in Aurora PostgreSQL ... 3854
Utilizzo dello switchover o failover per un Database globale Aurora ... 3869
Pianificazione della strategia BCDR ... 3870
Esecuzione di switchover ... 3871

Ripristino da un’interruzione non pianificata ... 3878
Gestione degli RPO (Aurora PostgreSQL) ... 3889
Resilienza tra Regioni per i cluster secondari ... 3895
Monitoraggio di un database globale Aurora ... 3895
Monitoraggio di un database globale Aurora con Performance Insights ... 3897
Monitoraggio dei database globali Aurora con i flussi di attività di database ... 3898
Monitoraggio dei database globali basati su Aurora MySQL ... 3898
Monitoraggio dei database globali Aurora basati su PostgreSQL ... 3902
Utilizzo delle Blue/Green distribuzioni per Aurora Global Database ... 3905
Vantaggi dell'utilizzo delle Blue/Green implementazioni con Aurora Global Database ... 3906
Come funzionano Blue/Green le implementazioni con Aurora Global Database ... 3906
Utilizzo di database globali Aurora con altri servizi AWS ... 3909
Aggiornamento di un database globale Amazon Aurora ... 3910
Aggiornamenti di una versione principale ... 3911
Aggiornamenti della versione secondaria ... 3912
Server proxy per Amazon RDS ... 3915
Disponibilità di regioni e versioni ... 3917
Quote e limiti ... 3917
Limitazioni di MySQL ... 3919
Limitazioni di PostgreSQL ... 3920
Pianificazione sull'utilizzo di RDS Proxy ... 3921
Concetti e terminologia RDS Proxy ... 3923
Panoramica dei concetti RDS Proxy ... 3923
Pooling di connessioni ... 3925
Sicurezza ... 3925
Failover ... 3928
Transazioni ... 3929
Nozioni di base su RDS Proxy ... 3930
Configurazione di una rete proxy ... 3930
Impostazione delle credenziali del database ... 3936
Configurazione dell'autenticazione IAM ... 3940
Creazione di un proxy ... 3946
Visualizzazione di un proxy ... 3959
Collegamento tramite RDS Proxy ... 3961
Gestire un RDS Proxy ... 3965
Modifica di un RDS Proxy ... 3965

Aggiunta di un utente di database ... 3973
Passaggio all'autenticazione end-to-end IAM ... 3975
Considerazioni sulla connessione a Server proxy per RDS ... 3978
Evitare di effettuare il pinning di Server proxy per RDS ... 3982
Eliminazione di un RDS Proxy ... 3988
Utilizzo degli endpoint RDS Proxy ... 3989
Panoramica degli endpoint proxy ... 3990
Limitazioni per gli endpoint proxy ... 3991
Utilizzo degli endpoint di lettura con cluster Aurora ... 3991
Accesso ai database Aurora RDS VPCs ... 3996
Creazione di un endpoint proxy ... 3997
Visualizzazione degli endpoint proxy ... 4000
Modifica di un endpoint proxy ... 4002
Eliminazione di un endpoint proxy ... 4003
Monitoraggio di RDS Proxy con CloudWatch ... 4004
Utilizzo degli eventi RDS Proxy ... 4013
Eventi RDS Proxy ... 4014
Esempi di RDS Proxy ... 4016
Risoluzione dei problemi RDS Proxy ... 4019
Verifica della connettività a un proxy ... 4020
Problemi e soluzioni comuni ... 4022
Risoluzione dei problemi relativi a RDS per MySQL ... 4023
Risoluzione dei problemi per RDS per PostgreSQL ... 4025
Utilizzo di RDS Proxy con AWS CloudFormation ... 4031
Utilizzo del Server proxy per RDS con i database globali Aurora ... 4032
Limitazioni di Server proxy per RDS con i database globali ... 4033
Come funzionano gli endpoint Server proxy per RDS con i database globali ... 4033
Integrazioni Zero-ETL ... 4035
Vantaggi ... 4037
Concetti chiave ... 4037
Limitazioni ... 4038
Limitazioni generali ... 4038
Aurora MySQL ... 4039
Limitazioni di Aurora PostgreSQL ... 4040
Limitazioni di Amazon Redshift ... 4040
Amazon SageMaker AI limitazioni di Lakehouse ... 4041

Quote 4041
Regioni supportate 4041
Guida introduttiva alle integrazioni Zero-ETL 4041
Fase 1: creazione di un gruppo di parametri del cluster di database personalizzato 4042
Passaggio 2: seleziona o crea un cluster di database di origine 4044
Passaggio 3a: creazione di un data warehouse di destinazione 4044
Configura un'integrazione utilizzando AWS SDKs 4046
Fase 3b: creare un AWS Glue catalogo per l'integrazione con Amazon SageMaker AI Zero-ETL 4056
Fasi successive 4059
Creazione di integrazioni Zero-ETL con Amazon Redshift 4059
Prerequisiti 4060
Autorizzazioni richieste 4060
Creazione di integrazioni Zero-ETL 4063
Integrazioni crittografate 4067
Fasi successive 4069
Creazione di integrazioni Zero-ETL con un lakehouse Amazon SageMaker 4070
Prerequisiti 4070
Autorizzazioni richieste 4070
Creazione di integrazioni Zero-ETL con un lakehouse Amazon SageMaker 4073
Integrazioni crittografate 4078
Fasi successive 4080
Filtro dei dati per le integrazioni Zero-ETL 4080
Formato di un filtro dei dati 4081
Logica dei filtri 4084
Precedenza dei filtri 4085
Esempi di Aurora MySQL 4085
Esempi di Aurora PostgreSQL 4086
Aggiunta di filtri dei dati 4087
Rimozione di filtri dei dati 4089
Aggiunta di dati ed esecuzione di query 4089
Creazione di un database di destinazione 4090
Aggiunta di dati al cluster di database di origine 4090
Interrogazione dei dati Aurora in Amazon Redshift 4091
Differenze dei tipi di dati 4093
Operazioni DDL per Aurora PostgreSQL 4105

Visualizzazione e monitoraggio delle integrazioni Zero-ETL 4108
Visualizzazione delle integrazioni 4108
Monitoraggio tramite tabelle di sistema 4110
Monitoraggio con EventBridge 4111
Modifica delle integrazioni Zero-ETL 4111
Eliminazione delle integrazioni Zero-ETL 4112
Risoluzione dei problemi delle integrazioni Zero-ETL 4113
Non riesco a creare un'integrazione Zero-ETL 4114
La mia integrazione è bloccata in uno stato di `Syncing` 4115
Le mie tabelle non si replicano su Amazon Redshift 4115
Una o più tabelle Amazon Redshift richiedono una risincronizzazione 4116
Problemi di integrazione non riuscita per le integrazioni Zero-ETL di Amazon SageMaker AI Lakehouse 4119
Le operazioni DDL che non sono complete vengono visualizzate in Amazon Redshift 4120
Uso di Aurora Serverless v2 4121
Casi d'uso di Aurora Serverless v2 4121
Conversione dei carichi di lavoro con provisioning 4123
Vantaggi di Aurora Serverless v2 4124
Funzionamento di Aurora Serverless v2 4125
Panoramica 4126
Configurazioni cluster 4127
Capacità 4128
Dimensionamento 4131
Elevata disponibilità 4134
Storage 4135
Parametri di configurazione 4136
Requisiti e limitazioni per Aurora Serverless v2 4136
Disponibilità di regioni e versioni 4137
I cluster che utilizzano Aurora Serverless v2 devono avere un intervallo di capacità specificato 4137
Aurora Serverless v2 non suppporta alcune caratteristiche di provisioning 4138
Aurora Serverless v2 presenta alcuni aspetti diversi rispetto a Aurora Serverless v1 4139
Creazione di un cluster di database Aurora Serverless v2 4139
Impostazioni 4139
Creazione di un cluster di database Aurora Serverless v2 4141
Creazione di un'istanza Aurora Serverless v2 di scrittura 4146

Gestione di Aurora Serverless v2 4147
Impostazione dell'intervallo di capacità di Aurora Serverless v2 per un cluster 4148
Verifica dell'intervallo di capacità per Aurora Serverless v2 4155
Aggiunta di un'istanza Aurora Serverless v2 di lettura 4156
Conversione da istanza database con provisioning in Aurora Serverless v2 4158
Conversione da istanza Aurora Serverless v2 in istanza con provisioning 4159
Messa in pausa delle istanze database Aurora Serverless v2 4160
Scelta del livello di promozione per un'istanza Aurora Serverless v2 di lettura 4160
Utilizzo di TLS/SSL con Aurora Serverless v2 4161
Visualizzazione di istanze Aurora Serverless v2 di scrittura e lettura 4163
Registrazione per Aurora Serverless v2 4164
Prestazioni e dimensionamento per Aurora Serverless v2 4171
Scelta dell'intervallo di capacità 4172
Uso di gruppi di parametri per Aurora Serverless v2 4187
Esclusione di errori dovuti all'esaurimento della memoria 4192
Parametri di CloudWatch importanti 4194
Monitoraggio delle prestazioni di Aurora Serverless v2 con Approfondimenti sulle prestazioni 4198
Risoluzione dei problemi di capacità di Aurora Serverless v2 4199
Dimensionamento a 0 ACU con pausa e ripresa automatiche 4202
Panoramica della pausa automatica 4202
Prerequisiti e limitazioni 4204
Attivazione e disattivazione della pausa automatica 4205
Come funziona la pausa automatica 4209
Pausa automatica e configurazioni dei cluster Aurora 4213
Monitoraggio della pausa automatica 4217
Risoluzione dei problemi relativi alla pausa automatica 4221
Progettazione delle applicazioni per la pausa automatica 4223
Migrazione a Aurora Serverless v2 4224
Utilizzo di Aurora Serverless v2 con un cluster esistente 4225
Passaggio di un cluster con provisioning 4228
Confronto tra Aurora Serverless v2 e Aurora Serverless v1 4234
Aggiornamento da Aurora Serverless v1 a Aurora Serverless v2 4246
Migrazione di un database on-premise a Aurora Serverless v2 4249
Utilizzo Aurora Serverless v1 4250
Disponibilità di Regioni e versioni per Aurora Serverless v1 4251

Vantaggi di Aurora Serverless v1 ... 4251
Casi d'uso per Aurora Serverless v1 ... 4252
Limitazioni di Aurora Serverless v1 ... 4252
Requisiti di configurazione per Aurora Serverless v1 ... 4255
Utilizzo di TLS/SSL con Aurora Serverless v1 ... 4256
Suite di crittografia supportate per connessioni a cluster di database Aurora Serverless v1 4259
Funzionamento di Aurora Serverless v1 ... 4259
Architettura di Aurora Serverless v1 ... 4260
Auto Scaling ... 4261
Operazione di timeout ... 4262
Sospendi e riprendi ... 4264
Determinazione di max_connections ... 4265
Gruppi di parametri ... 4268
Registrazione di log ... 4271
Manutenzione ... 4275
Failover ... 4276
Snapshot ... 4276
Creazione di un cluster database Aurora Serverless v1 ... 4276
Ripristino di un cluster database Aurora Serverless v1 ... 4281
Modifica di un cluster di database Aurora Serverless v1 ... 4284
Modifica della configurazione di dimensionamento ... 4285
Aggiornamento della versione principale ... 4287
Conversione da istanza Aurora Serverless v1 in istanza con provisioning ... 4290
Considerazioni sulla conversione da un cluster di database Aurora Serverless v1 a un cluster con provisioning ... 4291
Dimensionamento manuale della capacità del cluster database Aurora Serverless v1 ... 4293
Visualizzazione dei cluster database Aurora Serverless v1 ... 4296
Monitoraggio dei cluster database Aurora Serverless v1 con CloudWatch ... 4300
Eliminazione di un cluster di database Aurora Serverless v1 ... 4300
Versioni del motore del database di Aurora Serverless v1 e Aurora ... 4304
Aurora MySQL Serverless ... 4305
Aurora PostgreSQL Serverless ... 4305
Aggiornamenti a versioni secondarie automatiche ... 4305
Utilizzo dell'API dati di Amazon RDS ... 4307
Disponibilità per Regione e versione dell'API dati di Amazon RDS ... 4308
IPv6 supporto ... 4308

IPv6 supporto per endpoint 4309
Utilizzo degli endpoint IPv6 4309
Utilizzo con AWS PrivateLink IPv6 4312
Considerazioni sulla migrazione 4312
Risoluzione dei problemi IPv6 di connessione 4313
Limitazioni 4313
API dati con Aurora Serverless v2 rispetto ad Aurora Serverless v1 4314
Numero massimo di richieste al secondo 4315
Abilitazione o disabilitazione dell'API dati di Amazon RDS su un database esistente 4315
Eventi CloudTrail 4315
Supporto per istruzioni multiple 4316
Richieste simultanee per lo stesso ID transazione 4317
Comportamento di BatchExecuteStatement 4318
Comportamento di ExecuteSQL 4318
Comportamento di ExecuteStatement 4318
Comportamento del parametro Schema 4319
Autenticazione e autorizzazione 4320
Autorizzazione basata su tag 4321
Archiviazione delle credenziali in un segreto di Secrets Manager 4323
Abilitazione dell'API dati 4324
Abilitazione dell'API dati di RDS durante la creazione di un database 4325
Abilitazione dell'API dati RDS su un database esistente 4326
Creazione di un endpoint Amazon VPC (PrivateLink) 4329
Chiamata dell'API dati 4332
Riferimento alle operazioni 4333
Chiamata da AWS CLI 4337
Chiamata dalle app Python 4348
Chiamata da app Java 4352
Controllo del comportamento del timeout 4357
Libreria client Java 4359
Download della libreria client Java per l'API dati 4359
Esempi di librerie client Java 4359
Elaborazione dei risultati delle query in formato JSON 4361
Recupero dei risultati delle query in formato JSON 4362
Mappatura dei tipi di dati 4362
Risoluzione dei problemi 4363

Esempi 4364
Risoluzione dei problemi relativi all'API dati 4369
La transazione <transaction_ID> non è stata trovata 4369
Il pacchetto per la query è troppo grande 4370
La risposta del database è andata oltre il limite delle dimensioni 4370
HttpEndpointnon è abilitato per il cluster <cluster_ID> 4370
DatabaseErrorException: Transaction sta ancora eseguendo una query 4371
Eccezione per risultato non supportato 4371
Le istruzioni multiple non sono supportate 4371
Il parametro Schema non è supportato 4372
IPv6 problemi di connettività 4372
Registrazione di log delle chiamate API dati 4373
Utilizzo delle informazioni dell'API dati in CloudTrail 4373
Inclusione ed esclusione di eventi API dati da un trail CloudTrail 4374
Informazioni sulle voci dei file di registro dell'API dei dati 4377
Monitoraggio delle query dell'API dati 4379
Come vengono rappresentate le query dell'API dati RDS in Approfondimenti sulle prestazioni 4379
Utilizzo dell'editor della query 4380
Disponibilità dell'editor di query 4380
Autorizzazione di accesso 4380
Esecuzione di query 4382
Riferimento all'API DBQMS 4386
CreateFavoriteQuery 4387
CreateQueryHistory 4387
CreateTab 4387
DeleteFavoriteQueries 4387
DeleteQueryHistory 4387
DeleteTab 4387
DescribeFavoriteQueries 4387
DescribeQueryHistory 4388
DescribeTabs 4388
GetQueryString 4388
UpdateFavoriteQuery 4388
UpdateQueryHistory 4388
UpdateTab 4388

Utilizzo di machine learning di Aurora 4389
Utilizzo di machine learning di Aurora con Aurora MySQL 4390
Requisiti per l'utilizzo di machine learning di Aurora 4391
Disponibilità di regioni e versioni 4392
Funzionalità supportate e limitazioni 4393
Configurazione del cluster Aurora per machine learning di Aurora 4394
Utilizzo di Amazon Bedrock con il cluster di database Aurora MySQL 4408
Utilizzo di Amazon Comprehend con il cluster database Aurora MySQL 4410
Utilizzo di SageMaker AI con il cluster di database Aurora MySQL 4412
Considerazioni sulle prestazioni 4416
Monitoraggio 4419
Utilizzo del machine learning di Aurora con Aurora PostgreSQL 4420
Requisiti per l'utilizzo del machine learning di Aurora 4421
Funzionalità e limitazioni supportate 4422
Configurazione del cluster database Aurora per utilizzare il machine learning di Aurora 4422
Utilizzo di Amazon Bedrock con il cluster di database Aurora PostgreSQL 4435
Utilizzo di Amazon Comprehend con il cluster database Aurora PostgreSQL 4438
Utilizzo dell' SageMaker intelligenza artificiale con il cluster Aurora PostgreSQL DB 4440
Esportazione di dati in Amazon S3 SageMaker per la formazione su modelli di intelligenza artificiale (Advanced) 4444
Considerazioni sulle prestazioni 4445
Monitoraggio 4451
Esempi di codice 4453
Nozioni di base 4464
Hello Aurora 4465
Informazioni di base 4475
Azioni 4645
Scenari 4849
Creazione di una REST API per la libreria di prestiti 4850
Creazione di un tracciatore di elementi di lavoro di Aurora Serverless 4851
Best practice con Aurora 4856
Linee guida operative di base per Amazon Aurora 4856
Suggerimenti relativi alla RAM per un'istanza di database 4857
Driver di database AWS 4858
Monitoraggio di Amazon Aurora 4858
Uso di gruppi di parametri database e gruppi di parametri cluster database 4859

Video sulle best practice Amazon Aurora 4859
Esecuzione di un proof of concept di Aurora 4860
Panoramica di un proof of concept di Aurora 4860
1. Individuare gli obiettivi 4861
2. Comprendere le caratteristiche del carico di lavoro 4862
3. Fare pratica con la console o la CLI 4863
Pratica con la console 4863
Pratica con la AWS CLI 4864
4. Creare il cluster Aurora 4864
5. Configurare lo schema 4866
6. Importare i dati 4867
7. Trasferire il codice SQL 4867
8. Specificare le impostazioni di configurazione 4868
9. Connettere ad Aurora 4869
10. Eseguire il carico di lavoro 4871
11. Misurare le prestazioni 4871
12. Fare pratica con la disponibilità elevata di Aurora 4875
13. Cosa fare in seguito 4877
Sicurezza 4879
Database authentication (Autenticazione del database) 4881
Autenticazione password 4882
Autenticazione del database IAM 4883
Autenticazione Kerberos 4883
Gestione delle password con Aurora e Secrets Manager 4885
Disponibilità di regioni e versioni 4885
Limitazioni 4885
Panoramica di 4886
Vantaggi 4886
Permissions 4887
Implementazione della gestione da parte di Aurora 4888
Gestione della password per un cluster di database 4889
Segreti rotanti per cluster di database 4893
Visualizzazione dei dettagli di un segreto per un cluster di database 4895
Protezione dei dati 4898
Crittografia dei dati 4899
Riservatezza del traffico Internet 4929

Gestione dell’identità e degli accessi ... 4931
Destinatari ... 4931
Autenticazione con identità ... 4932
Gestione dell’accesso tramite policy ... 4935
Funzionamento di Amazon Aurora con IAM ... 4937
Esempi di policy basate sull’identità ... 4944
Policy gestite da AWS ... 4967
Aggiornamenti alle policy ... 4973
Prevenzione del problema "confused deputy" tra servizi ... 4985
Autenticazione del database IAM ... 4987
Risoluzione dei problemi ... 5038
Registrazione di log e monitoraggio ... 5040
Convalida della conformità ... 5043
Resilienza ... 5044
Backup e ripristino ... 5044
Replica ... 5045
Failover ... 5045
Sicurezza dell'infrastruttura ... 5046
Gruppi di sicurezza ... 5046
Public accessibility (Accesso pubblico) ... 5046
Endpoint VPC (AWS PrivateLink) ... 5048
Considerazioni ... 1171
Disponibilità ... 1171
Creazione di un endpoint VPC dell'interfaccia ... 1171
Creazione di una policy di endpoint VPC ... 1172
Best practice relative alla sicurezza ... 5052
Controllo dell'accesso con i gruppi di sicurezza ... 5053
Panoramica dei gruppi di sicurezza VPC ... 5053
Scenario del gruppo di sicurezza ... 5054
Creazione di un gruppo di sicurezza VPC ... 5056
Associazione a un cluster database ... 5056
Privilegi dell'account utente master ... 5056
Ruoli collegati ai servizi ... 5059
Autorizzazioni del ruolo collegato ai servizi per Amazon Aurora ... 5059
Autorizzazioni del ruolo collegato ai servizi per Amazon RDS Beta ... 5062
Ruolo collegato ai servizi per Amazon RDS Preview ... 5063

Uso di Amazon Aurora con Amazon VPC ... 5065
Uso di un cluster database in un VPC ... 5065
Scenari per accedere a un cluster database in un VPC ... 5083
Tutorial: Creazione di un Amazon VPC da utilizzare con un cluster database (solo IPv4) ... 5090
Tutorial: Creazione di un VPC per l'utilizzo con un cluster database (modalità dual-stack) . 5098
Quote e vincoli ... 5109
Quote in Amazon Aurora ... 5109
Vincoli per la denominazione in Amazon Aurora ... 5115
Limiti di dimensione Amazon Aurora ... 5116
Risoluzione dei problemi ... 5118
Impossibile connettersi all'istanza database di ... 5118
Test della connessione a un'istanza di database ... 5121
Risoluzione di problemi di autenticazione della connessione ... 5122
Problemi relativi alla sicurezza ... 5122
Messaggio di errore "Impossibile recuperare gli attributi dell'account, alcune funzioni della console potrebbero non essere attive." ... 5122
Reimpostazione della password del ruolo di proprietario dell'istanza di database ... 5122
Errore o riavvio di un'istanza di database ... 5123
Modifiche ai parametri che non hanno effetto ... 5124
Problemi di memoria liberabile Aurora ... 5124
Problemi di replica di Aurora MySQL ... 5125
Diagnosi e risoluzione del ritardo tra repliche di lettura ... 5126
Diagnosi e risoluzione di un errore relativo alla replica di lettura MySQL ... 5128
Errore di replica interrotta ... 5130
La replica di lettura non riesce a inizializzare la struttura dei metadati ... 5130
Documentazione di riferimento dell'API Amazon RDS ... 5132
Uso dell'API query ... 5132
Parametri di query ... 5132
Autenticazione delle richieste di query ... 5133
Risoluzione dei problemi delle applicazioni ... 5133
Errore durante il recupero ... 5133
Suggerimenti per la risoluzione dei problemi ... 5134
Cronologia dei documenti ... 5135
AWSGlossario ... 5243
... 5244

Che cos'è Amazon Aurora?

Amazon Aurora (Aurora) è un motore del database relazionale completamente gestito compatibile con MySQL e PostgreSQL. Sei già al corrente di come MySQL e PostgreSQL combinano la velocità e l'affidabilità dei database commerciali di fascia alta e la semplicità e la convenienza dei database open source. Il codice, gli strumenti e le applicazioni che attualmente utilizzi con i database MySQL e PostgreSQL esistenti possono essere usati anche con Aurora. Con alcuni carichi di lavoro, Aurora assicura un throughput fino a cinque volte superiore a MySQL e fino al triplo di PostgreSQL e non richiede alcuna modifica alla maggior parte delle applicazioni esistenti.

Aurora include un sottosistema di storage ad alte prestazioni. I relativi motori di database compatibili con MySQL e PostgreSQL sono personalizzati per beneficiare dello storage distribuito e rapido. Lo storage sottostante cresce automaticamente in funzione delle necessità. Un volume del cluster Aurora può raggiungere una dimensione massima di 128 tebibytes (TiB). Aurora inoltre automatizza ed esegue la standardizzazione del clustering e della replica di database, ovvero due aspetti in genere tra i più impegnativi nell'ambito della configurazione e dell'amministrazione dei database.

Aurora fa parte del servizio di database gestito Amazon Relational Database Service (Amazon RDS). Amazon RDS semplifica la configurazione, l'uso e la scalabilità dei database relazionali nel cloud. Se è la prima volta che utilizzi Amazon RDS, consulta la Guida per l'utente di Amazon Relational Database Service. Per ulteriori informazioni sulle varie opzioni di database disponibili su Amazon Web Services, consulta Choosing the right database for your organization on AWS (Scelta del database appropriato per l'organizzazione su AWS).

Argomenti

- Modello di responsabilità condivisa di Amazon RDS
- Funzionamento di Amazon Aurora con Amazon RDS
- Cluster database Amazon Aurora
- Versioni di Amazon Aurora
- Regioni e zone di disponibilità
- Funzionalità supportate in Amazon Aurora by e Regione AWS Aurora DB engine
- Connessioni degli endpoint Amazon Aurora
- Classi di istanze database Amazon Aurora
- Archiviazione Amazon Aurora

- Affidabilità Amazon Aurora
- Sicurezza di Amazon Aurora
- Elevata disponibilità di Amazon Aurora
- Replica con Amazon Aurora
- Fatturazione delle istanze database per Aurora

Modello di responsabilità condivisa di Amazon RDS

Amazon RDS è responsabile dell'hosting dei componenti software e dell'infrastruttura delle istanze database e dei cluster di database. Tu sei responsabile dell'ottimizzazione delle query, ovvero il processo di ottimizzazione delle query SQL per migliorare le prestazioni. Le prestazioni delle query dipendono fortemente dalla progettazione del database, dalla dimensione dei dati, dalla distribuzione dei dati, dal carico di lavoro dell'applicazione e dai modelli di query, che possono variare notevolmente. Il monitoraggio e l'ottimizzazione sono processi altamente personalizzati che puoi usare per i tuoi database RDS. È possibile utilizzare Approfondimenti sulle prestazioni di Amazon RDS e altri strumenti per identificare le query problematiche.

Funzionamento di Amazon Aurora con Amazon RDS

Gli elementi seguenti illustrano in che modo Amazon Aurora è correlato ai motori MySQL e PostgreSQL standard disponibili in Amazon RDS.

- Aurora MySQL o Aurora PostgreSQL viene scelto come opzione di motore del database durante la configurazione di nuovi server di database mediante Amazon RDS.
- Aurora sfrutta le caratteristiche note di Amazon Relational Database Service (Amazon RDS) per la gestione e l'amministrazione. Aurora utilizza l'interfaccia Console di gestione AWS di Amazon RDS, i comandi AWS CLI e le operazioni API per gestire le attività di database di routine, come il provisioning, l'applicazione di patch, il backup, il ripristino, il rilevamento degli errori e la correzione.
- Le operazioni di gestione Aurora in genere implicano cluster interi di server di database sincronizzati mediante operazioni di replica anziché singole istanze database. Le funzioni automatiche di clustering, replica e allocazione dello storage semplificano e rendono più conveniente dal punto di vista dei costi la configurazione, l'utilizzo e il dimensionamento delle implementazioni MySQL e PostgreSQL più importanti.
- Puoi trasferire dati da Amazon RDS for MySQL e Amazon RDS for PostgreSQL in Aurora creando e ripristinando snapshot o configurando una replica unidirezionale. Puoi utilizzare strumenti di

migrazione per convertire, con la semplice pressione di un pulsante, le applicazioni RDS per MySQL e RDS per PostgreSQL in Aurora.

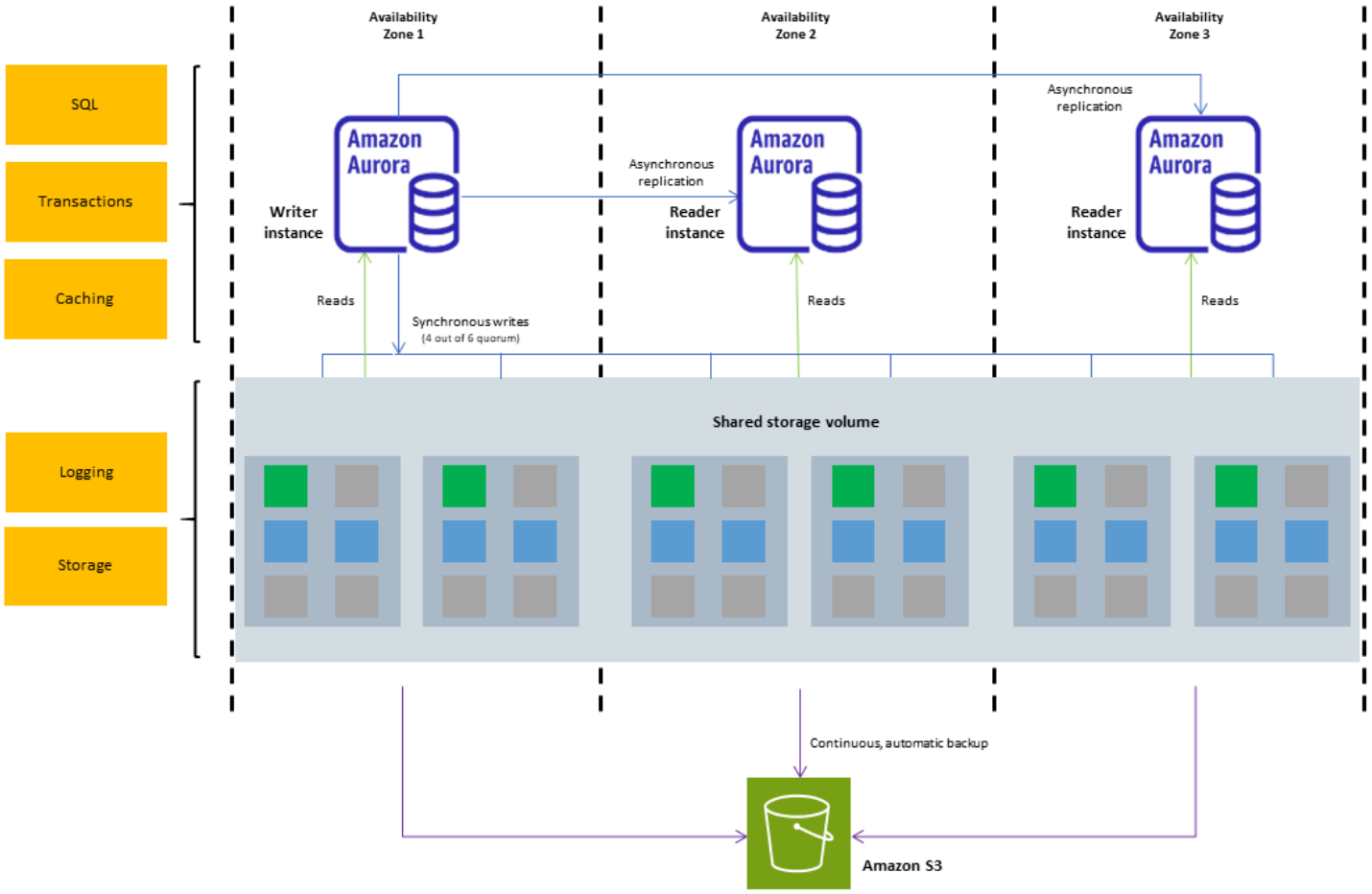
Prima di utilizzare Amazon Aurora, completa le fasi indicate in Configurazione dell'ambiente per Amazon Aurora e quindi esamina i concetti e le caratteristiche di Aurora in Cluster database Amazon Aurora.

Cluster database Amazon Aurora

Un cluster database Amazon Aurora è composto da una o più istanze database e da un volume del cluster che gestisce i dati per tali istanze. Un volume del cluster Aurora è uno storage di database virtuale che si estende su più zone di disponibilità, ciascuna delle quali include una copia dei dati del cluster database. Un cluster database Aurora è formato da due tipi di istanze:

- Istanza database primaria (di scrittura): supporta operazioni di lettura e scrittura ed esegue tutte le modifiche ai dati del volume del cluster. Ciascun cluster database Aurora ha un'istanza database primaria.
- Replica Aurora (istanza database di lettura): si connette allo stesso volume di archiviazione dell'istanza database primaria ma supporta solo operazioni di lettura. Oltre all'istanza database primaria, ciascun cluster database Aurora può avere fino a 15 repliche di Aurora. È possibile mantenere l'alta disponibilità posizionando le repliche Aurora in zone di disponibilità separate. Aurora esegue automaticamente il failover in una replica di Aurora nel caso in cui l'istanza database principale non sia più disponibile. È possibile specificare la priorità di failover per le repliche di Aurora. Le repliche di Aurora possono anche effettuare l'offload dei carichi di lavoro in lettura dall'istanza database primaria.

Il diagramma seguente mostra la relazione tra il volume del cluster, l'istanza database di scrittura e le istanze database di lettura in un cluster di database Aurora.



> **Note**
>
> Le informazioni precedenti si applicano a tutti i cluster di database Aurora: con provisioning, con query parallela, Database globale Aurora, Aurora Serverless, compatibili con Aurora MySQL e compatibili con Aurora PostgreSQL.

Il cluster di database Aurora illustra la separazione tra capacità di calcolo e archiviazione. Ad esempio, una configurazione Aurora con una sola istanza database è comunque un cluster, poiché il volume dello storage sottostante riguarda più nodi di storage distribuiti in più zone di disponibilità (AZ).

Le operazioni di input/output (I/O) nei cluster database Aurora vengono conteggiate allo stesso modo, indipendentemente dal fatto che si trovino su un'istanza database di scrittura o lettura. Per ulteriori informazioni, consulta Configurazioni dell'archiviazione per i cluster database Amazon Aurora.

Versioni di Amazon Aurora

Con Amazon Aurora, puoi scegliere il motore di database relazionale supportato che meglio si adatta ai requisiti delle tue applicazioni, mantenendo al contempo la compatibilità con i motori sottostanti. Aurora riutilizza il codice dei motori di database supportati, in modo da poter sfruttare le funzionalità, gli strumenti e le librerie esistenti per tali motori. Quando crei un cluster, devi specificare la versione del motore di database Amazon Aurora che desideri utilizzare. La versione scelta determina la compatibilità e le funzionalità disponibili.

Questa documentazione spiega i punti in comune e le differenze tra Amazon Aurora e i motori di database corrispondenti. Sulla base di queste informazioni, puoi determinare la versione del software da selezionare e capire come verificare le funzionalità disponibili e le correzioni di bug in ciascuna versione. Puoi utilizzare questo riferimento anche per determinare la cadenza di aggiornamento appropriata e pianificare l'aggiornamento.

Motori di database supportati per i cluster di database Amazon Aurora

I seguenti database relazionali sono disponibili su Amazon Aurora. Aurora riutilizza il codice e mantiene la compatibilità con i motori di database sottostanti. Tuttavia, Aurora ha specifici numeri di versione, cicli di rilascio e tempistiche per dichiarazione delle versioni come obsolete. Ogni nuova versione di Aurora è dotata di note di rilascio che elencano le nuove funzionalità, le correzioni, nonché altri miglioramenti e cambiamenti che si applicano a ciascuna versione.

Database Aurora	Guida per l'utente	Versioni disponibili	Note di rilascio
Amazon Aurora edizione compatibile con MySQL	Utilizzo di Amazon Aurora MySQL	Aggiornamenti del motore del database per Amazon Aurora MySQL	Note di rilascio di Aurora MySQL
Amazon Aurora PostgreSQL-Compatible Edition	Utilizzo di Amazon Aurora PostgreSQL	Aggiornamenti del motore di database Amazon Aurora PostgreSQL per 02/06/2020	Note di rilascio per Aurora PostgreSQL

Specifica della versione del database Amazon Aurora per il cluster database

Quando crei un nuovo cluster di database con l'operazione Crea database nella Console di gestione AWS o nell'interfaccia AWS CLI oppure l'operazione API `CreateDBCluster`, puoi specificare qualsiasi versione attualmente disponibile (principale o secondaria).

Per ulteriori informazioni su come creare cluster Aurora, consulta Creazione di un cluster database Amazon Aurora. Per ulteriori informazioni su come modificare la versione di un cluster Aurora esistente, consulta Modifica di un cluster database Amazon Aurora.

> **Note**
>
> Non tutte le versioni dei database Aurora sono disponibili in ogni Regione AWS. Per ulteriori informazioni sulle Regioni e sulle versioni disponibili in ciascuna Regione AWS, consulta Regioni e zone di disponibilità e Regioni e motori di database supportati per i database globali Aurora.

Controllo delle versioni di Amazon Aurora

Le versioni di Amazon Aurora sono diverse dai database di Community upstream con cui sono compatibili. Per aiutarti a garantire la compatibilità delle applicazioni e a sfruttare le funzionalità più recenti del motore di database, le sezioni seguenti spiegano le convenzioni di controllo delle versioni di Aurora e come le versioni di Aurora vengono mappate ai rispettivi database di Community.

Per un elenco dei database relazionali disponibili su Amazon Aurora, consulta Motori di database supportati per i cluster di database Amazon Aurora.

Differenze nei numeri di versione tra database della community e Aurora

Ogni versione di Amazon Aurora è compatibile con una specifica versione di database di Community corrispondente. Puoi trovare la versione Community del tuo database con la funzione `version` e la versione di Aurora utilizzando la funzione `aurora_version`.

Gli esempi seguenti mostrano come trovare la versione Community del database per Aurora MySQL e Aurora PostgreSQL.

Aurora MySQL

La funzione `version` restituisce la versione Community del database per Aurora MySQL.

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

Esempio di output:

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

E la funzione `aurora_version` restituisce la versione di Aurora:

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

Esempio di output:

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

Aurora PostgreSQL

La funzione `version` restituisce la versione Community del database per Aurora PostgreSQL.

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

Esempio di output:

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

E la funzione `aurora_version` restituisce la versione di Aurora:

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

Esempio di output:

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

Per ulteriori informazioni, consultare Controllo delle versioni Aurora MySQL utilizzando SQL e Identificazione delle versioni di Amazon Aurora PostgreSQL.

Versioni predefinite di Amazon Aurora

La versione predefinita è quella che Aurora sceglie automaticamente per la creazione o l'aggiornamento del database quando non specifichi manualmente una versione del motore di destinazione. Ad esempio, il comando seguente mostra la versione predefinita del motore per Aurora PostgreSQL (output di esempio incluso).

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

16.4
```

A ogni versione principale corrisponde una versione secondaria predefinita. Pertanto, la versione secondaria predefinita è 16.n per Aurora PostgreSQL 16, con il numero di versione n che cambia quando Aurora rilascia nuove versioni secondarie predefinite. In genere, ogni anno Aurora rilascia due versioni secondarie predefinite per ogni versione principale. Il seguente script della shell bash mostra le versioni secondarie predefinite per un set di versioni principali di Aurora PostgreSQL (output di esempio incluso).

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

```
done

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

Se l'opzione Aggiornamento automatico versione secondaria è abilitata per il cluster di database Aurora, Aurora utilizza la versione secondaria predefinita oppure una versione secondaria più recente per una determinata versione principale. Ad esempio, se la versione secondaria predefinita per Aurora PostgreSQL 15 è la 15.8 ed è disponibile anche la versione più recente 15.10, Aurora può eseguire automaticamente l'aggiornamento a 15.8 o 15.10.

Versioni principali di Amazon Aurora

Le versioni di Aurora usano lo schema *major*.*minor*.*patch*. Versione principale di Aurora si riferisce alla versione principale della community MySQL o PostgreSQL con cui Aurora è compatibile. Il supporto standard delle versioni principali di Aurora MySQL e Aurora PostgreSQL è disponibile almeno fino alla fine del ciclo di vita della community per la versione della community corrispondente. Puoi continuare a eseguire una versione principale a pagamento dopo la data di fine del supporto standard per Aurora. Per ulteriori informazioni, consulta Supporto esteso di Amazon RDS con Amazon Aurora and Prezzi di Amazon Aurora.

Per ulteriori informazioni sulle versioni principali e sul calendario dei rilasci per Aurora MySQL e Aurora PostgreSQL, consulta le pagine seguenti nelle rispettive note di rilascio:

- Calendario dei rilasci per le versioni principali di Aurora MySQL
- Calendario dei rilasci per le versioni principali di Aurora PostgreSQL

È inoltre possibile visualizzare informazioni sulle date di supporto per le versioni principali del motore eseguendo il comando describe-db-major-engine-versions dell'interfaccia AWS CLI o utilizzando l'operazione DescribeDBMajorEngineVersions dell'API RDS.

> **Note**
>
> Il supporto esteso per Amazon RDS per Aurora MySQL versione 2 inizia il 1° novembre 2024, ma non verrà addebitato alcun costo fino al 1° dicembre 2024. Tra il 1° novembre e

il 30 novembre 2024, tutti i cluster di database Aurora MySQL versione 2 sono coperti dal supporto esteso per Amazon RDS. Per ulteriori informazioni, consulta Supporto esteso per Amazon RDS per versioni di Aurora selezionate.

Per quanto tempo le versioni principali di Amazon Aurora rimangono disponibili

Le versioni principali di Amazon Aurora restano disponibili almeno fino alla fine del ciclo di vita della comunità per la versione della community corrispondente. Puoi utilizzare le date di fine del supporto standard per Aurora per pianificare i cicli di test e aggiornamento. Queste date rappresentano la prima data in cui potrebbe essere richiesto un aggiornamento a una versione più recente. Per ulteriori informazioni sulle date, consulta Versioni principali di Amazon Aurora.

Prima di chiederti di eseguire l'aggiornamento a una versione principale più recente e per aiutarti nella pianificazione, Aurora invia un promemoria con almeno 12 mesi di anticipo. I promemoria comunicano quanto segue in relazione al processo di aggiornamento.

- Le tempistiche di determinate tappe
- L'impatto sui cluster di database
- Azioni consigliate

Si consiglia di testare accuratamente le applicazioni con le nuove versioni del database prima di eseguire il cluster a una nuova versione principale.

Dopo che la versione principale raggiunge la fine del supporto standard di Aurora, ogni cluster di database su cui è ancora in esecuzione la versione precedente viene automaticamente aggiornato a una versione con supporto esteso durante una finestra di manutenzione pianificata. Potrebbero essere applicati costi relativi al supporto esteso. Per ulteriori informazioni sul supporto esteso per Amazon RDS, consulta Utilizzo del supporto esteso per Amazon RDS.

Versioni secondarie di Amazon Aurora

Le versioni di Aurora usano lo schema *major*.*minor*.*patch*. Una versione secondaria di Aurora fornisce al servizio miglioramenti incrementali specifici per la community e Aurora, ad esempio nuove funzionalità e correzioni.

Per ulteriori informazioni sulle versioni secondarie e sul calendario dei rilasci per Aurora MySQL e Aurora PostgreSQL, consulta le pagine seguenti nelle rispettive note di rilascio:

- Calendario dei rilasci per le versioni secondarie di Aurora MySQL
- Calendario dei rilasci per le versioni secondarie di Aurora PostgreSQL

Le sezioni seguenti descrivono i dettagli sulla cadenza e la durata che puoi aspettarti dalle versioni secondarie di Aurora.

Argomenti

- Quanto spesso vengono rilasciate le versioni secondarie di Amazon Aurora
- Per quanto tempo le versioni secondarie di Amazon Aurora rimangono disponibili

Quanto spesso vengono rilasciate le versioni secondarie di Amazon Aurora

In generale, le versioni secondarie di Amazon Aurora vengono rilasciate ogni trimestre. La pianificazione del rilascio potrebbe variare in base a funzionalità aggiuntive o correzioni.

Per quanto tempo le versioni secondarie di Amazon Aurora rimangono disponibili

Solitamente, Amazon Aurora rende disponibile ogni versione secondaria di una specifica versione principale per almeno 12 mesi. Al termine di questo periodo, Aurora potrebbe aggiornare automaticamente il database alla versione secondaria predefinita o a una versione successiva. Aurora avvia l'aggiornamento durante la finestra di manutenzione pianificata per qualsiasi cluster di database su cui è ancora in esecuzione la versione secondaria precedente.

In alcuni casi, Aurora potrebbe sostituire una versione secondaria di una specifica versione principale prima del normale periodo di 12 mesi. Le ragioni possono includere problemi di sicurezza critici o la data di fine supporto per una versione principale.

Prima di avviare gli aggiornamenti automatici delle versioni secondarie che si stanno avvicinando alla fine del ciclo di vita, generalmente Aurora invia un promemoria con tre mesi di anticipo. Aurora comunica quanto segue in relazione al processo di aggiornamento.

- Le tempistiche di determinate tappe
- L'impatto sui cluster di database
- Azioni consigliate

Le notifiche con un preavviso inferiore a tre mesi indicano questioni critiche, come problemi di sicurezza, che richiedono un'azione più rapida.

Se l’opzione Aggiornamento automatico versione secondaria è abilitata, viene visualizzato un promemoria ma nessuna notifica di evento RDS. Aurora aggiorna il database entro una finestra di manutenzione una volta raggiunta la scadenza obbligatoria dell’aggiornamento.

Se l’opzione Aggiornamento automatico versione secondaria non è abilitata, viene visualizzato un promemoria e un evento del cluster di database Amazon RDS con la categoria `maintenance` e l’ID `RDS-EVENT-0156`. Aurora aggiorna il database entro la successiva finestra di manutenzione.

Una volta che una versione secondaria ha raggiunto la fine del supporto standard di Aurora, non verranno rilasciate ulteriori versioni di patch per la versione secondaria in questione. Per ricevere CVE o correzioni di bug critici, è necessario eseguire l’aggiornamento a una versione secondaria con supporto standard.

Per ulteriori informazioni sugli aggiornamenti automatici a versioni secondarie, consulta Aggiornamenti automatici delle versioni secondarie per i cluster di database Aurora.

Versioni delle patch di Amazon Aurora

Le versioni di Aurora usano lo schema *`major`*`.`*`minor`*`.`*`patch`*. La versione di patch di Aurora include importanti correzioni aggiunte a una versione secondaria dopo la versione iniziale (ad esempio, Aurora MySQL 3.04.0, 3.04.1, ..., 3.04.3). Mentre ogni nuova versione secondaria fornisce nuove funzionalità Aurora, le nuove versioni di patch all'interno di una specifica versione secondaria vengono utilizzate principalmente per risolvere problemi importanti.

Per ulteriori informazioni sull'applicazione di patch, consulta Manutenzione di un cluster di database Amazon Aurora.

Aggiornamento dei cluster di database Amazon Aurora

Con Amazon Aurora, puoi controllare e testare gli aggiornamenti dei cluster di database. Amazon Aurora offre opzioni per gli aggiornamenti automatici alle versioni secondarie, il controllo manuale degli aggiornamenti, gli aggiornamenti richiesti e i test pre-aggiornamento. Puoi mantenere i cluster aggiornati con la versione secondaria più recente, rimandando gli aggiornamenti non critici, forzando gli aggiornamenti per problemi gravi e convalidando il comportamento di aggiornamento in ambienti non di produzione. Le seguenti sezioni descrivono in dettaglio come gestire e testare gli aggiornamenti del cluster di database Aurora utilizzando queste funzionalità.

Aggiornamenti automatici a versioni secondarie per Aurora

Gli aggiornamenti automatici a versioni secondarie aggiornano periodicamente il database in uso alle versioni recenti del motore di database. Tuttavia, l'aggiornamento potrebbe non includere sempre la versione più recente del motore di database. Se desideri che i database rimangano a specifiche versioni in determinati momenti, è consigliabile eseguire manualmente l'aggiornamento alle versioni del database necessarie in base alla pianificazione richiesta. In caso di problemi di sicurezza critici o quando una versione raggiunge la data di fine del supporto, Amazon Aurora potrebbe applicare un aggiornamento a una versione secondaria anche se non hai abilitato l'opzione Aggiornamento automatico della versione secondaria. Per ulteriori informazioni, consulta la documentazione relativa agli aggiornamenti per il tuo specifico motore di database.

Consulta Aggiornamento della versione secondaria o del livello di patch di un cluster di database Aurora MySQL e Esecuzione di un aggiornamento a una versione secondaria.

Puoi rimanere aggiornato con le versioni secondarie di Aurora attivando Aggiornamento automatico versione secondaria per ogni istanza database nel cluster Aurora. Aurora esegue l'aggiornamento automatico solo se tutte le istanze database nel cluster hanno questa impostazione abilitata.

Se l'opzione Aggiornamento automatico versione secondaria è impostata su Sì per il cluster di database in uso, Aurora esegue l'aggiornamento automaticamente alla versione secondaria predefinita o a una versione secondaria più recente. Ad esempio, se la versione secondaria predefinita è 15.8 per Aurora PostgreSQL 15 ed esiste la versione 15.10, la destinazione per l'aggiornamento automatico potrebbe essere la 15.8 o la 15.10.

In genere Aurora pianifica gli aggiornamenti automatici due volte l'anno per i cluster di database che hanno l'opzione Aggiornamento automatico versione secondaria abilitata. Questi aggiornamenti vengono eseguiti durante la finestra di manutenzione specificata per il cluster. Per ulteriori informazioni, consulta Aggiornamenti automatici delle versioni secondarie per i cluster di database Aurora.

Gli aggiornamenti automatici delle versioni secondarie vengono comunicati in anticipo tramite un evento cluster database Amazon RDS con una categoria di `maintenance` e un ID di `RDS-EVENT-0156`. Per ulteriori informazioni, consulta Categorie di eventi e messaggi di evento di Amazon RDS per Aurora.

Controllo manuale degli aggiornamenti dei cluster di database a nuove versioni

Se l'opzione Aggiornamento automatico versione secondaria è abilitata, Aurora aggiorna automaticamente il cluster di database alla versione secondaria predefinita o a una versione

secondaria più recente. In genere Aurora pianifica gli aggiornamenti automatici due volte l'anno per i cluster di database che hanno l'opzione Aggiornamento automatico versione secondaria abilitata. Questi aggiornamenti vengono avviati durante le finestre di manutenzione specificate dal cliente.

Per disattivare gli aggiornamenti automatici della versione secondaria, disabilita l'opzione Aggiornamento automatico versione secondaria per qualsiasi istanza database all'interno di un cluster Aurora. Aurora esegue un aggiornamento automatico a una versione secondaria solo se tutte le istanze database nel cluster hanno questa impostazione abilitata.

> **Note**
>
> Per gli aggiornamenti obbligatori come la fine del ciclo di vita della versione secondaria, Aurora aggiorna il cluster di database anche se l'opzione Aggiornamento automatico versione secondaria è disabilitata. Riceverai un promemoria ma nessuna notifica di evento RDS. Gli aggiornamenti di Aurora per il cluster vengono eseguiti entro una finestra di manutenzione una volta raggiunta la scadenza obbligatoria dell'aggiornamento.

Poiché presentano rischi relativi alla compatibilità, gli aggiornamenti delle versioni principali non vengono eseguiti automaticamente. È necessario avviarli, tranne quando una versione principale diventa obsoleta. Si consiglia di testare accuratamente le applicazioni con le nuove versioni del database prima di eseguire il cluster a una nuova versione principale.

Per ulteriori informazioni sull'aggiornamento di un cluster database a una nuova versione principale di Aurora, consulta Aggiornamento dei cluster database Amazon Aurora MySQL e Aggiornamento dei cluster database Amazon Aurora PostgreSQL.

Aggiornamenti obbligatori di Amazon Aurora

Per alcune correzioni critiche, Aurora potrebbe eseguire un aggiornamento gestito a un livello di patch all'interno della stessa versione secondaria. In questo caso, Aurora aggiorna il cluster anche se l'opzione Aggiornamento automatico versione secondaria è disattivata. Prima di farlo, Aurora comunica il processo di aggiornamento dettagliato. I dettagli includono la tempistica di alcune fasi cardine, l'impatto sui cluster di database e le azioni consigliate. Tali aggiornamenti gestiti vengono eseguiti automaticamente nella finestra di manutenzione del cluster.

Test del cluster database con una nuova versione di Aurora prima di eseguire l'aggiornamento

È possibile testare il processo di aggiornamento e il funzionamento della nuova versione con l'applicazione e il carico di lavoro. Selezionare uno dei seguenti metodi:

- Clona il tuo cluster utilizzando la funzione di clonazione rapida del database di Amazon Aurora. Esegui l'aggiornamento ed eventuali test post-aggiornamento sul nuovo cluster.
- Esegui il ripristino da uno snapshot del cluster per creare un nuovo cluster Aurora. Puoi creare uno snapshot del cluster da un cluster Aurora esistente. Aurora crea automaticamente snapshot periodici per ogni cluster. È quindi possibile avviare un aggiornamento della versione per il nuovo cluster. Puoi sperimentare la copia aggiornata del cluster prima di decidere se aggiornare il cluster originale.

Per ulteriori informazioni sui modi per creare nuovi cluster per il test, consulta Clonazione di un volume per un cluster di database Amazon Aurora e Creazione di uno snapshot del cluster database.

Supporto per le versioni di Amazon Aurora

Se un database Amazon Aurora ha dipendenze complesse da specifici comportamenti dei motori di database, si consiglia di eseguire test approfonditi prima di eseguire l'aggiornamento a versioni più recenti dei motori di database. Esistono opzioni di supporto a lungo termine che consentono di mantenere i cluster di database di determinate versioni di motori Aurora anche dopo che sono state sostituite da versioni più recenti. Nelle seguenti sezioni vengono spiegate le opzioni di supporto a lungo termine per i cluster di database Aurora.

Supporto a lungo termine per versioni secondarie selezionate di Amazon Aurora

Per ogni versione principale di Aurora, alcune versioni secondarie sono designate come versioni di supporto a lungo termine (LTS) e restano disponibili per almeno tre anni. In altre parole, almeno una versione secondaria per versione principale è disponibile per più di 12 mesi. Solitamente, Aurora invia un promemoria sei mesi prima della fine di tale periodo. Aurora comunica quanto segue in relazione al processo di aggiornamento.

- Le tempistiche di determinate tappe
- L'impatto sui cluster di database
- Azioni consigliate

Le notifiche con un preavviso inferiore a sei mesi comunicano questioni critiche, come problemi di sicurezza, che richiedono un'azione più rapida.

Le versioni secondarie LTS includono solo correzioni critiche (attraverso versioni di patch). Una versione LTS non include nuove funzionalità rilasciate dopo la sua introduzione. Una volta all'anno, ai cluster database in esecuzione su una versione secondaria LTS viene assegnata una patch con versione più recente della versione LTS. Aurora applica le patch per i cluster in modo da trarre vantaggio dalle correzioni cumulative per la sicurezza e la stabilità. Nel caso in cui fossero presenti correzioni critiche, ad esempio per la sicurezza, che devono essere applicate, potrebbero essere applicate patch a una versione secondaria LTS in modo più frequente.

> **Note**
>
> Per rimanere su una versione secondaria LTS per tutta la durata del relativo ciclo di vita, assicurati di disabilitare l'aggiornamento automatico della versione secondaria per le istanze database. Per evitare di aggiornare automaticamente il cluster di database dalla versione secondaria LTS, deseleziona la casella di controllo Abilita aggiornamento automatico versione secondaria per tutte le istanze database nel cluster Aurora.

Per i numeri di versione di tutte le versioni Aurora LTS, consulta Versioni con supporto a lungo termine (Long-Term Support, LTS) di Aurora MySQL e Utilizzo del rilascio con supporto a lungo termine (LTS) di Aurora PostgreSQL.

Supporto esteso per Amazon RDS per versioni di Aurora selezionate

Con il supporto esteso per Amazon RDS, puoi continuare a eseguire il tuo database su una versione principale del motore dopo la data di fine del supporto standard per Aurora a un costo aggiuntivo. Durante il periodo di supporto esteso per RDS, Amazon RDS fornirà patch per CVE critiche e a elevata vulnerabilità, come definito nelle valutazioni di gravità CVSS del National Vulnerability Database (NVD). Per ulteriori informazioni, consulta Supporto esteso di Amazon RDS con Amazon Aurora.

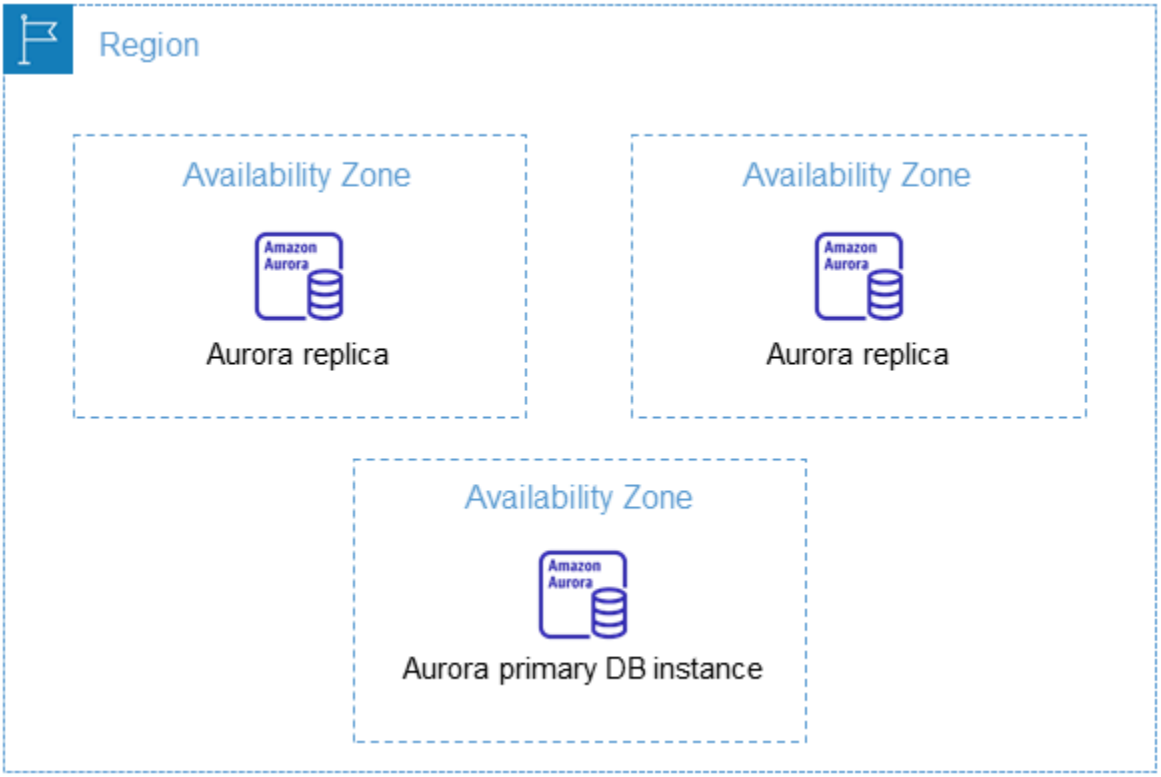
Il supporto esteso per RDS è disponibile solo su determinate versioni di Aurora. Per ulteriori informazioni, consulta Versioni principali di Amazon Aurora.

Regioni e zone di disponibilità

Le risorse di cloud computing Amazon sono ospitate in più ubicazioni in tutto il mondo. Queste località sono composte da AWS regioni e zone di disponibilità. Ciascuna AWSregione è un'area geografica distinta. Ogni AWS regione ha più località isolate note come zone di disponibilità.

> **Note**
>
> Per informazioni su come trovare le zone di disponibilità per una AWS regione, consulta Descrivi le tue zone di disponibilità nella EC2 documentazione di Amazon.

Amazon gestisce state-of-the-art data center ad alta disponibilità. Sebbene rari, possono verificarsi dei guasti che influiscano sulla disponibilità delle istanze database che si trovano nello stesso luogo. Se tutte le istanze database sono ospitate in un singolo luogo interessato da questo errore, nessuna delle istanze database sarà disponibile.



È importante ricordare che ogni AWS regione è completamente indipendente. Qualsiasi attività Amazon RDS avviata (ad esempio, la creazione di istanze di database o l'elenco delle istanze di database disponibili) viene eseguita solo nella regione predefinita corrente. AWS La AWS

regione predefinita può essere modificata nella console o impostando la variabile di ambiente. `AWS_DEFAULT_REGION` Oppure può essere sovrascritta utilizzando il `--region` parametro con AWS Command Line Interface ()AWS CLI. Per ulteriori informazioni, consulta la pagina relativa alla configurazione dell'AWS Command Line Interface, in particolare le sezioni relative alle variabili di ambiente e opzioni della riga di comando.

Amazon RDS supporta AWS regioni speciali chiamateAWS GovCloud (US). sono progettate per consentire ai clienti e agli enti governativi degli Stati Uniti di spostare nel cloud i carichi di lavoro più sensibili. Le regioni AWS GovCloud (US) fanno riferimento ai requisiti normativi e di compliance specifici del governo degli Stati Uniti. Per ulteriori informazioni, consulta Cos'èAWS GovCloud (US)?

Per creare o utilizzare un'istanza database Amazon RDS in una AWS regione specifica, utilizza l'endpoint di servizio regionale corrispondente.

> **Note**
>
> Aurora non supporta Local Zones

AWSRegioni

Ogni AWS regione è progettata per essere isolata dalle altre AWS regioni. Questo progetto permette di raggiungere la maggiore stabilità e tolleranza ai guasti possibile.

Quando si visualizzano le risorse, vengono visualizzate solo le risorse legate alla AWS regione specificata. Questo perché AWS le regioni sono isolate l'una dall'altra e non replichiamo automaticamente le risorse tra le AWS regioni.

Disponibilità nelle regioni

Quando si lavora con un cluster di database Aurora utilizzando l'interfaccia a riga di comando o le operazioni API, assicurarsi di specificare il relativo endpoint regionale.

Argomenti

- Disponibilità nelle regioni Aurora MySQL
- Disponibilità nelle regioni Aurora PostgreSQL

Disponibilità nelle regioni Aurora MySQL

La tabella seguente mostra le AWS regioni in cui Aurora MySQL è attualmente disponibile e l'endpoint per ciascuna regione.

Nome della regione	Regione	Endpoint	Protocollo	
US East (Ohio)	us-east-2	rds.us-east-2.amazonaws.com	HTTPS	
US East (N. Virginia)	us-east-1	rds.us-east-1.amazonaws.com	HTTPS	
Stati Uniti occidentali (California settentrionale)	us-west-1	rds.us-west-1.amazonaws.com	HTTPS	
US West (Oregon)	us-west-2	rds.us-west-2.amazonaws.com	HTTPS	
Africa (Città del Capo)	af-south-1	rds.af-south-1.amazonaws.com	HTTPS	
Asia Pacifico (Hong Kong)	ap-east-1	rds.ap-east-1.amazonaws.com	HTTPS	
Asia Pacific	ap-south-2	rds.ap-south-2.amazonaws.com	HTTPS	

Nome della regione	Regione	Endpoint	Protocollo	
(Hyderabad)				
Asia Pacifico (Giacarta)	ap-southeast-3	rds.ap-southeast-3.amazonaws.com	HTTPS	
Asia Pacifico (Malesia)	ap-southeast-5	rds.ap-southeast-5.amazonaws.com	HTTPS	
Asia Pacifico (Melbourne)	ap-southeast-4	rds.ap-southeast-4.amazonaws.com	HTTPS	
Asia Pacifico (Mumbai)	ap-south-1	rds.ap-south-1.amazonaws.com	HTTPS	
Asia Pacifico (Nuova Zelanda)	ap-southeast-6	rds.ap-southeast-6.amazonaws.com	HTTPS	
Asia Pacifico (Osaka-Locale)	ap-northeast-3	rds.ap-northeast-3.amazonaws.com	HTTPS	
Asia Pacifico (Seoul)	ap-northeast-2	rds.ap-northeast-2.amazonaws.com	HTTPS	

Nome della regione	Regione	Endpoint	Protocollo	
Asia Pacifico (Singapore)	ap-southeast-1	rds.ap-southeast-1.amazonaws.com	HTTPS	
Asia Pacifico (Sydney)	ap-southeast-2	rds.ap-southeast-2.amazonaws.com	HTTPS	
Asia Pacifico (Taipei)	ap-east-2	rds.ap-east-2.amazonaws.com	HTTPS	
Asia Pacifico (Thailandia)	ap-southeast-7	rds.ap-southeast-7.amazonaws.com	HTTPS	
Asia Pacifico (Tokyo)	ap-northeast-1	rds.ap-northeast-1.amazonaws.com	HTTPS	
Canada (Centrale)	ca-central-1	rds.ca-central-1.amazonaws.com	HTTPS	
Canada occidentale (Calgary)	ca-west-1	rds.ca-west-1.amazonaws.com	HTTPS	
Europa (Francoforte)	eu-central-1	rds.eu-central-1.amazonaws.com	HTTPS	

Nome della regione	Regione	Endpoint	Protocollo	
Europa (Irlanda)	eu-west-1	rds.eu-west-1.amazonaws.com	HTTPS	
Europa (Londra)	eu-west-2	rds.eu-west-2.amazonaws.com	HTTPS	
Europa (Milano)	eu-south-1	rds.eu-south-1.amazonaws.com	HTTPS	
Europa (Parigi)	eu-west-3	rds.eu-west-3.amazonaws.com	HTTPS	
Europa (Spagna)	eu-south-2	rds.eu-south-2.amazonaws.com	HTTPS	
Europa (Stoccolma)	eu-north-1	rds.eu-north-1.amazonaws.com	HTTPS	
Europa (Zurigo)	eu-central-2	rds.eu-central-2.amazonaws.com	HTTPS	
Israele (Tel Aviv)	il-central-1	rds.il-central-1.amazonaws.com	HTTPS	
Messico (Centrale)	mx-central-1	rds.mx-central-1.amazonaws.com	HTTPS	
Medio Oriente (Bahrein)	me-south-1	rds.me-south-1.amazonaws.com	HTTPS	

Nome della regione	Regione	Endpoint	Protocollo	
Medio Oriente (Emirati Arabi Uniti)	me-central-1	rds.me-central-1.amazonaws.com	HTTPS	
Sud America (São Paulo)	sa-east-1	rds.sa-east-1.amazonaws.com	HTTPS	
AWSGovCloud (Stati Uniti orientali)	us-gov-east-1	rds.us-gov-east-1.amazonaws.com	HTTPS	
AWSGovCloud (Stati Uniti occidentali)	us-gov-west-1	rds.us-gov-west-1.amazonaws.com	HTTPS	

Disponibilità nelle regioni Aurora PostgreSQL

La tabella seguente mostra le AWS regioni in cui Aurora PostgreSQL è attualmente disponibile e l'endpoint per ciascuna regione.

Nome della regione	Regione	Endpoint	Protocollo	
US East (Ohio)	us-east-2	rds.us-east-2.amazonaws.com	HTTPS	

Nome della regione	Regione	Endpoint	Protocollo	
US East (N. Virginia)	us-east-1	rds.us-east-1.amazonaws.com	HTTPS	
Stati Uniti occidentali (California settentrionale)	us-west-1	rds.us-west-1.amazonaws.com	HTTPS	
US West (Oregon)	us-west-2	rds.us-west-2.amazonaws.com	HTTPS	
Africa (Cape Town)	af-south-1	rds.af-south-1.amazonaws.com	HTTPS	
Asia Pacifico (Hong Kong)	ap-east-1	rds.ap-east-1.amazonaws.com	HTTPS	
Asia Pacific (Hyderabad)	ap-south-2	rds.ap-south-2.amazonaws.com	HTTPS	
Asia Pacifico (Giacarta)	ap-southeast-3	rds.ap-southeast-3.amazonaws.com	HTTPS	

Nome della regione	Regione	Endpoint	Protocollo	
Asia Pacifico (Malesia)	ap-southeast-5	rds.ap-southeast-5.amazonaws.com	HTTPS	
Asia Pacifico (Melbourne)	ap-southeast-4	rds.ap-southeast-4.amazonaws.com	HTTPS	
Asia Pacifico (Mumbai)	ap-south-1	rds.ap-south-1.amazonaws.com	HTTPS	
Asia Pacifico (Nuova Zelanda)	ap-southeast-6	rds.ap-southeast-6.amazonaws.com	HTTPS	
Asia Pacifico (Osaka-Locale)	ap-northeast-3	rds.ap-northeast-3.amazonaws.com	HTTPS	
Asia Pacifico (Seoul)	ap-northeast-2	rds.ap-northeast-2.amazonaws.com	HTTPS	
Asia Pacifico (Singapore)	ap-southeast-1	rds.ap-southeast-1.amazonaws.com	HTTPS	

Nome della regione	Regione	Endpoint	Protocollo	
Asia Pacifico (Sydney)	ap-southeast-2	rds.ap-southeast-2.amazonaws.com	HTTPS	
Asia Pacifico (Taipei)	ap-east-2	rds.ap-east-2.amazonaws.com	HTTPS	
Asia Pacifico (Thailandia)	ap-southeast-7	rds.ap-southeast-7.amazonaws.com	HTTPS	
Asia Pacifico (Tokyo)	ap-northeast-1	rds.ap-northeast-1.amazonaws.com	HTTPS	
Canada (Centrale)	ca-central-1	rds.ca-central-1.amazonaws.com	HTTPS	
Canada occidentale (Calgary)	ca-west-1	rds.ca-west-1.amazonaws.com	HTTPS	
Europa (Francoforte)	eu-central-1	rds.eu-central-1.amazonaws.com	HTTPS	
Europa (Irlanda)	eu-west-1	rds.eu-west-1.amazonaws.com	HTTPS	
Europa (Londra)	eu-west-2	rds.eu-west-2.amazonaws.com	HTTPS	

Nome della regione	Regione	Endpoint	Protocollo	
Europa (Milano)	eu-south-1	rds.eu-south-1.amazonaws.com	HTTPS	
Europa (Parigi)	eu-west-3	rds.eu-west-3.amazonaws.com	HTTPS	
Europa (Spagna)	eu-south-2	rds.eu-south-2.amazonaws.com	HTTPS	
Europa (Stoccolma)	eu-north-1	rds.eu-north-1.amazonaws.com	HTTPS	
Europa (Zurigo)	eu-central-2	rds.eu-central-2.amazonaws.com	HTTPS	
Israele (Tel Aviv)	il-central-1	rds.il-central-1.amazonaws.com	HTTPS	
Messico (Centrale)	mx-central-1	rds.mx-central-1.amazonaws.com	HTTPS	
Medio Oriente (Bahrein)	me-south-1	rds.me-south-1.amazonaws.com	HTTPS	
Medio Oriente (Emirati Arabi Uniti)	me-central-1	rds.me-central-1.amazonaws.com	HTTPS	

Nome della regione	Regione	Endpoint	Protocollo	
Sud America (São Paulo)	sa-east-1	rds.sa-east-1.amazonaws.com	HTTPS	
AWSGovCloud (Stati Uniti orientali)	us-gov-east-1	rds.us-gov-east-1.amazonaws.com	HTTPS	
AWSGovCloud (Stati Uniti occidentali)	us-gov-west-1	rds.us-gov-west-1.amazonaws.com	HTTPS	

Zone di disponibilità

Una zona di disponibilità è una località isolata in una data Regione AWS specificata. Ogni regione ha più zone di disponibilità (AZ) progettate per fornire un'elevata disponibilità per la regione. Una AZ è identificata dal codice AWS regionale seguito da una lettera identificativa (ad esempio,`us-east-1a`). Se crei il tuo VPC e le sottoreti anziché utilizzare il VPC di default, definisci ogni sottorete in una AZ specifica. Quando crei un cluster database Aurora, Aurora crea l'istanza primaria in una delle sottoreti del gruppo di sottoreti database del VPC, associando così l'istanza a una specifica zona di disponibilità scelta da Aurora.

Ogni cluster Aurora DB ospita copie del proprio storage in tre file separati AZs selezionati automaticamente da Aurora dal gruppo di sottoreti del AZs database. Ogni istanza DB del cluster deve trovarsi in una di queste tre istanze. AZs

Quando crei un'istanza database nel cluster, se non specifichi una zona di disponibilità, Aurora sceglie automaticamente una zona di disponibilità appropriata per quell'istanza.

Utilizza il EC2 comando describe-availability-zonesAmazon come segue per descrivere le zone di disponibilità all'interno della regione specificata abilitate per il tuo account.

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

Ad esempio, per descrivere le zone di disponibilità all'interno della regione Stati Uniti orientali (Virginia settentrionale) (us-east-1) abilitate per il tuo account, esegui il comando seguente:

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

Per informazioni su come specificare l'AZ quando si crea un cluster o vi si aggiungono istanze, consulta Configurazione della rete per il cluster database.

Fuso orario locale per i cluster DB Amazon Aurora

Per impostazione predefinita, il fuso orario di un cluster DB di Amazon Aurora è in formato UTC (Universal Time Coordinated). Tuttavia, puoi impostare il fuso orario delle istanze del cluster DB sul fuso orario locale dell'applicazione.

Per modificare il fuso orario locale per un cluster database, imposta il parametro del fuso orario su uno dei valori supportati. Imposti questo parametro nel gruppo di parametri del tuo cluster database.

- Per Aurora MySQL, il nome di questo parametro è `time_zone`. Per informazioni sulle best practice per l'impostazione del parametro `time_zone`, consulta Ottimizzazione delle operazioni di timestamp.
- Per Aurora PostgreSQL, il nome di questo parametro è `timezone`.

Quando imposti il parametro di fuso orario per un cluster database, tutte le istanze nel cluster database cambiano per utilizzare il nuovo fuso orario locale. In alcuni casi, è possibile che altri cluster database Aurora utilizzino lo stesso gruppo di parametri del cluster. In tal caso, tutte le istanze in tali cluster database cambiano per utilizzare il nuovo fuso orario locale. Per ulteriori informazioni sui parametri a livello di cluster, consulta Parametri dell'istanza database e del cluster database di Amazon Aurora.

Dopo aver impostato il fuso orario locale, tutte le nuove connessioni al database riflettono la modifica. In alcuni casi, potresti avere connessioni aperte al database quando modifichi il fuso orario locale. In tal caso, non vedrai l'aggiornamento del fuso orario locale finché non avrai chiuso la connessione e aperto una nuova connessione.

Se si esegue la replica tra AWS regioni, il cluster DB di origine della replica e la replica utilizzano gruppi di parametri diversi. I gruppi di parametri sono unici per una regione. AWS Per utilizzare lo stesso fuso orario locale per ogni istanza, assicurati di impostare il parametro del fuso orario nei gruppi di parametri sia per l'origine della replica che per la replica.

Quando ripristini cluster database da una snapshot cluster database, il fuso orario locale è impostato su UTC. Puoi aggiornare il fuso orario impostandolo sul fuso orario locale dopo il completamento del ripristino. In alcuni casi, è possibile eseguire un ripristino point-in-time del cluster database. In tal caso, il fuso orario locale per il cluster database ripristinato corrisponde all'impostazione del fuso orario per il gruppo di parametri cluster database ripristinato.

Nella tabella seguente sono elencati alcuni dei valori che puoi usare per impostare il fuso orario locale. Per elencare tutti i fusi orari disponibili, puoi utilizzare le seguenti query SQL:

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

> **Note**
>
> Per alcuni fusi orari, i valori di tempo di alcuni intervalli di date potrebbero essere riportati in modo non corretto, come indicato nella tabella. Per i fusi orari australiani, l'abbreviazione restituita riporta un valore obsoleto, come indicato nella tabella.

Time zone (Fuso orario)	Note
`Africa/Harare`	Questa impostazione del fuso orario potrebbe restituire valori non corretti dal 28/02/1903 21:49:40 GMT al 28/02/1903 21:55:48 GMT.
`Africa/Monrovia`	
`Africa/Nairobi`	Questa impostazione del fuso orario potrebbe restituire valori non corretti dal 31/12/1939 21:30:00 GMT al 31/12/1959 21:15:15 GMT.
`Africa/Windhoek`	

Time zone (Fuso orario)	Note
`America/Bogota`	Questa impostazione del fuso orario potrebbe restituire valori non corretti dal 23/11/1914 04:56:16 GMT al 23/11/1914 04:56:20 GMT.
`America/Caracas`	
`America/Chihuahua`	
`America/Cuiaba`	
`America/Denver`	
`America/Fortaleza`	In alcuni casi, per un cluster database nella Regione Sud America (San Paolo), l'ora non viene visualizzata correttamente per un fuso orario del Brasile modificato di recente. In tal caso, reimposta il parametro del fuso orario del cluster database su `America/Fortaleza` .
`America/Guatemala`	
`America/Halifax`	Questa impostazione del fuso orario potrebbe restituire valori non corretti dal 27/10/1918 05:00:00 GMT al 31/10/1918 05:00:00 GMT.
`America/Manaus`	Se il cluster DB si trova nel fuso orario del Sud America (Cuiaba) e l'ora prevista non viene visualizzata correttamente per il fuso orario del Brasile modificato di recente, reimpostare il parametro fuso orario del cluster DB su `America/Manaus` .
`America/Matamoros`	
`America/Monterrey`	
`America/Montevideo`	

Time zone (Fuso orario)	Note
America/Noronha	
America/Phoenix	
America/Tijuana	
Asia/Ashgabat	
Asia/Baghdad	
Asia/Baku	
Asia/Bangkok	
Asia/Beirut	
Asia/Calcutta	
Asia/Kabul	
Asia/Karachi	
Asia/Kathmandu	
Asia/Muscat	Questa impostazione del fuso orario potrebbe restituire valori non corretti dal 31/12/1919 20:05:36 GMT al 31/12/1919 20:05:40 GMT.
Asia/Riyadh	Questa impostazione del fuso orario potrebbe restituire valori non corretti dal 13/03/1947 20:53:08 GMT al 31/12/1949 20:53:08 GMT.
Asia/Seoul	Questa impostazione del fuso orario potrebbe restituire valori non corretti dal 30/11/1904 15:30:00 GMT al 07/09/1945 15:00:00 GMT.
Asia/Shanghai	Questa impostazione del fuso orario potrebbe restituire valori non corretti dal 31/12/1927 15:54:08 GMT al 02/06/1940 16:00:00 GMT.
Asia/Singapore	

Time zone (Fuso orario)	Note
Asia/Taipei	Questa impostazione del fuso orario potrebbe restituire valori non corretti dal 30/09/1937 16:00:00 GMT al 29/09/1979 15:00:00 GMT.
Asia/Tehran	
Asia/Tokyo	Questa impostazione del fuso orario potrebbe restituire valori non corretti dal 30/09/1937 15:00:00 GMT al 31/12/1937 15:00:00 GMT.
Asia/Ulaanbaatar	
Atlantic/Azores	Questa impostazione del fuso orario potrebbe restituire valori non corretti dal 24/05/1911 01:54:32 GMT al 01/01/1912 01:54:32 GMT.
Australia/Adelaide	L'abbreviazione di questo fuso orario viene restituita come CST anziché ACDT/ACST.
Australia/Brisbane	L'abbreviazione di questo fuso orario viene restituita come EST anziché AEDT/AEST.
Australia/Darwin	L'abbreviazione di questo fuso orario viene restituita come CST anziché ACDT/ACST.
Australia/Hobart	L'abbreviazione di questo fuso orario viene restituita come EST anziché AEDT/AEST.
Australia/Perth	L'abbreviazione di questo fuso orario viene restituita come WST anziché AWDT/. AWST
Australia/Sydney	L'abbreviazione di questo fuso orario viene restituita come EST anziché AEDT/AEST.
Brazil/East	
Canada/Saskatchewan	Questa impostazione del fuso orario potrebbe restituire valori non corretti dal 27/10/1918 08:00:00 GMT al 31/10/1918 08:00:00 GMT.

Time zone (Fuso orario)	Note
Europe/Amsterdam	
Europe/Athens	
Europe/Dublin	
Europe/Helsinki	Questa impostazione del fuso orario potrebbe restituire valori non corretti dal 30/04/1921 22:20:08 GMT al 30/04/1921 22:20:11 GMT.
Europe/Paris	
Europe/Prague	
Europe/Sarajevo	
Pacific/Auckland	
Pacific/Guam	
Pacific/Honolulu	Questa impostazione del fuso orario potrebbe restituire valori non corretti dal 21/05/1933 11:30:00 GMT al 30/09/1945 11:30:00 GMT.
Pacific/Samoa	Questa impostazione del fuso orario potrebbe restituire valori non corretti dal 01/01/1911 11:22:48 GMT al 01/01/1950 11:30:00 GMT.
US/Alaska	
US/Central	
US/Eastern	
US/East-Indiana	
US/Pacific	
UTC	

Funzionalità supportate in Amazon Aurora by e Regione AWS Aurora DB engine

I motori di database compatibili con Aurora MySQL e PostgreSQL supportano diverse funzionalità e opzioni di Amazon Aurora e Amazon RDS. Il supporto varia a seconda delle versioni specifiche di ciascun motore del database e delle Regioni AWS. Per identificare il supporto e la disponibilità della versione del motore di database Aurora per una determinata funzionalità Regione AWS, è possibile utilizzare le seguenti sezioni.

Alcune di queste funzionalità sono esclusive di Aurora. Ad esempio, Aurora Serverless, i database globali Aurora e il supporto per l'integrazione con i servizi di machine learning di AWS non sono supportati da Amazon RDS. Altre funzionalità, come Amazon RDS Proxy, sono supportate sia da Amazon Aurora che da Amazon RDS.

Regioni e motori di database supportati

- Convenzioni tabella
- Regioni e motori di database Aurora supportati per implementazioni blu/verde
- Regioni e motori di database Aurora supportati per configurazioni dell’archiviazione dei cluster
- Regioni e motori di database Aurora supportati per flussi di attività del database
- Regioni e motori di database Aurora supportati per l’esportazione dei dati del cluster in Amazon S3
- Regioni e motori di database Aurora supportati per l’esportazione dei dati dello snapshot in Amazon S3
- Regioni e motori di database supportati per i database globali Aurora
- Regioni e motori di database Aurora supportati per l’autenticazione del database IAM
- Regioni e motori di database Aurora supportati per l’autenticazione Kerberos
- Regioni e motori di database supportati per il machine learning Aurora
- Regioni e motori di database Aurora supportati per Approfondimenti sulle prestazioni
- Regioni e motori di database Aurora supportati per le integrazioni Zero-ETL
- Regioni e motori di database Aurora supportati per Server proxy per Amazon RDS
- Regioni e motori di database Aurora supportati per l’integrazione di Secrets Manager
- Regioni e motori di database Aurora supportati per Aurora Serverless v2
- Aurora Serverless v1
- Regioni e motori di database Aurora supportati per l’API dati RDS

- Regioni e motori di database Aurora supportati per l’applicazione di patch senza tempo di inattività (ZDP, Zero-Downtime Patching)
- Regioni supportate in Aurora PostgreSQL Limitless Database
- Regioni e motori di database supportati per le funzionalità native del motore Aurora

Convenzioni tabella

Le tabelle nelle sezioni delle caratteristiche utilizzano i seguenti modelli per specificare i numeri di versione e il livello di supporto:

- Versione x.y – È supportata la sola versione specifica.
- Versione x.y e successive: sono supportate la versione specificata e tutte le relative versioni secondarie successive. Ad esempio, "versione 10.11 e successive" significa che sono supportate le versioni 10.11, 10.11.1 e 10.12.

Regioni e motori di database Aurora supportati per implementazioni blu/verde

Una blue/green distribuzione copia un ambiente di database di produzione in un ambiente di staging separato e sincronizzato. Utilizzando Amazon RDS Blue/Green Deployments, puoi apportare modifiche al database nell'ambiente di staging senza influire sull'ambiente di produzione. Ad esempio, è possibile aggiornare la versione principale o secondaria del motore di database, modificare i parametri del database o apportare modifiche allo schema nell'ambiente di gestione temporanea. Quando sei pronto, puoi promuovere l'ambiente di gestione temporanea nel nuovo ambiente di database di produzione. Per ulteriori informazioni, consulta Utilizzo di Amazon Blue/Green Aurora Deployments per gli aggiornamenti del database.

Implementazioni blu/verde con Aurora MySQL

La funzionalità Blue/Green Deployments è disponibile per tutte le versioni di Aurora MySQL in generale, Regioni AWS inclusi i cluster Aurora MySQL configurati come Aurora Global Database.

Implementazioni blu/verde con Aurora PostgreSQL

Le seguenti regioni e versioni del motore sono disponibili per le Blue/Green distribuzioni con Aurora PostgreSQL, inclusi i cluster Aurora PostgreSQL configurati come Aurora Global Database.

Region	Aurora PostgreSQL 17	Aurora PostgreSQL 16	Aurora PostgreSQL 15	Aurora PostgreSQL 14	Aurora PostgreSQL 13	Aurora PostgreSQL 12	Aurora PostgreSQL 11
Tutti Regioni AWS	Versione 17.4 e successive	Versione 16.1 e successive	Versione 15.4 e successive	Versione 14.9 e successive	Versione 13.12 e successive	Versione 12.16 e successive	Versione 11.21 e successive

Regioni e motori di database Aurora supportati per configurazioni dell'archiviazione dei cluster

Amazon Aurora dispone di due configurazioni dell'archiviazione dei cluster database, ovvero Aurora I/O-Optimized e Aurora Standard. Per ulteriori informazioni, consulta Configurazioni dell'archiviazione per i cluster database Amazon Aurora.

Aurora I/O-Optimized

Aurora I/O-Optimized è disponibile in tutte le Regioni AWS per le seguenti versioni di Amazon Aurora:

- Aurora MySQL versione 3.03.1 e versioni successive
- Aurora PostgreSQL versioni 17.4 e successive, 16.1 e successive, 15.2 e successive, 14.7 e successive e 13.10 e successive

Aurora Standard

Aurora Standard è disponibile in tutte le Regioni AWS per tutte le versioni di Aurora MySQL e Aurora PostgreSQL.

Regioni e motori di database Aurora supportati per flussi di attività del database

Utilizzando i flussi di attività del database in Aurora, è possibile monitorare e impostare gli allarmi per l'attività di audit nel database Oracle in uso. Per ulteriori informazioni, consulta Monitoraggio di Amazon Aurora tramite i flussi di attività del database.

I flussi di attività del database non sono supportati per le seguenti funzionalità:

- Aurora Serverless v1
- Aurora Serverless v2
- Babelfish per Aurora PostgreSQL

Argomenti

- Flussi di attività del database con Amazon MySQL
- Flussi di attività del database con Aurora PostgreSQL

Flussi di attività del database con Amazon MySQL

Di seguito sono riportate le versioni di motore e le regioni disponibili per i flussi di attività del database con Aurora MySQL.

Region	Aurora MySQL versione 3	Aurora MySQL versione 2
Stati Uniti orientali (Virginia settentrionale)	Tutte le versioni disponibili	Aurora versione 2.11 e successive
Stati Uniti orientali (Ohio)	Tutte le versioni disponibili	Aurora versione 2.11 e successive
Stati Uniti occidentali (California settentrionale)	Tutte le versioni disponibili	Aurora versione 2.11 e successive
Stati Uniti occidentali (Oregon)	Tutte le versioni disponibili	Aurora versione 2.11 e successive
Africa (Città del Capo)	Tutte le versioni disponibili	Aurora versione 2.11 e successive
Asia Pacifico (Hong Kong)	Tutte le versioni disponibili	Aurora versione 2.11 e successive
Asia Pacific (Hyderabad)	Tutte le versioni disponibili	Aurora versione 2.11 e successive

Region	Aurora MySQL versione 3	Aurora MySQL versione 2
Asia Pacifico (Giacarta)	Tutte le versioni disponibili	Aurora versione 2.11 e successive
Asia Pacifico (Malesia)	Non disponibile	Non disponibile
Asia Pacifico (Melbourne)	Non disponibile	Non disponibile
Asia Pacifico (Mumbai)	Tutte le versioni disponibili	Aurora versione 2.11 e successive
Asia Pacifico (Nuova Zelanda)	Non disponibile	Non disponibile
Asia Pacifico (Osaka-Locale)	Tutte le versioni disponibili	Aurora versione 2.11 e successive
Asia Pacifico (Seul)	Tutte le versioni disponibili	Aurora versione 2.11 e successive
Asia Pacifico (Singapore)	Tutte le versioni disponibili	Aurora versione 2.11 e successive
Asia Pacifico (Sydney)	Tutte le versioni disponibili	Aurora versione 2.11 e successive
Asia Pacifico (Taipei)	Non disponibile	Non disponibile
Asia Pacifico (Thailandia)	Non disponibile	Non disponibile
Asia Pacifico (Tokyo)	Tutte le versioni disponibili	Aurora versione 2.11 e successive
Canada (Centrale)	Tutte le versioni disponibili	Aurora versione 2.11 e successive
Canada occidentale (Calgary)	Non disponibile	Non disponibile
Cina (Pechino)	Non disponibile	Non disponibile
Cina (Ningxia)	Non disponibile	Non disponibile

Region	Aurora MySQL versione 3	Aurora MySQL versione 2
Europa (Francoforte)	Tutte le versioni disponibili	Aurora versione 2.11 e successive
Europa (Irlanda)	Tutte le versioni disponibili	Aurora versione 2.11 e successive
Europa (Londra)	Tutte le versioni disponibili	Aurora versione 2.11 e successive
Europa (Milano)	Tutte le versioni disponibili	Aurora versione 2.11 e successive
Europa (Parigi)	Tutte le versioni disponibili	Aurora versione 2.11 e successive
Europa (Spagna)	Tutte le versioni disponibili	Aurora versione 2.11 e successive
Europa (Stoccolma)	Tutte le versioni disponibili	Aurora versione 2.11 e successive
Europa (Zurigo)	Non disponibile	Non disponibile
Israele (Tel Aviv)	Non disponibile	Non disponibile
Medio Oriente (Bahrein)	Tutte le versioni disponibili	Aurora versione 2.11 e successive
Messico (centrale)	Non disponibile	Non disponibile
Medio Oriente (Emirati Arabi Uniti)	Tutte le versioni disponibili	Aurora versione 2.11 e successive
Sud America (San Paolo)	Tutte le versioni disponibili	Aurora versione 2.11 e successive

Flussi di attività del database con Aurora PostgreSQL

Di seguito sono riportate le versioni di motore e le regioni disponibili per i flussi di attività del database con Aurora PostgreSQL.

Region	Aurora PostgreSQL 17	Aurora PostgreSQL 16	Aurora PostgreSQL 15	Aurora PostgreSQL 14	Aurora PostgreSQL 13	Aurora PostgreSQL 12	Aurora PostgreSQL 11
Stati Uniti orientali (Virginia settentrionale)	Versione 17.4 e successive	Versione 16.1 e successive	Versione 15.2 e successive	Tutte le versioni disponibili	Tutte le versioni disponibili	Tutte le versioni disponibili	Versione 11.9 e versione 11.13 e successive
Stati Uniti orientali (Ohio)	Versione 17.4 e successive	Versione 16.1 e successive	Versione 15.2 e successive	Tutte le versioni disponibili	Tutte le versioni disponibili	Tutte le versioni disponibili	Versione 11.9 e versione 11.13 e successive
Stati Uniti occidentali (California settentrionale)	Versione 17.4 e successive	Versione 16.1 e successive	Versione 15.2 e successive	Tutte le versioni disponibili	Tutte le versioni disponibili	Tutte le versioni disponibili	Versione 11.9 e versione 11.13 e successive
Stati Uniti occidentali (Oregon)	Versione 17.4 e successive	Versione 16.1 e successive	Versione 15.2 e successive	Tutte le versioni disponibili	Tutte le versioni disponibili	Tutte le versioni disponibili	Versione 11.9 e versione 11.13 e successive

Region	Aurora PostgreSQL 17	Aurora PostgreSQL 16	Aurora PostgreSQL 15	Aurora PostgreSQL 14	Aurora PostgreSQL 13	Aurora PostgreSQL 12	Aurora PostgreSQL 11
Africa (Città del Capo)	Versione 17.4 e successive	Versione 16.1 e successive	Versione 15.2 e successive	Tutte le versioni disponibili	Tutte le versioni disponibili	Tutte le versioni disponibili	Versione 11.9 e versione 11.13 e successive
Asia Pacifico (Hong Kong)	Versione 17.4 e successive	Versione 16.1 e successive	Versione 15.2 e successive	Tutte le versioni disponibili	Tutte le versioni disponibili	Tutte le versioni disponibili	Versione 11.9 e versione 11.13 e successive
Asia Pacific (Hyderabad)	Versione 17.4 e successive	Versione 16.1 e successive	Versione 15.2 e successive	Tutte le versioni disponibili	Tutte le versioni disponibili	Tutte le versioni disponibili	Versione 11.9 e versione 11.13 e successive
Asia Pacifico (Giacarta)	Versione 17.4 e successive	Versione 16.1 e successive	Versione 15.2 e successive	Tutte le versioni disponibili	Tutte le versioni disponibili	Tutte le versioni disponibili	Versione 11.9 e versione 11.13 e successive
Asia Pacifico (Malesia)	Non disponibile	Non disponibile	Non disponibile	Non disponibile	Non disponibile	Non disponibile	

Region	Aurora PostgreSQL 17	Aurora PostgreSQL 16	Aurora PostgreSQL 15	Aurora PostgreSQL 14	Aurora PostgreSQL 13	Aurora PostgreSQL 12	Aurora PostgreSQL 11
Asia Pacifico (Melbourne)	Non disponibile	Non disponibile	Non disponibile	Non disponibile	Non disponibile	Non disponibile	Non disponibile
Asia Pacifico (Mumbai)	Versione 17.4 e successive	Versione 16.1 e successive	Versione 15.2 e successive	Tutte le versioni disponibili	Tutte le versioni disponibili	Tutte le versioni disponibili	Versione 11.9 e versione 11.13 e successive
Asia Pacifico (Nuova Zelanda)	Non disponibile	Non disponibile	Non disponibile	Non disponibile	Non disponibile	Non disponibile	Non disponibile
Asia Pacifico (Osaka-Locale)	Versione 17.4 e successive	Versione 16.1 e successive	Versione 15.2 e successive	Tutte le versioni disponibili	Tutte le versioni disponibili	Tutte le versioni disponibili	Versione 11.9 e versione 11.13 e successive
Asia Pacifico (Seul)	Versione 17.4 e successive	Versione 16.1 e successive	Versione 15.2 e successive	Tutte le versioni disponibili	Tutte le versioni disponibili	Tutte le versioni disponibili	Versione 11.9 e versione 11.13 e successive

Region	Aurora PostgreSQL 17	Aurora PostgreSQL 16	Aurora PostgreSQL 15	Aurora PostgreSQL 14	Aurora PostgreSQL 13	Aurora PostgreSQL 12	Aurora PostgreSQL 11
Asia Pacifico (Singapore)	Versione 17.4 e successive	Versione 16.1 e successive	Versione 15.2 e successive	Tutte le versioni disponibili	Tutte le versioni disponibili	Tutte le versioni disponibili	Versione 11.9 e versione 11.13 e successive
Asia Pacifico (Sydney)	Versione 17.4 e successive	Versione 16.1 e successive	Versione 15.2 e successive	Tutte le versioni disponibili	Tutte le versioni disponibili	Tutte le versioni disponibili	Versione 11.9 e versione 11.13 e successive
Asia Pacifico (Taipei)	Non disponibile	Non disponibile	Non disponibile	Non disponibile	Non disponibile	Non disponibile	Non disponibile
Asia Pacifico (Thailandia)	Non disponibile	Non disponibile	Non disponibile	Non disponibile	Non disponibile	Non disponibile	Non disponibile
Asia Pacifico (Tokyo)	Versione 17.4 e successive	Versione 16.1 e successive	Versione 15.2 e successive	Tutte le versioni disponibili	Tutte le versioni disponibili	Tutte le versioni disponibili	Versione 11.9 e versione 11.13 e successive

Region	Aurora PostgreSQL 17	Aurora PostgreSQL 16	Aurora PostgreSQL 15	Aurora PostgreSQL 14	Aurora PostgreSQL 13	Aurora PostgreSQL 12	Aurora PostgreSQL 11
Canada (Centrale)	Versione 17.4 e successive	Versione 16.1 e successive	Versione 15.2 e successive	Tutte le versioni disponibili	Tutte le versioni disponibili	Tutte le versioni disponibili	Versione 11.9 e versione 11.13 e successive
Canada occidentale (Calgary)	Non disponibile	Non disponibile	Non disponibile	Non disponibile	Non disponibile	Non disponibile	Non disponibile
Cina (Pechino)	Non disponibile	Non disponibile	Non disponibile	Non disponibile	Non disponibile	Non disponibile	Non disponibile
Cina (Ningxia)	Non disponibile	Non disponibile	Non disponibile	Non disponibile	Non disponibile	Non disponibile	Non disponibile
Europa (Francoforte)	Versione 17.4 e successive	Versione 16.1 e successive	Versione 15.2 e successive	Tutte le versioni disponibili	Tutte le versioni disponibili	Tutte le versioni disponibili	Versione 11.9 e versione 11.13 e successive
Europa (Irlanda)	Versione 17.4 e successive	Versione 16.1 e successive	Versione 15.2 e successive	Tutte le versioni disponibili	Tutte le versioni disponibili	Tutte le versioni disponibili	Versione 11.9 e versione 11.13 e successive

Region	Aurora PostgreSQL 17	Aurora PostgreSQL 16	Aurora PostgreSQL 15	Aurora PostgreSQL 14	Aurora PostgreSQL 13	Aurora PostgreSQL 12	Aurora PostgreSQL 11
Europa (Londra)	Versione 17.4 e successive	Versione 16.1 e successive	Versione 15.2 e successive	Tutte le versioni disponibili	Tutte le versioni disponibili	Tutte le versioni disponibili	Versione 11.9 e versione 11.13 e successive
Europa (Milano)	Versione 17.4 e successive	Versione 16.1 e successive	Versione 15.2 e successive	Tutte le versioni disponibili	Tutte le versioni disponibili	Tutte le versioni disponibili	Versione 11.9 e versione 11.13 e successive
Europa (Parigi)	Versione 17.4 e successive	Versione 16.1 e successive	Versione 15.2 e successive	Tutte le versioni disponibili	Tutte le versioni disponibili	Tutte le versioni disponibili	Versione 11.9 e versione 11.13 e successive
Europa (Spagna)	Versione 17.4 e successive	Versione 16.1 e successive	Versione 15.2 e successive	Tutte le versioni disponibili	Tutte le versioni disponibili	Tutte le versioni disponibili	Versione 11.9 e versione 11.13 e successive

Region	Aurora PostgreSQL 17	Aurora PostgreSQL 16	Aurora PostgreSQL 15	Aurora PostgreSQL 14	Aurora PostgreSQL 13	Aurora PostgreSQL 12	Aurora PostgreSQL 11
Europa (Stoccolma)	Versione 17.4 e successive	Versione 16.1 e successive	Versione 15.2 e successive	Tutte le versioni disponibili	Tutte le versioni disponibili	Tutte le versioni disponibili	Versione 11.9 e versione 11.13 e successive
Europa (Zurigo)	Non disponibile	Non disponibile	Non disponibile	Non disponibile	Non disponibile	Non disponibile	Non disponibile
Israele (Tel Aviv)	Non disponibile	Non disponibile	Non disponibile	Non disponibile	Non disponibile	Non disponibile	Non disponibile
Messico (centrale)	Non disponibile	Non disponibile	Non disponibile	Non disponibile	Non disponibile	Non disponibile	Non disponibile
Medio Oriente (Bahrein)	Versione 17.4 e successive	Versione 16.1 e successive	Versione 15.2 e successive	Tutte le versioni disponibili	Tutte le versioni disponibili	Tutte le versioni disponibili	Versione 11.9 e versione 11.13 e successive
Medio Oriente (Emirati Arabi Uniti)	Versione 17.4 e successive	Versione 16.1 e successive	Versione 15.2 e successive	Tutte le versioni disponibili	Tutte le versioni disponibili	Tutte le versioni disponibili	Versione 11.9 e versione 11.13 e successive

Region	Aurora PostgreSQL 17	Aurora PostgreSQL 16	Aurora PostgreSQL 15	Aurora PostgreSQL 14	Aurora PostgreSQL 13	Aurora PostgreSQL 12	Aurora PostgreSQL 11
Sud America (San Paolo)	Versione 17.4 e successive	Versione 16.1 e successive	Versione 15.2 e successive	Tutte le versioni disponibili	Tutte le versioni disponibili	Tutte le versioni disponibili	Versione 11.9 e versione 11.13 e successive

Regioni e motori di database Aurora supportati per l'esportazione dei dati del cluster in Amazon S3

È possibile esportare i dati del cluster database Aurora in un bucket Amazon S3. Dopo l'esportazione dei dati, è possibile analizzare i dati esportati direttamente mediante strumenti quali Amazon Athena o Amazon Redshift Spectrum. Per ulteriori informazioni, consulta Esportazione dei dati del cluster database in Amazon S3.

L'esportazione dei dati del cluster in S3 è disponibile nelle seguenti Regioni AWS:

- Stati Uniti orientali (Virginia settentrionale)
- Stati Uniti orientali (Ohio)
- Stati Uniti occidentali (California settentrionale)
- Stati Uniti occidentali (Oregon)
- Asia Pacifico (Hong Kong)
- Asia Pacific (Hyderabad)
- Asia Pacifico (Giacarta)
- Asia Pacifico (Melbourne)
- Asia Pacifico (Mumbai)
- Asia Pacifico (Nuova Zelanda)
- Asia Pacifico (Osaka-Locale)
- Asia Pacific (Seul)

- Asia Pacifico (Singapore)
- Asia Pacifico (Sydney)
- Asia Pacifico (Tokyo)
- Canada (Centrale)
- Canada occidentale (Calgary)
- Cina (Ningxia)
- Europa (Francoforte)
- Europa (Irlanda)
- Europe (London)
- Europa (Parigi)
- Europa (Spagna)
- Europa (Stoccolma)
- Europa (Zurigo)
- Israele (Tel Aviv)
- Medio Oriente (Emirati Arabi Uniti)
- Sud America (San Paolo)

Argomenti

- Esportazione dei dati del cluster in S3 con Aurora MySQL
- Esportazione di dati del cluster in S3 con Aurora PostgreSQL

Esportazione dei dati del cluster in S3 con Aurora MySQL

Tutte le versioni del motore Aurora MySQL attualmente disponibili supportano l'esportazione del cluster database in Amazon S3. Per ulteriori informazioni sulle versioni, consulta Note di rilascio di Aurora MySQL.

Esportazione di dati del cluster in S3 con Aurora PostgreSQL

Tutte le versioni del motore Aurora PostgreSQL attualmente disponibili supportano l'esportazione dei dati del cluster database in Amazon S3. Per ulteriori informazioni sulle versioni, consulta Release Notes for Aurora PostgreSQL.

Regioni e motori di database Aurora supportati per l’esportazione dei dati dello snapshot in Amazon S3

È possibile esportare i dati dello snapshot del cluster database Aurora in un bucket Amazon S3. È possibile esportare snapshot manuali e snapshot automatici di sistema. Dopo l'esportazione dei dati, è possibile analizzare i dati esportati direttamente mediante strumenti quali Amazon Athena o Amazon Redshift Spectrum. Per ulteriori informazioni, consulta Esportazione dei dati dello snapshot del cluster di database in Amazon S3.

L'esportazione di istantanee in S3 è disponibile in tutte le versioni tranne le seguenti: Regioni AWS

- Asia Pacifico (Malesia)
- Asia Pacifico (Nuova Zelanda)
- Asia Pacifico (Taipei)
- Asia Pacifico (Thailandia)
- Messico (centrale)
- AWS GovCloud (Stati Uniti orientali)
- AWS GovCloud (Stati Uniti occidentali)

Argomenti

- Esportazione dei dati dello snapshot in S3 con Aurora MySQL
- Esportazione dei dati dello snapshot in S3 con Aurora PostgreSQL

Esportazione dei dati dello snapshot in S3 con Aurora MySQL

Tutte le versioni del motore Aurora MySQL attualmente disponibili supportano l'esportazione dei dati dello snapshot del cluster database in Amazon S3. Per ulteriori informazioni sulle versioni, consulta Note di rilascio di Aurora MySQL.

Esportazione dei dati dello snapshot in S3 con Aurora PostgreSQL

Tutte le versioni del motore Aurora PostgreSQL attualmente disponibili supportano l'esportazione dei dati dello snapshot del cluster database in Amazon S3. Per ulteriori informazioni sulle versioni, consulta Release Notes for Aurora PostgreSQL.

Regioni e motori di database supportati per i database globali Aurora

Un database globale Aurora è un singolo database che si estende su più databaseRegioni AWS, che consente letture globali a bassa latenza e disaster recovery da qualsiasi interruzione a livello regionale. Fornisce una tolleranza agli errori integrata per la distribuzione perché l'istanza DB non si basa su una singolaRegione AWS, ma su più regioni e diverse zone di disponibilità. Per ulteriori informazioni, consulta Utilizzo di Database globale Amazon Aurora.

Argomenti

- Database globali Aurora con Aurora MySQL
- Database globali Aurora con Aurora PostgreSQL

Database globali Aurora con Aurora MySQL

Le regioni e le versioni del motore riportate di seguito sono disponibili per i database globali Aurora con Aurora MySQL.

Region	Aurora MySQL versione 3	Aurora MySQL versione 2
Stati Uniti orientali (Virginia settentrionale)	Versione 3.01.0 e successive	Versione 2.07.0 e versioni successive
Stati Uniti orientali (Ohio)	Versione 3.01.0 e successive	Versione 2.07.0 e versioni successive
Stati Uniti occidentali (California settentrionale)	Versione 3.01.0 e successive	Versione 2.07.0 e versioni successive
Stati Uniti occidentali (Oregon)	Versione 3.01.0 e successive	Versione 2.07.0 e versioni successive
Africa (Città del Capo)	Versione 3.01.0 e successive	Versione 2.07.1 e versioni successive
Asia Pacifico (Hong Kong)	Versione 3.01.0 e successive	Versione 2.07.1 e versioni successive
Asia Pacifico (Hyderabad)	Versione 3.02.0 e successive	Versione 2.11.2 e successive

Region	Aurora MySQL versione 3	Aurora MySQL versione 2
Asia Pacifico (Giacarta)	Versione 3.01.0 e successive	Versione 2.07.6 e successive
Asia Pacifico (Malesia)	Versione 3.04.0 e successive	Versione 2.07.6 e successive
Asia Pacifico (Melbourne)	Versione 3.03.0 e successive	Versione 2.07.6 e successive
Asia Pacifico (Mumbai)	Versione 3.01.0 e successive	Versione 2.07.0 e versioni successive
Asia Pacifico (Nuova Zelanda)	Versione 3.01.0 e successive	Versione 2.07.0 e versioni successive
Asia Pacifico (Osaka)	Versione 3.01.0 e successive	Versione 2.07.3 e versioni successive
Asia Pacifico (Seoul)	Versione 3.01.0 e successive	Versione 2.07.0 e versioni successive
Asia Pacifico (Singapore)	Versione 3.01.0 e successive	Versione 2.07.0 e versioni successive
Asia Pacifico (Sydney)	Versione 3.01.0 e successive	Versione 2.07.0 e versioni successive
Asia Pacifico (Taipei)	Versione 3.01.0 e successive	Versione 2.07.0 e versioni successive
Asia Pacifico (Thailandia)	Versione 3.01.0 e successive	Versione 2.07.0 e versioni successive
Asia Pacifico (Tokyo)	Versione 3.01.0 e successive	Versione 2.07.0 e versioni successive
Canada (Centrale)	Versione 3.01.0 e successive	Versione 2.07.0 e versioni successive
Canada occidentale (Calgary)	Versione 3.01.0 e successive	Versione 2.07.0 e versioni successive

Region	Aurora MySQL versione 3	Aurora MySQL versione 2
Cina (Pechino)	Versione 3.01.0 e successive	Versione 2.07.2 e versioni successive
Cina (Ningxia)	Versione 3.01.0 e successive	Versione 2.07.2 e versioni successive
Europa (Francoforte)	Versione 3.01.0 e successive	Versione 2.07.0 e versioni successive
Europa (Irlanda)	Versione 3.01.0 e successive	Versione 2.07.0 e versioni successive
Europe (Londra)	Versione 3.01.0 e successive	Versione 2.07.0 e versioni successive
Europa (Milano)	Versione 3.01.0 e successive	Versione 2.07.1 e versioni successive
Europa (Parigi)	Versione 3.01.0 e successive	Versione 2.07.0 e versioni successive
Europa (Spagna)	Versione 3.02.0 e successive	Versione 2.07.6 e successive
Europa (Stoccolma)	Versione 3.01.0 e successive	Versione 2.07.0 e versioni successive
Europa (Zurigo)	Versione 3.02.0 e successive	Versione 2.07.6 e successive
Israele (Tel Aviv)	Versione 3.02.0 e successive	Versione 2.07.6 e successive
Messico (centrale)	Versione 3.02.0 e successive	Versione 2.07.6 e successive
Medio Oriente (Bahrein)	Versione 3.01.0 e successive	Versione 2.07.1 e versioni successive
Medio Oriente (Emirati Arabi Uniti)	Versione 3.02.0 e successive	Versione 2.07.6 e successive

Region	Aurora MySQL versione 3	Aurora MySQL versione 2
Sud America (San Paolo)	Versione 3.01.0 e successive	Versione 2.07.1 e versioni successive
AWSGovCloud (Stati Uniti orientali)	Versione 3.01.0 e successive	Versione 2.07.0 e versioni successive
AWSGovCloud (Stati Uniti occidentali)	Versione 3.01.0 e successive	Versione 2.07.0 e versioni successive

Database globali Aurora con Aurora PostgreSQL

Le regioni e le versioni del motore riportate di seguito sono disponibili per i database globali Aurora con Aurora PostgreSQL.

Region	Aurora PostgreSQL 17	Aurora PostgreSQL 16	Aurora PostgreSQL 15	Aurora PostgreSQL 14	Aurora PostgreSQL 13	Aurora PostgreSQL 12	Aurora PostgreSQL 11
Stati Uniti orientali (Virginia settentrionale)	Versione 17.4 e successive	Versione 16.1 e successive	Versione 15.2 e successive	Versione 14.3 e versioni successive	Versione 13.4 e successive	Versione 12.8 e successive	Versione 11.9 e versione 11.13 e successive
Stati Uniti orientali (Ohio)	Versione 17.4 e successive	Versione 16.1 e successive	Versione 15.2 e successive	Versione 14.3 e versioni successive	Versione 13.4 e successive	Versione 12.8 e successive	Versione 11.9 e versione 11.13 e successive
Stati Uniti occidentali	Versione 17.4 e	Versione 16.1 e	Versione 15.2 e	Versione 14.3 e versioni	Versione 13.4 e	Versione 12.8 e	Versione 11.9 e versione

Region	Aurora PostgreSQL 17	Aurora PostgreSQL 16	Aurora PostgreSQL 15	Aurora PostgreSQL 14	Aurora PostgreSQL 13	Aurora PostgreSQL 12	Aurora PostgreSQL 11
(California settentrionale)	successive	successive	successive	successive	successive	successive	11.13 e successive
Stati Uniti occidentali (Oregon)	Versione 17.4 e successive	Versione 16.1 e successive	Versione 15.2 e successive	Versione 14.3 e versioni successive	Versione 13.4 e successive	Versione 12.8 e successive	Versione 11.9 e versione 11.13 e successive
Africa (Città del Capo)	Versione 17.4 e successive	Versione 16.1 e successive	Versione 15.2 e successive	Versione 14.3 e versioni successive	Versione 13.4 e successive	Versione 12.8 e successive	Versione 11.9 e versione 11.13 e successive
Asia Pacifico (Hong Kong)	Versione 17.4 e successive	Versione 16.1 e successive	Versione 15.2 e successive	Versione 14.3 e versioni successive	Versione 13.4 e successive	Versione 12.8 e successive	Versione 11.9 e versione 11.13 e successive
Asia Pacifico (Hyderabad)	Versione 17.4 e successive	Versione 16.1 e successive	Versione 15.2 e successive	Versione 14.3 e versioni successive	Versione 13.4 e successive	Versione 12.8 e successive	Versione 11.9 e versione 11.13 e successive

Region	Aurora PostgreSQL 17	Aurora PostgreSQL 16	Aurora PostgreSQL 15	Aurora PostgreSQL 14	Aurora PostgreSQL 13	Aurora PostgreSQL 12	Aurora PostgreSQL 11
Asia Pacifico (Giacarta)	Versione 17.4 e successive	Versione 16.1 e successive	Versione 15.2 e successive	Versione 14.3 e versioni successive	Versione 13.4 e successive	Versione 12.8 e successive	Versione 11.9 e versione 11.13 e successive
Asia Pacifico (Malesia)	Versione 17.4 e successive	Versione 16.1 e successive	Versione 15.2 e successive	Versione 14.3 e versioni successive	Versione 13.4 e successive	Versione 12.8 e successive	Versione 11.9 e versione 11.13 e successive
Asia Pacifico (Melbourne)	Versione 17.4 e successive	Versione 16.1 e successive	Versione 15.2 e successive	Versione 14.3 e versioni successive	Versione 13.4 e successive	Versione 12.8 e successive	Versione 11.9 e versione 11.13 e successive
Asia Pacifico (Mumbai)	Versione 17.4 e successive	Versione 16.1 e successive	Versione 15.2 e successive	Versione 14.3 e versioni successive	Versione 13.4 e successive	Versione 12.8 e successive	Versione 11.9 e versione 11.13 e successive

Region	Aurora PostgreSQL 17	Aurora PostgreSQL 16	Aurora PostgreSQL 15	Aurora PostgreSQL 14	Aurora PostgreSQL 13	Aurora PostgreSQL 12	Aurora PostgreSQL 11
Asia Pacifico (Nuova Zelanda)	Versione 17.4 e successive	Versione 16.1 e successive	Versione 15.2 e successive	Versione 14.3 e versioni successive	Versione 13.4 e successive	Versione 12.8 e successive	Versione 11.9 e versione 11.13 e successive
Asia Pacifico (Osaka)	Versione 17.4 e successive	Versione 16.1 e successive	Versione 15.2 e successive	Versione 14.3 e versioni successive	Versione 13.4 e successive	Versione 12.8 e successive	Versione 11.9 e versione 11.13 e successive
Asia Pacifico (Seoul)	Versione 17.4 e successive	Versione 16.1 e successive	Versione 15.2 e successive	Versione 14.3 e versioni successive	Versione 13.4 e successive	Versione 12.8 e successive	Versione 11.9 e versione 11.13 e successive
Asia Pacifico (Singapore)	Versione 17.4 e successive	Versione 16.1 e successive	Versione 15.2 e successive	Versione 14.3 e versioni successive	Versione 13.4 e successive	Versione 12.8 e successive	Versione 11.9 e versione 11.13 e successive

Region	Aurora PostgreSQL 17	Aurora PostgreSQL 16	Aurora PostgreSQL 15	Aurora PostgreSQL 14	Aurora PostgreSQL 13	Aurora PostgreSQL 12	Aurora PostgreSQL 11
Asia Pacifico (Sydney)	Versione 17.4 e successive	Versione 16.1 e successive	Versione 15.2 e successive	Versione 14.3 e versioni successive	Versione 13.4 e successive	Versione 12.8 e successive	Versione 11.9 e versione 11.13 e successive
Asia Pacifico (Sydney)	Tutte le versioni	Tutte le versioni	Tutte le versioni	Tutte le versioni	Tutte le versioni	Tutte le versioni	Tutte le versioni
Asia Pacifico (Taipei)	Versione 17.4 e successive	Versione 16.1 e successive	Versione 15.2 e successive	Versione 14.3 e versioni successive	Versione 13.4 e successive	Versione 12.8 e successive	Versione 11.9 e versione 11.13 e successive
Asia Pacifico (Thailandia)	Versione 17.4 e successive	Versione 16.1 e successive	Versione 15.2 e successive	Versione 14.3 e versioni successive	Versione 13.4 e successive	Versione 12.8 e successive	Versione 11.9 e versione 11.13 e successive
Asia Pacifico (Tokyo)	Versione 17.4 e successive	Versione 16.1 e successive	Versione 15.2 e successive	Versione 14.3 e versioni successive	Versione 13.4 e successive	Versione 12.8 e successive	Versione 11.9 e versione 11.13 e successive

Region	Aurora PostgreSQL 17	Aurora PostgreSQL 16	Aurora PostgreSQL 15	Aurora PostgreSQL 14	Aurora PostgreSQL 13	Aurora PostgreSQL 12	Aurora PostgreSQL 11
Canada (Centrale)	Versione 17.4 e successive	Versione 16.1 e successive	Versione 15.2 e successive	Versione 14.3 e versioni successive	Versione 13.4 e successive	Versione 12.8 e successive	Versione 11.9 e versione 11.13 e successive
Canada occidentale (Calgary)	Versione 17.4 e successive	Versione 16.1 e successive	Versione 15.2 e successive	Versione 14.3 e versioni successive	Versione 13.4 e successive	Versione 12.8 e successive	Versione 11.9 e versione 11.13 e successive
Cina (Pechino)	Versione 17.4 e successive	Versione 16.1 e successive	Versione 15.2 e successive	Versione 14.3 e versioni successive	Versione 13.4 e successive	Versione 12.8 e successive	Versione 11.9 e versione 11.13 e successive
Cina (Ningxia)	Versione 17.4 e successive	Versione 16.1 e successive	Versione 15.2 e successive	Versione 14.3 e versioni successive	Versione 13.4 e successive	Versione 12.8 e successive	Versione 11.9 e versione 11.13 e successive

Region	Aurora PostgreSQL 17	Aurora PostgreSQL 16	Aurora PostgreSQL 15	Aurora PostgreSQL 14	Aurora PostgreSQL 13	Aurora PostgreSQL 12	Aurora PostgreSQL 11
Europa (Francoforte)	Versione 17.4 e successive	Versione 16.1 e successive	Versione 15.2 e successive	Versione 14.3 e versioni successive	Versione 13.4 e successive	Versione 12.8 e successive	Versione 11.9 e versione 11.13 e successive
Europa (Irlanda)	Versione 17.4 e successive	Versione 16.1 e successive	Versione 15.2 e successive	Versione 14.3 e versioni successive	Versione 13.4 e successive	Versione 12.8 e successive	Versione 11.9 e versione 11.13 e successive
Europe (Londra)	Versione 17.4 e successive	Versione 16.1 e successive	Versione 15.2 e successive	Versione 14.3 e versioni successive	Versione 13.4 e successive	Versione 12.8 e successive	Versione 11.9 e versione 11.13 e successive
Europa (Milano)	Versione 17.4 e successive	Versione 16.1 e successive	Versione 15.2 e successive	Versione 14.3 e versioni successive	Versione 13.4 e successive	Versione 12.8 e successive	Versione 11.9 e versione 11.13 e successive

Region	Aurora PostgreSQL 17	Aurora PostgreSQL 16	Aurora PostgreSQL 15	Aurora PostgreSQL 14	Aurora PostgreSQL 13	Aurora PostgreSQL 12	Aurora PostgreSQL 11
Europa (Parigi)	Versione 17.4 e successive	Versione 16.1 e successive	Versione 15.2 e successive	Versione 14.3 e versioni successive	Versione 13.4 e successive	Versione 12.8 e successive	Versione 11.9 e versione 11.13 e successive
Europa (Spagna)	Versione 17.4 e successive	Versione 16.1 e successive	Versione 15.2 e successive	Versione 14.3 e versioni successive	Versione 13.4 e successive	Versione 12.8 e successive	Versione 11.9 e versione 11.13 e successive
Europa (Stoccolma)	Versione 17.4 e successive	Versione 16.1 e successive	Versione 15.2 e successive	Versione 14.3 e versioni successive	Versione 13.4 e successive	Versione 12.8 e successive	Versione 11.9 e versione 11.13 e successive
Europa (Zurigo)	Versione 17.4 e successive	Versione 16.1 e successive	Versione 15.2 e successive	Versione 14.3 e versioni successive	Versione 13.4 e successive	Versione 12.8 e successive	Versione 11.9 e versione 11.13 e successive

Region	Aurora PostgreSQL 17	Aurora PostgreSQL 16	Aurora PostgreSQL 15	Aurora PostgreSQL 14	Aurora PostgreSQL 13	Aurora PostgreSQL 12	Aurora PostgreSQL 11
Israele (Tel Aviv)	Versione 17.4 e successive	Versione 16.1 e successive	Versione 15.2 e successive	Versione 14.3 e versioni successive	Versione 13.4 e successive	Versione 12.8 e successive	Versione 11.9 e versione 11.13 e successive
Messico (centrale)	Versione 17.4 e successive	Versione 16.1 e successive	Versione 15.2 e successive	Versione 14.3 e versioni successive	Versione 13.4 e successive	Versione 12.8 e successive	Versione 11.9 e versione 11.13 e successive
Medio Oriente (Bahrein)	Versione 17.4 e successive	Versione 16.1 e successive	Versione 15.2 e successive	Versione 14.3 e versioni successive	Versione 13.4 e successive	Versione 12.8 e successive	Versione 11.9 e versione 11.13 e successive
Medio Oriente (Emirati Arabi Uniti)	Versione 17.4 e successive	Versione 16.1 e successive	Versione 15.2 e successive	Versione 14.3 e versioni successive	Versione 13.4 e successive	Versione 12.8 e successive	Versione 11.9 e versione 11.13 e successive

Region	Aurora PostgreSQL 17	Aurora PostgreSQL 16	Aurora PostgreSQL 15	Aurora PostgreSQL 14	Aurora PostgreSQL 13	Aurora PostgreSQL 12	Aurora PostgreSQL 11
Sud America (San Paolo)	Versione 17.4 e successive	Versione 16.1 e successive	Versione 15.2 e successive	Versione 14.3 e versioni successive	Versione 13.4 e successive	Versione 12.8 e successive	Versione 11.9 e versione 11.13 e successive
AWSGovCloud (Stati Uniti orientali)	Versione 17.4 e successive	Versione 16.1 e successive	Versione 15.2 e successive	Versione 14.3 e versioni successive	Versione 13.4 e successive	Versione 12.8 e successive	Versione 11.9 e versione 11.13 e successive
AWSGovCloud (Stati Uniti occidentali)	Versione 17.4 e successive	Versione 16.1 e successive	Versione 15.2 e successive	Versione 14.3 e versioni successive	Versione 13.4 e successive	Versione 12.8 e successive	Versione 11.9 e versione 11.13 e successive

Regioni e motori di database Aurora supportati per l’autenticazione del database IAM

Con l'autenticazione del database IAM in Aurora, è possibile eseguire l'autenticazione al cluster database utilizzando l'autenticazione AWS Identity and Access Management (IAM). Questo metodo di autenticazione non richiede l'uso di una password quando si esegue la connessione al cluster database. Utilizzi invece un token di autenticazione. Per ulteriori informazioni, consulta Autenticazione del database IAM .

Argomenti

- Autenticazione database IAM con Aurora MySQL

- Autenticazione database IAM con Aurora PostgreSQL

Autenticazione database IAM con Aurora MySQL

L'autenticazione database IAM con Aurora MySQL è disponibile in tutte le regioni per le seguenti versioni:

- Aurora MySQL 3, tutte le versioni disponibili
- Aurora MySQL 2, tutte le versioni disponibili

Autenticazione database IAM con Aurora PostgreSQL

L'autenticazione database IAM con Aurora PostgreSQL è disponibile in tutte le regioni per le seguenti versioni del motore:

- Aurora PostgreSQL 17: tutte le versioni disponibili
- Aurora PostgreSQL 16: tutte le versioni disponibili
- Aurora PostgreSQL 15: tutte le versioni disponibili
- Aurora PostgreSQL 14, tutte le versioni disponibili
- Aurora PostgreSQL 13, tutte le versioni disponibili
- Aurora PostgreSQL 12, tutte le versioni disponibili
- Aurora PostgreSQL 11, tutte le versioni disponibili

Regioni e motori di database Aurora supportati per l’autenticazione Kerberos

Se si utilizza l'autenticazione Kerberos con Aurora, è possibile supportare l'autenticazione esterna degli utenti del database mediante Kerberos e Microsoft Active Directory. L'utilizzo di Kerberos e Active Directory offre i vantaggi dell'autenticazione Single Sign-On centralizzata degli utenti dei database. Kerberos e Active Directory sono disponibili con AWS Directory Service for Microsoft Active Directory, una funzionalità di. Directory Service Per ulteriori informazioni, consulta Autenticazione Kerberos.

Argomenti

- Autenticazione Kerberos con Aurora MySQL

- Autenticazione Kerberos con Aurora PostgreSQL
- Gruppi di sicurezza Active Directory (AD) con Aurora PostgreSQL

Autenticazione Kerberos con Aurora MySQL

Di seguito sono riportate le versioni di motore e le regioni disponibili per l'autenticazione Kerberos con Aurora MySQL.

Region	Aurora MySQL versione 3
Stati Uniti orientali (Virginia settentrionale)	Versione 3.03.0 e successive
Stati Uniti orientali (Ohio)	Versione 3.03.0 e successive
Stati Uniti occidentali (California settentrionale)	Versione 3.03.0 e successive
Stati Uniti occidentali (Oregon)	Versione 3.03.0 e successive
Africa (Città del Capo)	Non disponibile
Asia Pacifico (Hong Kong)	Non disponibile
Asia Pacifico (Giacarta)	Non disponibile
Asia Pacifico (Malesia)	Non disponibile
Asia Pacifico (Melbourne)	Non disponibile
Asia Pacifico (Mumbai)	Versione 3.03.0 e successive
Asia Pacifico (Nuova Zelanda)	Non disponibile
Asia Pacifico (Osaka-Locale)	Non disponibile
Asia Pacifico (Seul)	Versione 3.03.0 e successive
Asia Pacifico (Singapore)	Versione 3.03.0 e successive
Asia Pacifico (Sydney)	Versione 3.03.0 e successive
Asia Pacifico (Taipei)	Non disponibile

Region	Aurora MySQL versione 3
Asia Pacifico (Thailandia)	Non disponibile
Asia Pacifico (Tokyo)	Versione 3.03.0 e successive
Canada (Centrale)	Versione 3.03.0 e successive
Canada occidentale (Calgary)	Non disponibile
Cina (Pechino)	Versione 3.03.0 e successive
Cina (Ningxia)	Versione 3.03.0 e successive
Europa (Francoforte)	Versione 3.03.0 e successive
Europa (Irlanda)	Versione 3.03.0 e successive
Europa (Londra)	Versione 3.03.0 e successive
Europa (Milano)	Non disponibile
Europa (Parigi)	Versione 3.03.0 e successive
Europa (Spagna)	Non disponibile
Europa (Stoccolma)	Versione 3.03.0 e successive
Europa (Zurigo)	Non disponibile
Israele (Tel Aviv)	Non disponibile
Messico (centrale)	Non disponibile
Medio Oriente (Bahrein)	Non disponibile
Medio Oriente (Emirati Arabi Uniti)	Non disponibile
Sud America (San Paolo)	Versione 3.03.0 e successive
AWS GovCloud (Stati Uniti orientali)	Versione 3.03.0 e successive

Region	Aurora MySQL versione 3
AWS GovCloud (Stati Uniti occidentali)	Versione 3.03.0 e successive

Autenticazione Kerberos con Aurora PostgreSQL

Di seguito sono riportate le versioni di motore e le regioni disponibili per l'autenticazione Kerberos con Aurora PostgreSQL.

Region	Aurora PostgreSQL 17	Aurora PostgreSQL 16	Aurora PostgreSQL 15	Aurora PostgreSQL 14	Aurora PostgreSQL 13	Aurora PostgreSQL 12	Aurora PostgreSQL 11
Stati Uniti orientali (Virginia settentrionale)	Tutte le versioni	Tutte le versioni	Tutte le versioni	Tutte le versioni	Tutte le versioni	Tutte le versioni	Tutte le versioni
Stati Uniti orientali (Ohio)	Tutte le versioni	Tutte le versioni	Tutte le versioni	Tutte le versioni	Tutte le versioni	Tutte le versioni	Tutte le versioni
Stati Uniti occidentali (California settentrionale)	Tutte le versioni	Tutte le versioni	Tutte le versioni	Tutte le versioni	Tutte le versioni	Tutte le versioni	Tutte le versioni
Stati Uniti occidentali (Oregon)	Tutte le versioni	Tutte le versioni	Tutte le versioni	Tutte le versioni	Tutte le versioni	Tutte le versioni	Tutte le versioni

Region	Aurora PostgreSQL 17	Aurora PostgreSQL 16	Aurora PostgreSQL 15	Aurora PostgreSQL 14	Aurora PostgreSQL 13	Aurora PostgreSQL 12	Aurora PostgreSQL 11
Africa (Città del Capo)	Tutte le versioni	Tutte le versioni	Tutte le versioni	Tutte le versioni	Tutte le versioni	Tutte le versioni	Tutte le versioni
Asia Pacifico (Hong Kong)	Tutte le versioni	Tutte le versioni	Tutte le versioni	Tutte le versioni	Tutte le versioni	Tutte le versioni	Tutte le versioni
Asia Pacific (Hyderabad)	Tutte le versioni	Tutte le versioni	Tutte le versioni	Tutte le versioni	Tutte le versioni	Tutte le versioni	Tutte le versioni
Asia Pacifico (Giacarta)	Tutte le versioni	Tutte le versioni	Tutte le versioni	Tutte le versioni	Tutte le versioni	Tutte le versioni	Tutte le versioni
Asia Pacifico (Malesia)	Non disponibile	Non disponibile	Non disponibile	Non disponibile	Non disponibile	Non disponibile	Non disponibile
Asia Pacifico (Melbourne)	Tutte le versioni	Tutte le versioni	Tutte le versioni	Tutte le versioni	Tutte le versioni	Tutte le versioni	Tutte le versioni
Asia Pacifico (Mumbai)	Tutte le versioni	Tutte le versioni	Tutte le versioni	Tutte le versioni	Tutte le versioni	Tutte le versioni	Tutte le versioni

Region	Aurora PostgreSQL 17	Aurora PostgreSQL 16	Aurora PostgreSQL 15	Aurora PostgreSQL 14	Aurora PostgreSQL 13	Aurora PostgreSQL 12	Aurora PostgreSQL 11
Asia Pacifico (Nuova Zelanda)	Non disponibile	Non disponibile	Non disponibile	Non disponibile	Non disponibile	Non disponibile	Non disponibile
Asia Pacifico (Osaka-Locale)	Non disponibile	Non disponibile	Non disponibile	Non disponibile	Non disponibile	Non disponibile	Non disponibile
Asia Pacifico (Seul)	Tutte le versioni	Tutte le versioni	Tutte le versioni	Tutte le versioni	Tutte le versioni	Tutte le versioni	Tutte le versioni
Asia Pacifico (Singapore)	Tutte le versioni	Tutte le versioni	Tutte le versioni	Tutte le versioni	Tutte le versioni	Tutte le versioni	Tutte le versioni
Asia Pacifico (Sydney)	Tutte le versioni	Tutte le versioni	Tutte le versioni	Tutte le versioni	Tutte le versioni	Tutte le versioni	Tutte le versioni
Asia Pacifico (Taipei)	Non disponibile	Non disponibile	Non disponibile	Non disponibile	Non disponibile	Non disponibile	Non disponibile
Asia Pacifico (Thailandia)	Non disponibile	Non disponibile	Non disponibile	Non disponibile	Non disponibile	Non disponibile	Non disponibile

Region	Aurora PostgreSQL 17	Aurora PostgreSQL 16	Aurora PostgreSQL 15	Aurora PostgreSQL 14	Aurora PostgreSQL 13	Aurora PostgreSQL 12	Aurora PostgreSQL 11
Asia Pacifico (Tokyo)	Tutte le versioni	Tutte le versioni	Tutte le versioni	Tutte le versioni	Tutte le versioni	Tutte le versioni	Tutte le versioni
Canada (Centrale)	Tutte le versioni	Tutte le versioni	Tutte le versioni	Tutte le versioni	Tutte le versioni	Tutte le versioni	Tutte le versioni
Canada occidentale (Calgary)	Tutte le versioni	Tutte le versioni	Tutte le versioni	Tutte le versioni	Tutte le versioni	Tutte le versioni	Tutte le versioni
Cina (Pechino)	Tutte le versioni	Tutte le versioni	Tutte le versioni	Tutte le versioni	Tutte le versioni	Tutte le versioni	Tutte le versioni
Cina (Ningxia)	Tutte le versioni	Tutte le versioni	Tutte le versioni	Tutte le versioni	Tutte le versioni	Tutte le versioni	Tutte le versioni
Europa (Francoforte)	Tutte le versioni	Tutte le versioni	Tutte le versioni	Tutte le versioni	Tutte le versioni	Tutte le versioni	Tutte le versioni
Europa (Irlanda)	Tutte le versioni	Tutte le versioni	Tutte le versioni	Tutte le versioni	Tutte le versioni	Tutte le versioni	Tutte le versioni
Europa (Londra)	Tutte le versioni	Tutte le versioni	Tutte le versioni	Tutte le versioni	Tutte le versioni	Tutte le versioni	Tutte le versioni
Europa (Milano)	Tutte le versioni	Tutte le versioni	Tutte le versioni	Tutte le versioni	Tutte le versioni	Tutte le versioni	Tutte le versioni
Europa (Parigi)	Tutte le versioni	Tutte le versioni	Tutte le versioni	Tutte le versioni	Tutte le versioni	Tutte le versioni	Tutte le versioni

Region	Aurora PostgreSQL 17	Aurora PostgreSQL 16	Aurora PostgreSQL 15	Aurora PostgreSQL 14	Aurora PostgreSQL 13	Aurora PostgreSQL 12	Aurora PostgreSQL 11
Europa (Spagna)	Tutte le versioni	Tutte le versioni	Tutte le versioni	Tutte le versioni	Tutte le versioni	Tutte le versioni	Tutte le versioni
Europa (Stoccolma)	Tutte le versioni	Tutte le versioni	Tutte le versioni	Tutte le versioni	Tutte le versioni	Tutte le versioni	Tutte le versioni
Europa (Zurigo)	Tutte le versioni	Tutte le versioni	Tutte le versioni	Tutte le versioni	Tutte le versioni	Tutte le versioni	Tutte le versioni
Israele (Tel Aviv)	Tutte le versioni	Tutte le versioni	Tutte le versioni	Tutte le versioni	Tutte le versioni	Tutte le versioni	Tutte le versioni
Messico (centrale)	Non disponibile	Non disponibile	Non disponibile	Non disponibile	Non disponibile	Non disponibile	Non disponibile
Medio Oriente (Bahrein)	Tutte le versioni	Tutte le versioni	Tutte le versioni	Tutte le versioni	Tutte le versioni	Tutte le versioni	Tutte le versioni
Medio Oriente (Emirati Arabi Uniti)	Tutte le versioni	Tutte le versioni	Tutte le versioni	Tutte le versioni	Tutte le versioni	Tutte le versioni	Tutte le versioni
Sud America (San Paolo)	Tutte le versioni	Tutte le versioni	Tutte le versioni	Tutte le versioni	Tutte le versioni	Tutte le versioni	Tutte le versioni

Region	Aurora PostgreSQL 17	Aurora PostgreSQL 16	Aurora PostgreSQL 15	Aurora PostgreSQL 14	Aurora PostgreSQL 13	Aurora PostgreSQL 12	Aurora PostgreSQL 11
AWS GovCloud (Stati Uniti orientali)	Tutte le versioni	Tutte le versioni	Tutte le versioni	Tutte le versioni	Tutte le versioni	Tutte le versioni	Tutte le versioni
AWS GovCloud (Stati Uniti occidentali)	Tutte le versioni	Tutte le versioni	Tutte le versioni	Tutte le versioni	Tutte le versioni	Tutte le versioni	Tutte le versioni

Gruppi di sicurezza Active Directory (AD) con Aurora PostgreSQL

Le seguenti regioni e versioni del motore sono disponibili per ActiveDirectory Aurora PostgreSQL.

Region	Aurora PostgreSQL 17	Aurora PostgreSQL 16	Aurora PostgreSQL 15	Aurora PostgreSQL 14	Aurora PostgreSQL 13	Aurora PostgreSQL 12	Aurora PostgreSQL 11
Stati Uniti orientali (Virginia settentrionale)	Tutte le versioni	Tutte le versioni	Tutte le versioni	Tutte le versioni	Tutte le versioni	Tutte le versioni	Tutte le versioni
Stati Uniti orientali (Ohio)	Tutte le versioni	Tutte le versioni	Tutte le versioni	Tutte le versioni	Tutte le versioni	Tutte le versioni	Tutte le versioni
Stati Uniti occidenta	Tutte le versioni	Tutte le versioni	Tutte le versioni	Tutte le versioni	Tutte le versioni	Tutte le versioni	Tutte le versioni

Region	Aurora PostgreSQL 17	Aurora PostgreSQL 16	Aurora PostgreSQL 15	Aurora PostgreSQL 14	Aurora PostgreSQL 13	Aurora PostgreSQL 12	Aurora PostgreSQL 11
li (California settentrionale)							
Stati Uniti occidentali (Oregon)	Tutte le versioni	Tutte le versioni	Tutte le versioni	Tutte le versioni	Tutte le versioni	Tutte le versioni	Tutte le versioni
Africa (Città del Capo)	Tutte le versioni	Tutte le versioni	Tutte le versioni	Tutte le versioni	Tutte le versioni	Tutte le versioni	Tutte le versioni
Asia Pacifico (Hong Kong)	Tutte le versioni	Tutte le versioni	Tutte le versioni	Tutte le versioni	Tutte le versioni	Tutte le versioni	Tutte le versioni
Asia Pacific (Hyderabad)	Tutte le versioni	Tutte le versioni	Tutte le versioni	Tutte le versioni	Tutte le versioni	Tutte le versioni	Tutte le versioni
Asia Pacifico (Giacarta)	Tutte le versioni	Tutte le versioni	Tutte le versioni	Tutte le versioni	Tutte le versioni	Tutte le versioni	Tutte le versioni
Asia Pacifico (Malesia)	Non disponibile	Non disponibile	Non disponibile	Non disponibile	Non disponibile	Non disponibile	Non disponibile

Region	Aurora PostgreSQL 17	Aurora PostgreSQL 16	Aurora PostgreSQL 15	Aurora PostgreSQL 14	Aurora PostgreSQL 13	Aurora PostgreSQL 12	Aurora PostgreSQL 11
Asia Pacifico (Melbourne)	Tutte le versioni	Tutte le versioni	Tutte le versioni	Tutte le versioni	Tutte le versioni	Tutte le versioni	Tutte le versioni
Asia Pacifico (Mumbai)	Tutte le versioni	Tutte le versioni	Tutte le versioni	Tutte le versioni	Tutte le versioni	Tutte le versioni	Tutte le versioni
Asia Pacifico (Nuova Zelanda)	Non disponibile	Non disponibile	Non disponibile	Non disponibile	Non disponibile	Non disponibile	Non disponibile
Asia Pacifico (Osaka-Locale)	Non disponibile	Non disponibile	Non disponibile	Non disponibile	Non disponibile	Non disponibile	Non disponibile
Asia Pacifico (Seul)	Tutte le versioni	Tutte le versioni	Tutte le versioni	Tutte le versioni	Tutte le versioni	Tutte le versioni	Tutte le versioni
Asia Pacifico (Singapore)	Tutte le versioni	Tutte le versioni	Tutte le versioni	Tutte le versioni	Tutte le versioni	Tutte le versioni	Tutte le versioni
Asia Pacifico (Sydney)	Tutte le versioni	Tutte le versioni	Tutte le versioni	Tutte le versioni	Tutte le versioni	Tutte le versioni	Tutte le versioni

Region	Aurora PostgreSQL 17	Aurora PostgreSQL 16	Aurora PostgreSQL 15	Aurora PostgreSQL 14	Aurora PostgreSQL 13	Aurora PostgreSQL 12	Aurora PostgreSQL 11
Asia Pacifico (Taipei)	Non disponibile	Non disponibile	Non disponibile	Non disponibile	Non disponibile	Non disponibile	Non disponibile
Asia Pacifico (Thailandia)	Non disponibile	Non disponibile	Non disponibile	Non disponibile	Non disponibile	Non disponibile	Non disponibile
Asia Pacifico (Tokyo)	Tutte le versioni	Tutte le versioni	Tutte le versioni	Tutte le versioni	Tutte le versioni	Tutte le versioni	Tutte le versioni
Canada (Centrale)	Tutte le versioni	Tutte le versioni	Tutte le versioni	Tutte le versioni	Tutte le versioni	Tutte le versioni	Tutte le versioni
Canada occidentale (Calgary)	Non disponibile	Non disponibile	Non disponibile	Non disponibile	Non disponibile	Non disponibile	Non disponibile
Cina (Pechino)	Tutte le versioni	Tutte le versioni	Tutte le versioni	Tutte le versioni	Tutte le versioni	Tutte le versioni	Tutte le versioni
Cina (Ningxia)	Tutte le versioni	Tutte le versioni	Tutte le versioni	Tutte le versioni	Tutte le versioni	Tutte le versioni	Tutte le versioni
Europa (Francoforte)	Tutte le versioni	Tutte le versioni	Tutte le versioni	Tutte le versioni	Tutte le versioni	Tutte le versioni	Tutte le versioni
Europa (Irlanda)	Tutte le versioni	Tutte le versioni	Tutte le versioni	Tutte le versioni	Tutte le versioni	Tutte le versioni	Tutte le versioni

Region	Aurora PostgreSQL 17	Aurora PostgreSQL 16	Aurora PostgreSQL 15	Aurora PostgreSQL 14	Aurora PostgreSQL 13	Aurora PostgreSQL 12	Aurora PostgreSQL 11
Europa (Londra)	Tutte le versioni	Tutte le versioni	Tutte le versioni	Tutte le versioni	Tutte le versioni	Tutte le versioni	Tutte le versioni
Europa (Milano)	Tutte le versioni	Tutte le versioni	Tutte le versioni	Tutte le versioni	Tutte le versioni	Tutte le versioni	Tutte le versioni
Europa (Parigi)	Tutte le versioni	Tutte le versioni	Tutte le versioni	Tutte le versioni	Tutte le versioni	Tutte le versioni	Tutte le versioni
Europa (Spagna)	Tutte le versioni	Tutte le versioni	Tutte le versioni	Tutte le versioni	Tutte le versioni	Tutte le versioni	Tutte le versioni
Europa (Stoccolma)	Tutte le versioni	Tutte le versioni	Tutte le versioni	Tutte le versioni	Tutte le versioni	Tutte le versioni	Tutte le versioni
Europa (Zurigo)	Tutte le versioni	Tutte le versioni	Tutte le versioni	Tutte le versioni	Tutte le versioni	Tutte le versioni	Tutte le versioni
Israele (Tel Aviv)	Tutte le versioni	Tutte le versioni	Tutte le versioni	Tutte le versioni	Tutte le versioni	Tutte le versioni	Tutte le versioni
Messico (centrale)	Non disponibile	Non disponibile	Non disponibile	Non disponibile	Non disponibile	Non disponibile	Non disponibile
Medio Oriente (Bahrein)	Tutte le versioni	Tutte le versioni	Tutte le versioni	Tutte le versioni	Tutte le versioni	Tutte le versioni	Tutte le versioni
Medio Oriente (Emirati Arabi Uniti)	Tutte le versioni	Tutte le versioni	Tutte le versioni	Tutte le versioni	Tutte le versioni	Tutte le versioni	Tutte le versioni

Region	Aurora PostgreSQL 17	Aurora PostgreSQL 16	Aurora PostgreSQL 15	Aurora PostgreSQL 14	Aurora PostgreSQL 13	Aurora PostgreSQL 12	Aurora PostgreSQL 11
Sud America (San Paolo)	Tutte le versioni	Tutte le versioni	Tutte le versioni	Tutte le versioni	Tutte le versioni	Tutte le versioni	Tutte le versioni
AWS GovCloud (Stati Uniti orientali)	Tutte le versioni	Tutte le versioni	Tutte le versioni	Tutte le versioni	Tutte le versioni	Tutte le versioni	Tutte le versioni
AWS GovCloud (Stati Uniti occidentali)	Tutte le versioni	Tutte le versioni	Tutte le versioni	Tutte le versioni	Tutte le versioni	Tutte le versioni	Tutte le versioni

Regioni e motori di database supportati per il machine learning Aurora

Utilizzando l'apprendimento automatico di Amazon Aurora, puoi integrare il tuo cluster Aurora DB con uno dei seguenti servizi di AWS machine learning, a seconda delle tue esigenze. Ciascuno supporta casi d’uso di machine learning specifici.

Amazon Bedrock è un servizio completamente gestito che rende disponibili i principali modelli di fondazione delle aziende di intelligenza artificiale tramite un’API, insieme a strumenti per sviluppatori per creare e scalare applicazioni di IA generativa.

Amazon Comprehend è un servizio di elaborazione del linguaggio naturale che viene utilizzato per estrarre informazioni dai documenti. L'uso di machine learning di Aurora con Amazon Comprehend consente di determinare il sentiment del testo nelle tabelle del database.

SageMaker L'intelligenza artificiale è un servizio di machine learning completo. I data scientist utilizzano Amazon SageMaker AI per creare, addestrare e testare modelli di machine learning

per una varietà di attività di inferenza, come il rilevamento delle frodi. Utilizzando l'apprendimento automatico di Aurora con SageMaker AI, gli sviluppatori di database possono richiamare la funzionalità SageMaker AI nel codice SQL.

Non tutti Regioni AWS supportano tutti i servizi di machine learning. Solo alcune Regioni AWS supportano il machine learning Aurora e quindi forniscono l'accesso a questi servizi da un cluster di database Aurora. Anche il processo di integrazione per machine learning di Aurora differisce dal motore di database. Per ulteriori informazioni, consulta Utilizzo di machine learning di Amazon Aurora.

Argomenti

- Utilizzo di machine learning di Aurora con Aurora MySQL
- Utilizzo della funzionalità Machine Learning di Aurora con Aurora PostgreSQL

Utilizzo di machine learning di Aurora con Aurora MySQL

Amazon Bedrock è supportato solo su Aurora MySQL 3.06 e versioni successive. Per informazioni sulla disponibilità delle Regioni per Amazon Bedrock, consulta Model support by Regione AWS nella Amazon Bedrock User Guide.

L'apprendimento automatico Aurora con Amazon Comprehend e SageMaker Amazon AI è supportato per Aurora MySQL nell'elenco riportato nella tabella. Regioni AWS Oltre a disporre della versione di Aurora MySQL in uso, è Regione AWS necessario supportare anche il servizio che si desidera utilizzare. Per un elenco delle aree Regioni AWS in cui è disponibile Amazon SageMaker AI, consulta gli endpoint e le quote di Amazon SageMaker AI nel. Riferimenti generali di Amazon Web Services Per un elenco delle aree Regioni AWS in cui è disponibile Amazon Comprehend, consulta gli endpoint e le quote di Amazon Comprehend nel. Riferimenti generali di Amazon Web Services

Region	Aurora MySQL versione 3	Aurora MySQL versione 2
Stati Uniti orientali (Virginia settentrionale)	Versione 3.01.0 e successive	Versione 2.07 e versioni successive
Stati Uniti orientali (Ohio)	Versione 3.01.0 e successive	Versione 2.07 e versioni successive
Stati Uniti occidentali (California settentrionale)	Versione 3.01.0 e successive	Versione 2.07 e versioni successive

Region	Aurora MySQL versione 3	Aurora MySQL versione 2
Stati Uniti occidentali (Oregon)	Versione 3.01.0 e successive	Versione 2.07 e versioni successive
Africa (Città del Capo)	Non disponibile	Non disponibile
Asia Pacifico (Hong Kong)	Versione 3.01.0 e successive	Versione 2.07 e versioni successive
Asia Pacific (Hyderabad)	Versione 3.01.0 e successive	Versione 2.07 e versioni successive
Asia Pacifico (Giacarta)	Versione 3.01.0 e successive	Versione 2.07 e versioni successive
Asia Pacifico (Malesia)	Versione 3.04.0 e successive	Non disponibile
Asia Pacifico (Melbourne)	Versione 3.01.0 e successive	Versione 2.07 e versioni successive
Asia Pacifico (Mumbai)	Versione 3.01.0 e successive	Versione 2.07 e versioni successive
Asia Pacifico (Nuova Zelanda)	Non disponibile	Non disponibile
Asia Pacifico (Osaka-Locale)	Versione 3.01.0 e successive	Versione 2.07.3 e versioni successive
Asia Pacifico (Seul)	Versione 3.01.0 e successive	Versione 2.07 e versioni successive
Asia Pacifico (Singapore)	Versione 3.01.0 e successive	Versione 2.07 e versioni successive
Asia Pacifico (Sydney)	Versione 3.01.0 e successive	Versione 2.07 e versioni successive
Asia Pacifico (Taipei)	Non disponibile	Non disponibile

Region	Aurora MySQL versione 3	Aurora MySQL versione 2
Asia Pacifico (Thailandia)	Non disponibile	Non disponibile
Asia Pacifico (Tokyo)	Versione 3.01.0 e successive	Versione 2.07 e versioni successive
Canada (Centrale)	Versione 3.01.0 e successive	Versione 2.07 e versioni successive
Canada occidentale (Calgary)	Versione 3.01.0 e successive	Versione 2.07 e versioni successive
Cina (Pechino)	Versione 3.01.0 e successive	Versione 2.07 e versioni successive
Cina (Ningxia)	Versione 3.01.0 e successive	Versione 2.07 e versioni successive
Europa (Francoforte)	Versione 3.01.0 e successive	Versione 2.07 e versioni successive
Europa (Irlanda)	Versione 3.01.0 e successive	Versione 2.07 e versioni successive
Europa (Londra)	Versione 3.01.0 e successive	Versione 2.07 e versioni successive
Europa (Milano)	Non disponibile	Non disponibile
Europa (Parigi)	Versione 3.01.0 e successive	Versione 2.07 e versioni successive
Europa (Spagna)	Versione 3.01.0 e successive	Versione 2.07 e versioni successive
Europa (Stoccolma)	Versione 3.01.0 e successive	Versione 2.07 e versioni successive

Region	Aurora MySQL versione 3	Aurora MySQL versione 2
Europa (Zurigo)	Versione 3.01.0 e successive	Versione 2.07 e versioni successive
Israele (Tel Aviv)	Versione 3.01.0 e successive	Versione 2.07 e versioni successive
Messico (centrale)	Non disponibile	Non disponibile
Medio Oriente (Bahrein)	Versione 3.01.0 e successive	Versione 2.07 e versioni successive
Medio Oriente (Emirati Arabi Uniti)	Versione 3.01.0 e successive	Versione 2.07 e versioni successive
Sud America (San Paolo)	Versione 3.01.0 e successive	Versione 2.07 e versioni successive
AWS GovCloud (Stati Uniti orientali)	Versione 3.01.0 e successive	Versione 2.07 e versioni successive
AWS GovCloud (Stati Uniti occidentali)	Versione 3.01.0 e successive	Versione 2.07 e versioni successive

Utilizzo della funzionalità Machine Learning di Aurora con Aurora PostgreSQL

Per informazioni sul supporto delle versioni per Amazon Bedrock su Aurora PostgreSQL, consulta Utilizzo di Aurora PostgreSQL come Knowledge Base per Amazon Bedrock.

L'apprendimento automatico Aurora con Amazon Comprehend e SageMaker Amazon AI è supportato per Aurora PostgreSQL nell'elenco riportato nella tabella. Regioni AWS Oltre a disporre della versione di Aurora PostgreSQL in uso, è Regione AWS necessario supportare anche il servizio che si desidera utilizzare. Per un elenco delle aree Regioni AWS in cui è disponibile Amazon SageMaker AI, consulta gli endpoint e le quote di Amazon SageMaker AI nel. Riferimenti generali di Amazon Web Services Per un elenco delle aree Regioni AWS in cui è disponibile Amazon Comprehend, consulta gli endpoint e le quote di Amazon Comprehend nel. Riferimenti generali di Amazon Web Services

Di seguito sono riportate le versioni di motore e le regioni disponibili per la funzionalità machine learning di Aurora con Aurora PostgreSQL.

Region	Aurora PostgreSQL 17	Aurora PostgreSQL 16	Aurora PostgreSQL 15	Aurora PostgreSQL 14	Aurora PostgreSQL 13	Aurora PostgreSQL 12	Aurora PostgreSQL 11
Stati Uniti orientali (Virginia settentrionale)	Versione 17.4 e successive	Versione 16.1 e successive	Versione 15.2 e successive	Versione 14.3 e versioni successive	Versione 13.3 e versioni successive	Versione 12.4 e versioni successive	Versione 11.9, 11.12 e versioni successive
Stati Uniti orientali (Ohio)	Versione 17.4 e successive	Versione 16.1 e successive	Versione 15.2 e successive	Versione 14.3 e versioni successive	Versione 13.3 e versioni successive	Versione 12.4 e versioni successive	Versione 11.9, 11.12 e versioni successive
Stati Uniti occidentali (California settentrionale)	Versione 17.4 e successive	Versione 16.1 e successive	Versione 15.2 e successive	Versione 14.3 e versioni successive	Versione 13.3 e versioni successive	Versione 12.4 e versioni successive	Versione 11.9, 11.12 e versioni successive
Stati Uniti occidentali (Oregon)	Versione 17.4 e successive	Versione 16.1 e successive	Versione 15.2 e successive	Versione 14.3 e versioni successive	Versione 13.3 e versioni successive	Versione 12.4 e versioni successive	Versione 11.9, 11.12 e versioni successive

Region	Aurora PostgreSQL 17	Aurora PostgreSQL 16	Aurora PostgreSQL 15	Aurora PostgreSQL 14	Aurora PostgreSQL 13	Aurora PostgreSQL 12	Aurora PostgreSQL 11
Africa (Città del Capo)	Non disponibile	Non disponibile	Non disponibile	Non disponibile	Non disponibile	Non disponibile	Non disponibile
Asia Pacifico (Hong Kong)	Versione 17.4 e successive	Versione 16.1 e successive	Versione 15.2 e successive	Versione 14.3 e versioni successive	Versione 13.3 e versioni successive	Versione 12.4 e versioni successive	Versione 11.9, 11.12 e versioni successive
Asia Pacific (Hyderabad)	Versione 17.4 e successive	Versione 16.1 e successive	Versione 15.2 e successive	Versione 14.3 e versioni successive	Versione 13.3 e versioni successive	Versione 12.4 e versioni successive	Versione 11.9, 11.12 e versioni successive
Asia Pacifico (Giacarta)	Versione 17.4 e successive	Versione 16.1 e successive	Versione 15.2 e successive	Versione 14.3 e versioni successive	Versione 13.3 e versioni successive	Versione 12.4 e versioni successive	Versione 11.9, 11.12 e versioni successive
Asia Pacifico (Malesia)	Versione 17.4 e successive	Versione 16.1 e successive	Versione 15.2 e successive	Versione 14.3 e versioni successive	Versione 13.3 e versioni successive	Versione 12.4 e versioni successive	Versione 11.9, 11.12 e versioni successive

Region	Aurora PostgreSQL 17	Aurora PostgreSQL 16	Aurora PostgreSQL 15	Aurora PostgreSQL 14	Aurora PostgreSQL 13	Aurora PostgreSQL 12	Aurora PostgreSQL 11
Asia Pacifico (Melbourne)	Versione 17.4 e successive	Versione 16.1 e successive	Versione 15.2 e successive	Versione 14.3 e versioni successive	Versione 13.3 e versioni successive	Versione 12.4 e versioni successive	Versione 11.9, 11.12 e versioni successive
Asia Pacifico (Mumbai)	Versione 17.4 e successive	Versione 16.1 e successive	Versione 15.2 e successive	Versione 14.3 e versioni successive	Versione 13.3 e versioni successive	Versione 12.4 e versioni successive	Versione 11.9, 11.12 e versioni successive
Asia Pacifico (Nuova Zelanda)	Versione 17.4 e successive	Versione 16.1 e successive	Versione 15.2 e successive	Versione 14.3 e versioni successive	Versione 13.3 e versioni successive	Versione 12.4 e versioni successive	Versione 11.9, 11.12 e versioni successive
Asia Pacifico (Osaka-Locale)	Versione 17.4 e successive	Versione 16.1 e successive	Versione 15.2 e successive	Versione 14.3 e versioni successive	Versione 13.3 e versioni successive	Versione 12.4 e versioni successive	Versione 11.9, 11.12 e versioni successive

Region	Aurora PostgreSQL 17	Aurora PostgreSQL 16	Aurora PostgreSQL 15	Aurora PostgreSQL 14	Aurora PostgreSQL 13	Aurora PostgreSQL 12	Aurora PostgreSQL 11
Asia Pacifico (Seul)	Versione 17.4 e successive	Versione 16.1 e successive	Versione 15.2 e successive	Versione 14.3 e versioni successive	Versione 13.3 e versioni successive	Versione 12.4 e versioni successive	Versione 11.9, 11.12 e versioni successive
Asia Pacifico (Singapore)	Versione 17.4 e successive	Versione 16.1 e successive	Versione 15.2 e successive	Versione 14.3 e versioni successive	Versione 13.3 e versioni successive	Versione 12.4 e versioni successive	Versione 11.9, 11.12 e versioni successive
Asia Pacifico (Sydney)	Versione 17.4 e successive	Versione 16.1 e successive	Versione 15.2 e successive	Versione 14.3 e versioni successive	Versione 13.3 e versioni successive	Versione 12.4 e versioni successive	Versione 11.9, 11.12 e versioni successive
Asia Pacifico (Taipei)	Versione 17.4 e successive	Versione 16.1 e successive	Versione 15.2 e successive	Versione 14.3 e versioni successive	Versione 13.3 e versioni successive	Versione 12.4 e versioni successive	Versione 11.9, 11.12 e versioni successive

Region	Aurora PostgreSQL 17	Aurora PostgreSQL 16	Aurora PostgreSQL 15	Aurora PostgreSQL 14	Aurora PostgreSQL 13	Aurora PostgreSQL 12	Aurora PostgreSQL 11
Asia Pacifico (Thailandia)	Versione 17.4 e successive	Versione 16.1 e successive	Versione 15.2 e successive	Versione 14.3 e versioni successive	Versione 13.3 e versioni successive	Versione 12.4 e versioni successive	Versione 11.9, 11.12 e versioni successive
Asia Pacifico (Tokyo)	Versione 17.4 e successive	Versione 16.1 e successive	Versione 15.2 e successive	Versione 14.3 e versioni successive	Versione 13.3 e versioni successive	Versione 12.4 e versioni successive	Versione 11.9, 11.12 e versioni successive
Canada (Centrale)	Versione 17.4 e successive	Versione 16.1 e successive	Versione 15.2 e successive	Versione 14.3 e versioni successive	Versione 13.3 e versioni successive	Versione 12.4 e versioni successive	Versione 11.9, 11.12 e versioni successive
Canada occidentale (Calgary)	Versione 17.4 e successive	Versione 16.1 e successive	Versione 15.2 e successive	Versione 14.3 e versioni successive	Versione 13.3 e versioni successive	Versione 12.4 e versioni successive	Versione 11.9, 11.12 e versioni successive

Region	Aurora PostgreSQL 17	Aurora PostgreSQL 16	Aurora PostgreSQL 15	Aurora PostgreSQL 14	Aurora PostgreSQL 13	Aurora PostgreSQL 12	Aurora PostgreSQL 11
Cina (Pechino)	Versione 17.4 e successive	Versione 16.1 e successive	Versione 15.2 e successive	Versione 14.3 e versioni successive	Versione 13.3 e versioni successive	Versione 12.4 e versioni successive	Versione 11.9, 11.12 e versioni successive
Cina (Ningxia)	Versione 17.4 e successive	Versione 16.1 e successive	Versione 15.2 e successive	Versione 14.3 e versioni successive	Versione 13.3 e versioni successive	Versione 12.4 e versioni successive	Versione 11.9, 11.12 e versioni successive
Europa (Francoforte)	Versione 17.4 e successive	Versione 16.1 e successive	Versione 15.2 e successive	Versione 14.3 e versioni successive	Versione 13.3 e versioni successive	Versione 12.4 e versioni successive	Versione 11.9, 11.12 e versioni successive
Europa (Irlanda)	Versione 17.4 e successive	Versione 16.1 e successive	Versione 15.2 e successive	Versione 14.3 e versioni successive	Versione 13.3 e versioni successive	Versione 12.4 e versioni successive	Versione 11.9, 11.12 e versioni successive

Region	Aurora PostgreSQL 17	Aurora PostgreSQL 16	Aurora PostgreSQL 15	Aurora PostgreSQL 14	Aurora PostgreSQL 13	Aurora PostgreSQL 12	Aurora PostgreSQL 11
Europa (Londra)	Versione 17.4 e successive	Versione 16.1 e successive	Versione 15.2 e successive	Versione 14.3 e versioni successive	Versione 13.3 e versioni successive	Versione 12.4 e versioni successive	Versione 11.9, 11.12 e versioni successive
Europa (Milano)	Non disponibile	Non disponibile	Non disponibile	Non disponibile	Non disponibile	Non disponibile	Non disponibile
Europa (Parigi)	Versione 17.4 e successive	Versione 16.1 e successive	Versione 15.2 e successive	Versione 14.3 e versioni successive	Versione 13.3 e versioni successive	Versione 12.4 e versioni successive	Versione 11.9, 11.12 e versioni successive
Europa (Spagna)	Versione 17.4 e successive	Versione 16.1 e successive	Versione 15.2 e successive	Versione 14.3 e versioni successive	Versione 13.3 e versioni successive	Versione 12.4 e versioni successive	Versione 11.9, 11.12 e versioni successive
Europa (Stoccolma)	Versione 17.4 e successive	Versione 16.1 e successive	Versione 15.2 e successive	Versione 14.3 e versioni successive	Versione 13.3 e versioni successive	Versione 12.4 e versioni successive	Versione 11.9, 11.12 e versioni successive

Region	Aurora PostgreSQL 17	Aurora PostgreSQL 16	Aurora PostgreSQL 15	Aurora PostgreSQL 14	Aurora PostgreSQL 13	Aurora PostgreSQL 12	Aurora PostgreSQL 11
Europa (Zurigo)	Versione 17.4 e successive	Versione 16.1 e successive	Versione 15.2 e successive	Versione 14.3 e versioni successive	Versione 13.3 e versioni successive	Versione 12.4 e versioni successive	Versione 11.9, 11.12 e versioni successive
Israele (Tel Aviv)	Versione 17.4 e successive	Versione 16.1 e successive	Versione 15.2 e successive	Versione 14.3 e versioni successive	Versione 13.3 e versioni successive	Versione 12.4 e versioni successive	Versione 11.9, 11.12 e versioni successive
Messico (centrale)	Non disponibile	Non disponibile	Non disponibile	Non disponibile	Non disponibile	Non disponibile	Non disponibile
Medio Oriente (Bahrein)	Versione 17.4 e successive	Versione 16.1 e successive	Versione 15.2 e successive	Versione 14.3 e versioni successive	Versione 13.3 e versioni successive	Versione 12.4 e versioni successive	Versione 11.9, 11.12 e versioni successive
Medio Oriente (Emirati Arabi Uniti)	Versione 17.4 e successive	Versione 16.1 e successive	Versione 15.2 e successive	Versione 14.3 e versioni successive	Versione 13.3 e versioni successive	Versione 12.4 e versioni successive	Versione 11.9, 11.12 e versioni successive

Region	Aurora PostgreSQL 17	Aurora PostgreSQL 16	Aurora PostgreSQL 15	Aurora PostgreSQL 14	Aurora PostgreSQL 13	Aurora PostgreSQL 12	Aurora PostgreSQL 11
Sud America (San Paolo)	Versione 17.4 e successive	Versione 16.1 e successive	Versione 15.2 e successive	Versione 14.3 e versioni successive	Versione 13.3 e versioni successive	Versione 12.4 e versioni successive	Versione 11.9, 11.12 e versioni successive
AWS GovCloud (Stati Uniti orientali)	Versione 17.4 e successive	Versione 16.1 e successive	Versione 15.2 e successive	Versione 14.3 e versioni successive	Versione 13.3 e versioni successive	Versione 12.4 e versioni successive	Versione 11.9, 11.12 e versioni successive
AWS GovCloud (Stati Uniti occidentali)	Versione 17.4 e successive	Versione 16.1 e successive	Versione 15.2 e successive	Versione 14.3 e versioni successive	Versione 13.3 e versioni successive	Versione 12.4 e versioni successive	Versione 11.9, 11.12 e versioni successive

Regioni e motori di database Aurora supportati per Approfondimenti sulle prestazioni

Important

AWS ha annunciato la end-of-life data di Performance Insights: 30 giugno 2026. Dopo questa data, Amazon RDS non supporterà più l'esperienza della console Approfondimenti sulle prestazioni, i periodi di conservazione flessibili (1-24 mesi) e i relativi prezzi. L'API Approfondimenti sulle prestazioni continuerà a esistere senza modifiche ai prezzi. I costi per l'API Performance Insights verranno visualizzati nella AWS fattura insieme al costo di CloudWatch Database Insights.

Si consiglia di aggiornare tutte DB Clusters di database utilizzando il livello a pagamento di Performance Insights alla modalità Advanced di Database Insights prima del 30 giugno 2026. Per informazioni sull'aggiornamento alla modalità avanzata di Database Insights, consulta Attivazione della modalità avanzata di Database Insights per Amazon Aurora.
Se non si esegue alcuna azione, i cluster di database che utilizzano Approfondimenti sulle prestazioni, per impostazione predefinita useranno la modalità standard di Database Insights. Con la modalità standard di Database Insights, è possibile perdere l'accesso alla cronologia dei dati sulle prestazioni di oltre 7 giorni e non essere in grado di utilizzare i piani di esecuzione e le funzionalità di analisi on demand nella console Amazon RDS. Dopo il 30 giugno 2026 solo la modalità Advanced di Database Insights supporterà i piani di esecuzione e l'analisi su richiesta.
Con CloudWatch Database Insights, puoi monitorare il carico del database per la tua flotta di database e analizzare e risolvere i problemi delle prestazioni su larga scala. Per ulteriori informazioni su Database Insights, consulta Monitoraggio dei database Amazon Aurora con CloudWatch Database Insights. Per informazioni sui prezzi, consulta la pagina CloudWatch dei prezzi di Amazon.

Performance Insights espande le sue potenzialità grazie alle funzionalità di monitoraggio esistenti di Amazon RDS a supporto delle operazioni di analisi delle prestazioni del database. Il pannello di controllo di Performance Insights consente di visualizzare il carico del database sull'istanza database Amazon RDS e filtrare il carico in base alle attese, alle istruzioni, agli host o agli utenti. Per ulteriori informazioni, consulta Panoramica di Performance Insights su Amazon Aurora.

Per informazioni sull'assistenza alla regione, al motore di database e alla classe di istanza per le funzionalità Performance Insights, consulta Supporto di classe di istanza, regione e motore di database Amazon Aurora per funzionalità Performance Insights.

Argomenti

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

Performance Insights con Aurora MySQL

> **Note**
>
> Il supporto della versione del motore è diverso per Performance Insights con Aurora MySQL se la query parallela è attivata. Per ulteriori informazioni sulla query parallela, consultare Query parallela per Amazon Aurora MySQL.

Argomenti

- Performance Insights con Aurora MySQL e query parallela disattivata
- Performance Insights con Aurora MySQL e query parallela attivata

Performance Insights con Aurora MySQL e query parallela disattivata

Di seguito sono riportate le versioni di motore e le regioni disponibili per Approfondimenti sulle prestazioni con Aurora MySQL e la query parallela disattivata.

Region	Aurora MySQL versione 3	Aurora MySQL versione 2
Stati Uniti orientali (Virginia settentrionale)	Tutte le versioni	Tutte le versioni
Stati Uniti orientali (Ohio)	Tutte le versioni	Tutte le versioni
Stati Uniti occidentali (California settentrionale)	Tutte le versioni	Tutte le versioni
Stati Uniti occidentali (Oregon)	Tutte le versioni	Tutte le versioni
Africa (Città del Capo)	Tutte le versioni	Tutte le versioni
Asia Pacifico (Hong Kong)	Tutte le versioni	Tutte le versioni
Asia Pacific (Hyderabad)	Tutte le versioni	Tutte le versioni
Asia Pacifico (Giacarta)	Tutte le versioni	Tutte le versioni
Asia Pacifico (Malesia)	Tutte le versioni	Tutte le versioni

Region	Aurora MySQL versione 3	Aurora MySQL versione 2
Asia Pacifico (Melbourne)	Tutte le versioni	Tutte le versioni
Asia Pacifico (Mumbai)	Tutte le versioni	Tutte le versioni
Asia Pacifico (Nuova Zelanda)	Tutte le versioni	Tutte le versioni
Asia Pacifico (Osaka-Locale)	Tutte le versioni	Tutte le versioni
Asia Pacifico (Seul)	Tutte le versioni	Tutte le versioni
Asia Pacifico (Singapore)	Tutte le versioni	Tutte le versioni
Asia Pacifico (Sydney)	Tutte le versioni	Tutte le versioni
Asia Pacifico (Taipei)	Tutte le versioni	Tutte le versioni
Asia Pacifico (Thailandia)	Tutte le versioni	Tutte le versioni
Asia Pacifico (Tokyo)	Tutte le versioni	Tutte le versioni
Canada (Centrale)	Tutte le versioni	Tutte le versioni
Canada occidentale (Calgary)	Tutte le versioni	Tutte le versioni
Cina (Pechino)	Tutte le versioni	Tutte le versioni
Cina (Ningxia)	Tutte le versioni	Tutte le versioni
Europa (Francoforte)	Tutte le versioni	Tutte le versioni
Europa (Irlanda)	Tutte le versioni	Tutte le versioni
Europa (Londra)	Tutte le versioni	Tutte le versioni
Europa (Milano)	Tutte le versioni	Tutte le versioni
Europa (Parigi)	Tutte le versioni	Tutte le versioni
Europa (Spagna)	Tutte le versioni	Tutte le versioni

Region	Aurora MySQL versione 3	Aurora MySQL versione 2
Europa (Stoccolma)	Tutte le versioni	Tutte le versioni
Europa (Zurigo)	Tutte le versioni	Tutte le versioni
Israele (Tel Aviv)	Tutte le versioni	Tutte le versioni
Messico (centrale)	Tutte le versioni	Tutte le versioni
Medio Oriente (Bahrein)	Tutte le versioni	Tutte le versioni
Medio Oriente (Emirati Arabi Uniti)	Tutte le versioni	Tutte le versioni
Sud America (San Paolo)	Tutte le versioni	Tutte le versioni
AWS GovCloud (Stati Uniti orientali)	Tutte le versioni	Tutte le versioni
AWS GovCloud (Stati Uniti occidentali)	Tutte le versioni	Tutte le versioni

Performance Insights con Aurora MySQL e query parallela attivata

Di seguito sono riportate le versioni di motore e le regioni disponibili per Approfondimenti sulle prestazioni con Aurora MySQL e la query parallela attivata.

Region	Aurora MySQL versione 3	Aurora MySQL versione 2
Stati Uniti orientali (Virginia settentrionale)	Non disponibile	Versione 2.09.0 e versioni successive
Stati Uniti orientali (Ohio)	Non disponibile	Versione 2.09.0 e versioni successive
Stati Uniti occidentali (California settentrionale)	Non disponibile	Versione 2.09.0 e versioni successive

Region	Aurora MySQL versione 3	Aurora MySQL versione 2
Stati Uniti occidentali (Oregon)	Non disponibile	Versione 2.09.0 e versioni successive
Africa (Città del Capo)	Non disponibile	Versione 2.09.0 e versioni successive
Asia Pacifico (Hong Kong)	Non disponibile	Versione 2.09.0 e versioni successive
Asia Pacific (Hyderabad)	Non disponibile	Tutte le versioni
Asia Pacifico (Giacarta)	Non disponibile	Versione 2.09.0 e versioni successive
Asia Pacifico (Malesia)	Non disponibile	Versione 2.09.0 e versioni successive
Asia Pacifico (Melbourne)	Non disponibile	Versione 2.09.0 e versioni successive
Asia Pacifico (Mumbai)	Non disponibile	Versione 2.09.0 e versioni successive
Asia Pacifico (Nuova Zelanda)	Non disponibile	Versione 2.09.0 e versioni successive
Asia Pacifico (Osaka-Locale)	Non disponibile	Versione 2.09.0 e versioni successive
Asia Pacifico (Seul)	Non disponibile	Versione 2.09.0 e versioni successive
Asia Pacifico (Singapore)	Non disponibile	Versione 2.09.0 e versioni successive
Asia Pacifico (Sydney)	Non disponibile	Versione 2.09.0 e versioni successive

Region	Aurora MySQL versione 3	Aurora MySQL versione 2
Asia Pacifico (Taipei)	Non disponibile	Versione 2.09.0 e versioni successive
Asia Pacifico (Thailandia)	Non disponibile	Versione 2.09.0 e versioni successive
Asia Pacifico (Tokyo)	Non disponibile	Versione 2.09.0 e versioni successive
Canada (Centrale)	Non disponibile	Versione 2.09.0 e versioni successive
Canada occidentale (Calgary)	Non disponibile	Versione 2.09.0 e versioni successive
Cina (Pechino)	Non disponibile	Versione 2.09.0 e versioni successive
Cina (Ningxia)	Non disponibile	Versione 2.09.0 e versioni successive
Europa (Francoforte)	Non disponibile	Versione 2.09.0 e versioni successive
Europa (Irlanda)	Non disponibile	Versione 2.09.0 e versioni successive
Europa (Londra)	Non disponibile	Versione 2.09.0 e versioni successive
Europa (Milano)	Non disponibile	Versione 2.09.0 e versioni successive
Europa (Parigi)	Non disponibile	Versione 2.09.0 e versioni successive
Europa (Spagna)	Non disponibile	Versione 2.09.0 e versioni successive

Region	Aurora MySQL versione 3	Aurora MySQL versione 2
Europa (Stoccolma)	Non disponibile	Versione 2.09.0 e versioni successive
Europa (Zurigo)	Non disponibile	Versione 2.09.0 e versioni successive
Israele (Tel Aviv)	Non disponibile	Versione 2.09.0 e versioni successive
Messico (centrale)	Non disponibile	Versione 2.09.0 e versioni successive
Medio Oriente (Bahrein)	Non disponibile	Versione 2.09.0 e versioni successive
Medio Oriente (Emirati Arabi Uniti)	Non disponibile	Versione 2.09.0 e versioni successive
Sud America (San Paolo)	Non disponibile	Versione 2.09.0 e versioni successive
AWS GovCloud (Stati Uniti orientali)	Non disponibile	Versione 2.09.0 e versioni successive
AWS GovCloud (Stati Uniti occidentali)	Non disponibile	Versione 2.09.0 e versioni successive

Performance Insights con Aurora PostgreSQL

Di seguito sono riportate le versioni di motore e le regioni disponibili per Approfondimenti sulle prestazioni con Aurora PostgreSQL.

Region	Aurora PostgreSQL 17	Aurora PostgreSQL 16	Aurora PostgreSQL 15	Aurora PostgreSQL 14	Aurora PostgreSQL 13	Aurora PostgreSQL 12	Aurora PostgreSQL 11	Aurora PostgreSQL 10
Stati Uniti orientali (Virginia settentrionale)	Tutte le versioni	Tutte le versioni	Tutte le versioni	Tutte le versioni	Tutte le versioni	Tutte le versioni	Tutte le versioni	Tutte le versioni
Stati Uniti orientali (Ohio)	Tutte le versioni	Tutte le versioni	Tutte le versioni	Tutte le versioni	Tutte le versioni	Tutte le versioni	Tutte le versioni	Tutte le versioni
Stati Uniti occidentali (California settentrionale)	Tutte le versioni	Tutte le versioni	Tutte le versioni	Tutte le versioni	Tutte le versioni	Tutte le versioni	Tutte le versioni	Tutte le versioni
Stati Uniti occidentali (Oregon)	Tutte le versioni	Tutte le versioni	Tutte le versioni	Tutte le versioni	Tutte le versioni	Tutte le versioni	Tutte le versioni	Tutte le versioni
Africa (Città del Capo)	Tutte le versioni	Tutte le versioni	Tutte le versioni	Tutte le versioni	Tutte le versioni	Tutte le versioni	Tutte le versioni	Tutte le versioni
Asia Pacifico	Tutte le versioni	Tutte le versioni	Tutte le versioni	Tutte le versioni	Tutte le versioni	Tutte le versioni	Tutte le versioni	Tutte le versioni

Region	Aurora PostgreSQL 17	Aurora PostgreSQL 16	Aurora PostgreSQL 15	Aurora PostgreSQL 14	Aurora PostgreSQL 13	Aurora PostgreSQL 12	Aurora PostgreSQL 11	Aurora PostgreSQL 10
(Hong Kong)								
Asia Pacific (Hyderabad)	Tutte le versioni	Tutte le versioni	Tutte le versioni	Tutte le versioni	Tutte le versioni	Tutte le versioni	Tutte le versioni	Tutte le versioni
Asia Pacifico (Giacarta)	Tutte le versioni	Tutte le versioni	Tutte le versioni	Tutte le versioni	Tutte le versioni	Tutte le versioni	Tutte le versioni	Tutte le versioni
Asia Pacifico (Malesia)	Tutte le versioni	Tutte le versioni	Tutte le versioni	Tutte le versioni	Tutte le versioni	Tutte le versioni	Tutte le versioni	Tutte le versioni
Asia Pacifico (Melbourne)	Tutte le versioni	Tutte le versioni	Tutte le versioni	Tutte le versioni	Tutte le versioni	Tutte le versioni	Tutte le versioni	Tutte le versioni
Asia Pacifico (Mumbai)	Tutte le versioni	Tutte le versioni	Tutte le versioni	Tutte le versioni	Tutte le versioni	Tutte le versioni	Tutte le versioni	Tutte le versioni
Asia Pacifico (Nuova Zelanda)	Tutte le versioni	Tutte le versioni	Tutte le versioni	Tutte le versioni	Tutte le versioni	Tutte le versioni	Tutte le versioni	Tutte le versioni
Asia Pacifico (Osaka-Locale)	Tutte le versioni	Tutte le versioni	Tutte le versioni	Tutte le versioni	Tutte le versioni	Tutte le versioni	Tutte le versioni	Tutte le versioni

Region	Aurora PostgreSQL 17	Aurora PostgreSQL 16	Aurora PostgreSQL 15	Aurora PostgreSQL 14	Aurora PostgreSQL 13	Aurora PostgreSQL 12	Aurora PostgreSQL 11	Aurora PostgreSQL 10
Asia Pacifico (Seul)	Tutte le versioni	Tutte le versioni	Tutte le versioni	Tutte le versioni	Tutte le versioni	Tutte le versioni	Tutte le versioni	Tutte le versioni
Asia Pacifico (Singapore)	Tutte le versioni	Tutte le versioni	Tutte le versioni	Tutte le versioni	Tutte le versioni	Tutte le versioni	Tutte le versioni	Tutte le versioni
Asia Pacifico (Sydney)	Tutte le versioni	Tutte le versioni	Tutte le versioni	Tutte le versioni	Tutte le versioni	Tutte le versioni	Tutte le versioni	Tutte le versioni
Asia Pacifico (Taipei)	Tutte le versioni	Tutte le versioni	Tutte le versioni	Tutte le versioni	Tutte le versioni	Tutte le versioni	Tutte le versioni	Tutte le versioni
Asia Pacifico (Thailandia)	Tutte le versioni	Tutte le versioni	Tutte le versioni	Tutte le versioni	Tutte le versioni	Tutte le versioni	Tutte le versioni	Tutte le versioni
Asia Pacifico (Tokyo)	Tutte le versioni	Tutte le versioni	Tutte le versioni	Tutte le versioni	Tutte le versioni	Tutte le versioni	Tutte le versioni	Tutte le versioni
Canada (Centrale)	Tutte le versioni	Tutte le versioni	Tutte le versioni	Tutte le versioni	Tutte le versioni	Tutte le versioni	Tutte le versioni	Tutte le versioni
Canada occidentale (Calgary)	Tutte le versioni	Tutte le versioni	Tutte le versioni	Tutte le versioni	Tutte le versioni	Tutte le versioni	Tutte le versioni	Tutte le versioni

Region	Aurora PostgreSQL 17	Aurora PostgreSQL 16	Aurora PostgreSQL 15	Aurora PostgreSQL 14	Aurora PostgreSQL 13	Aurora PostgreSQL 12	Aurora PostgreSQL 11	Aurora PostgreSQL 10
Cina (Pechino)	Tutte le versioni	Tutte le versioni	Tutte le versioni	Tutte le versioni	Tutte le versioni	Tutte le versioni	Tutte le versioni	Tutte le versioni
Cina (Ningxia)	Tutte le versioni	Tutte le versioni	Tutte le versioni	Tutte le versioni	Tutte le versioni	Tutte le versioni	Tutte le versioni	Tutte le versioni
Europa (Francoforte)	Tutte le versioni	Tutte le versioni	Tutte le versioni	Tutte le versioni	Tutte le versioni	Tutte le versioni	Tutte le versioni	Tutte le versioni
Europa (Irlanda)	Tutte le versioni	Tutte le versioni	Tutte le versioni	Tutte le versioni	Tutte le versioni	Tutte le versioni	Tutte le versioni	Tutte le versioni
Europa (Londra)	Tutte le versioni	Tutte le versioni	Tutte le versioni	Tutte le versioni	Tutte le versioni	Tutte le versioni	Tutte le versioni	Tutte le versioni
Europa (Milano)	Tutte le versioni	Tutte le versioni	Tutte le versioni	Tutte le versioni	Tutte le versioni	Tutte le versioni	Tutte le versioni	Tutte le versioni
Europa (Parigi)	Tutte le versioni	Tutte le versioni	Tutte le versioni	Tutte le versioni	Tutte le versioni	Tutte le versioni	Tutte le versioni	Tutte le versioni
Europa (Spagna)	Tutte le versioni	Tutte le versioni	Tutte le versioni	Tutte le versioni	Tutte le versioni	Tutte le versioni	Tutte le versioni	Tutte le versioni
Europa (Stoccolma)	Tutte le versioni	Tutte le versioni	Tutte le versioni	Tutte le versioni	Tutte le versioni	Tutte le versioni	Tutte le versioni	Tutte le versioni
Europa (Zurigo)	Tutte le versioni	Tutte le versioni	Tutte le versioni	Tutte le versioni	Tutte le versioni	Tutte le versioni	Tutte le versioni	Tutte le versioni
Israele (Tel Aviv)	Tutte le versioni	Tutte le versioni	Tutte le versioni	Tutte le versioni	Tutte le versioni	Tutte le versioni	Tutte le versioni	Tutte le versioni

Region	Aurora PostgreSQL 17	Aurora PostgreSQL 16	Aurora PostgreSQL 15	Aurora PostgreSQL 14	Aurora PostgreSQL 13	Aurora PostgreSQL 12	Aurora PostgreSQL 11	Aurora PostgreSQL 10
Messico (centrale)	Tutte le versioni	Tutte le versioni	Tutte le versioni	Tutte le versioni	Tutte le versioni	Tutte le versioni	Tutte le versioni	Tutte le versioni
Medio Oriente (Bahrein)	Tutte le versioni	Tutte le versioni	Tutte le versioni	Tutte le versioni	Tutte le versioni	Tutte le versioni	Tutte le versioni	Tutte le versioni
Medio Oriente (Emirati Arabi Uniti)	Tutte le versioni	Tutte le versioni	Tutte le versioni	Tutte le versioni	Tutte le versioni	Tutte le versioni	Tutte le versioni	Tutte le versioni
Sud America (San Paolo)	Tutte le versioni	Tutte le versioni	Tutte le versioni	Tutte le versioni	Tutte le versioni	Tutte le versioni	Tutte le versioni	Tutte le versioni
AWS GovCloud (Stati Uniti orientali)	Tutte le versioni	Tutte le versioni	Tutte le versioni	Tutte le versioni	Tutte le versioni	Tutte le versioni	Tutte le versioni	Tutte le versioni
AWS GovCloud (Stati Uniti occidentali)	Tutte le versioni	Tutte le versioni	Tutte le versioni	Tutte le versioni	Tutte le versioni	Tutte le versioni	Tutte le versioni	Tutte le versioni

Performance Insights con Aurora Serverless

Aurora Serverless v2 supporta Performance Insights per tutte le versioni compatibili con MySQL e compatibili con PostgreSQL. Si consiglia di impostare la capacità minima su almeno 2 unità di capacità Aurora ()ACUs.

Aurora Serverless v1 non supporta Performance Insights.

Regioni e motori di database Aurora supportati per le integrazioni Zero-ETL

Le integrazioni Zero-ETL di Amazon Aurora sono una soluzione completamente gestita per rendere disponibili i dati transazionali in Amazon Redshift o Amazon SageMaker dopo la scrittura su un cluster Aurora. Per ulteriori informazioni, consulta Integrazioni Zero-ETL.

Argomenti

- Integrazioni Zero-ETL di Aurora MySQL
- Integrazioni Zero-ETL di Aurora PostgreSQL

Integrazioni Zero-ETL di Aurora MySQL

Le seguenti Regioni e versioni del motore sono disponibili per le integrazioni Zero-ETL di Aurora MySQL con Amazon Redshift e Amazon SageMaker.

Region	Integrazione Zero-ETL con Amazon Redshift per Aurora MySQL versione 3	Integrazione Zero-ETL con Amazon SageMaker per Aurora MySQL versione 3
Stati Uniti orientali (Virginia settentrionale)	Versione 3.05.2 e successive	Versione 3.05.2 e successive
Stati Uniti orientali (Ohio)	Versione 3.05.2 e successive	Versione 3.05.2 e successive
Stati Uniti occidentali (California settentrionale)	Versione 3.05.2 e successive	Non disponibile
Stati Uniti occidentali (Oregon)	Versione 3.05.2 e successive	Versione 3.05.2 e successive
Africa (Città del Capo)	Versione 3.05.2 e successive	Non disponibile

Region	Integrazione Zero-ETL con Amazon Redshift per Aurora MySQL versione 3	Integrazione Zero-ETL con Amazon SageMaker per Aurora MySQL versione 3
Asia Pacifico (Hong Kong)	Versione 3.05.2 e successive	Versione 3.05.2 e successive
Asia Pacific (Hyderabad)	Versione 3.05.2 e successive	Non disponibile
Asia Pacifico (Giacarta)	Versione 3.05.2 e successive	Non disponibile
Asia Pacifico (Malesia)	Non disponibile	Non disponibile
Asia Pacifico (Melbourne)	Versione 3.05.2 e successive	Non disponibile
Asia Pacifico (Mumbai)	Versione 3.05.2 e successive	Non disponibile
Asia Pacifico (Nuova Zelanda)	Non disponibile	Non disponibile
Asia Pacifico (Osaka-Locale)	Versione 3.05.2 e successive	Non disponibile
Asia Pacifico (Seul)	Versione 3.05.2 e successive	Versione 3.05.2 e successive
Asia Pacifico (Singapore)	Versione 3.05.2 e successive	Versione 3.05.2 e successive
Asia Pacifico (Sydney)	Versione 3.05.2 e successive	Versione 3.05.2 e successive
Asia Pacifico (Taipei)	Non disponibile	Non disponibile
Asia Pacifico (Thailandia)	Non disponibile	Non disponibile
Asia Pacifico (Tokyo)	Versione 3.05.2 e successive	Versione 3.05.2 e successive
Canada (Centrale)	Versione 3.05.2 e successive	Versione 3.05.2 e successive
Canada occidentale (Calgary)	Versione 3.05.2 e successive	Non disponibile
Cina (Pechino)	Versione 3.05.2 e successive	Non disponibile
Cina (Ningxia)	Versione 3.05.2 e successive	Non disponibile
Europa (Francoforte)	Versione 3.05.2 e successive	Versione 3.05.2 e successive

Region	Integrazione Zero-ETL con Amazon Redshift per Aurora MySQL versione 3	Integrazione Zero-ETL con Amazon SageMaker per Aurora MySQL versione 3
Europa (Irlanda)	Versione 3.05.2 e successive	Versione 3.05.2 e successive
Europa (Londra)	Versione 3.05.2 e successive	Versione 3.05.2 e successive
Europa (Milano)	Versione 3.05.2 e successive	Non disponibile
Europa (Parigi)	Versione 3.05.2 e successive	Non disponibile
Europa (Spagna)	Versione 3.05.2 e successive	Non disponibile
Europa (Stoccolma)	Versione 3.05.2 e successive	Versione 3.05.2 e successive
Europa (Zurigo)	Versione 3.05.2 e successive	Non disponibile
Israele (Tel Aviv)	Versione 3.05.2 e successive	Non disponibile
Messico (centrale)	Non disponibile	Non disponibile
Medio Oriente (Bahrein)	Versione 3.05.2 e successive	Non disponibile
Medio Oriente (Emirati Arabi Uniti)	Versione 3.05.2 e successive	Non disponibile
Sud America (San Paolo)	Versione 3.05.2 e successive	Versione 3.05.2 e successive
AWS GovCloud (Stati Uniti orientali)	Non disponibile	Non disponibile
AWS GovCloud (Stati Uniti occidentali)	Non disponibile	Non disponibile

Integrazioni Zero-ETL di Aurora PostgreSQL

Le seguenti regioni e versioni del motore sono disponibili per le integrazioni Aurora PostgreSQL zero-ETL con Amazon Redshift e. Amazon Sagemaker

Region	Integrazione zero-ETL con Amazon Redshift per Aurora PostgreSQL versione 17	Integrazione zero-ETL con Amazon SageMaker Aurora PostgreSQL versione 17	Integrazione zero-ETL con Amazon Redshift per Aurora PostgreSQL versione 16	Integrazione zero-ETL con Amazon SageMaker Aurora PostgreSQL versione 16
Stati Uniti orientali (Ohio)	Versione 17.4 e successive	Versione 17.4 e successive	Versione 16.4 e successive	Versione 16.4 e successive
Stati Uniti orientali (Virginia settentrionale)	Versione 17.4 e successive	Versione 17.4 e successive	Versione 16.4 e successive	Versione 16.4 e successive
Stati Uniti occidentali (California settentrionale)	Versione 17.4 e successive	Non disponibile	Versione 16.4 e successive	Non disponibile
Stati Uniti occidentali (Oregon)	Versione 17.4 e successive	Versione 17.4 e successive	Versione 16.4 e successive	Versione 16.4 e successive
Africa (Città del Capo)	Versione 17.4 e successive	Non disponibile	Versione 16.4 e successive	Non disponibile
Asia Pacifico (Hong Kong)	Versione 17.4 e successive	Versione 17.4 e successive	Versione 16.4 e successive	Versione 16.4 e successive
Asia Pacific (Hyderabad)	Versione 17.4 e successive	Versione 16.4 e successive	Non disponibile	Non disponibile
Asia Pacifico (Giacarta)	Versione 17.4 e successive	Non disponibile	Versione 16.4 e successive	Non disponibile
Asia Pacifico (Melbourne)	Versione 17.4 e successive	Non disponibile	Versione 16.4 e successive	Non disponibile

Region	Integrazione zero-ETL con Amazon Redshift per Aurora PostgreSQL versione 17	Integrazione zero-ETL con Amazon SageMaker Aurora PostgreSQL versione 17	Integrazione zero-ETL con Amazon Redshift per Aurora PostgreSQL versione 16	Integrazione zero-ETL con Amazon SageMaker Aurora PostgreSQL versione 16
Asia Pacifico (Malesia)	Non disponibile	Non disponibile		
Asia Pacifico (Malesia)	Non disponibile	Non disponibile		
Asia Pacifico (Mumbai)	Versione 17.4 e successive	Non disponibile	Versione 16.4 e successive	Non disponibile
Asia Pacifico (Nuova Zelanda)	Non disponibile	Non disponibile		
Asia Pacifico (Osaka-Locale)	Versione 17.4 e successive	Non disponibile	Versione 16.4 e successive	Non disponibile
Asia Pacifico (Seul)	Versione 17.4 e successive	Versione 17.4 e successive	Versione 16.4 e successive	Versione 16.4 e successive
Asia Pacifico (Singapore)	Versione 17.4 e successive	Versione 17.4 e successive	Versione 16.4 e successive	Versione 16.4 e successive
Asia Pacifico (Sydney)	Versione 17.4 e successive	Versione 17.4 e successive	Versione 16.4 e successive	Versione 16.4 e successive
Asia Pacifico (Taipei)	Non disponibile	Non disponibile		
Asia Pacifico (Thailandia)	Non disponibile	Non disponibile		

Region	Integrazione zero-ETL con Amazon Redshift per Aurora PostgreSQL versione 17	Integrazione zero-ETL con Amazon SageMaker Aurora PostgreSQL versione 17	Integrazione zero-ETL con Amazon Redshift per Aurora PostgreSQL versione 16	Integrazione zero-ETL con Amazon SageMaker Aurora PostgreSQL versione 16
Asia Pacifico (Taipei)	Non disponibile	Non disponibile		
Asia Pacifico (Thailandia)	Non disponibile	Non disponibile		
Asia Pacifico (Tokyo)	Versione 17.4 e successive	Versione 17.4 e successive	Versione 16.4 e successive	Versione 16.4 e successive
Canada (Centrale)	Versione 17.4 e successive	Versione 17.4 e successive	Versione 16.4 e successive	Versione 16.4 e successive
Canada occidentale (Calgary)	Versione 17.4 e successive	Non disponibile	Versione 16.4 e successive	Non disponibile
Cina (Pechino)	Versione 17.4 e successive	Non disponibile	Versione 16.4 e successive	Non disponibile
Cina (Ningxia)	Versione 17.4 e successive	Non disponibile	Versione 16.4 e successive	Non disponibile
Europa (Francoforte)	Versione 17.4 e successive	Versione 17.4 e successive	Versione 16.4 e successive	Versione 16.4 e successive
Europa (Irlanda)	Versione 17.4 e successive	Versione 17.4 e successive	Versione 16.4 e successive	Versione 16.4 e successive
Europa (Londra)	Versione 17.4 e successive	Versione 17.4 e successive	Versione 16.4 e successive	Versione 16.4 e successive

Region	Integrazione zero-ETL con Amazon Redshift per Aurora PostgreSQL versione 17	Integrazione zero-ETL con Amazon SageMaker Aurora PostgreSQL versione 17	Integrazione zero-ETL con Amazon Redshift per Aurora PostgreSQL versione 16	Integrazione zero-ETL con Amazon SageMaker Aurora PostgreSQL versione 16
Europa (Milano)	Versione 17.4 e successive	Non disponibile	Versione 16.4 e successive	Non disponibile
Europa (Parigi)	Versione 17.4 e successive	Non disponibile	Versione 16.4 e successive	Non disponibile
Europa (Spagna)	Versione 17.4 e successive	Non disponibile	Versione 16.4 e successive	Non disponibile
Europa (Stoccolma)	Versione 17.4 e successive	Versione 17.4 e successive	Versione 16.4 e successive	Versione 16.4 e successive
Europa (Zurigo)	Versione 17.4 e successive	Non disponibile	Versione 16.4 e successive	Non disponibile
Israele (Tel Aviv)	Versione 17.4 e successive	Non disponibile	Versione 16.4 e successive	Non disponibile
Messico (centrale)	Non disponibile	Non disponibile		
Messico (centrale)	Non disponibile	Non disponibile		
Medio Oriente (Bahrein)	Versione 17.4 e successive	Non disponibile	Versione 16.4 e successive	Non disponibile
Medio Oriente (Emirati Arabi Uniti)	Versione 17.4 e successive	Non disponibile	Versione 16.4 e successive	Non disponibile

Region	Integrazione zero-ETL con Amazon Redshift per Aurora PostgreSQL versione 17	Integrazione zero-ETL con Amazon SageMaker Aurora PostgreSQL versione 17	Integrazione zero-ETL con Amazon Redshift per Aurora PostgreSQL versione 16	Integrazione zero-ETL con Amazon SageMaker Aurora PostgreSQL versione 16
Sud America (San Paolo)	Versione 17.4 e successive	Versione 17.4 e successive	Versione 16.4 e successive	Versione 16.4 e successive
AWS GovCloud (Stati Uniti orientali)	Non disponibile	Non disponibile	Non disponibile	Non disponibile
AWS GovCloud (Stati Uniti occidentali)	Non disponibile	Non disponibile	Non disponibile	Non disponibile

Regioni e motori di database Aurora supportati per Server proxy per Amazon RDS

Amazon RDS Proxy è un proxy di database completamente gestito e ad alta disponibilità che rende le applicazioni più scalabili mediante il pooling e la condivisione di connessioni di database consolidate. Per ulteriori informazioni su RDS Proxy, consulta Server proxy per Amazon RDS per Aurora.

Argomenti

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

Amazon RDS Proxy con Aurora MySQL

Di seguito sono riportate le versioni di motore e le regioni disponibili per RDS Proxy con Aurora MySQL.

Region	Aurora MySQL versione 3	Aurora MySQL versione 2
Stati Uniti orientali (Virginia settentrionale)	Versione 3.01.0 e successive	Versione 2.07 e versione 2.11 e successive
Stati Uniti orientali (Ohio)	Versione 3.01.0 e successive	Versione 2.07 e versione 2.11 e successive
Stati Uniti occidentali (California settentrionale)	Versione 3.01.0 e successive	Versione 2.07 e versione 2.11 e successive
Stati Uniti occidentali (Oregon)	Versione 3.01.0 e successive	Versione 2.07 e versione 2.11 e successive
Africa (Città del Capo)	Versione 3.01.0 e successive	Versione 2.07 e versione 2.11 e successive
Asia Pacifico (Hong Kong)	Versione 3.01.0 e successive	Versione 2.07 e versione 2.11 e successive
Asia Pacific (Hyderabad)	Versione 3.01.0 e successive	Versione 2.07 e versione 2.11 e successive
Asia Pacifico (Giacarta)	Versione 3.01.0 e successive	Versione 2.07 e versione 2.11 e successive
Asia Pacifico (Malesia)	Versione 3.01.0 e successive	Versione 2.07 e versione 2.11 e successive
Asia Pacifico (Melbourne)	Versione 3.01.0 e successive	Versione 2.07 e versione 2.11 e successive
Asia Pacifico (Mumbai)	Versione 3.01.0 e successive	Versione 2.07 e versione 2.11 e successive
Asia Pacifico (Nuova Zelanda)	Non disponibile	Non disponibile
Asia Pacifico (Osaka-Locale)	Versione 3.01.0 e successive	Versione 2.07 e versione 2.11 e successive

Region	Aurora MySQL versione 3	Aurora MySQL versione 2
Asia Pacifico (Seul)	Versione 3.01.0 e successive	Versione 2.07 e versione 2.11 e successive
Asia Pacifico (Singapore)	Versione 3.01.0 e successive	Versione 2.07 e versione 2.11 e successive
Asia Pacifico (Sydney)	Versione 3.01.0 e successive	Versione 2.07 e versione 2.11 e successive
Asia Pacifico (Taipei)	Non disponibile	Non disponibile
Asia Pacifico (Thailandia)	Versione 3.01.0 e successive	Versione 2.07 e versione 2.11 e successive
Asia Pacifico (Tokyo)	Versione 3.01.0 e successive	Versione 2.07 e versione 2.11 e successive
Canada (Centrale)	Versione 3.01.0 e successive	Versione 2.07 e versione 2.11 e successive
Canada occidentale (Calgary)	Versione 3.01.0 e successive	Versione 2.07 e versione 2.11 e successive
Cina (Pechino)	Versione 3.01.0 e successive	Versione 2.07 e versione 2.11 e successive
Cina (Ningxia)	Versione 3.01.0 e successive	Versione 2.07 e versione 2.11 e successive
Europa (Francoforte)	Versione 3.01.0 e successive	Versione 2.07 e versione 2.11 e successive
Europa (Irlanda)	Versione 3.01.0 e successive	Versione 2.07 e versione 2.11 e successive
Europa (Londra)	Versione 3.01.0 e successive	Versione 2.07 e versione 2.11 e successive

Region	Aurora MySQL versione 3	Aurora MySQL versione 2
Europa (Milano)	Versione 3.01.0 e successive	Versione 2.07 e versione 2.11 e successive
Europa (Parigi)	Versione 3.01.0 e successive	Versione 2.07 e versione 2.11 e successive
Europa (Spagna)	Versione 3.01.0 e successive	Versione 2.07 e versione 2.11 e successive
Europa (Stoccolma)	Versione 3.01.0 e successive	Versione 2.07 e versione 2.11 e successive
Europa (Zurigo)	Versione 3.01.0 e successive	Versione 2.07 e versione 2.11 e successive
Israele (Tel Aviv)	Versione 3.01.0 e successive	Versione 2.07 e versione 2.11 e successive
Messico (centrale)	Versione 3.01.0 e successive	Versione 2.07 e versione 2.11 e successive
Medio Oriente (Bahrein)	Versione 3.01.0 e successive	Versione 2.07 e versione 2.11 e successive
Medio Oriente (Emirati Arabi Uniti)	Versione 3.01.0 e successive	Versione 2.07 e versione 2.11 e successive
Sud America (San Paolo)	Versione 3.01.0 e successive	Versione 2.07 e versione 2.11 e successive
AWS GovCloud (Stati Uniti orientali)	Versione 3.01.0 e successive	Versione 2.07 e versione 2.11 e successive
AWS GovCloud (Stati Uniti occidentali)	Versione 3.01.0 e successive	Versione 2.07 e versione 2.11 e successive

Amazon RDS Proxy con Aurora PostgreSQL

Di seguito sono riportate le versioni di motore e le regioni disponibili per RDS Proxy con Aurora PostgreSQL.

Region	Aurora PostgreSQL 17	Aurora PostgreSQL 16	Aurora PostgreSQL 15	Aurora PostgreSQL 14	Aurora PostgreSQL 13	Aurora PostgreSQL 12	Aurora PostgreSQL 11
Stati Uniti orientali (Virginia settentrionale)	Versione 17.4 e successive	Versione 16.1 e successive	Versione 15.2 e successive	Versione 14.3 e versioni successive	Versione 13.4 e successive	Versione 12.8 e successive	Versione 11.9 e versione 11.13 e successive
Stati Uniti orientali (Ohio)	Versione 17.4 e successive	Versione 16.1 e successive	Versione 15.2 e successive	Versione 14.3 e versioni successive	Versione 13.4 e successive	Versione 12.8 e successive	Versione 11.9 e versione 11.13 e successive
Stati Uniti occidentali (California settentrionale)	Versione 17.4 e successive	Versione 16.1 e successive	Versione 15.2 e successive	Versione 14.3 e versioni successive	Versione 13.4 e successive	Versione 12.8 e successive	Versione 11.9 e versione 11.13 e successive
Stati Uniti occidentali (Oregon)	Versione 17.4 e successive	Versione 16.1 e successive	Versione 15.2 e successive	Versione 14.3 e versioni successive	Versione 13.4 e successive	Versione 12.8 e successive	Versione 11.9 e versione 11.13 e successive

Region	Aurora PostgreSQL 17	Aurora PostgreSQL 16	Aurora PostgreSQL 15	Aurora PostgreSQL 14	Aurora PostgreSQL 13	Aurora PostgreSQL 12	Aurora PostgreSQL 11
Africa (Città del Capo)	Versione 17.4 e successive	Versione 16.1 e successive	Versione 15.2 e successive	Versione 14.3 e versioni successive	Versione 13.4 e successive	Versione 12.8 e successive	Versione 11.9 e versione 11.13 e successive
Asia Pacifico (Hong Kong)	Versione 17.4 e successive	Versione 16.1 e successive	Versione 15.2 e successive	Versione 14.3 e versioni successive	Versione 13.4 e successive	Versione 12.8 e successive	Versione 11.9 e versione 11.13 e successive
Asia Pacific (Hyderabad)	Versione 17.4 e successive	Versione 16.1 e successive	Versione 15.2 e successive	Versione 14.3 e versioni successive	Versione 13.4 e successive	Versione 12.8 e successive	Versione 11.9 e versione 11.13 e successive
Asia Pacifico (Giacarta)	Versione 17.4 e successive	Versione 16.1 e successive	Versione 15.2 e successive	Versione 14.3 e versioni successive	Versione 13.4 e successive	Versione 12.8 e successive	Versione 11.9 e versione 11.13 e successive

Region	Aurora PostgreSQL 17	Aurora PostgreSQL 16	Aurora PostgreSQL 15	Aurora PostgreSQL 14	Aurora PostgreSQL 13	Aurora PostgreSQL 12	Aurora PostgreSQL 11
Asia Pacifico (Malesia)	Versione 17.4 e successive	Versione 16.1 e successive	Versione 15.2 e successive	Versione 14.3 e versioni successive	Versione 13.4 e successive	Versione 12.8 e successive	Versione 11.9 e versione 11.13 e successive
Asia Pacifico (Melbourne)	Versione 17.4 e successive	Versione 16.1 e successive	Versione 15.2 e successive	Versione 14.3 e versioni successive	Versione 13.4 e successive	Versione 12.8 e successive	Versione 11.9 e versione 11.13 e successive
Asia Pacifico (Mumbai)	Versione 17.4 e successive	Versione 16.1 e successive	Versione 15.2 e successive	Versione 14.3 e versioni successive	Versione 13.4 e successive	Versione 12.8 e successive	Versione 11.9 e versione 11.13 e successive
Asia Pacifico (Nuova Zelanda)	Non disponibile	Non disponibile	Non disponibile	Non disponibile	Non disponibile	Non disponibile	Non disponibile
Asia Pacifico (Osaka-Locale)	Versione 17.4 e successive	Versione 16.1 e successive	Versione 15.2 e successive	Versione 14.3 e versioni successive	Versione 13.4 e successive	Versione 12.8 e successive	Versione 11.9 e versione 11.13 e successive

Region	Aurora PostgreSQL 17	Aurora PostgreSQL 16	Aurora PostgreSQL 15	Aurora PostgreSQL 14	Aurora PostgreSQL 13	Aurora PostgreSQL 12	Aurora PostgreSQL 11
Asia Pacifico (Seul)	Versione 17.4 e successive	Versione 16.1 e successive	Versione 15.2 e successive	Versione 14.3 e versioni successive	Versione 13.4 e successive	Versione 12.8 e successive	Versione 11.9 e versione 11.13 e successive
Asia Pacifico (Singapore)	Versione 17.4 e successive	Versione 16.1 e successive	Versione 15.2 e successive	Versione 14.3 e versioni successive	Versione 13.4 e successive	Versione 12.8 e successive	Versione 11.9 e versione 11.13 e successive
Asia Pacifico (Sydney)	Versione 17.4 e successive	Versione 16.1 e successive	Versione 15.2 e successive	Versione 14.3 e versioni successive	Versione 13.4 e successive	Versione 12.8 e successive	Versione 11.9 e versione 11.13 e successive
Asia Pacifico (Taipei)	Non disponibile	Non disponibile	Non disponibile	Non disponibile	Non disponibile	Non disponibile	Non disponibile
Asia Pacifico (Thailandia)	Versione 17.4 e successive	Versione 16.1 e successive	Versione 15.2 e successive	Versione 14.3 e versioni successive	Versione 13.4 e successive	Versione 12.8 e successive	Versione 11.9 e versione 11.13 e successive

Region	Aurora PostgreSQL 17	Aurora PostgreSQL 16	Aurora PostgreSQL 15	Aurora PostgreSQL 14	Aurora PostgreSQL 13	Aurora PostgreSQL 12	Aurora PostgreSQL 11
Asia Pacifico (Tokyo)	Versione 17.4 e successive	Versione 16.1 e successive	Versione 15.2 e successive	Versione 14.3 e versioni successive	Versione 13.4 e successive	Versione 12.8 e successive	Versione 11.9 e versione 11.13 e successive
Canada (Centrale)	Versione 17.4 e successive	Versione 16.1 e successive	Versione 15.2 e successive	Versione 14.3 e versioni successive	Versione 13.4 e successive	Versione 12.8 e successive	Versione 11.9 e versione 11.13 e successive
Canada occidentale (Calgary)	Versione 17.4 e successive	Versione 16.1 e successive	Versione 15.2 e successive	Versione 14.3 e versioni successive	Versione 13.4 e successive	Versione 12.8 e successive	Versione 11.9 e versione 11.13 e successive
Cina (Pechino)	Versione 17.4 e successive	Versione 16.1 e successive	Versione 15.2 e successive	Versione 14.3 e versioni successive	Versione 13.4 e successive	Versione 12.8 e successive	Versione 11.9 e versione 11.13 e successive

Region	Aurora PostgreSQL 17	Aurora PostgreSQL 16	Aurora PostgreSQL 15	Aurora PostgreSQL 14	Aurora PostgreSQL 13	Aurora PostgreSQL 12	Aurora PostgreSQL 11
Cina (Ningxia)	Versione 17.4 e successive	Versione 16.1 e successive	Versione 15.2 e successive	Versione 14.3 e versioni successive	Versione 13.4 e successive	Versione 12.8 e successive	Versione 11.9 e versione 11.13 e successive
Europa (Francoforte)	Versione 17.4 e successive	Versione 16.1 e successive	Versione 15.2 e successive	Versione 14.3 e versioni successive	Versione 13.4 e successive	Versione 12.8 e successive	Versione 11.9 e versione 11.13 e successive
Europa (Irlanda)	Versione 17.4 e successive	Versione 16.1 e successive	Versione 15.2 e successive	Versione 14.3 e versioni successive	Versione 13.4 e successive	Versione 12.8 e successive	Versione 11.9 e versione 11.13 e successive
Europa (Londra)	Versione 17.4 e successive	Versione 16.1 e successive	Versione 15.2 e successive	Versione 14.3 e versioni successive	Versione 13.4 e successive	Versione 12.8 e successive	Versione 11.9 e versione 11.13 e successive

Region	Aurora PostgreSQL 17	Aurora PostgreSQL 16	Aurora PostgreSQL 15	Aurora PostgreSQL 14	Aurora PostgreSQL 13	Aurora PostgreSQL 12	Aurora PostgreSQL 11
Europa (Milano)	Versione 17.4 e successive	Versione 16.1 e successive	Versione 15.2 e successive	Versione 14.3 e versioni successive	Versione 13.4 e successive	Versione 12.8 e successive	Versione 11.9 e versione 11.13 e successive
Europa (Parigi)	Versione 17.4 e successive	Versione 16.1 e successive	Versione 15.2 e successive	Versione 14.3 e versioni successive	Versione 13.4 e successive	Versione 12.8 e successive	Versione 11.9 e versione 11.13 e successive
Europa (Spagna)	Versione 17.4 e successive	Versione 16.1 e successive	Versione 15.2 e successive	Versione 14.3 e versioni successive	Versione 13.4 e successive	Versione 12.8 e successive	Versione 11.9 e versione 11.13 e successive
Europa (Stoccolma)	Versione 17.4 e successive	Versione 16.1 e successive	Versione 15.2 e successive	Versione 14.3 e versioni successive	Versione 13.4 e successive	Versione 12.8 e successive	Versione 11.9 e versione 11.13 e successive

Region	Aurora PostgreSQL 17	Aurora PostgreSQL 16	Aurora PostgreSQL 15	Aurora PostgreSQL 14	Aurora PostgreSQL 13	Aurora PostgreSQL 12	Aurora PostgreSQL 11
Europa (Zurigo)	Versione 17.4 e successive	Versione 16.1 e successive	Versione 15.2 e successive	Versione 14.3 e versioni successive	Versione 13.4 e successive	Versione 12.8 e successive	Versione 11.9 e versione 11.13 e successive
Israele (Tel Aviv)	Versione 17.4 e successive	Versione 16.1 e successive	Versione 15.2 e successive	Versione 14.3 e versioni successive	Versione 13.4 e successive	Versione 12.8 e successive	Versione 11.9 e versione 11.13 e successive
Messico (centrale)	Versione 17.4 e successive	Versione 16.1 e successive	Versione 15.2 e successive	Versione 14.3 e versioni successive	Versione 13.4 e successive	Versione 12.8 e successive	Versione 11.9 e versione 11.13 e successive
Medio Oriente (Bahrein)	Versione 17.4 e successive	Versione 16.1 e successive	Versione 15.2 e successive	Versione 14.3 e versioni successive	Versione 13.4 e successive	Versione 12.8 e successive	Versione 11.9 e versione 11.13 e successive

Region	Aurora PostgreSQL 17	Aurora PostgreSQL 16	Aurora PostgreSQL 15	Aurora PostgreSQL 14	Aurora PostgreSQL 13	Aurora PostgreSQL 12	Aurora PostgreSQL 11
Medio Oriente (Emirati Arabi Uniti)	Versione 17.4 e successive	Versione 16.1 e successive	Versione 15.2 e successive	Versione 14.3 e versioni successive	Versione 13.4 e successive	Versione 12.8 e successive	Versione 11.9 e versione 11.13 e successive
Sud America (San Paolo)	Versione 17.4 e successive	Versione 16.1 e successive	Versione 15.2 e successive	Versione 14.3 e versioni successive	Versione 13.4 e successive	Versione 12.8 e successive	Versione 11.9 e versione 11.13 e successive
AWS GovCloud (Stati Uniti orientali)	Versione 17.4 e successive	Versione 16.1 e successive	Versione 15.2 e successive	Versione 14.3 e versioni successive	Versione 13.4 e successive	Versione 12.8 e successive	Versione 11.9 e versione 11.13 e successive
AWS GovCloud (Stati Uniti occidentali)	Versione 17.4 e successive	Versione 16.1 e successive	Versione 15.2 e successive	Versione 14.3 e versioni successive	Versione 13.4 e successive	Versione 12.8 e successive	Versione 11.9 e versione 11.13 e successive

Regioni e motori di database Aurora supportati per l’integrazione di Secrets Manager

Con Gestione dei segreti AWS, puoi sostituire le credenziali con codifica fissa (incluse le password del database) con una chiamata API a Secrets Manager in modo da recuperare il segreto a livello di codice. Per ulteriori informazioni su Secrets Manager, consulta la Guida per l'utente di Gestione dei segreti AWS.

Puoi specificare che Amazon Aurora gestisca la password dell'utente master in Secrets Manager per un cluster database Aurora. Aurora genera la password, la memorizza in Secrets Manager e la ruota regolarmente. Per ulteriori informazioni, consulta Gestione delle password con Amazon Aurora e Gestione dei segreti AWS.

L’integrazione di Secrets Manager è disponibile in tutte le Regioni AWS.

Regioni e motori di database Aurora supportati per Aurora Serverless v2

Aurora Serverless v2 è una funzionalità con scalabilità automatica on demand progettata per fornire un approccio conveniente all'esecuzione di carichi di lavoro intermittenti o imprevedibili su Amazon Aurora. Si dimensiona automaticamente la capacità in base alle esigenze dell'applicazione. Il dimensionamento è più veloce e più granulare rispetto a quello di Aurora Serverless v1. Con Aurora Serverless v2, ogni cluster può contenere un'istanza database scrittore e più istanze database lettore. È possibile combinare Aurora Serverless v2 e le istanze database con provisioning tradizionali all'interno dello stesso cluster. Per ulteriori informazioni, consulta Uso di Aurora Serverless v2.

Argomenti

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

Aurora Serverless v2 con Aurora MySQL

Di seguito sono riportate le versioni di motore e le regioni disponibili per Aurora Serverless v2 con Aurora MySQL.

Region	Aurora MySQL versione 3
Stati Uniti orientali (Virginia settentrionale)	Versione 3.02.0 e successive

Region	Aurora MySQL versione 3
Stati Uniti orientali (Ohio)	Versione 3.02.0 e successive
Stati Uniti occidentali (California settentrionale)	Versione 3.02.0 e successive
Stati Uniti occidentali (Oregon)	Versione 3.02.0 e successive
Africa (Città del Capo)	Versione 3.02.0 e successive
Asia Pacifico (Hong Kong)	Versione 3.02.0 e successive
Asia Pacific (Hyderabad)	Versione 3.02.3 e successive
Asia Pacifico (Giacarta)	Versione 3.02.0 e successive
Asia Pacifico (Malesia)	Versioni 3.04.3, 3.05.2, 3.06.1, 3.07.1 e successive
Asia Pacifico (Melbourne)	Versione 3.02.3 e successive
Asia Pacifico (Mumbai)	Versione 3.02.0 e successive
Asia Pacifico (Nuova Zelanda)	Non disponibile
Asia Pacifico (Osaka-Locale)	Versione 3.02.0 e successive
Asia Pacifico (Seul)	Versione 3.02.0 e successive
Asia Pacifico (Singapore)	Versione 3.02.0 e successive
Asia Pacifico (Sydney)	Versione 3.02.0 e successive
Asia Pacifico (Taipei)	Non disponibile
Asia Pacifico (Thailandia)	Versioni 3.04.3 e successive, 3.08.0 e successive
Asia Pacifico (Tokyo)	Versione 3.02.0 e successive
Canada (Centrale)	Versione 3.02.0 e successive

Region	Aurora MySQL versione 3
Canada occidentale (Calgary)	Versione 3.04.0 e successive
Cina (Pechino)	Versione 3.02.2 e successive
Cina (Ningxia)	Versione 3.02.2 e successive
Europa (Francoforte)	Versione 3.02.0 e successive
Europa (Irlanda)	Versione 3.02.0 e successive
Europa (Londra)	Versione 3.02.0 e successive
Europa (Milano)	Versione 3.02.0 e successive
Europa (Parigi)	Versione 3.02.0 e successive
Europa (Spagna)	Versione 3.02.3 e successive
Europa (Stoccolma)	Versione 3.02.0 e successive
Europa (Zurigo)	Versione 3.02.3 e successive
Israele (Tel Aviv)	Versioni 3.02.3 e successive, 3.03.1 e successive
Messico (centrale)	Non disponibile
Medio Oriente (Bahrein)	Versione 3.02.0 e successive
Medio Oriente (Emirati Arabi Uniti)	Versione 3.02.3 e successive
Sud America (San Paolo)	Versione 3.02.0 e successive
AWS GovCloud (Stati Uniti orientali)	Versione 3.02.2 e successive
AWS GovCloud (Stati Uniti occidentali)	Versione 3.02.2 e successive

I limiti ACU massimo e minimo per la capacità di Aurora Serverless v2 possono variare a seconda della versione del motore. Per informazioni dettagliate, vedi Capacità di Aurora Serverless v2.

Aurora Serverless v2 con Aurora PostgreSQL

Di seguito sono riportate le versioni di motore e le regioni disponibili per Aurora Serverless v2 con Aurora PostgreSQL.

Region	Aurora PostgreSQL 17	Aurora PostgreSQL 16	Aurora PostgreSQL 15	Aurora PostgreSQL 14	Aurora PostgreSQL 13
Stati Uniti orientali (Virginia settentrionale)	Versione 17.4 e successive	Versione 16.1 e successive	Versione 15.2 e successive	Versione 14.3 e versioni successive	Versione 13.6 e versioni successive
Stati Uniti orientali (Ohio)	Versione 17.4 e successive	Versione 16.1 e successive	Versione 15.2 e successive	Versione 14.3 e versioni successive	Versione 13.6 e versioni successive
Stati Uniti occidentali (California settentrionale)	Versione 17.4 e successive	Versione 16.1 e successive	Versione 15.2 e successive	Versione 14.3 e versioni successive	Versione 13.6 e versioni successive
Stati Uniti occidentali (Oregon)	Versione 17.4 e successive	Versione 16.1 e successive	Versione 15.2 e successive	Versione 14.3 e versioni successive	Versione 13.6 e versioni successive
Africa (Città del Capo)	Versione 17.4 e successive	Versione 16.1 e successive	Versione 15.2 e successive	Versione 14.3 e versioni successive	Versione 13.6 e versioni successive
Asia Pacifico (Hong Kong)	Versione 17.4 e successive	Versione 16.1 e successive	Versione 15.2 e successive	Versione 14.3 e versioni successive	Versione 13.6 e versioni successive

Region	Aurora PostgreSQL 17	Aurora PostgreSQL 16	Aurora PostgreSQL 15	Aurora PostgreSQL 14	Aurora PostgreSQL 13
Asia Pacific (Hyderabad)	Versione 17.4 e successive	Versione 16.1 e successive	Versione 15.2 e successive	Versione 14.6 e successive	Versione 13.9 e successive
Asia Pacifico (Giacarta)	Versione 17.4 e successive	Versione 16.1 e successive	Versione 15.2 e successive	Versione 14.3 e versioni successive	Versione 13.6 e versioni successive
Asia Pacifico (Malesia)	Versione 17.4 e successive	Versione 16.1 e successive	Versione 15.4 e successive	Versione 14.6, 14.9 e successive	Versione 13.9, 13.12 e successive
Asia Pacifico (Melbourne)	Versione 17.4 e successive	Versione 16.1 e successive	Versione 15.2 e successive	Versione 14.6 e successive	Versione 13.9 e successive
Asia Pacifico (Mumbai)	Versione 17.4 e successive	Versione 16.1 e successive	Versione 15.2 e successive	Versione 14.3 e versioni successive	Versione 13.6 e versioni successive
Asia Pacifico (Nuova Zelanda)	Non disponibile	Non disponibile	Non disponibile	Non disponibile	Non disponibile
Asia Pacifico (Osaka-Locale)	Versione 17.4 e successive	Versione 16.1 e successive	Versione 15.2 e successive	Versione 14.3 e versioni successive	Versione 13.6 e versioni successive
Asia Pacifico (Seul)	Versione 17.4 e successive	Versione 16.1 e successive	Versione 15.2 e successive	Versione 14.3 e versioni successive	Versione 13.6 e versioni successive
Asia Pacifico (Singapore)	Versione 17.4 e successive	Versione 16.1 e successive	Versione 15.2 e successive	Versione 14.3 e versioni successive	Versione 13.6 e versioni successive

Region	Aurora PostgreSQL 17	Aurora PostgreSQL 16	Aurora PostgreSQL 15	Aurora PostgreSQL 14	Aurora PostgreSQL 13
Asia Pacifico (Taipei)	Non disponibile	Non disponibile	Non disponibile	Non disponibile	Non disponibile
Asia Pacifico (Sydney)	Versione 17.4 e successive	Versione 16.1 e successive	Versione 15.2 e successive	Versione 14.3 e versioni successive	Versione 13.6 e versioni successive
Asia Pacifico (Thailandia)	Versione 17.4 e successive	Versione 16.4 e successive	Versione 15.8 e successive	Versione 14.13 e successive	Non disponibile
Asia Pacifico (Tokyo)	Versione 17.4 e successive	Versione 16.1 e successive	Versione 15.2 e successive	Versione 14.3 e versioni successive	Versione 13.6 e versioni successive
Canada (Centrale)	Versione 17.4 e successive	Versione 16.1 e successive	Versione 15.2 e successive	Versione 14.3 e versioni successive	Versione 13.6 e versioni successive
Canada occidentale (Calgary)	Versione 17.4 e successive	Versione 16.1 e successive	Versione 15.3 e successive	Versione 14.6, 14.8 e successive	Versione 13.9, 13.11 e successive
Cina (Pechino)	Versione 17.4 e successive	Versione 16.1 e successive	Versione 15.2 e successive	Versione 14.3 e versioni successive	Versione 13.6 e versioni successive
Cina (Ningxia)	Versione 17.4 e successive	Versione 16.1 e successive	Versione 15.2 e successive	Versione 14.3 e versioni successive	Versione 13.6 e versioni successive
Europa (Francoforte)	Versione 17.4 e successive	Versione 16.1 e successive	Versione 15.2 e successive	Versione 14.3 e versioni successive	Versione 13.6 e versioni successive

Region	Aurora PostgreSQL 17	Aurora PostgreSQL 16	Aurora PostgreSQL 15	Aurora PostgreSQL 14	Aurora PostgreSQL 13
Europa (Irlanda)	Versione 17.4 e successive	Versione 16.1 e successive	Versione 15.2 e successive	Versione 14.3 e versioni successive	Versione 13.6 e versioni successive
Europa (Londra)	Versione 17.4 e successive	Versione 16.1 e successive	Versione 15.2 e successive	Versione 14.3 e versioni successive	Versione 13.6 e versioni successive
Europa (Milano)	Versione 17.4 e successive	Versione 16.1 e successive	Versione 15.2 e successive	Versione 14.3 e versioni successive	Versione 13.6 e versioni successive
Europa (Parigi)	Versione 17.4 e successive	Versione 16.1 e successive	Versione 15.2 e successive	Versione 14.3 e versioni successive	Versione 13.6 e versioni successive
Europa (Spagna)	Versione 17.4 e successive	Versione 16.1 e successive	Versione 15.2 e successive	Versione 14.6 e successive	Versione 13.9 e successive
Europa (Stoccolma)	Versione 17.4 e successive	Versione 16.1 e successive	Versione 15.2 e successive	Versione 14.3 e versioni successive	Versione 13.6 e versioni successive
Europa (Zurigo)	Versione 17.4 e successive	Versione 16.1 e successive	Versione 15.2 e successive	Versione 14.6 e successive	Versione 13.9 e successive
Israele (Tel Aviv)	Versione 17.4 e successive	Versione 16.1 e successive	Versione 15.2 e successive	Versione 14.6 e successive	Versione 13.9 e successive
Messico (centrale)	Non disponibile	Non disponibile	Non disponibile	Non disponibile	Non disponibile
Medio Oriente (Bahrein)	Versione 17.4 e successive	Versione 16.1 e successive	Versione 15.2 e successive	Versione 14.3 e versioni successive	Versione 13.6 e versioni successive

Region	Aurora PostgreSQL 17	Aurora PostgreSQL 16	Aurora PostgreSQL 15	Aurora PostgreSQL 14	Aurora PostgreSQL 13
Medio Oriente (Emirati Arabi Uniti)	Versione 17.4 e successive	Versione 16.1 e successive	Versione 15.2 e successive	Versione 14.6 e successive	Versione 13.9 e successive
Sud America (San Paolo)	Versione 17.4 e successive	Versione 16.1 e successive	Versione 15.2 e successive	Versione 14.3 e versioni successive	Versione 13.6 e versioni successive
AWS GovCloud (Stati Uniti orientali)	Versione 17.4 e successive	Versione 16.1 e successive	Versione 15.2 e successive	Versione 14.3 e versioni successive	Versione 13.6 e versioni successive
AWS GovCloud (Stati Uniti occidentali)	Versione 17.4 e successive	Versione 16.1 e successive	Versione 15.2 e successive	Versione 14.3 e versioni successive	Versione 13.6 e versioni successive

I limiti ACU massimo e minimo per la capacità di Aurora Serverless v2 possono variare a seconda della versione del motore. Per informazioni dettagliate, vedi Capacità di Aurora Serverless v2.

Aurora Serverless v1

Important

AWS ha annunciato la end-of-life data delAurora Serverless v1: 31 marzo 2025. È fortemente consigliabile aggiornare i cluster di database Aurora Serverless v1 a Aurora Serverless v2 prima di tale data. L'aggiornamento può comportare una modifica del numero di versione principale del motore di database. Pertanto, è importante pianificare, testare e implementare questo passaggio prima della data. end-of-life A partire dall'8 gennaio 2025, i clienti non saranno più in grado di creare nuovi Aurora Serverless v1 cluster o istanze né con la

CLI. Console di gestione AWS Per informazioni sul processo di migrazione, consulta Aggiornamento da un cluster Aurora Serverless v1 ad Aurora Serverless v2.
Aurora Serverless v2 scala più rapidamente e in modo più granulare. Aurora Serverless v2 garantisce inoltre una maggiore compatibilità con altre funzionalità Aurora, come le istanze database di lettura. Per maggiori informazioni su Aurora Serverless v2, consulta Uso di Aurora Serverless v2.

Aurora Serverless v1 è una funzionalità con scalabilità automatica on demand progettata per fornire un approccio conveniente all'esecuzione di carichi di lavoro intermittenti o imprevedibili su Amazon Aurora. Avvia, arresta e dimensiona automaticamente la capacità in base alle esigenze dell'applicazione usando una singola istanza database in ogni cluster. Per ulteriori informazioni, consulta Utilizzo di Amazon Aurora Serverless v1.

Argomenti

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

Aurora Serverless v1 con Aurora MySQL

Di seguito sono riportate le versioni di motore e le regioni disponibili per Aurora Serverless v1 con Aurora MySQL.

Region	Aurora MySQL versione 3	Aurora MySQL versione 2
Stati Uniti orientali (Virginia settentrionale)	Non disponibile	Versione 2.11.4
Stati Uniti orientali (Ohio)	Non disponibile	Versione 2.11.4
Stati Uniti occidentali (Californ ia settentrionale)	Non disponibile	Versione 2.11.4
Stati Uniti occidentali (Oregon)	Non disponibile	Versione 2.11.4
Africa (Città del Capo)	Non disponibile	Non disponibile
Asia Pacifico (Hong Kong)	Non disponibile	Non disponibile

Region	Aurora MySQL versione 3	Aurora MySQL versione 2
Asia Pacific (Hyderabad)	Non disponibile	Non disponibile
Asia Pacifico (Giacarta)	Non disponibile	Non disponibile
Asia Pacifico (Malesia)	Non disponibile	Non disponibile
Asia Pacifico (Melbourne)	Non disponibile	Non disponibile
Asia Pacifico (Mumbai)	Non disponibile	Versione 2.11.4
Asia Pacifico (Osaka-Locale)	Non disponibile	Non disponibile
Asia Pacifico (Seul)	Non disponibile	Versione 2.11.4
Asia Pacifico (Singapore)	Non disponibile	Versione 2.11.4
Asia Pacifico (Sydney)	Non disponibile	Versione 2.11.4
Asia Pacifico (Thailandia)	Non disponibile	Non disponibile
Asia Pacifico (Tokyo)	Non disponibile	Versione 2.11.4
Canada (Centrale)	Non disponibile	Versione 2.11.4
Canada occidentale (Calgary)	Non disponibile	Non disponibile
Cina (Pechino)	Non disponibile	Non disponibile
Cina (Ningxia)	Non disponibile	Versione 2.11.4
Europa (Francoforte)	Non disponibile	Versione 2.11.4
Europa (Irlanda)	Non disponibile	Versione 2.11.4
Europa (Londra)	Non disponibile	Versione 2.11.4
Europa (Milano)	Non disponibile	Non disponibile
Europa (Parigi)	Non disponibile	Versione 2.11.4

Region	Aurora MySQL versione 3	Aurora MySQL versione 2
Europa (Spagna)	Non disponibile	Non disponibile
Europa (Stoccolma)	Non disponibile	Non disponibile
Europa (Zurigo)	Non disponibile	Non disponibile
Israele (Tel Aviv)	Non disponibile	Non disponibile
Medio Oriente (Bahrein)	Non disponibile	Non disponibile
Medio Oriente (Emirati Arabi Uniti)	Non disponibile	Non disponibile
Sud America (San Paolo)	Non disponibile	Non disponibile
AWS GovCloud (Stati Uniti orientali)	Non disponibile	Non disponibile
AWS GovCloud (Stati Uniti occidentali)	Non disponibile	Non disponibile

Aurora Serverless v1 con Aurora PostgreSQL

Di seguito sono riportate le versioni di motore e le regioni disponibili per Aurora Serverless v1 con Aurora PostgreSQL.

Region	Aurora PostgreSQL 13
Stati Uniti orientali (Virginia settentrionale)	Versione 13.12
Stati Uniti orientali (Ohio)	Versione 13.12
Stati Uniti occidentali (California settentrionale)	Versione 13.12
Stati Uniti occidentali (Oregon)	Versione 13.12
Africa (Città del Capo)	Non disponibile

Region	Aurora PostgreSQL 13
Asia Pacifico (Hong Kong)	Non disponibile
Asia Pacific (Hyderabad)	Non disponibile
Asia Pacifico (Giacarta)	Non disponibile
Asia Pacifico (Malesia)	Non disponibile
Asia Pacifico (Melbourne)	Non disponibile
Asia Pacifico (Mumbai)	Versione 13.12
Asia Pacifico (Osaka-Locale)	Non disponibile
Asia Pacifico (Seul)	Versione 13.12
Asia Pacifico (Singapore)	Versione 13.12
Asia Pacifico (Sydney)	Versione 13.12
Asia Pacifico (Thailandia)	Non disponibile
Asia Pacifico (Tokyo)	Versione 13.12
Canada (Centrale)	Versione 13.12
Canada occidentale (Calgary)	Non disponibile
Cina (Pechino)	Non disponibile
Cina (Ningxia)	Non disponibile
Europa (Francoforte)	Versione 13.12
Europa (Irlanda)	Versione 13.12
Europa (Londra)	Versione 13.12
Europa (Milano)	Non disponibile

Region	Aurora PostgreSQL 13
Europa (Parigi)	Versione 13.12
Europa (Spagna)	Non disponibile
Europa (Stoccolma)	Non disponibile
Europa (Zurigo)	Non disponibile
Israele (Tel Aviv)	Non disponibile
Medio Oriente (Bahrein)	Non disponibile
Medio Oriente (Emirati Arabi Uniti)	Non disponibile
Sud America (San Paolo)	Non disponibile
AWS GovCloud (Stati Uniti orientali)	Non disponibile
AWS GovCloud (Stati Uniti occidentali)	Non disponibile

Regioni e motori di database Aurora supportati per l'API dati RDS

L'API dati RDS fornisce un'interfaccia di servizi web a un cluster di database Amazon Aurora. Anziché gestire le connessioni al database dalle applicazioni client, è possibile eseguire comandi SQL su un endpoint HTTPS. Per ulteriori informazioni, consulta Utilizzo dell'API dati di Amazon RDS.

Argomenti

- API dati con Aurora PostgreSQL Serverless v2 e cluster allocati
- API dati con Aurora MySQL Serverless v2 e cluster allocati
- API dati con Aurora PostgreSQL Serverless v1
- API dati con Aurora MySQL Serverless v1

API dati con Aurora PostgreSQL Serverless v2 e cluster allocati

Di seguito sono riportate le Regioni e le versioni del motore disponibili per l'API dati con Aurora PostgreSQL Serverless v2 e cluster di database allocati.

Region	Aurora PostgreSQL 17	Aurora PostgreSQL 16	Aurora PostgreSQL 15	Aurora PostgreSQL 14	Aurora PostgreSQL 13
Stati Uniti orientali (Virginia settentrionale)	Versione 17.4 e successive	Versione 16.1 e successive	Versione 15.3 e successive	Versione 14.8 e successive	Versione 13.11 e successive
Stati Uniti orientali (Ohio)	Versione 17.4 e successive	Versione 16.1 e successive	Versione 15.3 e successive	Versione 14.8 e successive	Versione 13.11 e successive
Stati Uniti occidentali (California settentrionale)	Versione 17.4 e successive	Versione 16.1 e successive	Versione 15.3 e successive	Versione 14.8 e successive	Versione 13.11 e successive
Stati Uniti occidentali (Oregon)	Versione 17.4 e successive	Versione 16.1 e successive	Versione 15.3 e successive	Versione 14.8 e successive	Versione 13.11 e successive
Africa (Città del Capo)	Versione 17.4 e successive	Versione 16.1 e successive	Versione 15.3 e successive	Versione 14.8 e successive	Versione 13.11 e successive
Asia Pacifico (Hong Kong)	Versione 17.4 e successive	Versione 16.1 e successive	Versione 15.3 e successive	Versione 14.8 e successive	Versione 13.11 e successive
Asia Pacific (Hyderabad)	Non disponibile	Non disponibile	Non disponibile	Non disponibile	Non disponibile
Asia Pacifico (Giacarta)	Versione 17.4 e successive	Versione 16.1 e successive	Versione 15.3 e successive	Versione 14.8 e successive	Versione 13.11 e successive

Region	Aurora PostgreSQL 17	Aurora PostgreSQL 16	Aurora PostgreSQL 15	Aurora PostgreSQL 14	Aurora PostgreSQL 13
Asia Pacifico (Malesia)	Versione 17.4 e successive	Versione 16.1 e successive	Versione 15.3 e successive	Versione 14.8 e successive	Versione 13.11 e successive
Asia Pacifico (Melbourne)	Non disponibile	Non disponibile	Non disponibile	Non disponibile	Non disponibile
Asia Pacifico (Mumbai)	Versione 17.4 e successive	Versione 16.1 e successive	Versione 15.3 e successive	Versione 14.8 e successive	Versione 13.11 e successive
Asia Pacifico (Nuova Zelanda)	Non disponibile	Non disponibile	Non disponibile	Non disponibile	Non disponibile
Asia Pacifico (Osaka-Locale)	Versione 17.4 e successive	Versione 16.1 e successive	Versione 15.3 e successive	Versione 14.8 e successive	Versione 13.11 e successive
Asia Pacifico (Seul)	Versione 17.4 e successive	Versione 16.1 e successive	Versione 15.3 e successive	Versione 14.8 e successive	Versione 13.11 e successive
Asia Pacifico (Singapore)	Versione 17.4 e successive	Versione 16.1 e successive	Versione 15.3 e successive	Versione 14.8 e successive	Versione 13.11 e successive
Asia Pacifico (Sydney)	Versione 17.4 e successive	Versione 16.1 e successive	Versione 15.3 e successive	Versione 14.8 e successive	Versione 13.11 e successive
Asia Pacifico (Taipei)	Non disponibile	Non disponibile	Non disponibile	Non disponibile	Non disponibile

Region	Aurora PostgreSQL 17	Aurora PostgreSQL 16	Aurora PostgreSQL 15	Aurora PostgreSQL 14	Aurora PostgreSQL 13
Asia Pacifico (Thailandia)	Versione 17.4 e successive	Versione 16.1 e successive	Versione 15.3 e successive	Versione 14.8 e successive	Versione 13.11 e successive
Asia Pacifico (Tokyo)	Versione 17.4 e successive	Versione 16.1 e successive	Versione 15.3 e successive	Versione 14.8 e successive	Versione 13.11 e successive
Canada (Centrale)	Versione 17.4 e successive	Versione 16.1 e successive	Versione 15.3 e successive	Versione 14.8 e successive	Versione 13.11 e successive
Canada occidentale (Calgary)	Non disponibile	Non disponibile	Non disponibile	Non disponibile	Non disponibile
Cina (Pechino)	Non disponibile	Non disponibile	Non disponibile	Non disponibile	Non disponibile
Cina (Ningxia)	Non disponibile	Non disponibile	Non disponibile	Non disponibile	Non disponibile
Europa (Francoforte)	Versione 17.4 e successive	Versione 16.1 e successive	Versione 15.3 e successive	Versione 14.8 e successive	Versione 13.11 e successive
Europa (Irlanda)	Versione 17.4 e successive	Versione 16.1 e successive	Versione 15.3 e successive	Versione 14.8 e successive	Versione 13.11 e successive
Europa (Londra)	Versione 17.4 e successive	Versione 16.1 e successive	Versione 15.3 e successive	Versione 14.8 e successive	Versione 13.11 e successive

Region	Aurora PostgreSQL 17	Aurora PostgreSQL 16	Aurora PostgreSQL 15	Aurora PostgreSQL 14	Aurora PostgreSQL 13
Europa (Milano)	Versione 17.4 e successive	Versione 16.1 e successive	Versione 15.3 e successive	Versione 14.8 e successive	Versione 13.11 e successive
Europa (Parigi)	Versione 17.4 e successive	Versione 16.1 e successive	Versione 15.3 e successive	Versione 14.8 e successive	Versione 13.11 e successive
Europa (Spagna)	Versione 17.4 e successive	Versione 16.1 e successive	Versione 15.3 e successive	Versione 14.8 e successive	Versione 13.11 e successive
Europa (Stoccolma)	Versione 17.4 e successive	Versione 16.1 e successive	Versione 15.3 e successive	Versione 14.8 e successive	Versione 13.11 e successive
Europa (Zurigo)	Non disponibile	Non disponibile	Non disponibile	Non disponibile	Non disponibile
Israele (Tel Aviv)	Non disponibile	Non disponibile	Non disponibile	Non disponibile	Non disponibile
Medio Oriente (Bahrein)	Versione 17.4 e successive	Versione 16.1 e successive	Versione 15.3 e successive	Versione 14.8 e successive	Versione 13.11 e successive
Messico (centrale)	Non disponibile	Non disponibile	Non disponibile	Non disponibile	Non disponibile
Medio Oriente (Emirati Arabi Uniti)	Versione 17.4 e successive	Versione 16.1 e successive	Versione 15.3 e successive	Versione 14.8 e successive	Versione 13.11 e successive

Region	Aurora PostgreSQL 17	Aurora PostgreSQL 16	Aurora PostgreSQL 15	Aurora PostgreSQL 14	Aurora PostgreSQL 13
Sud America (San Paolo)	Versione 17.4 e successive	Versione 16.1 e successive	Versione 15.3 e successive	Versione 14.8 e successive	Versione 13.11 e successive
AWS GovCloud (Stati Uniti orientali)	Non disponibile	Non disponibile	Non disponibile	Non disponibile	Non disponibile
AWS GovCloud (Stati Uniti occidentali)	Non disponibile	Non disponibile	Non disponibile	Non disponibile	Non disponibile

API dati con Aurora MySQL Serverless v2 e cluster allocati

Di seguito sono riportate le Regioni e le versioni del motore disponibili per l'API dati con Aurora MySQL Serverless v2 e cluster di database allocati.

Region	Aurora MySQL versione 3
Stati Uniti orientali (Ohio)	Versione 3.07 e successive
Stati Uniti orientali (Virginia settentrionale)	Versione 3.07 e successive
Stati Uniti occidentali (California settentrionale)	Versione 3.07 e successive
Stati Uniti occidentali (Oregon)	Versione 3.07 e successive
Africa (Città del Capo)	Versione 3.07 e successive
Asia Pacifico (Hong Kong)	Versione 3.07 e successive
Asia Pacific (Hyderabad)	Non disponibile

Region	Aurora MySQL versione 3
Asia Pacifico (Giacarta)	Versione 3.07 e successive
Asia Pacifico (Malesia)	Versione 3.07 e successive
Asia Pacifico (Melbourne)	Non disponibile
Asia Pacifico (Mumbai)	Versione 3.07 e successive
Asia Pacifico (Nuova Zelanda)	Non disponibile
Asia Pacifico (Osaka-Locale)	Versione 3.07 e successive
Asia Pacifico (Seul)	Versione 3.07 e successive
Asia Pacifico (Singapore)	Versione 3.07 e successive
Asia Pacifico (Sydney)	Versione 3.07 e successive
Asia Pacifico (Taipei)	Non disponibile
Asia Pacifico (Thailandia)	Versione 3.07 e successive
Asia Pacifico (Tokyo)	Versione 3.07 e successive
Canada (Centrale)	Versione 3.07 e successive
Canada occidentale (Calgary)	Non disponibile
Cina (Pechino)	Non disponibile
Cina (Ningxia)	Non disponibile
Europa (Francoforte)	Versione 3.07 e successive
Europa (Irlanda)	Versione 3.07 e successive
Europa (Londra)	Versione 3.07 e successive
Europa (Milano)	Versione 3.07 e successive

Region	Aurora MySQL versione 3
Europa (Parigi)	Versione 3.07 e successive
Europa (Spagna)	Versione 3.07 e successive
Europa (Stoccolma)	Versione 3.07 e successive
Europa (Zurigo)	Non disponibile
Israele (Tel Aviv)	Non disponibile
Messico (centrale)	Non disponibile
Medio Oriente (Bahrein)	Versione 3.07 e successive
Medio Oriente (Emirati Arabi Uniti)	Versione 3.07 e successive
Sud America (San Paolo)	Versione 3.07 e successive
AWS GovCloud (Stati Uniti orientali)	Non disponibile
AWS GovCloud (Stati Uniti occidentali)	Non disponibile

API dati con Aurora PostgreSQL Serverless v1

Di seguito sono riportate le Regioni e le versioni del motore disponibili per l'API dati con Aurora PostgreSQL Serverless v1.

Region	Aurora PostgreSQL 13	Aurora PostgreSQL 11
Stati Uniti orientali (Virginia settentrionale)	Versione 13.9	Versione 11.18
Stati Uniti orientali (Ohio)	Versione 13.9	Versione 11.18
Stati Uniti occidentali (Californ ia settentrionale)	Versione 13.9	Versione 11.18
Stati Uniti occidentali (Oregon)	Versione 13.9	Versione 11.18

Region	Aurora PostgreSQL 13	Aurora PostgreSQL 11
Africa (Città del Capo)	Non disponibile	Non disponibile
Asia Pacifico (Hong Kong)	Non disponibile	Non disponibile
Asia Pacific (Hyderabad)	Non disponibile	Non disponibile
Asia Pacifico (Giacarta)	Non disponibile	Non disponibile
Asia Pacifico (Malesia)	Non disponibile	Non disponibile
Asia Pacifico (Melbourne)	Non disponibile	Non disponibile
Asia Pacifico (Mumbai)	Versione 13.9	Versione 11.18
Asia Pacifico (Osaka-Locale)	Non disponibile	Non disponibile
Asia Pacifico (Seul)	Versione 13.9	Versione 11.18
Asia Pacifico (Singapore)	Versione 13.9	Versione 11.18
Asia Pacifico (Sydney)	Versione 13.9	Versione 11.18
Asia Pacifico (Thailandia)	Non disponibile	Non disponibile
Asia Pacifico (Tokyo)	Versione 13.9	Versione 11.18
Canada (Centrale)	Versione 13.9	Versione 11.18
Cina (Pechino)	Non disponibile	Non disponibile
Cina (Ningxia)	Non disponibile	Non disponibile
Europa (Francoforte)	Versione 13.9	Versione 11.18
Europa (Irlanda)	Versione 13.9	Versione 11.18
Europa (Londra)	Versione 13.9	Versione 11.18
Europa (Milano)	Non disponibile	Non disponibile

Region	Aurora PostgreSQL 13	Aurora PostgreSQL 11
Europa (Parigi)	Versione 13.9	Versione 11.18
Europa (Spagna)	Versione 13.9	Versione 11.18
Europa (Stoccolma)	Non disponibile	Non disponibile
Europa (Zurigo)	Non disponibile	Non disponibile
Israele (Tel Aviv)	Non disponibile	Non disponibile
Medio Oriente (Bahrein)	Non disponibile	Non disponibile
Medio Oriente (Emirati Arabi Uniti)	Non disponibile	Non disponibile
Sud America (San Paolo)	Non disponibile	Non disponibile
AWS GovCloud (Stati Uniti orientali)	Non disponibile	Non disponibile
AWS GovCloud (Stati Uniti occidentali)	Non disponibile	Non disponibile

API dati con Aurora MySQL Serverless v1

Di seguito sono riportate le Regioni e le versioni del motore disponibili per l'API dati con Aurora MySQL Serverless v1.

Region	Aurora MySQL versione 2
Stati Uniti orientali (Virginia settentrionale)	versione 2.11.3
Stati Uniti orientali (Ohio)	versione 2.11.3
Stati Uniti occidentali (California settentrionale)	versione 2.11.3
Stati Uniti occidentali (Oregon)	versione 2.11.3

Region	Aurora MySQL versione 2
Africa (Città del Capo)	Non disponibile
Asia Pacifico (Hong Kong)	Non disponibile
Asia Pacific (Hyderabad)	Non disponibile
Asia Pacifico (Giacarta)	Non disponibile
Asia Pacifico (Malesia)	Non disponibile
Asia Pacifico (Melbourne)	Non disponibile
Asia Pacifico (Mumbai)	versione 2.11.3
Asia Pacifico (Osaka-Locale)	Non disponibile
Asia Pacifico (Seul)	versione 2.11.3
Asia Pacifico (Singapore)	versione 2.11.3
Asia Pacifico (Sydney)	versione 2.11.3
Asia Pacifico (Thailandia)	Non disponibile
Asia Pacifico (Tokyo)	versione 2.11.3
Canada (Centrale)	versione 2.11.3
Canada occidentale (Calgary)	Non disponibile
Cina (Pechino)	Non disponibile
Cina (Ningxia)	versione 2.11.3
Europa (Francoforte)	versione 2.11.3
Europa (Irlanda)	versione 2.11.3
Europa (Londra)	versione 2.11.3

Region	Aurora MySQL versione 2
Europa (Milano)	Non disponibile
Europa (Parigi)	versione 2.11.3
Europa (Spagna)	versione 2.11.3
Europa (Stoccolma)	Non disponibile
Europa (Zurigo)	Non disponibile
Israele (Tel Aviv)	Non disponibile
Medio Oriente (Bahrein)	Non disponibile
Medio Oriente (Emirati Arabi Uniti)	Non disponibile
Sud America (San Paolo)	Non disponibile
AWS GovCloud (Stati Uniti orientali)	Non disponibile
AWS GovCloud (Stati Uniti occidentali)	Non disponibile

Regioni e motori di database Aurora supportati per l’applicazione di patch senza tempo di inattività (ZDP, Zero-Downtime Patching)

L'esecuzione di aggiornamenti per i cluster database Aurora MySQL comporta la possibilità di un'interruzione quando il database viene arrestato e durante l'aggiornamento. Per impostazione predefinita, se si avvia l'aggiornamento mentre il database è occupato, si perdono tutte le connessioni e le transazioni elaborate dal cluster di database. Se si aspetta che il database sia inattivo per eseguire l'aggiornamento, potrebbe essere necessario attendere a lungo.

La funzionalità di applicazione di patch senza tempi di inattività si basa sul miglior tentativo per preservare le connessioni client attraverso un aggiornamento Aurora. Se l'applicazione di patch senza tempi di inattività viene completata correttamente, le sessioni delle applicazioni vengono conservate e il motore del database si riavvia mentre l'aggiornamento è in corso. Il riavvio del motore del database può causare un calo del throughput della durata di alcuni secondi fino a circa un minuto.

Per informazioni dettagliate sulle condizioni e sulle versioni del motore in cui la funzionalità di applicazione di patch senza tempi di inattività è disponibile per gli aggiornamenti di Aurora MySQL, consulta Utilizzo dell'applicazione di patch senza tempi di inattività.

Per informazioni dettagliate sulle condizioni e sulle versioni del motore in cui la funzionalità di applicazione di patch senza tempi di inattività è disponibile per gli aggiornamenti di Aurora PostgreSQL, consulta Aggiornamenti della versione secondaria e applicazione di patch senza tempi di inattività.

Regioni supportate in Aurora PostgreSQL Limitless Database

Amazon Aurora PostgreSQL Limitless Database offre un dimensionamento orizzontale automatizzato per elaborare milioni di transazioni di scrittura al secondo e gestisce petabyte di dati mantenendo la semplicità di funzionamento all'interno di un singolo database. Con Aurora PostgreSQL Limitless Database, è possibile dedicarsi alla creazione di applicazioni su larga scala senza dover creare e mantenere soluzioni complesse per scalare i dati su più istanze database per supportare i carichi di lavoro.

Per ulteriori informazioni, consulta Utilizzo di Amazon Aurora PostgreSQL Limitless Database.

Aurora PostgreSQL Limitless Database è disponibile in tutte le Regioni AWS ad eccezione di Asia Pacifico (Taipei).

Regioni e motori di database supportati per le funzionalità native del motore Aurora

I motori di database Aurora supportano inoltre caratteristiche e funzionalità aggiuntive specifiche per Aurora. Alcune funzionalità native del motore potrebbero avere supporto limitato o privilegi limitati per un particolare motore di database Aurora, versione o regione.

Argomenti

- Funzionalità native del motore per Aurora MySQL
- Funzionalità native del motore per Aurora PostgreSQL

Funzionalità native del motore per Aurora MySQL

Di seguito sono riportate le funzionalità native del motore per Aurora MySQL

- Auditing avanzato

- Backtrack
- Query di fault injection
- Inoltro di scrittura all'interno del cluster
- Query parallela

Funzionalità native del motore per Aurora PostgreSQL

Di seguito sono riportate le funzionalità native del motore per Aurora PostgreSQL

- Babelfish
- Query di fault injection
- Gestione del piano di query

Connessioni degli endpoint Amazon Aurora

Amazon Aurora in genere utilizza un cluster di istanze database anziché una singola istanza. Ogni connessione viene gestita da un'istanza database specifica. Quando ti connetti a un cluster Aurora, il nome host e la porta specificati puntano a un handler intermedio chiamato endpoint. Aurora utilizza il meccanismo endpoint per astrarre queste connessioni. Pertanto, non è necessario codificare tutti i nomi host o scrivere una propria logica per il bilanciamento e il reindirizzamento delle connessioni quando alcune istanze database non sono disponibili.

Per determinate attività di Aurora, le diverse istanze o i gruppi di istanze svolgono ruoli differenti. Ad esempio, l'istanza primaria gestisce tutte le istruzioni DDL (Data Definition Language) e DML (Data Manipulation Language) e fino a 15 repliche di Aurora gestiscono il traffico di query di sola lettura.

Argomenti

- Tipi di endpoint di Aurora
- Visualizzazione degli endpoint per un cluster Aurora
- Come gli endpoint Aurora funzionano con elevata disponibilità
- Endpoint del cluster per Amazon Aurora
- Endpoint di lettura per Amazon Aurora
- Endpoint di istanza per Amazon Aurora
- Endpoint personalizzati per Amazon Aurora

Tipi di endpoint di Aurora

Usando gli endpoint puoi associare ogni connessione all'istanza o al gruppo di istanze appropriato in base al caso d'uso. Ad esempio, per eseguire le istruzioni DDL puoi connetterti a qualsiasi istanza sia l'istanza primaria. Per eseguire le query, puoi connetterti all'endpoint di lettura mentre Aurora esegue automaticamente il bilanciamento delle connessioni tra tutte le repliche Aurora. Per i cluster con istanze database con capacità o configurazioni diverse, puoi connetterti agli endpoint personalizzati associati a diversi sottoinsiemi di istanze database. Per la diagnosi o l'ottimizzazione, puoi connetterti a un endpoint di istanza specifico per esaminare i dettagli su una determinata istanza database.

Un endpoint è rappresentato da un URL specifico di Aurora contenente un indirizzo host e una porta. Di seguito sono riportati i tipi di endpoint disponibili da un cluster database Aurora.

Endpoint del cluster

Connettiti all'istanza primaria del cluster per sviluppare e testare le applicazioni, nonché per eseguire trasformazioni come istruzioni `INSERT` e operazioni DDL, DML ed ETL. Trova la posizione dell'endpoint del cluster utilizzando la Console di gestione AWS, l'interfaccia AWS CLI o l'API Amazon RDS, come descritto in Visualizzazione degli endpoint per un cluster Aurora.

Per ulteriori informazioni sugli endpoint del cluster, consulta Endpoint del cluster per Amazon Aurora.

Endpoint di lettura

Esegui le query. Aurora esegue automaticamente il bilanciamento delle connessioni tra tutte le repliche Aurora. Trova la posizione dell'endpoint di lettura utilizzando la Console di gestione AWS, l'interfaccia AWS CLI o l'API Amazon RDS, come descritto in Visualizzazione degli endpoint per un cluster Aurora.

Per ulteriori informazioni sugli endpoint di lettura, consulta Endpoint di lettura per Amazon Aurora.

Endpoint dell'istanza

Esamina i dettagli di un'istanza database specifica per la diagnosi o l'ottimizzazione. Puoi trovare la posizione dell'endpoint per ciascuna istanza solo nella Console di gestione AWS, nella pagina dei dettagli dell'istanza in uso.

Per ulteriori informazioni sugli endpoint dell'istanza, consulta Endpoint di istanza per Amazon Aurora.

Endpoint personalizzato

Connettiti a diversi sottoinsiemi di istanze database sul cluster di database. Questo è utile quando si hanno diverse capacità e configurazioni di istanze all'interno del cluster di database. Trova le posizioni degli endpoint personalizzati utilizzando la Console di gestione AWS, l'interfaccia AWS CLI o l'API Amazon RDS, come descritto in Visualizzazione degli endpoint per un cluster Aurora.

Per ulteriori informazioni sugli endpoint personalizzati, consulta Endpoint personalizzati per Amazon Aurora.

Endpoint di scrittura per Database globali Aurora

Database globale Aurora dispone di un tipo speciale di endpoint che ha lo stesso scopo dell'endpoint di un cluster Aurora autonomo. Gestisce sia le richieste di scrittura che quelle di lettura. Quando un cluster secondario diventa il nuovo cluster primario a causa di uno switchover o di un failover, Aurora modifica automaticamente questo endpoint in modo che punti all'endpoint del nuovo cluster primario, nell'altra Regione AWS. In questo modo, non è necessario codificare la Regione AWS nella stringa di connessione dell'applicazione e neppure modificare la stringa di connessione quando cambia il layout del database globale. Aurora crea questo endpoint quando si configura un Database globale Aurora, ad esempio scegliendo Aggiungi Regione per un cluster Aurora nella Console di gestione AWS.

Per informazioni su come utilizzare questo tipo di endpoint con Database globale Aurora, consulta Connessione a Database globale Amazon Aurora.

Visualizzazione degli endpoint per un cluster Aurora

Sebbene sia possibile trovare la posizione dell'endpoint dell'istanza solo nella pagina dei dettagli dell'istanza nella Console di gestione AWS, puoi utilizzare la console, l'interfaccia AWS CLI o l'API Amazon RDS per trovare le posizioni di cluster, endpoint di lettura ed endpoint personalizzati.

Console

Nella Console di gestione AWS, l'endpoint del cluster, l'endpoint di lettura e gli eventuali endpoint personalizzati sono visualizzati nella pagina dei dettagli dell'istanza di ciascun cluster. L'endpoint dell'istanza viene visualizzato nella pagina dei dettagli di ogni istanza. Quando ti connetti, devi aggiungere al nome dell'endpoint, mostrato in questa pagina dei dettagli, il numero di porta associato seguito dai due punti.

AWS CLI

Con il comando AWS CLI, puoi trovare gli endpoint di scrittura, di lettura e personalizzati nell'output del comando describe-db-clusters. Ad esempio, il comando seguente mostra gli attributi dell'endpoint per tutti i cluster nella regione AWS corrente.

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

Amazon RDS API

Con l'API Amazon RDS, puoi recuperare gli endpoint chiamando l'operazione DescribeDBClusterEndpoints.

Come gli endpoint Aurora funzionano con elevata disponibilità

Per i cluster in cui è importante la disponibilità elevata, utilizza l'endpoint del cluster per le connessioni di lettura-scrittura o a scopo generale e l'endpoint di lettura per le connessioni di sola lettura. Gli endpoint di scrittura e lettura gestiscono il failover delle istanze DB meglio degli endpoint di istanza. A differenza degli endpoint istanza, gli endpoint di scrittura e lettura modificano automaticamente l'istanza database a cui si connettono se un'istanza database nel cluster diventa non disponibile. Per maggiori informazioni sugli endpoint del cluster e di lettura, consulta Endpoint del cluster per Amazon Aurora e Endpoint di lettura per Amazon Aurora.

In caso di errore dell'istanza database primaria di un cluster database, Aurora esegue automaticamente il failover su una nuova istanza database primaria. Questa operazione viene eseguita promuovendo una replica Aurora esistente in una nuova istanza database primaria oppure creando una nuova istanza database primaria. Se si verifica un failover, è possibile utilizzare l'endpoint del cluster per la riconnessione all'istanza database primaria appena creata o promossa oppure ricorrere all'endpoint di lettura per riconnettersi a una delle repliche Aurora nel cluster di database. Durante un failover, l'endpoint di lettura potrebbe dirigere le connessioni alla nuova istanza database primaria di un cluster database per un breve periodo di tempo dopo che una replica di Aurora viene promossa a nuova istanza database primaria.

Se progetti la tua logica applicativa per gestire le connessioni agli endpoint di istanza, puoi rilevare a livello di codice o manualmente il set risultante di istanze database disponibili nel cluster database. Utilizza il comando describe-db-clusters AWS CLI o l'operazione API RDS DescribeDBClusters per individuare il cluster di database e gli endpoint di lettura e le istanze database, oltre che per capire

se le istanze database sono istanze di lettura e per scoprire i relativi livelli di promozione. Puoi quindi confermare le classi di istanza dopo il failover e collegarti a un endpoint di istanza appropriato.

Per ulteriori informazioni sui failover, consulta Tolleranza ai guasti di un cluster DB Aurora.

Per ulteriori informazioni sulla disponibilità elevata in Amazon Aurora, consulta Elevata disponibilità di Amazon Aurora.

Endpoint del cluster per Amazon Aurora

Per endpoint del cluster (o endpoint di scrittura) si intende un endpoint per un cluster di database Aurora che si connette all'istanza database primaria corrente di quel cluster. Questo endpoint è l'unico in grado di eseguire operazioni di scrittura come le istruzioni DDL. Per questo motivo, l'endpoint del cluster è quello a cui ti connetti quando imposti un cluster per la prima volta o quando il cluster contiene una sola istanza database.

Ciascun cluster database Aurora ha un endpoint del cluster e un'istanza database primaria.

L'endpoint del cluster si usa per tutte le operazioni di scrittura sul cluster database, inclusi aggiornamenti, inserimenti, eliminazioni e modifiche DDL. Puoi anche utilizzare l'endpoint del cluster per le operazioni di lettura, come ad esempio le query.

L'endpoint del cluster fornisce un failover del cluster per connessioni di lettura-scrittura al cluster database. In caso di errore dell'istanza database primaria corrente di un cluster database, Aurora esegue automaticamente il failover su una nuova istanza database primaria. Durante un failover, il cluster database continua a servire le richieste di connessione all'endpoint del cluster dalla nuova istanza database primaria, riducendo al minimo l'interruzione del servizio.

L'esempio seguente mostra un endpoint del cluster per un cluster database Aurora MySQL.

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

Ogni cluster Aurora ha un singolo endpoint cluster integrato, il cui nome e altri attributi sono gestiti da Aurora. Non puoi creare, eliminare o modificare questo tipo di endpoint.

L'endpoint del cluster viene utilizzato per la gestione del cluster, l'esecuzione di operazioni di estrazione, trasformazione, caricamento (ETL) o di applicazioni di sviluppo e test. L'endpoint del cluster si connette all'istanza primaria del cluster. L'istanza primaria è l'unica istanza database in cui è possibile creare tabelle e indici, eseguire istruzioni `INSERT` e altre operazioni DDL e DML.

L'indirizzo IP fisico indicato dall'endpoint del cluster cambia quando il meccanismo di failover promuove una nuova istanza database come istanza primaria di lettura-scrittura per il cluster. Se utilizzi un pool di connessioni o altro tipo di multiplexing, preparati ad azzerare o ridurre il valore time-to-live per qualsiasi informazione DNS memorizzata nella cache. In tal modo si evita di provare a stabilire una connessione in lettura-scrittura a un'istanza database che non è più disponibile o che ora è di sola lettura a causa di un failover.

Endpoint di lettura per Amazon Aurora

L’endpoint di lettura per un cluster di database Aurora fornisce supporto per il bilanciamento delle connessioni di sola lettura al cluster di database Aurora. Puoi utilizzare l'endpoint di lettura per le operazioni di lettura, come ad esempio le query. Elaborando tali istruzioni nelle repliche Aurora di sola lettura, questo endpoint riduce il sovraccarico sull'istanza primaria. Consente inoltre al cluster di ridimensionare la capacità di gestire query `SELECT` simultanee, proporzionale al numero di repliche Aurora nel cluster. Ogni cluster database Aurora ha un endpoint di lettura.

Se il cluster contiene una o più repliche Aurora, l’endpoint di lettura bilancia ogni richiesta di connessione tra le repliche Aurora. In tal caso, è possibile eseguire solo istruzioni di sola lettura come `SELECT` in quella sessione. Se il cluster contiene solo un'istanza primaria e nessuna replica Aurora, l'endpoint di lettura si connette all'istanza primaria. In tal caso, è possibile eseguire operazioni di scrittura attraverso l'endpoint.

L'esempio seguente mostra un endpoint di lettura per un cluster database Aurora MySQL.

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

Usi l'endpoint di lettura fornisce per le connessioni di sola lettura al cluster Aurora. Questo endpoint utilizza un meccanismo di bilanciamento della connessione per consentire al cluster di gestire un carico di lavoro che implica numerose query. L'endpoint di lettura è quello che fornisci alle applicazioni che eseguono report o altre operazioni di sola lettura sul cluster.

L’endpoint di lettura bilancia le connessioni alle repliche Aurora disponibili in un cluster di database Aurora. Non può bilanciare le singole query. Se desideri bilanciare ogni query per distribuire il carico di lavoro di lettura per un cluster di database, apri una nuova connessione all’endpoint di lettura per ogni query.

Ogni cluster Aurora ha un singolo endpoint di lettura integrato, il cui nome e altri attributi sono gestiti da Aurora. Non puoi creare, eliminare o modificare questo tipo di endpoint.

Se il cluster contiene solo una destinazione primaria (istanza o gruppo di shard del database) e nessuna replica Aurora, l'endpoint di lettura si connette all'istanza primaria. In tal caso, è possibile eseguire operazioni di scrittura tramite questo endpoint.

> **Tip**
>
> Tramite RDS Proxy, è possibile creare ulteriori endpoint di sola lettura per un cluster Aurora. Questi endpoint eseguono lo stesso tipo di bilanciamento della connessione dell'endpoint di lettura Aurora. Le applicazioni possono riconnettersi più rapidamente agli endpoint proxy rispetto all'endpoint di lettura Aurora se le istanze del lettore non sono disponibili. Gli endpoint proxy possono anche sfruttare altre funzionalità proxy come il multiplexing. Per ulteriori informazioni, consulta Utilizzo degli endpoint di lettura con cluster Aurora.

Endpoint di istanza per Amazon Aurora

Un endpoint dell'istanza si connette a un'istanza database specifica all'interno di un cluster Aurora. Ogni istanza database in un cluster database dispone del proprio endpoint dell'istanza univoco. Pertanto esiste un endpoint dell'istanza per l'istanza database primaria corrente del cluster database e un endpoint dell'istanza per ciascuna replica di Aurora nel cluster database.

L'endpoint dell'istanza fornisce controllo diretto sulle connessioni al cluster database, per scenari in cui l'utilizzo dell'endpoint del cluster o dell'endpoint di lettura potrebbe non essere appropriato. Ad esempio, l'applicazione client potrebbe richiedere un maggiore bilanciamento della connessione granulare in base al tipo di carico di lavoro. In questo caso, è possibile configurare più client per connettersi alle repliche di Aurora in un cluster database per distribuire i carichi di lavoro in lettura. Per un esempio in cui si usano gli endpoint dell'istanza per migliorare la velocità di connessione dopo un failover per Aurora PostgreSQL, consulta Failover rapido con Amazon Aurora PostgreSQL. Per un esempio in cui vengono utilizzati gli endpoint dell'istanza per migliorare la velocità di connessione dopo un failover per Aurora MySQL, consultare MariaDB Connector/J failover support – case Amazon Aurora.

L'esempio seguente mostra un endpoint dell'istanza per un'istanza database in un cluster database Aurora MySQL.

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

Ogni istanza database in un cluster Aurora ha un proprio endpoint di istanza integrato, il cui nome e altri attributi sono gestiti da Aurora. Non puoi creare, eliminare o modificare questo tipo di endpoint. Potresti avere familiarità con gli endpoint di istanza se usi Amazon RDS. Tuttavia, in genere con Aurora si utilizzano gli endpoint di scrittura e lettura più spesso degli endpoint di istanza.

Nelle operazioni quotidiane di Aurora, il modo principale in cui si utilizzano gli endpoint di istanza consiste nel diagnosticare i problemi di capacità o di prestazioni che interessano una specifica istanza in un cluster Aurora. Durante la connessione a un'istanza specifica, puoi esaminare le variabili di stato, i parametri e così via. Ciò può aiutarti a determinare cosa sta succedendo di diverso per quell'istanza da ciò che accade per le altre istanze nel cluster.

Nei casi di utilizzo avanzati, puoi configurare alcune istanze database in modo diverso rispetto ad altre. In questo caso, utilizza l'endpoint dell'istanza per connetterti direttamente a un'istanza più piccola, più grande o con caratteristiche diverse rispetto alle altre. Inoltre, imposta la priorità di failover in modo che questa istanza database speciale sia l'ultima scelta da assumere come istanza primaria. In tali casi, consigliamo di utilizzare gli endpoint personalizzati anziché l'endpoint di istanza. Ciò semplifica la gestione delle connessioni e l'elevata disponibilità man mano che aggiungi altre istanze database al cluster.

Endpoint personalizzati per Amazon Aurora

L'endpoint personalizzato per un cluster Aurora rappresenta un set di istanze database selezionate. Quando ti connetti all’endpoint, Aurora esegue il bilanciamento della connessione e sceglie una delle istanze del gruppo per gestire la connessione. Puoi definire a quali istanze si riferisce questo endpoint e a quale scopo serve l'endpoint.

Un cluster database Aurora non ha endpoint personalizzati finché non ne viene creato uno. Puoi creare fino a cinque endpoint personalizzati per ogni cluster Aurora con provisioning o cluster Aurora Serverless v2. Non puoi utilizzare gli endpoint personalizzati per i cluster Aurora Serverless v1

L’endpoint personalizzato fornisce le connessioni ai database bilanciate sulla base di criteri diversi dalla capacità di sola lettura o di lettura/scrittura delle istanze database. Ad esempio, potresti definire un endpoint personalizzato per connetterti alle istanze che utilizzano una particolare classe di istanza AWS o un particolare gruppo di parametri database. Poi potresti indicare a particolari gruppi di utenti questo endpoint personalizzato. Ad esempio, potresti indirizzare gli utenti interni alle istanze a bassa capacità per la generazione di report o l'esecuzione di query ad hoc (una tantum) e indirizzare il traffico di produzione alle istanze ad alta capacità.

Dal momento che è possibile eseguire la connessione a qualsiasi istanza database associata all'endpoint personalizzato, è consigliabile assicurarsi che tutte le istanze database all'interno del gruppo condividano alcune caratteristiche simili. Ciò garantisce che le prestazioni, la capacità di memoria e altre funzionalità siano coerenti per tutti coloro che si connettono a quell'endpoint.

Questa caratteristica è progettata per gli utenti esperti con tipi specializzati di carichi di lavoro in cui non è pratico mantenere identiche tutte le repliche di Aurora nel cluster. Con gli endpoint personalizzati, puoi prevedere la capacità dell'istanza database utilizzata per ciascuna connessione. Quando usi gli endpoint personalizzati, in genere non utilizzi l'endpoint di lettura per il cluster.

L'esempio seguente mostra un endpoint personalizzato per un'istanza database in un cluster database Aurora MySQL.

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

Gli endpoint personalizzati vengono utilizzati per semplificare la gestione delle connessioni quando il cluster contiene istanze database con capacità e impostazioni di configurazione diverse.

In precedenza, potresti aver utilizzato il meccanismo CNAMES per impostare gli alias DNS (Domain Name Service) dal tuo dominio per ottenere risultati simili. Utilizzando gli endpoint personalizzati, puoi evitare di aggiornare i record CNAME quando il tuo cluster aumenta o si riduce. Gli endpoint personalizzati ti consentono anche di utilizzare connessioni Transport Layer Security/Secure Sockets Layer (TLS/SSL).

Invece di utilizzare un'istanza database per ogni scopo specifico e connetterti al relativo endpoint dell'istanza, puoi creare più gruppi di istanze database specializzate. In questo caso, ciascun gruppo ha il proprio endpoint personalizzato. In questo modo, Aurora può eseguire il bilanciamento della connessione tra tutte le istanze dedicate ad attività come la creazione di report o la gestione di query interne o di produzione. Gli endpoint personalizzati distribuiscono le connessioni tra le istanze in modo passivo, utilizzando il DNS per restituire l'indirizzo IP di una delle istanze in modo casuale. Se una delle istanze database all'interno di un gruppo diventa non disponibile, Aurora indirizza le successive connessioni dell'endpoint personalizzato a una delle altre istanze database associate allo stesso endpoint.

Argomenti

- Considerazioni sugli endpoint personalizzati di Amazon Aurora
- Creazione di un endpoint personalizzato
- Visualizzazione degli endpoint personalizzati

- Modifica di un endpoint personalizzato
- Eliminazione di un endpoint personalizzato
- Esempi dall'interfaccia AWS CLI di endpoint personalizzati per Amazon Aurora

Considerazioni sugli endpoint personalizzati di Amazon Aurora

Utilizza le seguenti sezioni per gestire gli endpoint personalizzati, nonché per specificarne le proprietà e utilizzare per essi le regole di appartenenza.

Argomenti

- Gestione degli endpoint personalizzati
- Configurazione delle proprietà degli endpoint personalizzati
- Regole di appartenenza degli endpoint personalizzati

Gestione degli endpoint personalizzati

Dal momento che i cluster Aurora appena creati non hanno endpoint personalizzati, devi creare e gestire questi oggetti autonomamente. Ciò è possibile tramite la Console di gestione AWS, AWS CLI o l'API Amazon RDS.

> **Note**
>
> Inoltre, devi creare e gestire gli endpoint personalizzati per i cluster Aurora ripristinati da snapshot. Gli endpoint personalizzati non sono inclusi nello snapshot. Li crei nuovamente dopo il ripristino e scegli nuovi nomi per gli endpoint se il cluster ripristinato si trova nella stessa regione di quello originale.

Per usare gli endpoint personalizzati dalla Console di gestione AWS, accedi alla pagina dei dettagli per il cluster Aurora e utilizza i controlli della sezione Endpoint personalizzati.

Per usare gli endpoint personalizzati dall'AWS CLI, puoi utilizzare queste operazioni:

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

- delete-db-cluster-endpoint

Per usare gli endpoint personalizzati tramite l'API Amazon RDS, puoi utilizzare le seguenti funzioni:

- CreateDBClusterEndpoint
- DescribeDBClusterEndpoints
- ModifyDBClusterEndpoint
- DeleteDBClusterEndpoint

Configurazione delle proprietà degli endpoint personalizzati

La lunghezza massima di un nome di endpoint personalizzato è 63 caratteri. Il formato del nome è il seguente:

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

Non è possibile riutilizzare lo stesso nome di endpoint personalizzato per più di un cluster nella stessa Regione AWS. L'identificatore DNS personalizzato è un identificatore univoco associato all'Account AWS in una particolare Regione AWS.

A ogni endpoint personalizzato è associato un tipo che determina quali istanze database possono essere associate all'endpoint. Attualmente, il tipo può essere `READER` o `ANY`. Le seguenti considerazioni si applicano ai tipi di endpoint personalizzato:

- Non puoi selezionare il tipo di endpoint personalizzato nella Console di gestione AWS. Tutti gli endpoint personalizzati creati tramite la Console di gestione AWS sono di tipo `ANY`.

  Puoi impostare e modificare il tipo di endpoint personalizzato utilizzando l'API AWS CLI o Amazon RDS.
- Solo le istanze database di lettura possono far parte di un endpoint personalizzato `READER`.
- Le istanze database di lettura e di scrittura possono entrambe fare parte di un endpoint personalizzato `ANY`. Aurora dirige le connessioni agli endpoint cluster con tipo `ANY` a qualsiasi istanza database associata con uguale probabilità. Il tipo `ANY` si applica ai cluster che utilizzano qualsiasi topologia di replica.
- Se tenti di creare un endpoint personalizzato con un tipo non appropriato in base alla configurazione di replica per un cluster, Aurora restituisce un errore.

Regole di appartenenza degli endpoint personalizzati

Quando aggiungi un'istanza database a un endpoint personalizzato o la si rimuove da un endpoint personalizzato, tutte le connessioni esistenti all'istanza database rimangono attive.

È possibile definire un elenco di istanze database da includere o escludere da un endpoint personalizzato. Ci riferiamo a questi elenchi come elenchi statici e di esclusioni, rispettivamente. Puoi utilizzare il meccanismo di inclusione/esclusione per suddividere ulteriormente i gruppi di istanze database e assicurarti che il set di endpoint personalizzati copra tutte le istanze database del cluster. Ogni endpoint personalizzato può contenere solo uno di questi tipi di elenchi.

Nella Console di gestione AWS:

- La scelta è rappresentata dalla casella di controllo Attach future instances added to this cluster (Collega le istanze aggiunte in futuro a questo cluster). Quando lasci deselezionata la casella di controllo, l'endpoint personalizzato utilizza un elenco statico contenente solo le istanze database specificate nella pagina. Quando selezioni la casella di controllo, l'endpoint personalizzato utilizza un elenco di esclusioni. In questo caso, l'endpoint personalizzato rappresenta tutte le istanze database nel cluster (incluse quelle che aggiungi in seguito) tranne quelle lasciate deselezionate nella pagina.
- La console non consente di specificare il tipo di endpoint. Qualsiasi endpoint personalizzato creato utilizzando la console è di tipo `ANY`.

  Pertanto, Aurora non modifica l'appartenenza all'endpoint personalizzato quando le istanze database cambiano ruolo tra istanza di scrittura e istanza di lettura a causa di un failover o di una promozione.

In AWS CLI e nell'API Amazon RDS:

- Puoi specificare il tipo di endpoint. Pertanto, quando il tipo di endpoint è impostato su `READER`, l'appartenenza dell'endpoint viene adeguata automaticamente durante i failover e le promozioni.

  Ad esempio, un endpoint personalizzato di tipo `READER` include una replica Aurora che poi viene promossa a istanza database di scrittura. La nuova istanza di scrittura non fa più parte dell'endpoint personalizzato.
- È possibile aggiungere singoli membri e rimuoverli dagli elenchi dopo che hanno cambiato ruolo. Utilizza il comando AWS CLI modify-db-cluster-endpoint o l'operazione API ModifyDBClusterEndpoint.

Puoi associare un'istanza database a più di un endpoint personalizzato. Ad esempio, supponiamo che aggiungi una nuova istanza database a un cluster o che Aurora aggiunga automaticamente un'istanza database tramite il meccanismo di Auto Scaling. In questi casi, l'istanza database viene aggiunta a tutti gli endpoint personalizzati per i quali è idonea. A quali endpoint viene aggiunta l'istanza database dipende dal tipo di endpoint personalizzato, `READER` o `ANY`, oltre che da eventuali elenchi statici o di esclusioni definiti per ciascun endpoint. Ad esempio, se l'endpoint include un elenco statico di istanze database, le repliche di Aurora aggiunte non vengono inserite nell'endpoint. Al contrario, se l'endpoint ha un elenco di esclusioni, le repliche di Aurora aggiunte vengono inserite nell'endpoint, se non sono citate nell'elenco di esclusioni e i ruoli corrispondono al tipo dell'endpoint personalizzato.

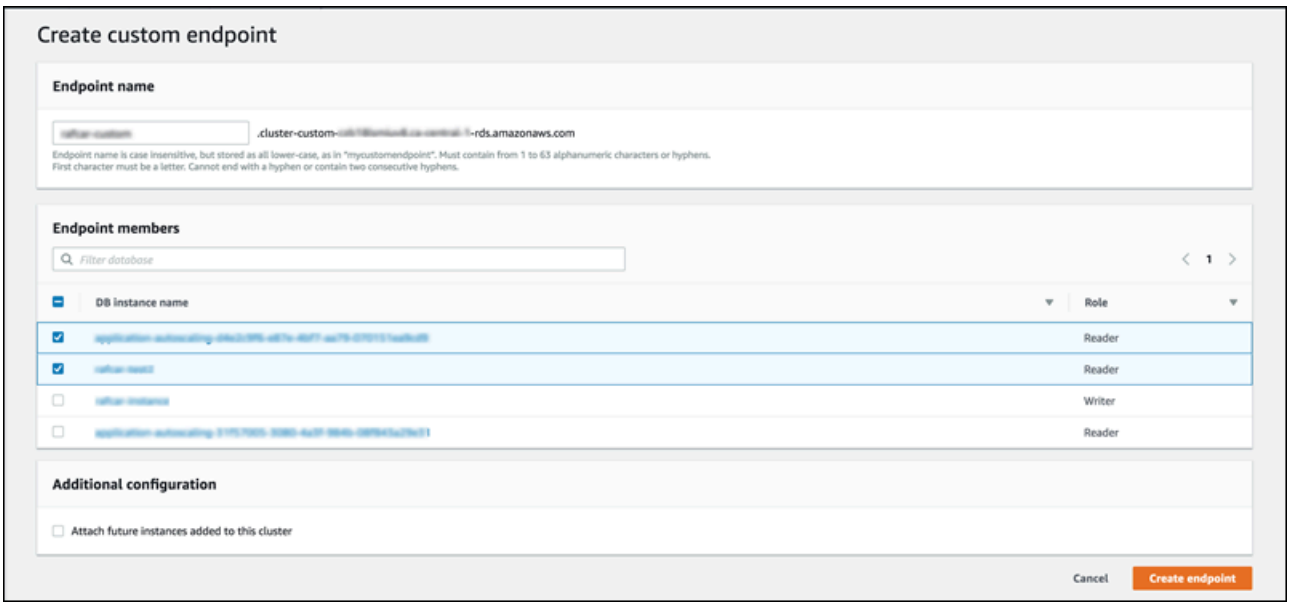
Se una replica di Aurora diventa non disponibile, rimane associata agli endpoint personalizzati. Ad esempio, continua a far parte dell'endpoint personalizzato quando non è integra, è stata interrotta, riavviata e così via. Tuttavia, non puoi connetterti alla replica tramite questi endpoint finché non diventa nuovamente disponibile.

Creazione di un endpoint personalizzato

Crea un endpoint personalizzato utilizzando la Console di gestione AWS, AWS CLI o l'API Amazon RDS.

Console

Per creare un endpoint personalizzato con la Console di gestione AWS, vai alla pagina dei dettagli del cluster e scegli l'operazione `Create custom endpoint` nella sezione Endpoint. Scegli un nome per l'endpoint personalizzato, univoco per l'ID utente e la regione. Per selezionare un elenco di istanze database che rimane invariato anche quando il cluster si espande, lascia deselezionata la casella di controllo Attach future instances added to this cluster (Collega le istanze che in futuro saranno aggiunte a questo cluster). Quando selezioni questa casella di controllo, l'endpoint personalizzato aggiunge dinamicamente nuove istanze man mano che le aggiungi al cluster.



Non puoi selezionare il tipo di endpoint personalizzato nella Console di gestione AWS. Tutti gli endpoint personalizzati creati tramite la Console di gestione AWS sono di tipo ANY.

AWS CLI

Per creare un endpoint personalizzato con l'AWS CLI, esegui il comando create-db-cluster-endpoint.

Il seguente comando crea un endpoint personalizzato collegato a un cluster specifico. Inizialmente, l'endpoint è associato a tutte le istanze di replica di Aurora nel cluster. Un comando successivo lo associa a un set specifico di istanze database del cluster.

Per Linux, macOS o Unix:

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

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

Per Windows:

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

```
  --endpoint-type reader ^
  --db-cluster-identifier cluster_id

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

API RDS

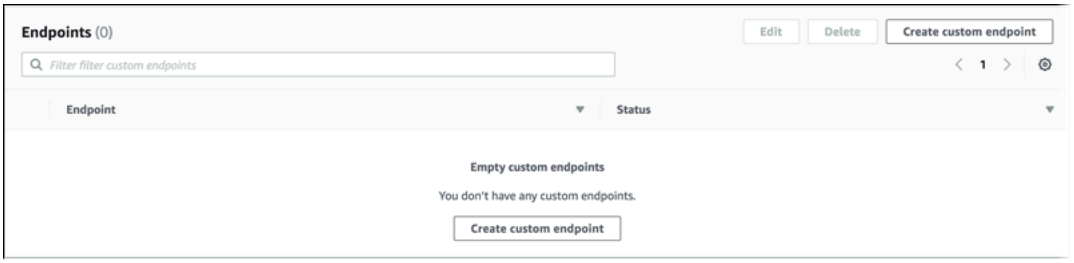
Per creare un endpoint personalizzato con l'API RDS, esegui l'operazione CreateDBClusterEndpoint.

Visualizzazione degli endpoint personalizzati

Console

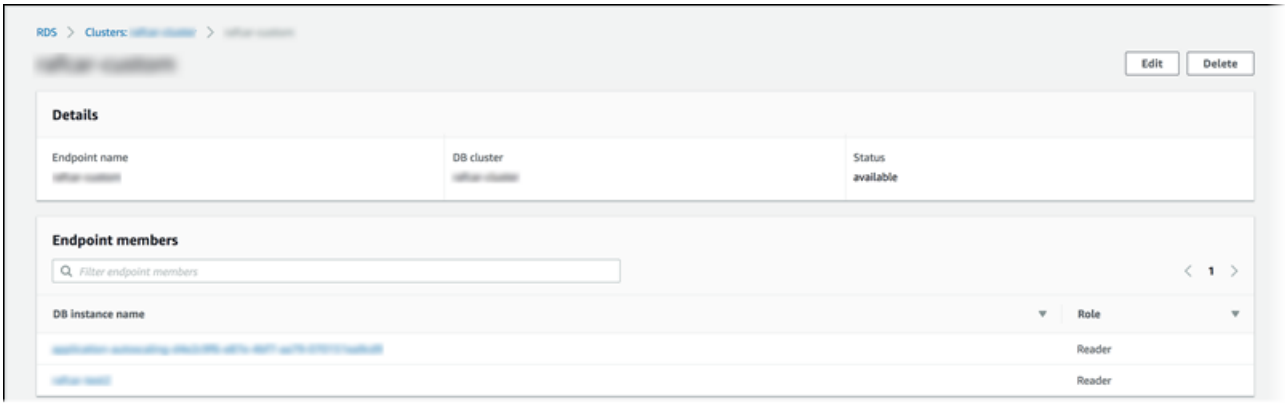
Per visualizzare gli endpoint personalizzati con la Console di gestione AWS, vai alla pagina dei dettagli del cluster per il cluster e guarda la sezione Endpoint. Questa sezione contiene solo informazioni relative agli endpoint personalizzati. I dettagli per gli endpoint integrati sono elencati nella sezione Dettagli principale. Per vedere i dettagli di uno specifico endpoint personalizzato, selezionane il nome per visualizzare la relativa pagina dei dettagli.

La seguente schermata mostra l'elenco degli endpoint personalizzati per un cluster Aurora che è inizialmente vuoto.



Una volta creati gli endpoint personalizzati per il cluster, vengono visualizzati nella sezione Endpoint.

Facendo clic sulla pagina dei dettagli vengono mostrate le istanze database a cui è attualmente associato l'endpoint.



Per visualizzare altri dettagli sulle nuove istanze database aggiunte al cluster che vengono aggiunte automaticamente anche all'endpoint, apri la pagina Edit (Modifica) per l'endpoint.

AWS CLI

Per visualizzare gli endpoint personalizzati con l'AWS CLI, esegui il comando describe-db-cluster-endpoints.

Il seguente comando mostra gli endpoint personalizzati associati a un cluster specificato in una regione specifica. L'output include gli endpoint integrati e tutti gli endpoint personalizzati.

Per Linux, macOS o Unix:

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

Per Windows:

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

Di seguito è illustrato un esempio di output di un comando `describe-db-cluster-endpoints`. `EndpointType` di `WRITER` o `READER` indica gli endpoint integrati di lettura-scrittura e di sola lettura per il cluster. L'`EndpointType` `CUSTOM` indica gli endpoint creati e scegli le istanze database associate. Il campo `StaticMembers` di uno degli endpoint non è vuoto, a indicare che è associato a un set preciso di istanze database. Il campo `ExcludedMembers` dell'altro endpoint non è vuoto, a indicare che l'endpoint è associato a tutte le istanze database che non sono elencate in `ExcludedMembers`. Questo secondo tipo di endpoint personalizzato aumenta per includere le nuove istanze man mano che le aggiungi al cluster.

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

```
      "custom-endpoint-demo-05",
      "custom-endpoint-demo-03",
      "custom-endpoint-demo-06",
      "custom-endpoint-demo-01"
    ],
    "DBClusterIdentifier": "custom-endpoint-demo",
    "Status": "available",
    "EndpointType": "CUSTOM",
    "Endpoint": "eight-and-higher.cluster-custom-123456789012.ca-
central-1.rds.amazonaws.com",
    "StaticMembers": [],
    "DBClusterEndpointResourceIdentifier": "cluster-endpoint-
W7PE3TLLFNSHYQKFU6J6NV5FHU",
    "DBClusterEndpointArn": "arn:aws:rds:ca-central-1:111122223333:cluster-
endpoint:eight-and-higher"
  }
 ]
}
```

API RDS

Per visualizzare gli endpoint personalizzati con l'API RDS, esegui l'operazione DescribeDBClusterEndpoints.html.

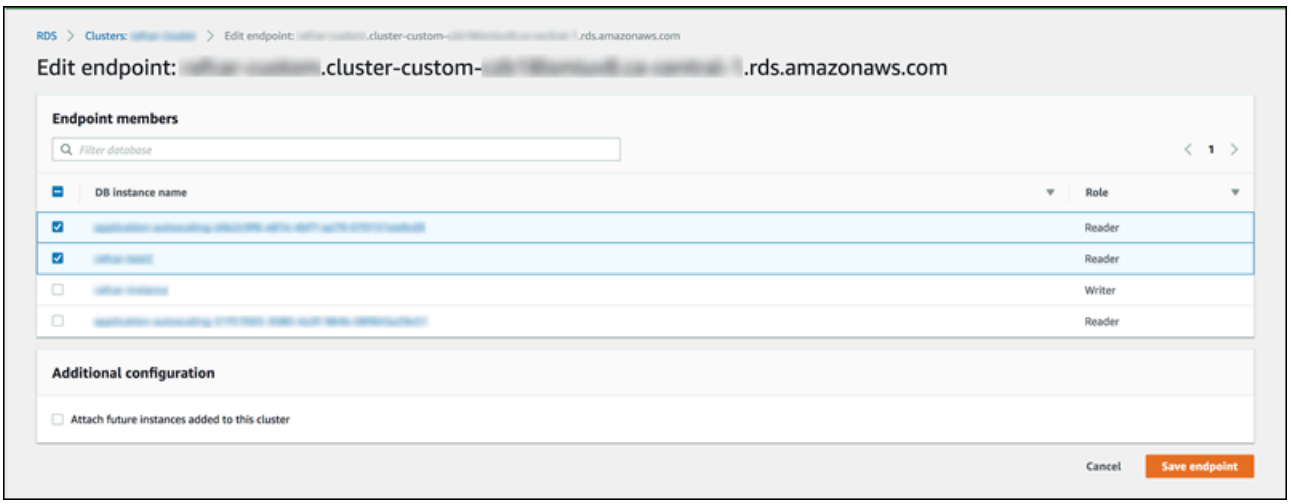
Modifica di un endpoint personalizzato

Puoi modificare le proprietà di un endpoint personalizzato per cambiare le istanze database associate all'endpoint. Puoi anche modificare un endpoint tra un elenco statico e un elenco di esclusioni. Per maggiori dettagli su queste proprietà dell'endpoint, vedi Regole di appartenenza degli endpoint personalizzati.

Puoi continuare a connetterti e utilizzare un endpoint personalizzato mentre sono in corso le modifiche di un'operazione di modifica.

Console

Per modificare un endpoint personalizzato con la Console di gestione AWS, puoi selezionare l'endpoint nella pagina dei dettagli del cluster o visualizzare la pagina dei dettagli per l'endpoint e scegliere l'operazione Modifica.



AWS CLI

Per modificare un endpoint personalizzato con l'AWS CLI, esegui il comando modify-db-cluster-endpoint.

I seguenti comandi modificano il set di istanze database che si applicano a un endpoint personalizzato e opzionalmente alterna il comportamento di elenco statico e di esclusione. I parametri `--static-members` e `--excluded-members` usano un elenco separato da spazi di identificatori istanze database.

Per Linux, macOS o Unix:

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

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

Per Windows:

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

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

API RDS

Per modificare un endpoint personalizzato con l'API RDS, esegui l'operazione ModifyDBClusterEndpoint.html.

Eliminazione di un endpoint personalizzato

Elimina un endpoint personalizzato utilizzando la Console di gestione AWS, la AWS CLI o l'API Amazon RDS.

Console

Per eliminare un endpoint personalizzato con la Console di gestione AWS, vai alla pagina dei dettagli del cluster e seleziona l'endpoint personalizzato appropriato, quindi scegli l'operazione Elimina.

AWS CLI

Per eliminare un endpoint personalizzato con la AWS CLI, esegui il comando delete-db-cluster-endpoint.

Il seguente comando elimina un endpoint personalizzato. Non è necessario specificare il cluster associato, ma devi specificare la regione.

Per Linux, macOS o Unix:

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

Per Windows:

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

API RDS

Per eliminare un endpoint personalizzato con l'API RDS, esegui l'operazione DeleteDBClusterEndpoint.

Esempi dall'interfaccia AWS CLI di endpoint personalizzati per Amazon Aurora

Il seguente tutorial usa gli esempi di AWS CLI con la sintassi della shell Unix per mostrare come potresti definire un cluster con diverse istanze database di piccole dimensioni e alcune istanze database di grandi dimensioni, e creare gli endpoint personalizzati per la connessione a ogni set di istanze database. Per eseguire comandi simili sul tuo sistema, devi avere familiarità con le nozioni di base dell'uso dei cluster Aurora e della AWS CLI per fornire i valori dei parametri come regione, gruppo di sottoreti e gruppo di sicurezza VPC.

Questo esempio dimostra le fasi della configurazione iniziale: la creazione di un cluster Aurora e l'aggiunta delle istanze database. Questo è un cluster eterogeneo, il che significa che non tutte le istanze database hanno la stessa capacità. La maggior parte delle istanze utilizza la classe di istanza AWS `db.r4.4xlarge`, ma le ultime due istanze database utilizzano `db.r4.16xlarge`. Ognuno di questi comandi `create-db-instance` di esempio invia l'output sullo schermo e salva una copia del JSON in un file per un'ispezione successiva.

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

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

```
    --region $REGION \
    | tee custom-endpoint-demo-${i}.json
done

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

Le istanze più grandi sono riservate a tipi specializzati di query per la creazione di report. Per rendere improbabile che vengano promosse all'istanza primaria, l'esempio seguente imposta il livello di promozione sulla priorità più bassa. In questo esempio è specificata l'opzione `--manage-master-user-password` per generare la password dell'utente master e gestirla in Secrets Manager. Per ulteriori informazioni, consulta Gestione delle password con Amazon Aurora e Gestione dei segreti AWS. In alternativa, puoi utilizzare l'opzione `--master-password` per specificare e gestire personalmente la password.

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

Supponiamo di voler utilizzare le due istanze "più grandi" solo per le query che richiedono più risorse. A tale scopo, puoi innanzitutto creare un endpoint di sola lettura personalizzato. Quindi puoi aggiungere un elenco statico di membri in modo che l'endpoint si connetta solo a tali istanze database. Queste istanze sono già nel livello di promozione più basso, rendendo improbabile che una di esse venga promossa a istanza primaria. Se una di esse venisse promossa a istanza primaria, diventerebbe irraggiungibile tramite questo endpoint perché abbiamo specificato il tipo `READER` anziché `ANY`.

Nell'esempio seguente viene illustrato il comando di creazione e modifica di endpoint e l'output JSON di esempio che mostra lo stato iniziale e modificato dell'endpoint personalizzato.

```
$ aws rds create-db-cluster-endpoint --region $REGION \
    --db-cluster-identifier custom-endpoint-demo \
```

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

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

L'endpoint READER predefinito per il cluster può connettersi alle istanze database piccole o grandi, rendendo poco pratico prevedere le prestazioni e la scalabilità delle query quando il cluster diventa occupato. Per dividere in modo pulito il carico di lavoro tra i gruppi di istanze database, puoi ignorare l'endpoint READER predefinito e creare un secondo endpoint personalizzato che si connette a tutte le altre istanze database. L'esempio seguente effettua ciò creando un endpoint personalizzato e

quindi aggiungendo un elenco di esclusioni. Qualsiasi altra istanza database aggiunta al cluster successivamente verrà inserita automaticamente in questo endpoint. Il tipo ANY indica che questo endpoint è associato a otto istanze in totale: l'istanza primaria e altre sette repliche di Aurora. Se l'esempio avesse specificato il tipo READER, l'endpoint personalizzato sarebbe associato solo alle sette repliche di Aurora.

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

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

```
    "Status": "modifying"
}
```

L'esempio seguente controlla lo stato degli endpoint per questo cluster. Il cluster ha ancora il suo endpoint del cluster originale, con `EndPointType` `WRITER`, che sarà comunque utilizzato per l'amministrazione, l'ETL e altre operazioni di scrittura. Ha ancora il suo endpoint `READER` originale, che non usi perché ogni connessione verrebbe diretta a una istanza database piccola o grande. Gli endpoint personalizzati rendono questo comportamento prevedibile, con connessioni garantite per l'utilizzo di una delle istanze database piccole o grandi in base all'endpoint specificato.

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

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

Gli esempi finali dimostrano come le successive connessioni del database agli endpoint personalizzati avvengono alle varie istanze database nel cluster Aurora. L'endpoint `small-instances` si connette sempre alle istanze database `db.r4.4xlarge` che sono gli host con un numero inferiore in questo cluster.

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

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

```
+--------------------------+
| custom-endpoint-demo-07 |
+--------------------------+

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

L'endpoint `big-instances` si connette sempre alle istanze database `db.r4.16xlarge` che sono i due host con il numero più alto in questo cluster.

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

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

Classi di istanze database Amazon Aurora

La classe di istanza database determina la capacità di calcolo e memoria di un'istanza database Amazon Aurora. La classe di istanza database di cui hai bisogno dipende dalla potenza di elaborazione e dai requisiti di memoria specifici.

Una classe di istanza database è costituita dal tipo di classe di istanza database e dalla dimensione. Ad esempio, db.r6g è un tipo di classe di istanza database con ottimizzazione della memoria con processori AWS Graviton2. Nel tipo di classe di istanza db.r6g, db.r6g.2xlarge è una classe di istanza database. La dimensione di questa classe è 2xlarge.

Per ulteriori informazioni sui prezzi delle classi di istanza, consulta Prezzi di Amazon RDS.

Per ulteriori informazioni sui tipi di classi di istanza database, sui motori di database supportati, sulle Regioni AWS supportate, o sulle specifiche hardware delle classi di istanza database, consulta le seguenti sezioni.

Argomenti

- Tipi di classi di istanza database
- Motori di database supportati per classi di istanza database
- Determinazione del supporto delle classi di istanza database nelle Regioni AWS
- Specifiche hardware per le classi di istanza database per Aurora

Tipi di classi di istanza database

Amazon Aurora supporta le classi di istanza database per i seguenti casi d'uso:

- Aurora Serverless v2
- Ottimizzato per la memoria
- Prestazioni espandibili
- Letture ottimizzate

Per ulteriori informazioni sui tipi di EC2 istanze Amazon, consulta Tipi di istanza nella EC2 documentazione di Amazon.

Tipo di classe di istanza Aurora Serverless v2

È disponibile il tipo Aurora Serverless v2 seguente:

- db.serverless: un tipo di classe di istanza database speciale utilizzato da Aurora Serverless v2. Aurora calibra dinamicamente le risorse di calcolo, memoria e di rete man mano che il carico di lavoro cambia. Per informazioni di utilizzo dettagliate, consulta Uso di Aurora Serverless v2.

Tipi di classe di istanza ottimizzata per la memoria

La famiglia X ottimizzata per la memoria supporta le seguenti classi di istanza:

- db.x2g — Classi di istanze ottimizzate per applicazioni a uso intensivo di memoria e alimentate da processori Graviton2. AWS Queste classi di istanza offrono un basso costo per GiB di memoria.

  È possibile modificare un'istanza DB per utilizzare una delle classi di istanze DB alimentate dai processori Graviton2. AWS Per farlo, esegui gli stessi passaggi utilizzati per qualsiasi altra modifica dell'istanza database.

La famiglia R ottimizzata per la memoria supporta i seguenti tipi di classi di istanza:

- db.r8g: classi di istanza con tecnologia basata su processori AWS Graviton4. Queste classi di istanza sono ideali per l'esecuzione di carichi di lavoro a uso intensivo di memoria in . Queste istanze offrono istanze di dimensioni maggiori con fino a 3 volte più v CPUs e memoria rispetto alle istanze db.r7g di settima generazione AWS basate su Graviton3. Sono alimentate dal sistema Nitro, una combinazione di hardware dedicato e hypervisor leggero. AWS
- È possibile modificare un'istanza DB per utilizzare una delle classi di istanze DB basate sui processori Graviton4. AWS Per farlo, esegui gli stessi passaggi utilizzati per qualsiasi altra modifica dell'istanza database.
- db.r7g — Classi di istanze basate su processori Graviton3. AWS Queste classi di istanza sono ideali per l'esecuzione di carichi di lavoro a uso intensivo di memoria in .

  È possibile modificare un'istanza DB per utilizzare una delle classi di istanze DB alimentate dai processori Graviton3. AWS Per farlo, esegui gli stessi passaggi utilizzati per qualsiasi altra modifica dell'istanza database. Sono alimentati dal sistema AWS Nitro, una combinazione di hardware dedicato e hypervisor leggero.

- db.r7i: classi di istanza con tecnologia basata su processori Intel Xeon scalabili di quarta generazione. Queste classi di istanza sono certificate SAP e ideali per l'esecuzione di carichi di lavoro a uso intensivo di memoria. È possibile modificare un'istanza database per utilizzare una delle classi di istanza database con processori Intel Xeon scalabili di quarta generazione. Per farlo, esegui gli stessi passaggi utilizzati per qualsiasi altra modifica dell'istanza database. Sono alimentati dal sistema AWS Nitro, una combinazione di hardware dedicato e hypervisor leggero.
- db.r6g: classi di istanza basate su processori AWS Graviton2. Queste classi di istanza sono ideali per l'esecuzione di carichi di lavoro a uso intensivo di memoria in locale ad alta velocità e bassa latenza. Sono alimentati dal sistema AWS Nitro, una combinazione di hardware dedicato e hypervisor leggero.
- È possibile modificare un'istanza DB per utilizzare una delle classi di istanze DB basate sui processori Graviton2. AWS Per farlo, esegui gli stessi passaggi utilizzati per qualsiasi altra modifica dell'istanza database.
- db.r6i: classi di istanza ottimizzate basate su processori Intel Xeon scalabili di terza generazione Queste classi di istanza sono certificate SAP e ideali per l'esecuzione di carichi di lavoro a uso intensivo di memoria .
- db.r4: classi di istanze ottimizzate per applicazioni a uso intensivo della memoria. Queste classi di istanza offrono reti migliorate e prestazioni Amazon Elastic Block Store (Amazon EBS). Sono alimentate dal sistema AWS Nitro, una combinazione di hardware dedicato e hypervisor leggero.
- db.r4: queste classi di istanza sono supportate solo per Aurora MySQL 2.x e per le versioni 11 e 12 di Aurora PostgreSQL. Per tutti i cluster di database Aurora che utilizzano classi di istanza database db.r4, è consigliabile eseguire l'aggiornamento a una classe di istanza database di generazione superiore non appena possibile.

  Le classi di istanza db.r4 non sono disponibili per la configurazione dell'archiviazione del cluster Aurora I/O-Optimized.

Tipi di classe di istanza a prestazioni espandibili

Sono disponibili i tipi di classe di istanza database a prestazioni espandibili seguenti:

- db.t4g — Classi di istanze generiche basate su processori Graviton2 basati su ARM. AWS Queste classi di istanza offrono prestazioni di prezzo migliori rispetto alle classi di istanza database con prestazioni espandibili della generazione precedente per un ampio set di carichi di lavoro espandibili. Le istanze Amazon RDS di tipo db.t4g sono configurate per la modalità illimitata.

Questo significa che possono espandersi oltre la linea di base in una finestra di 24 ore a un costo aggiuntivo.

È possibile modificare un'istanza DB per utilizzare una delle classi di istanze DB alimentate dai processori Graviton2. AWS Per farlo, esegui gli stessi passaggi utilizzati per qualsiasi altra modifica dell'istanza database.

- db.t3: classi di istanza che forniscono un livello di prestazioni di base, con la possibilità di burst per un utilizzo completo della CPU. Le istanze di tipo db.t3 sono configurate per la modalità illimitata. Queste classi di istanza forniscono più capacità di calcolo rispetto alle classi di istanza db.t2 precedenti. Sono basate sul nuovo sistema AWS Nitro, una combinazione di hardware dedicato e hypervisor leggeri. Consigliamo di usare queste classi di istanza solo per i server di sviluppo e test o altri server non di produzione.
- db.t2: classi di istanze che forniscono un livello di prestazioni di base, con la possibilità di burst per un utilizzo completo della CPU. Le istanze di tipo db.t2 sono configurate per la modalità illimitata. Consigliamo di usare queste classi di istanza solo per i server di sviluppo e test o altri server non di produzione.

  Le classi di istanza db.t2 non sono disponibili per la configurazione dell'archiviazione del cluster Aurora I/O-Optimized.

> **Note**
>
> Consigliamo di utilizzare le classi di istanza database T solo per i server di sviluppo e test o altri server non di produzione. Per informazioni dettagliate sulle classi di istanza T, consulta Utilizzo delle classi di istanza T per lo sviluppo e i test.

Per le specifiche dell'hardware della classe di istanza database, consultare Specifiche hardware per le classi di istanza database per Aurora.

Tipi di classe di istanza per letture ottimizzate

I tipi di classe di istanza per letture ottimizzate disponibili sono i seguenti:

- db.r8gd: classi di istanze basate su processori Graviton4. Queste classi di istanze sono ideali per eseguire carichi di lavoro che richiedono molta memoria e offrono storage SSD a livello di blocco NVMe su base locale per applicazioni che richiedono storage locale ad alta velocità e bassa

latenza. Offrono una memoria massima di 1,5 TiB e fino a 11,4 TB di storage SSD basato su collegamento diretto NVMe.

- db.r6gd — Classi di istanze basate su processori Graviton2. AWS Queste classi di istanze sono ideali per eseguire carichi di lavoro che richiedono molta memoria e offrono storage SSD a livello di blocco NVMe su base locale per applicazioni che richiedono storage locale ad alta velocità e bassa latenza.
- db.r6id: classi di istanza ottimizzate basate su processori Intel Xeon scalabili di terza generazione. Queste classi di istanza sono certificate SAP e ideali per l'esecuzione di carichi di lavoro con elevati requisiti di memoria. Offrono una memoria massima di 1 TiB e fino a 7,6 TB di storage SSD basato su collegamento diretto NVMe.

Motori di database supportati per classi di istanza database

Le tabelle seguenti mostrano le classi di istanza database supportate per i motori di database Amazon Aurora.

db.serverless: classe di istanza Aurora Serverless v2 con dimensionamento automatico della capacità

Classe di istanza	Aurora MySQL	Aurora PostgreSQL
db.serverless	Per informazioni, consulta Regioni e motori di database Aurora supportati per Aurora Serverless v2.	Per informazioni, consulta Regioni e motori di database Aurora supportati per Aurora Serverless v2.

db.x2g — classi di istanze ottimizzate per la memoria alimentate da processori Graviton2 AWS

Classe di istanza	Aurora MySQL	Aurora PostgreSQL
db.x2g.16xlarge	Tutte le versioni attualmente disponibili	17.4 e versioni successive, 16.1 e versioni successive, 15.2 e versioni successive, 14.3 e versioni successive, 13.3 e versioni successive, 12.8 e versioni

Classe di istanza	Aurora MySQL	Aurora PostgreSQL
		successive, 11.9, 11.12 e versioni successive
db.x2g.12xlarge	Tutte le versioni attualmente disponibili	17.4 e versioni successive, 16.1 e versioni successive, 15.2 e versioni successive, 14.3 e versioni successive, 13.3 e versioni successive, 12.8 e versioni successive, 11.9, 11.12 e versioni successive
db.x2g.8xlarge	Tutte le versioni attualmente disponibili	17.4 e versioni successive, 16.1 e versioni successive, 15.2 e versioni successive, 14.3 e versioni successive, 13.3 e versioni successive, 12.8 e versioni successive, 11.9, 11.12 e versioni successive
db.x2g.4xlarge	Tutte le versioni attualmente disponibili	17.4 e versioni successive, 16.1 e versioni successive, 15.2 e versioni successive, 14.3 e versioni successive, 13.3 e versioni successive, 12.8 e versioni successive, 11.9, 11.12 e versioni successive
db.x2g.2xlarge	Tutte le versioni attualmente disponibili	17.4 e versioni successive, 16.1 e versioni successive, 15.2 e versioni successive, 14.3 e versioni successive, 13.3 e versioni successive, 12.8 e versioni successive, 11.9, 11.12 e versioni successive

Classe di istanza	Aurora MySQL	Aurora PostgreSQL
db.x2g.xlarge	Tutte le versioni attualmente disponibili	17.4 e versioni successive, 16.1 e versioni successive, 15.2 e versioni successive, 14.3 e versioni successive, 13.3 e versioni successive, 12.8 e versioni successive, 11.9, 11.12 e versioni successive
db.x2g.large	Tutte le versioni attualmente disponibili	17.4 e versioni successive, 16.1 e versioni successive, 15.2 e versioni successive, 14.3 e versioni successive, 13.3 e versioni successive, 12.8 e versioni successive, 11.9, 11.12 e versioni successive

db.r6gd — Classi di istanze Optimized Reads basate su processori Graviton2 AWS

Classe di istanza	Aurora MySQL	Aurora PostgreSQL
db.r6gd.16xlarge	No	17.4 e versioni successive, 16.1 e versioni successive, 15.4 e versioni successive, 14.9 e versioni successive
db.r6gd.12xlarge	No	17.4 e versioni successive, 16.1 e versioni successive, 15.4 e versioni successive, 14.9 e versioni successive
db.r6gd.8xlarge	No	17.4 e versioni successive, 16.1 e versioni successive, 15.4 e versioni successive, 14.9 e versioni successive
db.r6gd.4xlarge	No	17.4 e versioni successive, 16.1 e versioni successive, 15.4 e versioni successive, 14.9 e versioni successive

Classe di istanza	Aurora MySQL	Aurora PostgreSQL
db.r6gd.2xlarge	No	17.4 e versioni successive, 16.1 e versioni successive, 15.4 e versioni successive, 14.9 e versioni successive
db.r6gd.xlarge	No	17.4 e versioni successive, 16.1 e versioni successive, 15.4 e versioni successive, 14.9 e versioni successive

db.r6i: classi di istanza letture ottimizzate

Classe di istanza	Aurora MySQL	Aurora PostgreSQL
db.r6id.32xlarge	No	17.4 e versioni successive, 16.1 e versioni successive, 15.4 e versioni successive, 14.9 e versioni successive
db.r6id.24xlarge	No	17.4 e versioni successive, 16.1 e versioni successive, 15.4 e versioni successive, 14.9 e versioni successive

db.r8g — classi di istanze ottimizzate per la memoria alimentate da processori Graviton4 AWS

Classe di istanza	Aurora MySQL	Aurora PostgreSQL
db.r8g.48xlarge	3.08.0 e versioni successive	17.4 e versioni successive, 16.3 e versioni successive, 15.7 e versioni successive, 14.12 e versioni successive, 13.15 e versioni successive
db.r8g.24xlarge	3.08.0 e versioni successive	17.4 e versioni successive, 16.3 e versioni successive, 15.7 e versioni successive, 14.12 e

Classe di istanza	Aurora MySQL	Aurora PostgreSQL
		versioni successive, 13.15 e versioni successive
db.r8g.16xlarge	3.08.0 e versioni successive	17.4 e versioni successive, 16.3 e versioni successive, 15.7 e versioni successive, 14.12 e versioni successive, 13.15 e versioni successive
db.r8g.12xlarge	3.08.0 e versioni successive	17.4 e versioni successive, 16.3 e versioni successive, 15.7 e versioni successive, 14.12 e versioni successive, 13.15 e versioni successive
db.r8g.8xlarge	3.08.0 e versioni successive	17.4 e versioni successive, 16.3 e versioni successive, 15.7 e versioni successive, 14.12 e versioni successive, 13.15 e versioni successive
db.r8g.4xlarge	3.08.0 e versioni successive	17.4 e versioni successive, 16.3 e versioni successive, 15.7 e versioni successive, 14.12 e versioni successive, 13.15 e versioni successive
db.r8g.2xlarge	3.08.0 e versioni successive	17.4 e versioni successive, 16.3 e versioni successive, 15.7 e versioni successive, 14.12 e versioni successive, 13.15 e versioni successive

Classe di istanza	Aurora MySQL	Aurora PostgreSQL
db.r8g.xlarge	3.08.0 e versioni successive	17.4 e versioni successive, 16.3 e versioni successive, 15.7 e versioni successive, 14.12 e versioni successive, 13.15 e versioni successive
db.r8g.large	3.08.0 e versioni successive	17.4 e versioni successive, 16.3 e versioni successive, 15.7 e versioni successive, 14.12 e versioni successive, 13.15 e versioni successive

db.r8gd — classi di istanze di lettura ottimizzate basate su processori Graviton4 AWS

Classe di istanza	Aurora MySQL	Aurora PostgreSQL
db.r8gd.48xlarge	No	17.4 e versioni successive, 16.3 e versioni successive, 15.7 e versioni successive, 14.12 e versioni successive
db.r8gd.24xlarge	No	17.4 e versioni successive, 16.3 e versioni successive, 15.7 e versioni successive, 14.12 e versioni successive
db.r8gd.16xlarge	No	17.4 e versioni successive, 16.3 e versioni successive, 15.7 e versioni successive, 14.12 e versioni successive
db.r8gd.12xlarge	No	17.4 e versioni successive, 16.3 e versioni successive, 15.7 e versioni successive, 14.12 e versioni successive

Classe di istanza	Aurora MySQL	Aurora PostgreSQL
db.r8gd.8xlarge	No	17.4 e versioni successive, 16.3 e versioni successive, 15.7 e versioni successive, 14.12 e versioni successive
db.r8gd.4xlarge	No	17.4 e versioni successive, 16.3 e versioni successive, 15.7 e versioni successive, 14.12 e versioni successive
db.r8gd.2xlarge	No	17.4 e versioni successive, 16.3 e versioni successive, 15.7 e versioni successive, 14.12 e versioni successive
db.r8g.xlarge	No	17.4 e versioni successive, 16.3 e versioni successive, 15.7 e versioni successive, 14.12 e versioni successive

db.r7g — classi di istanze ottimizzate per la memoria alimentate da processori Graviton3 AWS

Classe di istanza	Aurora MySQL	Aurora PostgreSQL
db.r7g.16xlarge	2.12.0 e versioni successive, 3.03.1 e versioni successive	17.4 e versioni successive, 16.1 e versioni successive, 15.2 e versioni successive, 14.7 e versioni successive, 13.10 e versioni successive
db.r7g.12xlarge	2.12.0 e versioni successive, 3.03.1 e versioni successive	17.4 e versioni successive, 16.1 e versioni successive, 15.2 e versioni successive, 14.7 e versioni successive, 13.10 e versioni successive

Classe di istanza	Aurora MySQL	Aurora PostgreSQL
db.r7g.8xlarge	2.12.0 e versioni successive, 3.03.1 e versioni successive	17.4 e versioni successive, 16.1 e versioni successive, 15.2 e versioni successive, 14.7 e versioni successive, 13.10 e versioni successive
db.r7g.4xlarge	2.12.0 e versioni successive, 3.03.1 e versioni successive	17.4 e versioni successive, 16.1 e versioni successive, 15.2 e versioni successive, 14.7 e versioni successive, 13.10 e versioni successive
db.r7g.2xlarge	2.12.0 e versioni successive, 3.03.1 e versioni successive	17.4 e versioni successive, 16.1 e versioni successive, 15.2 e versioni successive, 14.7 e versioni successive, 13.10 e versioni successive
db.r7g.xlarge	2.12.0 e versioni successive, 3.03.1 e versioni successive	17.4 e versioni successive, 16.1 e versioni successive, 15.2 e versioni successive, 14.7 e versioni successive, 13.10 e versioni successive
db.r7g.large	2.12.0 e versioni successive, 3.03.1 e versioni successive	17.4 e versioni successive, 16.1 e versioni successive, 15.2 e versioni successive, 14.7 e versioni successive, 13.10 e versioni successive

db.r7i: classi di istanze ottimizzate per la memoria

Classe di istanza	Aurora MySQL	Aurora PostgreSQL
db.r7i.48xlarge	3.08.0 e versioni successive	17.4 e versioni successive, 16.3 e versioni successive, 15.7 e versioni successive, 14.12 e versioni successive, 13.15 e versioni successive
db.r7i.24xlarge	3.08.0 e versioni successive	17.4 e versioni successive, 16.3 e versioni successive, 15.7 e versioni successive, 14.12 e versioni successive, 13.15 e versioni successive
db.r7i.16xlarge	3.08.0 e versioni successive	17.4 e versioni successive, 16.3 e versioni successive, 15.7 e versioni successive, 14.12 e versioni successive, 13.15 e versioni successive
db.r7i.12xlarge	3.08.0 e versioni successive	17.4 e versioni successive, 16.3 e versioni successive, 15.7 e versioni successive, 14.12 e versioni successive, 13.15 e versioni successive
db.r7i.8xlarge	3.08.0 e versioni successive	17.4 e versioni successive, 16.3 e versioni successive, 15.7 e versioni successive, 14.12 e versioni successive, 13.15 e versioni successive
db.r7i.4xlarge	3.08.0 e versioni successive	17.4 e versioni successive, 16.3 e versioni successive, 15.7 e versioni successive, 14.12

Classe di istanza	Aurora MySQL	Aurora PostgreSQL
		e versioni successive, 13.15 e versioni successive
db.r7i.2xlarge	3.08.0 e versioni successive	17.4 e versioni successive, 16.3 e versioni successive, 15.7 e versioni successive, 14.12 e versioni successive, 13.15 e versioni successive
db.r7i.xlarge	3.08.0 e versioni successive	17.4 e versioni successive, 16.3 e versioni successive, 15.7 e versioni successive, 14.12 e versioni successive, 13.15 e versioni successive
db.r7i.large	3.08.0 e versioni successive	17.4 e versioni successive, 16.3 e versioni successive, 15.7 e versioni successive, 14.12 e versioni successive, 13.15 e versioni successive

db.r6g — classi di istanze ottimizzate per la memoria alimentate da processori Graviton2 AWS

Classe di istanza	Aurora MySQL	Aurora PostgreSQL
db.r6g.16xlarge	Tutte le versioni attualmente disponibili	17.4 e versioni successive, 16.1 e versioni successive, 15.2 e versioni successive, 14.3 e versioni successive, 13.3 e versioni successive, 12.8 e versioni successive, 11.9, 11.12 e versioni successive
db.r6g.12xlarge	Tutte le versioni attualmente disponibili	17.4 e versioni successive, 16.1 e versioni successive,

Classe di istanza	Aurora MySQL	Aurora PostgreSQL
		15.2 e versioni successive, 14.3 e versioni successive, 13.3 e versioni successive, 12.8 e versioni successive, 11.9, 11.12 e versioni successive
db.r6g.8xlarge	Tutte le versioni attualmente disponibili	17.4 e versioni successive, 16.1 e versioni successive, 15.2 e versioni successive, 14.3 e versioni successive, 13.3 e versioni successive, 12.8 e versioni successive, 11.9, 11.12 e versioni successive
db.r6g.4xlarge	Tutte le versioni attualmente disponibili	17.4 e versioni successive, 16.1 e versioni successive, 15.2 e versioni successive, 14.3 e versioni successive, 13.3 e versioni successive, 12.8 e versioni successive, 11.9, 11.12 e versioni successive
db.r6g.2xlarge	Tutte le versioni attualmente disponibili	17.4 e versioni successive, 16.1 e versioni successive, 15.2 e versioni successive, 14.3 e versioni successive, 13.3 e versioni successive, 12.8 e versioni successive, 11.9, 11.12 e versioni successive

Classe di istanza	Aurora MySQL	Aurora PostgreSQL
db.r6g.xlarge	Tutte le versioni attualmente disponibili	17.4 e versioni successive, 16.1 e versioni successive, 15.2 e versioni successive, 14.3 e versioni successive, 13.3 e versioni successive, 12.8 e versioni successive, 11.9, 11.12 e versioni successive
db.r6g.large	Tutte le versioni attualmente disponibili	17.4 e versioni successive, 16.1 e versioni successive, 15.2 e versioni successive, 14.3 e versioni successive, 13.3 e versioni successive, 12.8 e versioni successive, 11.9, 11.12 e versioni successive

db.r6i: classi di istanza ottimizzate per la memoria

Classe di istanza	Aurora MySQL	Aurora PostgreSQL
db.r6i.32xlarge	Tutte le versioni attualmente disponibili	17.4 e versioni successive, 16.1 e versioni successive, 15.2 e versioni successive, 14.3 e versioni successive, 13.5 e versioni successive, 12.9 e versioni successive
db.r6i.24xlarge	Tutte le versioni attualmente disponibili	17.4 e versioni successive, 16.1 e versioni successive, 15.2 e versioni successive, 14.3 e versioni successive, 13.5 e versioni successive, 12.9 e versioni successive

Classe di istanza	Aurora MySQL	Aurora PostgreSQL
db.r6i.16xlarge	Tutte le versioni attualmente disponibili	17.4 e versioni successive, 16.1 e versioni successive, 15.2 e versioni successive, 14.3 e versioni successive, 13.5 e versioni successive, 12.9 e versioni successive
db.r6i.12xlarge	Tutte le versioni attualmente disponibili	17.4 e versioni successive, 16.1 e versioni successive, 15.2 e versioni successive, 14.3 e versioni successive, 13.5 e versioni successive, 12.9 e versioni successive
db.r6i.8xlarge	Tutte le versioni attualmente disponibili	17.4 e versioni successive, 16.1 e versioni successive, 15.2 e versioni successive, 14.3 e versioni successive, 13.5 e versioni successive, 12.9 e versioni successive
db.r6i.4xlarge	Tutte le versioni attualmente disponibili	17.4 e versioni successive, 16.1 e versioni successive, 15.2 e versioni successive, 14.3 e versioni successive, 13.5 e versioni successive, 12.9 e versioni successive
db.r6i.2xlarge	Tutte le versioni attualmente disponibili	17.4 e versioni successive, 16.1 e versioni successive, 15.2 e versioni successive, 14.3 e versioni successive, 13.5 e versioni successive, 12.9 e versioni successive

Classe di istanza	Aurora MySQL	Aurora PostgreSQL
db.r6i.xlarge	Tutte le versioni attualmente disponibili	17.4 e versioni successive, 16.1 e versioni successive, 15.2 e versioni successive, 14.3 e versioni successive, 13.5 e versioni successive, 12.9 e versioni successive
db.r6i.large	Tutte le versioni attualmente disponibili	17.4 e versioni successive, 16.1 e versioni successive, 15.2 e versioni successive, 14.3 e versioni successive, 13.5 e versioni successive, 12.9 e versioni successive

db.r5: classi di istanza ottimizzate per la memoria

Classe di istanza	Aurora MySQL	Aurora PostgreSQL
db.r5.24xlarge	Tutte le versioni attualmente disponibili	Tutte le versioni attualmente disponibili
db.r5.16xlarge	Tutte le versioni attualmente disponibili	Tutte le versioni attualmente disponibili
db.r5.12xlarge	Tutte le versioni attualmente disponibili	Tutte le versioni attualmente disponibili
db.r5.8xlarge	Tutte le versioni attualmente disponibili	Tutte le versioni attualmente disponibili
db.r5.4xlarge	Tutte le versioni attualmente disponibili	Tutte le versioni attualmente disponibili
db.r5.2xlarge	Tutte le versioni attualmente disponibili	Tutte le versioni attualmente disponibili

Classe di istanza	Aurora MySQL	Aurora PostgreSQL
db.r5.xlarge	Tutte le versioni attualmente disponibili	Tutte le versioni attualmente disponibili
db.r5.large	Tutte le versioni attualmente disponibili	Tutte le versioni attualmente disponibili

db.r4: classi di istanze ottimizzate per la memoria

Classe di istanza	Aurora MySQL	Aurora PostgreSQL
db.r4.16xlarge	Tutte le versioni 2.x; non supportato nelle versioni 3.x	Tutte le versioni 11 e 12 supportate
db.r4.8xlarge	Tutte le versioni 2.x; non supportato nelle versioni 3.x	Tutte le versioni 11 e 12 supportate
db.r4.4xlarge	Tutte le versioni 2.x; non supportato nelle versioni 3.x	Tutte le versioni 11 e 12 supportate
db.r4.2xlarge	Tutte le versioni 2.x; non supportato nelle versioni 3.x	Tutte le versioni 11 e 12 supportate
db.r4.xlarge	Tutte le versioni 2.x; non supportato nelle versioni 3.x	Tutte le versioni 11 e 12 supportate
db.r4.large	Tutte le versioni 2.x; non supportato nelle versioni 3.x	Tutte le versioni 11 e 12 supportate

db.t4g — classi di istanze a prestazioni eccezionali alimentate da processori Graviton2 AWS

Classe di istanza	Aurora MySQL	Aurora PostgreSQL
db.t4g.2xlarge	No	No
db.t4g.xlarge	No	No

Classe di istanza	Aurora MySQL	Aurora PostgreSQL
db.t4g.large	Tutte le versioni attualmente disponibili	17.4 e versioni successive, 16.1 e versioni successive, 15.2 e versioni successive, 14.3 e versioni successive, 13.3 e versioni successive, 12.7 e versioni successive, 11.12 e versioni successive
db.t4g.medium	Tutte le versioni attualmente disponibili	17.4 e versioni successive, 16.1 e versioni successive, 15.2 e versioni successive, 14.3 e versioni successive, 13.3 e versioni successive, 12.7 e versioni successive, 11.12 e versioni successive
db.t4g.small	No	No

db.t3: classi di istanze a prestazioni espandibili

Classe di istanza	Aurora MySQL	Aurora PostgreSQL
db.t3.2xlarge	No	No
db.t3.xlarge	No	No
db.t3.large	Tutte le versioni attualmente disponibili	17.4 e versioni successive, 16.1 e versioni successive, 15.2 e versioni successive, 14.3 e versioni successive, 13.3 e versioni successive, 12.7 e versioni successive, 11.12 e versioni successive

Classe di istanza	Aurora MySQL	Aurora PostgreSQL
db.t3.medium	Tutte le versioni attualmente disponibili	17.4 e versioni successive, 16.1 e versioni successive, 15.2 e versioni successive, 14.3 e versioni successive, 13.3 e versioni successive, 12.7 e versioni successive, 11.12 e versioni successive
db.t3.small	Tutte le versioni 2.x; non supportato nelle versioni 3.x	No
db.t3.micro	No	No

db.t2: classi di istanze a prestazioni espandibili

Classe di istanza	Aurora MySQL	Aurora PostgreSQL
db.t2.medium	Tutte le versioni 2.x; non supportato nelle versioni 3.x	No
db.t2.small	Tutte le versioni 2.x; non supportato nelle versioni 3.x	No

Determinazione del supporto delle classi di istanza database nelle Regioni AWS

Per determinare le classi di istanza database supportate da ciascun motore di database in una specifica Regione AWS, puoi adottare uno di diversi approcci. È possibile utilizzare la Console di gestione AWS, la pagina Prezzi di Amazon RDS o il comando AWS CLI describe-orderable-db-instance-options.

> **Note**
>
> Quando esegui operazioni con Console di gestione AWS, vengono visualizzate automaticamente le classi di istanza database supportate per un motore di database specifico, la versione del motore di database e la Regione AWS. Esempi di operazioni che è possibile eseguire includono la creazione e la modifica di un'istanza database.

Indice

- Utilizzo della pagina dei prezzi di Amazon RDS per determinare il supporto delle classi di istanza database nelle Regioni AWS
- Utilizzo di AWS CLI per determinare il supporto della classe di istanza database nelle Regioni AWS
  - Elenco delle classi di istanza database supportate da una versione specifica del motore database in una Regione AWS
  - Elenco delle versioni del motore DB che supportano una classe di istanza database specifica in una Regione AWS

Utilizzo della pagina dei prezzi di Amazon RDS per determinare il supporto delle classi di istanza database nelle Regioni AWS

Puoi utilizzare la pagina Prezzi di Amazon Aurora per determinare le classi di istanza database supportate da ciascun motore del database in una Regione AWS specifica.

Per utilizzare la pagina dei prezzi per determinare le classi di istanza database supportate da ciascun modulo di gestione in una regione

1. Vai a Prezzi di Amazon Aurora.
2. Scegli un motore Amazon Aurora nella sezione Calcolatore dei prezzi AWS.
3. In Scegli una regione, scegli una Regione AWS.
4. In Opzione di configurazione del cluster, scegli un'opzione di configurazione.
5. Utilizza la sezione dedicata alle istanze compatibili per visualizzare le classi di istanze database supportate.
6. (Facoltativo) Scegli altre opzioni nella calcolatrice, quindi scegli Salva e visualizza riepilogo o Salva e aggiungi servizio.

Utilizzo di AWS CLI per determinare il supporto della classe di istanza database nelle Regioni AWS

È possibile utilizzare AWS CLI per determinare quali classi di istanza database sono supportate per motori DB specifici e versioni del motore DB in una Regione AWS.

Per utilizzare gli esempi AWS CLI riportati di seguito, immetti i valori validi per il motore di database, la versione del motore di database, la classe di istanza database e la Regione AWS. Nella tabella seguente vengono illustrati i valori validi del motore DB.

Nome motore	Valore del motore nei comandi della CLI	Ulteriori informazioni sulle versioni
Compatibile con MySQL 5.7 e compatibile con Aurora 8.0	`aurora-mysql`	Aggiornamenti del motore del database per Amazon Aurora MySQL versione 2 e Aggiornamenti del motore del database per Amazon Aurora MySQL versione 3 nelle Note di rilascio per Aurora MySQL
Aurora PostgreSQL	`aurora-postgresql`	Note di rilascio per Aurora PostgreSQL

Per informazioni sui nomi delle Regione AWS, consulta AWSRegioni.

Negli esempi seguenti viene illustrato come determinare il supporto delle classi di istanza database in una Regione AWS utilizzando il comando AWS CLI describe-orderable-db-instance-options.

Argomenti

- Elenco delle classi di istanza database supportate da una versione specifica del motore database in una Regione AWS
- Elenco delle versioni del motore DB che supportano una classe di istanza database specifica in una Regione AWS

Elenco delle classi di istanza database supportate da una versione specifica del motore database in una Regione AWS

Per elencare le classi di istanza database supportate da una versione specifica del motore di DB in una regione Regione AWS, emettere il comando seguente.

Per Linux, macOS o Unix:

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

Per Windows:

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

L'output mostra anche le modalità motore supportate per ogni classe di istanza database.

Ad esempio, il comando seguente elenca le classi di istanza database supportate per la versione 13.6 del motore DB Aurora PostgreSQL in Stati Uniti orientali (Virginia settentrionale).

Per Linux, macOS o Unix:

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

Per Windows:

```
aws rds describe-orderable-db-instance-options --engine aurora-postgresql --engine-
version 15.3 ^
```

```
    --query "OrderableDBInstanceOptions[].
{DBInstanceClass:DBInstanceClass,SupportedEngineModes:SupportedEngineModes[0]}"  ^
    --output table ^
    --region us-east-1
```

Elenco delle versioni del motore DB che supportano una classe di istanza database specifica in una Regione AWS

Per elencare le versioni del motore DB che supportano una classe di istanza database specifica in una Regione AWS, emettere il comando seguente.

Per Linux, macOS o Unix:

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

Per Windows:

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

L'output mostra anche le modalità del motore supportate per ogni versione del motore DB.

Ad esempio, il comando seguente elenca le versioni del motore DB del motore DB Aurora PostgreSQL che supportano la classe di istanza database db.r5.large in Stati Uniti orientali (Virginia settentrionale).

Per Linux, macOS o Unix:

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

```
    --output table \
    --region us-east-1
```

Per Windows:

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

Specifiche hardware per le classi di istanza database per Aurora

Nella tabella di questa sezione, sono riportati i dettagli hardware relativi alle classi di istanza database Amazon RDS per Aurora.

Per informazioni sul supporto del motore del database di Aurora per ciascuna classe di istanza database, consulta Motori di database supportati per classi di istanza database.

Argomenti

- Terminologia hardware per le classi di istanza database per Aurora
- Specifiche hardware per le classi di istanza ottimizzata per la memoria
- Specifiche hardware per le classi di istanza a prestazioni espandibili

Terminologia hardware per le classi di istanza database per Aurora

La terminologia seguente viene utilizzata per descrivere le specifiche dell'hardware per le classi di istanza database:

VPCU

Il numero di unità di elaborazione centrale virtuali (CPUs). Una CPU virtuale è un'unità di capacità che puoi usare per confrontare le classi di istanza database. Invece di acquistare o affittare un determinato processore da utilizzare per vari mesi o anni, si affitta la capacità su base oraria. L'obiettivo è quello di rendere disponibile una quantità coerente e specifica di capacità di CPU, entro i limiti dell'hardware effettivo sottostante.

ECU

La misura relativa della potenza di elaborazione intera di un' EC2 istanza Amazon. Per consentire agli sviluppatori di confrontare facilmente la capacità della CPU tra diverse classi di istanze, abbiamo definito un'Amazon EC2 Compute Unit. La quantità di CPU allocata a una particolare istanza è espressa in termini di queste unità di EC2 calcolo. Attualmente una ECU fornisce una capacità della CPU equivalente a quella di un processore Opteron GHz 2007 Opteron 1.0-1,2 o 2007 Xeon.

Memoria (GiB)

La RAM, in gibibyte, allocata all'istanza database. Spesso c'è un rapporto costante tra memoria e vCPU. A titolo esemplificativo, prendi la classe di istanza db.r4, che ha una memoria in rapporto vCPU simile alla classe di istanza db.r5. Tuttavia, per la maggior parte dei casi d'uso la classe di istanza db.r5 fornisce prestazioni migliori e più costanti rispetto alla classe di istanza db.r4.

Quantità max Larghezza di banda EBS (Mbps)

La larghezza di banda EBS massima in megabit al secondo. Dividendo il valore per 8, puoi ottenere il throughput previsto in megabyte al secondo.

> **Note**
>
> Questa cifra si riferisce alla I/O larghezza di banda per l'archiviazione locale all'interno dell'istanza DB. Non si applica alla comunicazione con il volume Aurora del cluster.

Larghezza di banda di rete

La velocità di rete relativa ad altre classi di istanza database.

Per informazioni sull'utilizzo dei CloudWatch parametri di Amazon per monitorare il throughput dell'istanza Aurora DB, consulta e. Valutazione dell’utilizzo dell’istanza database per Aurora MySQL con le metriche di Amazon CloudWatch Valutazione dell'utilizzo delle istanze DB per Aurora CloudWatch PostgreSQL con metriche

Specifiche hardware per le classi di istanza ottimizzata per la memoria

Le tabelle seguenti mostrano le specifiche di calcolo, memoria, archiviazione e larghezza di banda per le classi di istanza ottimizzata per la memoria.

db.x2g: classi di istanze ottimizzate per la memoria con processori Graviton2 AWS

Classe di istanza	VPCU	ECU	Memoria (GiB)	Archiviazione delle istanze (GiB)	Quantità max Larghezza di banda EBS (Mbps)	Larghezza di banda della rete (Gbps)
db.x2g.16xlarge	64	—	1.024	Solo ottimizzata per EBS	19.000	25
db.x2g.12xlarge	48	—	768	Solo ottimizzata per EBS	14.250	20
db.x2g.8xlarge	32	—	512	Solo ottimizzata per EBS	9.500	12
db.x2g.4xlarge	16	—	256	Solo ottimizzata per EBS	4.750	Fino a 10
db.x2g.2xlarge	8	—	128	Solo ottimizzata per EBS	Fino a 4.750	Fino a 10
db.x2g.xlarge	4	—	64	Solo ottimizzata per EBS	Fino a 4.750	Fino a 10
db.x2g.large	2	—	32	Solo ottimizzata per EBS	Fino a 4.750	Fino a 10

db.r8gd — classi di istanze ottimizzate per la memoria basate su processori Graviton4 e storage SSD AWS

Classe di istanza	VPCU	ECU	Memoria (GiB)	Archiviazione delle istanze (GiB)	Quantità max Larghezza di banda EBS (Mbps)	Larghezza di banda della rete (Gbps)
db.r8gd.48xlarge	192	—	1536	SSD 6 x 1900 NVMe	40.000	50

Classe di istanza	VPCU	ECU	Memoria (GiB)	Archiviazione delle istanze (GiB)	Quantità max Larghezza di banda EBS (Mbps)	Larghezza di banda della rete (Gbps)
db.r8gd.24xlarge	96	—	768	SSD 3 x 1900 NVMe	30.000	40
db.r8gd.16xlarge	64	—	512	2 x SSD 1900 NVMe	20.000	30
db.r8gd.12xlarge	48	—	384	SSD 3 x 950 NVMe	15.000	22,5
db.r8gd.8xlarge	32	—	256	1 x SSD 1900 NVMe	10.000	15
db.r8gd.4xlarge	16	—	128	1 x SSD 950 NVMe	Fino a 10.000	Fino a 15
db.r8gd.2xlarge	8	—	64	1 x SSD 474 NVMe	Fino a 10.000	Fino a 15
db.r8gd.xlarge	4	—	32	1 x 237 SSD NVMe	Fino a 10.000	Fino a 12,5
db.r8gd.large	2	—	16	1 x SSD da 118 NVMe	Fino a 10.000	Fino a 12,5

db.r8g — classi di istanze ottimizzate per la memoria alimentate da processori Graviton4 AWS

Classe di istanza	VPCU	ECU	Memoria (GiB)	Archiviazione delle istanze (GiB)	Quantità max Larghezza di banda EBS (Mbps)	Larghezza di banda della rete (Gbps)
db.r8g.48xlarge	192	—	1536	Solo ottimizzata per EBS	40.000	50

Classe di istanza	VPCU	ECU	Memoria (GiB)	Archiviazione delle istanze (GiB)	Quantità max Larghezza di banda EBS (Mbps)	Larghezza di banda della rete (Gbps)
db.r8g.24xlarge	96	—	768	Solo ottimizzata per EBS	30.000	40
db.r8g.16xlarge	64	—	512	Solo ottimizzata per EBS	20.000	30
db.r8g.12xlarge	48	—	384	Solo ottimizzata per EBS	15.000	22.5
db.r8g.8xlarge	32	—	256	Solo ottimizzata per EBS	10.000	15
db.r8g.4xlarge	16	—	128	Solo ottimizzata per EBS	Fino a 10.000	Fino a 15
db.r8g.2xlarge	8	—	64	Solo ottimizzata per EBS	Fino a 10.000	Fino a 15
db.r8g.xlarge	4	—	32	Solo ottimizzata per EBS	Fino a 10.000	Fino a 12,5
db.r8g.large	2	—	16	Solo ottimizzata per EBS	Fino a 10.000	Fino a 12,5

db.r7i: classi di istanza ottimizzata per la memoria basate su processori Intel Xeon scalabili di quarta generazione

Classe di istanza	VPCU	ECU	Memoria (GiB)	Archiviazione delle istanze (GiB)	Quantità max Larghezza di banda EBS (Mbps)	Larghezza di banda della rete (Gbps)
db.r7i.48xlarge	192	—	1536	Solo ottimizzata per EBS	40.000	50
db.r7i.24xlarge	96	—	768	Solo ottimizzata per EBS	30.000	37,5
db.r7i.16xlarge	64	—	512	Solo ottimizzata per EBS	20.000	25
db.r7i.12xlarge	48	—	384	Solo ottimizzata per EBS	15.000	18,75
db.r7i.8xlarge	32	—	256	Solo ottimizzata per EBS	10.000	12,5
db.r7i.4xlarge	16	—	128	Solo ottimizzata per EBS	Fino a 10.000	Fino a 12,5
db.r7i.2xlarge	8	—	64	Solo ottimizzata per EBS	Fino a 10.000	Fino a 12,5
db.r7i.xlarge	4	—	32	Solo ottimizzata per EBS	Fino a 10.000	Fino a 12,5
db.r7i.large	2	—	16	Solo ottimizzata per EBS	Fino a 10.000	Fino a 12,5

db.r7g — classi di istanze ottimizzate per la memoria con processori Graviton3 AWS

Classe di istanza	VPCU	ECU	Memoria (GiB)	Archiviazione delle istanze (GiB)	Quantità max Larghezza di banda EBS (Mbps)	Larghezza di banda della rete (Gbps)
db.r7g.16xlarge	64	—	512	Solo ottimizzata per EBS	20.000	30
db.r7g.12xlarge	48	—	384	Solo ottimizzata per EBS	15.000	22.5
db.r7g.8xlarge	32	—	256	Solo ottimizzata per EBS	10.000	15
db.r7g.4xlarge	16	—	128	Solo ottimizzata per EBS	Fino a 10.000	Fino a 15
db.r7g.2xlarge	8	—	64	Solo ottimizzata per EBS	Fino a 10.000	Fino a 15
db.r7g.xlarge	4	—	32	Solo ottimizzata per EBS	Fino a 10.000	Fino a 12,5
db.r7g.large	2	—	16	Solo ottimizzata per EBS	Fino a 10.000	Fino a 12,5

db.r6id: classi di istanza ottimizzata per la memoria con processori Intel Xeon scalabili di terza generazione e archiviazione SSD

Classe di istanza	VPCU	ECU	Memoria (GiB)	Archiviazione delle istanze (GiB)	Quantità max Larghezza di banda EBS (Mbps)	Larghezza di banda della rete (Gbps)
db.r6id.32xlarge	128	—	1,024	SSD 4 x 1900 NVMe	40.000	50

Classe di istanza	VPCU	ECU	Memoria (GiB)	Archiviazione delle istanze (GiB)	Quantità max Larghezza di banda EBS (Mbps)	Larghezza di banda della rete (Gbps)
db.r6id.24xlarge	96	—	768	NVMe SSD 4 x 1425	30.000	37,5

db.r6gd — classi di istanze ottimizzate per la memoria con processori Graviton2 e storage SSD AWS

Classe di istanza	VPCU	ECU	Memoria (GiB)	Archiviazione delle istanze (GiB)	Quantità max Larghezza di banda EBS (Mbps)	Larghezza di banda della rete (Gbps)
db.r6gd.16xlarge	64	—	512	2 x NVMe 1900 SSD	19.000	25
db.r6gd.12xlarge	48	—	384	2 x SSD 1425 NVMe	13.500	20
db.r6gd.8xlarge	32	—	256	1 x SSD 1900 NVMe	9.000	12
db.r6gd.4xlarge	16	—	128	1 x SSD 950 NVMe	4.750	Fino a 10
db.r6gd.2xlarge	8	—	64	1 x SSD 474 NVMe	Fino a 4.750	Fino a 10
db.r6gd.xlarge	4	—	32	1 x 237 SSD NVMe	Fino a 4.750	Fino a 10

db.r6g — classi di istanze ottimizzate per la memoria con processori Graviton2 AWS

Classe di istanza	VPCU	ECU	Memoria (GiB)	Archiviazione delle istanze (GiB)	Quantità max Larghezza di banda EBS (Mbps)	Larghezza di banda della rete (Gbps)
db.r6g.16xlarge	64	—	512	Solo ottimizzata per EBS	19.000	25
db.r6g.12xlarge	48	—	384	Solo ottimizzata per EBS	13.500	20
db.r6g.8xlarge	32	—	256	Solo ottimizzata per EBS	9.000	12
db.r6g.4xlarge	16	—	128	Solo ottimizzata per EBS	4.750	Fino a 10
db.r6g.2xlarge	8	—	64	Solo ottimizzata per EBS	Fino a 4.750	Fino a 10
db.r6g.xlarge	4	—	32	Solo ottimizzata per EBS	Fino a 4.750	Fino a 10
db.r6g.large	2	—	16	Solo ottimizzata per EBS	Fino a 4.750	Fino a 10

db.r6i: classi di istanza ottimizzata per la memoria con processori Intel Xeon scalabili di terza generazione

Classe di istanza	VPCU	ECU	Memoria (GiB)	Archiviazione delle istanze (GiB)	Quantità max Larghezza di banda EBS (Mbps)	Larghezza di banda della rete (Gbps)
db.r6i.32xlarge	128	—	1,024	Solo ottimizzata per EBS	40.000	50

Classe di istanza	VPCU	ECU	Memoria (GiB)	Archiviazione delle istanze (GiB)	Quantità max Larghezza di banda EBS (Mbps)	Larghezza di banda della rete (Gbps)
db.r6i.24xlarge	96	—	768	Solo ottimizzata per EBS	30.000	37,5
db.r6i.16xlarge	64	—	512	Solo ottimizzata per EBS	20.000	25
db.r6i.12xlarge	48	—	384	Solo ottimizzata per EBS	15.000	18.75
db.r6i.8xlarge	32	—	256	Solo ottimizzata per EBS	10.000	12,5
db.r6i.4xlarge	16	—	128	Solo ottimizzata per EBS	Fino a 10.000	Fino a 12,5
db.r6i.2xlarge	8	—	64	Solo ottimizzata per EBS	Fino a 10.000	Fino a 12,5
db.r6i.xlarge	4	—	32	Solo ottimizzata per EBS	Fino a 10.000	Fino a 12,5
db.r6i.large	2	—	16	Solo ottimizzata per EBS	Fino a 10.000	Fino a 12,5

db.r5: classi di istanze ottimizzate per la memoria

Classe di istanza	VPCU	ECU	Memoria (GiB)	Archiviazione delle istanze (GiB)	Quantità max Larghezza di banda EBS (Mbps)	Larghezza di banda della rete (Gbps)
db.r5.24xlarge	96	347	768	Solo ottimizzata per EBS	19.000	25

Classe di istanza	VPCU	ECU	Memoria (GiB)	Archiviazione delle istanze (GiB)	Quantità max Larghezza di banda EBS (Mbps)	Larghezza di banda della rete (Gbps)
db.r5.16xlarge	64	264	512	Solo ottimizzata per EBS	13.600	20
db.r5.12xlarge	48	173	384	Solo ottimizzata per EBS	9.500	12
db.r5.8xlarge	32	132	256	Solo ottimizzata per EBS	6.800	10
db.r5.4xlarge	16	71	128	Solo ottimizzata per EBS	4.750	Fino a 10
db.r5.2xlarge	8	38	64	Solo ottimizzata per EBS	Fino a 4.750	Fino a 10
db.r5.xlarge	4	19	32	Solo ottimizzata per EBS	Fino a 4.750	Fino a 10
db.r5.large	2	10	16	Solo ottimizzata per EBS	Fino a 4.750	Fino a 10

db.r4: classi di istanza ottimizzata per la memoria con processori Intel Xeon scalabili

Classe di istanza	VPCU	ECU	Memoria (GiB)	Archiviazione delle istanze (GiB)	Quantità max Larghezza di banda EBS (Mbps)	Larghezza di banda della rete (Gbps)
db.r4.16xlarge	64	195	488	Solo ottimizzata per EBS	14.000	25
db.r4.8xlarge	32	99	244	Solo ottimizzata per EBS	7,000	10

Classe di istanza	VPCU	ECU	Memoria (GiB)	Archiviazione delle istanze (GiB)	Quantità max Larghezza di banda EBS (Mbps)	Larghezza di banda della rete (Gbps)
db.r4.4xlarge	16	53	122	Solo ottimizzata per EBS	3,500	Fino a 10
db.r4.2xlarge	8	27	61	Solo ottimizzata per EBS	1.700	Fino a 10
db.r4.xlarge	4	13,5	30,5	Solo ottimizzata per EBS	850	Fino a 10
db.r4.large	2	7	15,25	Solo ottimizzata per EBS	425	Fino a 10

Specifiche hardware per le classi di istanza a prestazioni espandibili

Le tabelle seguenti mostrano le specifiche di calcolo, memoria, archiviazione e larghezza di banda per le classi di istanza a prestazioni espandibili.

db.t4g — classi di istanze a prestazioni eccezionali alimentate da processori Graviton2 AWS

Classe di istanza	VPCU	ECU	Memoria (GiB)	Archiviazione delle istanze (GiB)	Quantità max Larghezza di banda EBS (Mbps)	Larghezza di banda della rete (Gbps)
db.t4g.large	2	—	8	Solo ottimizzata per EBS	Fino a 2.780	Fino a 5
db.t4g.medium	2	—	4	Solo ottimizzata per EBS	Fino a 2.085	Fino a 5

db.t3: classi di istanza a prestazioni espandibili

Classe di istanza	VPCU	ECU	Memoria (GiB)	Archiviazione delle istanze (GiB)	Quantità max Larghezza di banda EBS (Mbps)	Larghezza di banda della rete (Gbps)
db.t3.large	2	Variab	8	Solo ottimizzata per EBS	Fino a 2.048	Fino a 5
db.t3.medium	2	Variab	4	Solo ottimizzata per EBS	Fino a 1.536	Fino a 5
db.t3.small	2	Variab	2	Solo ottimizzata per EBS	Fino a 1.536	Fino a 5

db.t2: classi di istanza a prestazioni espandibili

Classe di istanza	VPCU	ECU	Memoria (GiB)	Archiviazione delle istanze (GiB)	Quantità max Larghezza di banda EBS (Mbps)	Larghezza di banda della rete (Gbps)
db.t2.medium	2	Variab	4	Solo EBS	—	Moderata
db.t2.small	1	Variab	2	Solo EBS	—	Bassa

Archiviazione Amazon Aurora

Di seguito, sono riportare informazioni sul sottosistema di archiviazione Aurora. Aurora utilizza un'architettura di archiviazione distribuita e condivisa che è un fattore importante in termini di prestazioni, scalabilità e affidabilità per i cluster Aurora.

Argomenti

- Panoramica dell'archiviazione di Amazon Aurora
- Contenuto dei volumi del cluster
- Configurazioni dell'archiviazione per i cluster database Amazon Aurora

- Ridimensionamento automatico dello storage Aurora
- Come viene fatturato lo storage dei dati Aurora

Panoramica dell'archiviazione di Amazon Aurora

I dati di Aurora sono archiviati nel volume del cluster che è un singolo volume virtuale che utilizza unità SSD (Solid State Drive). Un volume del cluster è composto da copie di dati distribuite su tre zone di disponibilità in una singola regione AWS. Poiché i dati vengono replicati automaticamente nelle zone di disponibilità, i dati risultano estremamente durevoli e poco soggetti ad andare perduti. La replica garantisce anche una maggiore disponibilità del database durante un failover perché le copie dei dati sono già presenti in altre zone di disponibilità e continuano a servire le richieste di dati alle istanze database del cluster database. La quantità di replica è indipendente dal numero di istanze database nel cluster.

Aurora utilizza lo storage locale separato per i file temporanei non persistenti. Sono inclusi i file utilizzati per scopi quali l'ordinamento di set di dati di grandi dimensioni durante l'elaborazione delle query e la creazione degli indici. Per ulteriori informazioni, consultare Limiti di storage temporaneo per Aurora MySQL e Limiti di storage temporaneo per Aurora PostgreSQL.

Contenuto dei volumi del cluster

Il volume del cluster Aurora contiene tutti i dati dell'utente, gli oggetti dello schema e i metadati interni come le tabelle di sistema e il log binario. Ad esempio, Aurora memorizza tutte le tabelle, gli indici, gli oggetti binari di grandi dimensioni (BLOB), le stored procedure e così via per un cluster Aurora nel volume del cluster.

L'architettura di storage condivisa di Aurora rende i dati indipendenti dalle istanze database nel cluster. Ad esempio, puoi aggiungere rapidamente un'istanza database perché Aurora non crea una nuova copia dei dati della tabella. L'istanza database si connette al volume condiviso che contiene già tutti i dati. Puoi rimuovere un'istanza database da un cluster senza rimuovere alcun dato sottostante dal cluster. Solo quando elimini l'intero cluster Aurora rimuove i dati.

Configurazioni dell'archiviazione per i cluster database Amazon Aurora

Amazon Aurora dispone di due configurazioni dell'archiviazione per i cluster database:

- Aurora I/O-Optimized: rapporto prezzo/prestazioni e prevedibilità migliorati per applicazioni con uso intensivo di I/O. I prezzi sono calcolati sono calcolati solo in base all'utilizzo e all'archiviazione dei cluster database, senza costi aggiuntivi per le operazioni di I/O di lettura e scrittura.

Aurora I/O-Optimized è la scelta migliore quando la spesa I/O è pari o superiore al 25% della spesa totale del database Aurora.

È possibile scegliere Aurora I/O-Optimized durante la creazione o la modifica di un cluster database con una versione del motore DB che supporta la configurazione del cluster Aurora I/O-Optimized. È possibile passare da Aurora I/O-Optimized a Aurora Standard in qualsiasi momento.

- Aurora Standard: prezzi convenienti per molte applicazioni con un utilizzo moderato di I/O. Oltre all'utilizzo e all'archiviazione dei cluster database, il prezzo è calcolato in base a una tariffa standard per 1 milione di richieste per le operazioni di I/O.

  Aurora Standard è la scelta migliore quando la spesa I/O è inferiore al 25% della spesa totale del database Aurora.

  È possibile passare da Aurora Standard a Aurora I/O-Optimized una volta ogni 30 giorni. Quando passi dall'opzione di archiviazione Aurora Standard a Aurora I/O-Optimized e viceversa per le istanze database non basate su NVMe, non si verificano tempi di inattività. Tuttavia, per le istanze database basate su NVMe, il passaggio dall'opzione di archiviazione Aurora I/O-Optimized a Aurora Standard e viceversa richiede il riavvio del motore di database, che può causare un breve tempo di inattività.

Per ulteriori informazioni sul supporto della Regione AWS e della versione, consultare Regioni e motori di database Aurora supportati per configurazioni dell'archiviazione dei cluster.

Per informazioni sui prezzi relativi alle configurazioni dell'archiviazione per Amazon Aurora, consulta la pagina relativa ai prezzi di Amazon Aurora.

Per informazioni sulla scelta della configurazione dello storage durante la creazione di un cluster database, consulta Creazione di un cluster di database. Per informazioni sulla modifica della configurazione dello storage per un cluster database, consulta Impostazioni per Amazon Aurora.

Ridimensionamento automatico dello storage Aurora

I volumi dei cluster Aurora aumentano automaticamente quando aumenta la quantità di dati nel database. Per informazioni sulle dimensioni massime dei volumi dei cluster Aurora per ogni versione del motore, consulta Limiti di dimensione Amazon Aurora. Questo dimensionamento automatico dello storage è combinato con un sottosistema di storage ad alte prestazioni e altamente distribuito. Ciò rende Aurora la scelta ideale per i dati aziendali importanti, quando i tuoi obiettivi principali sono l'affidabilità e l'elevata disponibilità.

Per visualizzare lo stato del volume, consulta Visualizzazione dello stato del volume per un cluster DB Aurora MySQL o Visualizzazione dello stato del volume per un cluster di database Aurora PostgreSQL . Per trovare metodi per bilanciare i costi di storage rispetto ad altre priorità, Dimensionamento dello storage descrive come monitorare i parametri di Amazon Aurora `AuroraVolumeBytesLeftTotal` e `VolumeBytesUsed` in CloudWatch .

Quando i dati di Aurora vengono rimossi, lo spazio allocato per tali dati viene liberato. Esempi di rimozione di dati sono l'eliminazione o il troncamento di una tabella. Questa riduzione automatica dell'utilizzo dello storage consente di contenere al minimo i costi di storage.

> **Note**
>
> I limiti di archiviazione e il comportamento di ridimensionamento dinamico indicati in questa pagina si applicano alle tabelle persistenti e ad altri dati archiviati nel volume del cluster.
> Per Aurora PostgreSQL, i dati delle tabelle temporanee vengono archiviati nell'istanza database locale.
> Per Aurora MySQL versione 2, i dati della tabella temporanea vengono archiviati per impostazione predefinita nel volume del cluster per le istanze di scrittura e nell'archiviazione locale per le istanze di lettura. Per ulteriori informazioni, consulta Motore di archiviazione per le tabelle temporanee su disco.
> Per Aurora MySQL versione 3, i dati della tabella temporanea vengono archiviati nell'istanza database locale o nel volume del cluster. Per ulteriori informazioni, consulta Nuovo comportamento della tabella temporanea in Aurora MySQL versione 3.
> La dimensione massima delle tabelle temporanee che risiedono nell'archiviazione locale è limitata dalla dimensione massima dello spazio di archiviazione locale dell'istanza database. La dimensione dell'archiviazione locale dipende dalla classe di istanza utilizzata. Per ulteriori informazioni, consultare Limiti di storage temporaneo per Aurora MySQL e Limiti di storage temporaneo per Aurora PostgreSQL.

Alcune funzionalità di archiviazione, ad esempio la dimensione massima del volume del cluster e il ridimensionamento automatico quando i dati vengono rimossi, dipendono dalla versione di Aurora del cluster. Per ulteriori informazioni, consulta Dimensionamento dello storage. Puoi inoltre scoprire come evitare problemi di storage e come monitorare lo spazio di storage allocato e lo spazio libero nel cluster.

Come viene fatturato lo storage dei dati Aurora

Anche se un volume del cluster Aurora può crescere fino al limite di 256 TiB per specifiche versioni del motore, il costo addebitato rimane limitato a quello dello spazio effettivamente utilizzato nel volume del cluster Aurora. Nelle versioni precedenti di Aurora, il volume del cluster poteva riutilizzare lo spazio liberato dalla rimozione dei dati, ma lo spazio di archiviazione allocato non diminuiva mai. Ora quando i dati di Aurora vengono rimossi, ad esempio eliminando una tabella o un database, lo spazio allocato complessivo diminuisce della quantità equivalente. Pertanto, puoi ridurre le spese di archiviazione eliminando tabelle, indici, database e così via che non sono più necessari.

> **Tip**
>
> Per le versioni precedenti senza la funzionalità di ridimensionamento dinamico, la reimpostazione dell'utilizzo dello storage per un cluster comportava l'esecuzione di un dump logico e il ripristino di un nuovo cluster. Tali operazioni possono richiedere molto tempo per un volume considerevole di dati. Se si verifica questa situazione, prendi in considerazione l'aggiornamento del cluster a una versione che supporta il ridimensionamento dinamico del volume.

Per informazioni su quali versioni di Aurora supportano il ridimensionamento dinamico e su come ridurre al minimo i costi di archiviazione monitorando l'utilizzo dello spazio di archiviazione del cluster, consulta Dimensionamento dello storage. Per informazioni sulla fatturazione dell'archiviazione di backup Aurora, consulta Informazioni sull'utilizzo dello storage di backup Amazon Aurora. Per informazioni sui prezzi dello storage dei dati di Aurora, consulta Prezzi di Amazon RDS for Aurora.

Affidabilità Amazon Aurora

Il design di Aurora assicura affidabilità, durevolezza e tolleranza agli errori. È possibile sfruttare varie opzioni per configurare l'architettura del cluster database Aurora in modo da ottimizzare la disponibilità, come aggiungere repliche di Aurora e posizionarle in zone di disponibilità diverse. Inoltre Aurora include una serie di caratteristiche automatiche che lo rendono una soluzione database estremamente affidabile.

Argomenti

- Riparazione automatica dello storage
- Cache delle pagine superstite

- Ripristino in seguito a riavvii non pianificati

Riparazione automatica dello storage

Dal momento che Aurora conserva varie copie dei dati in tre zone di disponibilità, il rischio di perdita dei dati a seguito di un problema con un disco è estremamente remoto. Aurora inoltre rileva automaticamente gli errori nei volumi dei dischi che formano il volume di cluster. In caso di errore in un volume del disco, Aurora ripara immediatamente il segmento. Quando Aurora ripara il segmento del disco, utilizza i dati in altri volumi che creano il volume del cluster per garantire che i dati del segmento riparato siano aggiornati. In questo modo, Aurora evita la perdita di dati e riduce la necessità di eseguire operazioni di ripristino point-in-time a seguito di un errore nel disco.

Cache delle pagine superstite

In Aurora, ogni cache delle pagine dell'istanza viene gestita con un processo separato dal database, che consente alla cache di pagina di sopravvivere in modo indipendente dal database. (La cache delle pagine è anche chiamata pool di buffer InnoDB su Aurora MySQL e la cache del buffer InnoDB su Aurora PostgreSQL.)

Nella remota eventualità di un errore del database, la cache delle pagine rimane in memoria. Questo consente di mantenere le pagine di dati "precaricate" nella cache delle pagine quando il database viene riavviato. Tale espediente consente di migliorare le prestazioni, perché elimina la necessità di query iniziali per eseguire operazioni di I/O di lettura per "precaricare" la cache delle pagine.

Per Aurora MySQL, il comportamento della cache delle pagine durante il riavvio e il failover è il seguente:

- È possibile riavviare l’istanza di scrittura senza riavviare le istanze di lettura.
  - Se le istanze di lettura non vengono riavviate al riavvio dell'istanza di scrittura, non perdono le cache delle pagine.
  - Se le istanze di lettura vengono riavviate al riavvio dell'istanza di scrittura, perdono le cache delle pagine.
- Quando un'istanza di lettura viene riavviata, entrambe le cache delle pagine sulle istanze di scrittura e lettura sopravvivono.
- Quando il cluster database esegue il failover, l'effetto è simile al riavvio di un'istanza di scrittura. Sulla nuova istanza di scrittura (in precedenza l'istanza di lettura) la cache delle pagine sopravvive, ma sull'istanza di lettura (in precedenza l'istanza di scrittura), la cache delle pagina non sopravvive.

Per Aurora PostgreSQL, è possibile utilizzare la gestione della cache del cluster per conservare la cache delle pagine di un'istanza di lettura designata che diventa l'istanza di scrittura dopo il failover. Per ulteriori informazioni, consulta Ripristino rapido dopo il failover con Cluster Cache Management per Aurora PostgreSQL.

Ripristino in seguito a riavvii non pianificati

Aurora è progettato per eseguire il ripristino quasi istantaneo dopo un riavvio non pianificato per continuare a fornire i dati dell'applicazione senza il log binario. Aurora esegue il ripristino in modo asincrono su thread paralleli, in modo che il database sia aperto e immediatamente disponibile dopo un riavvio non pianificato.

Per ulteriori informazioni, consultare Tolleranza ai guasti di un cluster DB Aurora e Ottimizzazioni per ridurre i tempi di riavvio del database.

Di seguito riportiamo alcune considerazioni sui log binari e il ripristino dopo un riavvio non pianificato di Aurora MySQL:

- L'attivazione dei log binari in Aurora incide direttamente sui tempi di ripristino dopo un riavvio non pianificato, perché obbliga l'istanza database a eseguire il ripristino dei log binari.
- Il tipo di log binari usati incide sulle dimensioni e sull'efficienza dell'operazione. Partendo dalla stessa quantità di attività del database, alcuni formati inseriscono nei log binari un numero maggiore di informazioni rispetto ad altri. Le seguenti impostazioni del parametro `binlog_format` modificano la quantità di dati inseriti nei log:
  - `ROW` – Massima quantità di dati
  - `STATEMENT` – Minima quantità di dati
  - `MIXED` – Una quantità di dati moderata che in genere rappresenta il compromesso ottimale fra integrità dei dati e prestazioni

  La quantità di dati dei log binari influisce sui tempi di ripristino. Con l'inserimento di una grande quantità di dati nei log binari, l'istanza database deve elaborare più dati, allungando i tempi di ripristino.
- Per ridurre il sovraccarico di calcolo e migliorare i tempi di ripristino con la registrazione di log binari, è possibile utilizzare il log binario avanzato. Il log binario avanzato migliora i tempi di ripristino del database fino al 99%. Per ulteriori informazioni, consulta Configurazione del file di log binario avanzato per Aurora MySQL.

- Con Aurora non è necessario che i log binari replichino i dati all'interno di un cluster database o che eseguano il ripristino point-in-time (PITR).
- Se non hai bisogno del log binario per la replica esterna (o di uno stream esterno del log binario), ti consigliamo di impostare il parametro `binlog_format` su `OFF` per disattivare i log binari. In questo modo, puoi abbreviare i tempi di ripristino.

Per ulteriori informazioni sui log binari e la replica in Aurora, consulta Replica con Amazon Aurora. Per ulteriori informazioni sulle implicazioni dei vari tipi di replica MySQL, consulta Advantages and Disadvantages of Statement-Based and Row-Based Replication nella documentazione di MySQL.

Sicurezza di Amazon Aurora

La sicurezza di Amazon Aurora viene gestita su tre livelli:

- Per controllare chi è in grado di eseguire le operazioni di gestione di Amazon RDS nei cluster database e nelle istanze database, viene utilizzato AWS Identity and Access Management (IAM). Quando esegui la connessione ad AWS usando le credenziali IAM, il tuo account AWS deve disporre di policy IAM per la concessione delle autorizzazioni richieste per eseguire le operazioni di gestione di Amazon RDS. Per ulteriori informazioni, consulta Gestione accessi e identità per Amazon Aurora.

  Se si utilizza IAM per accedere alla console Amazon RDS, è necessario in primo luogo accedere alla Console di gestione AWS con le credenziali utente, quindi passare alla console Amazon RDS all'indirizzo https://console.aws.amazon.com/rds.
- I cluster di database Aurora devono essere creati in un cloud privato virtuale (VPC) utilizzando il servizio Amazon VPC. Per controllare i dispositivi e le istanze Amazon EC2 che possono aprire le connessioni all'endpoint e alla porta dell'istanza database per i cluster di database Aurora in un VPC, è necessario utilizzare un gruppo di sicurezza VPC. Puoi creare queste connessioni di endpoint e porta tramite Transport Layer Security (TLS)/Secure Sockets Layer (SSL). Le regole del firewall aziendale possono inoltre determinare se i dispositivi in esecuzione nell'azienda possono aprire connessioni a un'istanza database. Per ulteriori informazioni sui VPC, consulta VPC Amazon e Amazon Aurora.
- Per autenticare gli accessi e le autorizzazioni per un cluster database Amazon Aurora puoi seguire uno degli approcci qui riportati oppure utilizzare una loro combinazione.
  - Puoi adottare lo stesso approccio utilizzato per un'istanza database standalone di MySQL o PostgreSQL.

Le tecniche per l'autenticazione degli accessi e delle autorizzazioni delle istanze database standalone di MySQL o PostgreSQL, come ad esempio l'uso di comandi SQL o la modifica delle tabelle degli schemi del database, funzionano anche con Aurora. Per ulteriori informazioni, consulta Sicurezza con Amazon Aurora MySQL o Sicurezza con Amazon Aurora PostgreSQL.

- È possibile utilizzare l'autenticazione database IAM.

  Con l'autenticazione database IAM, esegui l'autenticazione sul cluster database Aurora tramite un utente o un ruolo IAM e un token di autenticazione. Il token di autenticazione è un valore univoco, generato tramite il processo di firma Signature Version 4. Quando utilizzi l'autenticazione database IAM, puoi utilizzare le stesse credenziali per controllare l'accesso alle risorse AWS e ai database. Per ulteriori informazioni, consulta Autenticazione del database IAM .

- È possibile usare l'autenticazione Kerberos per Aurora PostgreSQL e Aurora MySQL.

  È possibile utilizzare Kerberos per autenticare gli utenti quando si connettono al cluster database Aurora PostgreSQL e Aurora MySQL. In questo caso, il cluster di database funziona con AWS Directory Service for Microsoft Active Directory per abilitare l'autenticazione Kerberos. AWS Directory Service for Microsoft Active Directory è anche noto come AWS Managed Microsoft AD. Mantenere tutte le credenziali nella stessa directory consente di ridurre il tempo e l'impegno. È disponibile una posizione centralizzata per archiviare e gestire le credenziali per più cluster di database. L'uso di una directory può inoltre migliorare il profilo di sicurezza complessivo. Per ulteriori informazioni, consultare Utilizzo di Autenticazione Kerberos con Aurora PostgreSQL e Utilizzo dell'autenticazione Kerberos per Aurora MySQL.

Per informazioni sulla configurazione della sicurezza, vedi Sicurezza in Amazon Aurora.

Utilizzo di SSL con i cluster di database di Aurora

I cluster database Amazon Aurora supportano le connessioni Secure Sockets Layer (SSL) da applicazioni che utilizzano lo stesso processo e la stessa chiave pubblica delle istanze database Amazon RDS. Per ulteriori informazioni, consulta Sicurezza con Amazon Aurora MySQL, Sicurezza con Amazon Aurora PostgreSQL o Utilizzo di TLS/SSL con Aurora Serverless v1.

Elevata disponibilità di Amazon Aurora

L'architettura Amazon Aurora prevede la separazione tra archiviazione e calcolo. Aurora include alcune funzionalità per l'alta disponibilità applicabili ai dati nel cluster database. I dati restano

protetti anche se alcune o tutte le istanze database nel cluster smettono di essere disponibili. Altre funzionalità per l'alta disponibilità sono applicabili alle istanze database. Queste funzionalità assicurano la presenza di una o più istanze database per la gestione delle richieste al database provenienti dall'applicazione.

Argomenti

- Elevata disponibilità di dati Aurora
- Architetture ad alta disponibilità per istanze database Aurora
- Elevata disponibilità nelle regioni AWS con database globali Aurora
- Tolleranza ai guasti di un cluster DB Aurora
- Elevata disponibilità con Amazon RDS Proxy

Elevata disponibilità di dati Aurora

Aurora archivia le copie dei dati in un cluster di database in più zone di disponibilità in una singola Regione AWS. L'archiviazione avviene indipendentemente dal fatto che le istanze nel cluster database siano estese su più zone di disponibilità. Per ulteriori informazioni su Aurora, consulta Gestione di un cluster DB Amazon Aurora.

Quando i dati vengono scritti nell'istanza database primaria, Aurora replica in modo sincrono i dati nelle zone di disponibilità in sei nodi di storage associati al volume cluster. Questa operazione fornisce la ridondanza dei dati, elimina i blocchi I/O e riduce al minimo i picchi di latenza durante i backup di sistema. Eseguendo un'istanza database con disponibilità elevata, è possibile migliorare la disponibilità durante la manutenzione pianificata del sistema e consentire di proteggere i database da errori e interruzioni relative alle zone di disponibilità. Per ulteriori informazioni sulle zone di disponibilità, consulta Regioni e zone di disponibilità.

Architetture ad alta disponibilità per istanze database Aurora

Dopo aver creato l'istanza primaria (scrittura), è possibile creare fino a 15 repliche di Aurora in sola lettura. Le repliche Aurora sono note anche come istanze di lettore. Le repliche Aurora utilizzano la replica asincrona per supportare l'elevata disponibilità senza influire sulle prestazioni dell'istanza primaria.

Durante le operazioni quotidiane, è possibile scaricare parte del lavoro per le applicazioni ad alta intensità di lettura utilizzando le istanze del lettore per elaborare le query `SELECT`. Quando un

problema riguarda l'istanza primaria, una di queste istanze del lettore prende il sopravvento come istanza primaria. Questo meccanismo è noto come failover. Molte funzionalità Aurora si applicano al meccanismo di failover. Ad esempio, Aurora rileva i problemi del database e attiva automaticamente il meccanismo di failover quando necessario. Aurora dispone inoltre di funzionalità che riducono i tempi di completamento del failover. In questo modo si riduce al minimo il tempo in cui il database non è disponibile per la scrittura durante un failover.

Aurora è progettato per eseguire il ripristino il più rapidamente possibile e il percorso più rapido per il ripristino è spesso il riavvio o il failover sulla stessa istanza database. Il riavvio è più rapido e comporta un sovraccarico inferiore rispetto al failover.

Per utilizzare una stringa di connessione che rimane invariata anche quando un failover promuove una nuova istanza primaria, è necessario connettersi all'endpoint del cluster. L'endpoint del cluster rappresenta sempre l'istanza primaria corrente nel cluster. Per ulteriori informazioni sull'endpoint del cluster, consulta Connessioni degli endpoint Amazon Aurora.

> **Tip**
>
> All'interno di ogni Regione AWS, le zone di disponibilità rappresentano posizioni distinte l'una dall'altra per fornire isolamento in caso di interruzioni. È consigliabile distribuire l'istanza primaria e le istanze di lettura nel cluster di database su più zone di disponibilità, in modo da migliorare la disponibilità del cluster di database. In questo modo, un problema che riguarda un'intera zona di disponibilità non causa un'interruzione del cluster.
> È possibile impostare un cluster di database Multi-AZ facendo una semplice scelta quando si crea il cluster. Puoi utilizzare il plugin Console di gestione AWS, la AWS CLI o l'API Amazon RDS. È inoltre possibile convertire un cluster di database Aurora esistente in un cluster di database Multi-AZ aggiungendo una nuova istanza database di lettura e specificando una zona di disponibilità diversa.

Elevata disponibilità nelle regioni AWS con database globali Aurora

Per un'elevata disponibilità in più Regioni AWS, è possibile configurare i database globali Aurora. Ciascun database globale Aurora si estende su più Regioni AWS, consentendo letture globali a bassa latenza e il disaster recovery da interruzioni a livello di Regione AWS. Aurora replica in modo asincrono tutti i dati e gli aggiornamenti dalla Regione AWS primaria in ciascuna delle Regioni secondarie. Per ulteriori informazioni, consulta Utilizzo di Database globale Amazon Aurora.

Tolleranza ai guasti di un cluster DB Aurora

In base alla progettazione, un cluster DB Aurora è tollerante ai guasti. Il volume del cluster è uno si estende su più zone di disponibilità di una singola Regione AWS e ciascuna di esse include una copia dei dati del volume del cluster. Questa funzionalità consente al cluster DB di tollerare il guasto di una zona di disponibilità senza perdita di dati e con solo una breve interruzione del servizio.

In caso di guasto dell'istanza primaria di un cluster di database, Aurora esegue automaticamente il failover su una nuova istanza primaria, adottando uno dei seguenti due metodi:

- Promuovendo una replica Aurora esistente come nuova istanza primaria.
- Creando una nuova istanza primaria

Se il cluster di database include una o più repliche di Aurora, una di questeAurora verrà promossa come istanza primaria durante un evento di errore. Un evento di errore ha come conseguenza una breve interruzione, durante la quale le operazioni di lettura e scrittura inviate all'istanza primaria falliscono con un'eccezione. Tuttavia, il servizio viene in genere ripristinato in meno di 60 secondi e spesso in meno di 30 secondi. Per aumentare la disponibilità del cluster DB, consigliamo di creare almeno una o più repliche Aurora in due o più due zone di disponibilità.

> **Tip**
>
> In Aurora MySQL, è possibile migliorare la disponibilità durante un failover se è presente più di un'istanza database di lettura in un cluster. In Aurora MySQL, Aurora riavvia solo l'istanza database di scrittura e l'istanza di lettura in cui si verifica il failover. Altre istanze di lettura nel cluster rimangono disponibili durante un failover per continuare a elaborare le query tramite connessioni all'endpoint di lettura.
> È inoltre possibile migliorare la disponibilità durante un failover utilizzando RDS Proxy con il cluster database di Aurora database. Per ulteriori informazioni, consulta Elevata disponibilità con Amazon RDS Proxy.

Puoi personalizzare l'ordine di promozione a istanza principale delle repliche Aurora dopo un guasto assegnando una priorità a ciascuna di esse. Le priorità vanno da 0, quella massima, a 15, quella minima. Se l'istanza primaria non va a buon fine, Amazon RDS promuove la replica di Aurora a istanza primaria con la massima priorità. Puoi modificare la priorità di una replica Aurora in qualsiasi momento. Modificando una priorità non attiverai un failover.

Più repliche Aurora possono condividere la stessa priorità, avendo come risultato livelli di promozione. Se due o più repliche di Aurora condividono la stessa priorità, Amazon RDS promuove quella di dimensioni maggiori. Se due o più repliche di Aurora condividono la stessa priorità e dimensioni, Amazon RDS ne promuove una arbitrariamente nello stesso livello di promozione.

> **Note**
>
> Nell'identificazione di una destinazione di failover intervengono diversi fattori. Dopo cinque tentativi di failover non riusciti, i livelli di promozione non vengono più presi in considerazione.

Se il cluster database non contiene repliche Aurora, l'istanza primaria viene ricreata nello stesso AZ durante un evento di errore. Un evento di errore ha come conseguenza un'interruzione, durante la quale le operazioni di lettura e scrittura inviate all'istanza primaria falliscono con un'eccezione. Il servizio viene ripristinato quando crei la nuova istanza primaria. In genere, ciò richiede meno di 10 minuti. La promozione di una replica Aurora a istanza primaria è un'operazione molto più veloce rispetto alla creazione di una nuova istanza primaria.

Si assuma che l'istanza primaria nel cluster non sia disponibile a causa di un'interruzione che influisce su una intera zona di disponibilità. In questo caso, il modo per portare online una nuova istanza primaria varia in base all'utilizzo del cluster di una configurazione Multi-AZ:

- Se il cluster Aurora Serverless v2 o con provisioning contiene istanze di lettore in altre zone di disponibilità, Aurora utilizza il meccanismo di failover per promuovere una di queste istanze di lettura come nuova istanza primaria.
- Se il cluster Aurora Serverless v2 o con provisioning contiene solo una singola istanza database o se l'istanza primaria e tutte le istanze del lettore si trovano nella stessa zona di disponibilità, dovrai creare manualmente una o più nuove istanze database in un'altra zona di disponibilità.
- Se il cluster utilizza Aurora Serverless v1, Aurora crea automaticamente una nuova istanza database in un'altra zona di disponibilità. Tuttavia, questo processo comporta una sostituzione dell'host e richiede quindi un tempo di failover maggiore.

> **Note**
>
> Amazon Aurora supporta anche la replica tramite un database MySQL esterno o un'istanza database RDS MySQL. Per ulteriori informazioni, consulta Replica tra Aurora e MySQL o tra Aurora e un altro cluster di database Aurora (replica dei log binari).

Elevata disponibilità con Amazon RDS Proxy

Con RDS Proxy, è possibile creare applicazioni in grado di tollerare in modo trasparente gli errori del database senza dover scrivere codice di gestione degli errori complesso. Il proxy instrada automaticamente il traffico verso una nuova istanza database preservando le connessioni alle applicazioni. Inoltre, ignora le cache DNS (Domain Name System) per ridurre i tempi di failover fino al 66% per i database Aurora Multi-AZ. Per ulteriori informazioni, consulta Server proxy per Amazon RDS per Aurora.

Replica con Amazon Aurora

Esistono varie opzioni di replica con Aurora. Ogni cluster di database Aurora dispone di una replica incorporata tra più istanze database nello stesso cluster. Inoltre, è possibile impostare la replica con il cluster Aurora come origine o destinazione. Quando si replicano dati all'interno o all'esterno di un cluster Aurora, è possibile scegliere tra funzionalità incorporate come database globali Aurora o meccanismi di replica tradizionali per i motori MySQL o PostgreSQL DB. È possibile scegliere le opzioni appropriate in base alle quali offrire la giusta combinazione di disponibilità elevata, convenienza e prestazioni per le esigenze specifiche. Le seguenti sezioni spiegano come e quando scegliere ogni tecnica.

Argomenti

- Repliche di Aurora
- Replica con Aurora MySQL
- Replica con Aurora PostgreSQL

Repliche di Aurora

Quando si crea una seconda istanza database (o una terza e così via) in un cluster di database con provisioning di Aurora, Aurora imposta automaticamente la replica dall'istanza database writer a

tutte le altre istanze database. Queste altre istanze database sono di sola lettura e sono note come repliche di Aurora. Vengono anche definite istanze reader in relazione ai modi in cui è possibile combinare istanze database writer e reader all'interno di un cluster.

Le repliche di Aurora hanno due scopi principali. È possibile inviare query per ridimensionare le operazioni di lettura per l'applicazione. Generalmente, si esegue la connessione all'endpoint di lettura del cluster. In questo modo, Aurora può distribuire il carico per le connessioni di sola lettura su tutte le repliche di Aurora presenti nel cluster. Le repliche di Aurora contribuiscono anche ad aumentare la disponibilità. Se l'istanza writer in un cluster non è disponibile, Aurora promuove automaticamente una delle istanze reader affinché prenda il suo posto come nuovo writer.

Un cluster di database di Aurora può contenere fino a 15 repliche di Aurora. È possibile distribuire le repliche di Aurora nelle zone di disponibilità sulle quali si estende un cluster di database in una regione AWS.

I dati nel cluster di database dispongono di funzionalità di elevata disponibilità e affidabilità, indipendentemente dalle istanze database nel cluster. Se hai familiarità con le funzionalità di storage di Aurora, consulta Panoramica dell'archiviazione di Amazon Aurora. Il volume del cluster di database si compone di più copie dei dati per il cluster di database. L'istanza primaria e le repliche di Aurora nel cluster di database vedono tutti i dati nel volume cluster come un singolo volume logico.

Di conseguenza, tutte le repliche di Aurora restituiscono gli stessi dati per i risultati di query con un ritardo di replica minimo. Questo ritardo di replica è solitamente molto inferiore a 100 millisecondi dopo che l'istanza primaria scrive un aggiornamento. Il ritardo di replica varia in base alla velocità di modifica del database. Pertanto, nei periodi in cui si verificano numerose operazioni di scrittura per il database, potresti riscontrare un aumento del ritardo di replica.

> **Note**
>
> La funzionalità di replica Aurora si riavvia automaticamente quando perde la comunicazione con l'istanza database di scrittura per più di 60 secondi nelle seguenti versioni di Aurora PostgreSQL:
>
> - 14.6 e versioni precedenti
> - 13.9 e versioni precedenti
> - 12.13 e versioni precedenti
> - Tutte le versioni di Aurora PostgreSQL 11

> Con la funzionalità di disponibilità in lettura, le repliche Aurora non si riavviano automaticamente. Per ulteriori informazioni sulla funzionalità della disponibilità di lettura e sulle versioni in cui è applicabile, consulta Miglioramento della disponibilità di lettura delle repliche Aurora.

Le repliche Aurora funzionano bene per il dimensionamento della lettura perché sono dedicate completamente a operazioni di lettura nel volume cluster. Le operazioni di lettura sono gestite dall'istanza primaria. Poiché il volume del cluster viene condiviso tra tutte le istanze database nel cluster di database, è necessaria solo una minima quantità di operazioni per replicare una copia dei dati per ciascuna replica di Aurora.

Per aumentare la disponibilità puoi servirti delle repliche di Aurora come target di failover. Pertanto se l'istanza primaria non va a buon fine, una replica di Aurora viene promossa a istanza primaria. Si verifica una breve interruzione durante la quale le richieste di lettura e scrittura inviate all'istanza primaria falliscono con un'eccezione.

Tuttavia, promuovere una replica di Aurora tramite failover è un'operazione molto più veloce rispetto a ricreare l'istanza primaria. Se il cluster di database Aurora non include nessuna replica Aurora, il cluster di database non sarà disponibile mentre l’istanza database esegue il ripristino dall’errore.

Quando si verifica il failover, alcune repliche di Aurora possono essere riavviate, a seconda della versione del motore di database. Ad esempio, in Aurora MySQL, durante un failover Aurora riavvia solo l’istanza database di scrittura e la destinazione del failover. Per ulteriori informazioni sul comportamento di riavvio di diverse versioni del motore di database Aurora, consulta Riavvio di un cluster Amazon Aurora DB o di un'istanza Amazon Aurora DB. Per informazioni su cosa succede alle cache delle pagine durante il riavvio o il failover, consulta Cache delle pagine superstite.

Per gli scenari di disponibilità elevata, è consigliato creare una o più repliche di Aurora Dovrebbero essere della stessa classe delle istanze database dell'istanza primaria e in diverse zone di disponibilità per il cluster DB Aurora. Per ulteriori informazioni sulle repliche Aurora come destinazioni di failover, consulta Tolleranza ai guasti di un cluster DB Aurora.

Non è possibile creare una replica Aurora crittografata per un cluster database Aurora non crittografato. Non è possibile creare una replica Aurora non crittografata per un cluster database Aurora crittografato.

> **Tip**
>
> È possibile utilizzare le repliche di Aurora all'interno di un cluster di Aurora come unica forma di replica per mantenere i dati altamente disponibili. È possibile anche combinare la replica di Aurora incorporata con gli altri tipi di replica. In questo modo è possibile fornire un livello superiore di disponibilità elevata e distribuzione geografica dei dati.

Per informazioni dettagliate su come creare una replica di Aurora, consulta Aggiunta di repliche di Aurora a un cluster di database.

Replica con Aurora MySQL

Oltre alle repliche di Aurora, sono disponibili le seguenti opzioni di replica con Aurora MySQL:

- I cluster database Aurora MySQL in diverse regioni AWS.
  - È possibile replicare i dati in più regioni utilizzando un database globale di Aurora. Per dettagli, consultare Elevata disponibilità nelle regioni AWS con database globali Aurora.
  - Puoi creare una replica di lettura Aurora di un cluster database Aurora MySQL in una regione AWS differente tramite la replica del log binario MySQL (binlog). Ogni cluster può avere fino a cinque repliche di lettura create in questo modo, ciascuna in una regione geografica diversa.
- Due cluster database Aurora MySQL nella stessa regione , utilizzando una replica del log binario (binlog) MySQL.
- Una istanza database RDS per MySQL come origine di dati e un cluster database Aurora MySQL, creando una replica di lettura Aurora di una istanza database RDS per MySQL. In genere, è possibile utilizzare questo approccio per la migrazione a Aurora MySQL piuttosto che per le repliche in corso.

Per ulteriori informazioni sulla replica con Aurora MySQL, consulta Replica con Amazon Aurora MySQL.

Replica con Aurora PostgreSQL

Oltre alle repliche di Aurora, sono disponibili le seguenti opzioni di replica con Aurora PostgreSQL:

- Tramite Database globale Aurora, è possibile avere un cluster di database primario Aurora in una Regione e un massimo di 10 cluster di database secondari di sola lettura in Regioni diverse. In

altre parole, Aurora PostgreSQL non supporta le repliche Aurora tra regioni. Tuttavia, è possibile utilizzare il database globale Aurora per dimensionare le capacità di lettura del cluster database Aurora PostgreSQL su più regioni AWS e soddisfare gli obiettivi di disponibilità. Per ulteriori informazioni, consulta Utilizzo di Database globale Amazon Aurora.

- Due cluster di database Aurora PostgreSQL nella stessa regione, utilizzando la caratteristica di replica logica di PostgreSQL.
- Un'istanza del database RDS for PostgreSQL come origine dei dati e un cluster di database Aurora PostgreSQL, creando una replica di lettura Aurora di un'istanza database RDS for PostgreSQL. In genere, è possibile utilizzare questo approccio per la migrazione a Aurora PostgreSQL piuttosto che per le repliche in corso.

Per ulteriori informazioni sulla replica con Aurora PostgreSQL, consulta Replica con Amazon Aurora PostgreSQL.

Fatturazione delle istanze database per Aurora

Le istanze Amazon RDS con provisioning in un cluster Amazon Aurora vengono fatturate in base ai componenti seguenti:

- Ore dell'istanza database (all'ora) – In base alla classe di istanza database (ad esempio, db.t2.small or db.m4.large). I prezzi sono calcolati in base a una tariffa oraria, mentre le fatture sono calcolate al secondo e mostrano i valori in formato decimale. L'utilizzo di RDS viene fatturato in incrementi di 1 secondo, con un minimo di 10 minuti. Per ulteriori informazioni, consulta Classi di istanze database Amazon Aurora.
- Storage (per GiB al mese) – Capacità di storage assegnata all'istanza database. Se la capacità di storage assegnata viene dimensionata nel corso del mese, l'addebito sarà ripartito proporzionalmente. Per ulteriori informazioni, consulta Archiviazione Amazon Aurora.
- Richieste di input/output (I/O) (per 1 milione di richieste): numero totale di richieste di I/O di archiviazione effettuate in un ciclo di fatturazione, solo per la configurazione del cluster database Aurora Standard.

  Per ulteriori informazioni sulla fatturazione I/O in Amazon Aurora, consulta Configurazioni dell'archiviazione per i cluster database Amazon Aurora.
- Storage di backup (per GiB al mese) – Lo storage di backup è lo storage associato ai backup di database automatici e a qualsiasi snapshot DB attivo acquisito. Estendendo il periodo di retention dei backup o creando ulteriori snapshot del database, si aumenta lo storage di backup consumato

dal database. La fatturazione per secondo non si applica allo storage di backup (misurato in GB/mese).

Per ulteriori informazioni, consulta Backup e ripristino di un cluster DB Amazon Aurora.

- Trasferimento dati (per GB): il trasferimento dati da e verso l'istanza database da o verso Internet e altre regioni AWS. Per esempi utili, consulta il post del blog AWSExploring Data Transfer Costs for AWS Managed Databases.

Amazon RDS offre le opzioni di acquisto seguenti per permetterti di ottimizzare i costi in base alle esigenze:

- Istanze on demand – Paghi all'ora per le ore dell'istanza database usate. I prezzi sono calcolati in base a una tariffa oraria, mentre le fatture sono calcolate al secondo e mostrano i valori in formato decimale. L'utilizzo di RDS ora viene fatturato in incrementi di 1 secondo, con un minimo di 10 minuti.
- Istanze riservate – Prenotando un'istanza database con durata di un anno o tre anni, puoi ricevere uno sconto significativo rispetto ai prezzi delle istanze database on demand. Grazie all'uso delle istanze riservate, è possibile avviare, eliminare e arrestare più istanze entro un'ora e ottenere il vantaggio delle istanze riservate per tutte le istanze.
- Aurora Serverless v2 – Aurora Serverless v2 fornisce capacità on demand in cui l'unità di fatturazione è l'ora dell'unità di capacità Aurora anziché l'ora dell'istanza database. La capacità di Aurora Serverless v2 aumenta e diminuisce, all'interno di un intervallo specificato, a seconda del carico del database. Puoi configurare un cluster in cui tutta la capacità è fornita da Aurora Serverless v2 oppure puoi configurare un cluster in cui la capacità è una combinazione di Aurora Serverless v2 e istanze con provisioning on demand o riservate. Per ulteriori informazioni sul funzionamento delle unità di capacità di Aurora Serverless v2, consulta Funzionamento di Aurora Serverless v2.
- Aurora PostgreSQL Limitless Database: Aurora PostgreSQL Limitless Database è una funzionalità di dimensionamento orizzontale automatizzata che scala il throughput di scrittura e i limiti di archiviazione di una singola istanza database. Limitless Database distribuisce il carico di lavoro su più istanze di scrittura Aurora, pur mantenendo la facilità di funzionamento come un unico database. Limitless Database fornisce capacità on demand in cui l'unità di fatturazione è costituita dalle ore dell'unità di capacità Aurora (ACU) in un gruppo di shard database. Per informazioni sul funzionamento delle unità di capacità Aurora di Limitless Database, consulta Creazione di un cluster di database che utilizza Aurora PostgreSQL Limitless Database.

Per informazioni sui prezzi di Aurora, consulta la pagina dei prezzi di Aurora.

Argomenti

- Istanze database on demand per Aurora
- Istanze database riservate per Amazon Aurora

Istanze database on demand per Aurora

Le istanze database on demand Amazon RDS vengono fatturate in base alla classe di istanza database, ad esempio db.t3.small o db.m5.large. Per informazioni sui prezzi di Amazon RDS, consulta la pagina del prodotto Amazon RDS.

La fatturazione per un'istanza database ha inizio non appena l'istanza database è disponibile. I prezzi sono calcolati in base a una tariffa oraria, mentre le fatture sono calcolate al secondo e mostrano i valori in formato decimale. L'utilizzo di Amazon RDS viene fatturato in incrementi di un secondo, con un minimo di 10 minuti. In caso di modifica della configurazione fatturabile, come l'elaborazione della scalabilità o la capacità di storage, viene effettuato l'addebito di un minimo di 10 minuti. La fatturazione continua finché l'istanza database non viene terminata, il che avviene quando elimini l'istanza database o in caso di errore dell'istanza database.

Se non vuoi più pagare per l'istanza database, devi arrestare o eliminarla in modo da evitare la fatturazione di altre ore dell'istanza database. Per ulteriori informazioni sugli stati dell'istanza database fatturata, consulta Visualizzazione dello stato dell’istanza database di in un cluster Aurora.

Istanze database arrestate

Mentre l'istanza database è arrestata, ti viene addebitato lo storage assegnato, incluso lo storage Provisioned IOPS. Ti viene addebitato anche lo storage dei backup, incluso quello per gli snapshot manuali e i backup automatici all'interno della finestra di retention specificata. Non è previsto alcun addebito per le ore dell'istanza database.

Istanze database Multi-AZ

Una configurazione Multi-AZ migliora la durabilità e la disponibilità dei dati effettuando automaticamente l’allocazione e mantenendo una replica in standby sincronizzata in un’altra zona di disponibilità. A causa delle risorse aggiuntive e della maggiore disponibilità, le implementazioni Multi-AZ hanno un prezzo più elevato rispetto alle implementazioni Single-AZ e possono costare circa il doppio per l’istanza in standby aggiuntiva e le risorse associate.

Considerare le seguenti importanti informazioni in merito ai prezzi delle implementazioni Multi-AZ:

- Costi di calcolo: fatturati per ora di istanza database sia per le istanze primarie sia per quelle in standby.
- Costi di archiviazione: fatturati per GB al mese per l’archiviazione allocata sia per l’istanza primaria sia per quella in standby.

- Costi di trasferimento dei dati: la replica tra l'istanza primaria e l'istanza in standby è già inclusa, ma potrebbero essere applicati altri costi di trasferimento dei dati in base all'utilizzo.

Per stimare con precisione i costi mensili in base al caso d'uso specifico e alla Regione AWS, è possibile utilizzare il Calcolatore dei prezzi AWS. Questo strumento consente di inserire i dettagli di configurazione e fornisce una ripartizione completa dei costi.

> **Note**
>
> I prezzi sono soggetti a modifiche. Consulta la pagina Prezzi di Amazon RDS per le informazioni più recenti.

Istanze database riservate per Amazon Aurora

Con le istanze database riservate puoi prenotare un'istanza database per un periodo di uno o tre anni. Le istanze database riservate offrono una notevole riduzione di prezzo rispetto alle istanze database on demand. Le istanze database riservate non sono istanze fisiche, ma piuttosto si tratta di uno sconto sulla fattura applicato all'uso di determinate istanze database nell'account. Gli sconti per le istanze database riservate sono legati al tipo di istanza e alla Regione AWS.

Il processo generale per l'uso delle istanze database riservate è il seguente: prima di tutto ottieni informazioni sulle offerte disponibili per le istanze database riservate, quindi acquisti un'offerta di istanza database riservata e infine ottieni informazioni sulle tue istanze database riservate esistenti.

Per informazioni sull'acquisto di istanze database riservate e sulla visualizzazione della fatturazione per le istanze database riservate, consulta le seguenti sezioni.

- Acquisto di istanze database riservate per Amazon Aurora
- Visualizzazione della fatturazione per le istanze database riservate per Amazon Aurora

Panoramica delle istanze database riservate

Quando acquisti un'istanza database riservata in Amazon RDS, acquisti un impegno che ti permette di ottenere una tariffa scontata per un tipo di istanza database specifico, per la durata dell'istanza database riservata. Per usare un'istanza database Amazon RDS riservata, devi creare un'istanza database con una procedura analoga a quella per la creazione di un'istanza on demand.

La nuova istanza database creata deve avere le stesse specifiche dell'istanza database riservata relativamente a quanto segue:

- Regione AWS
- Motore di database (non è necessario che il numero di versione del motore di database corrisponda)
- Tipo di istanza database

Se le specifiche della nuova istanza database corrispondono a un'istanza database riservata esistente per il tuo account, viene fatturata la tariffa scontata per l'istanza database riservata. In caso contrario, l'istanza database viene fatturata in base a una tariffa on demand.

Puoi modificare un'istanza database che usi come istanza database riservata. Se la modifica rientra nelle specifiche dell'istanza database riservata, la parte o la totalità dello sconto viene comunque applicata all'istanza database modificata. Se la modifica è al di fuori alle specifiche, ad esempio la modifica della classe di istanza, lo sconto non viene più applicato. Per ulteriori informazioni, consulta Istanze database riservate con dimensioni flessibili.

Argomenti

- Tipi offerta
- Flessibilità della configurazione del cluster database Aurora
- Istanze database riservate con dimensioni flessibili
- Esempi di fatturazione di istanze database riservate Aurora
- Eliminazione di un'istanza database riservata

Per ulteriori informazioni sulle istanze riservate e sui relativi prezzi, consulta Istanze riservate di Amazon RDS.

Tipi offerta

Le istanze database riservate disponibili sono di tre tipi, ovvero —nessun pagamento anticipato, pagamento anticipato parziale e pagamento anticipato dell'intero costo—per permetterti di ottimizzare i costi di Amazon RDS in base all'utilizzo previsto.

> **Note**
>
> Non tutte le classi di istanze RDS supportano tutti i tipi di offerta di istanze riservate. Ad esempio, alcune classi di istanze potrebbero non offrire l'opzione Nessun pagamento anticipato. Per confermare la disponibilità, consulta le offerte di istanze riservate in Console di gestione AWS o utilizza il comando. `describe-reserved-db-instances-offerings` AWS CLI

Nessun pagamento anticipato

Questa opzione permette di accedere a un'istanza database riservata senza un pagamento anticipato. L'istanza riservata senza pagamento anticipato viene fatturata applicando una tariffa oraria scontata per ogni ora durante il periodo della prenotazione, indipendentemente

dall'utilizzo, e non è richiesto alcun pagamento anticipato. Questa opzione è disponibile solo per le prenotazioni della durata di un anno.

Pagamento anticipato parziale

Questa opzione richiede il pagamento anticipato di una parte dell'istanza database riservata. Le ore rimanenti del periodo di prenotazione vengono fatturate a una tariffa oraria scontata, indipendentemente dall'utilizzo. Questa opzione sostituisce l'opzione precedente per utilizzo pesante.

Pagamento anticipato intero costo

Il pagamento viene effettuato per intero all'inizio del periodo della prenotazione e non vengono addebitati altri costi per il resto del periodo, indipendentemente dal numero di ore di utilizzo.

Se usi la fatturazione consolidata, tutti gli account all'interno dell'organizzazione vengono trattati come se fossero un account unico. Questo significa che tutti gli account di un'organizzazione possono usufruire del vantaggio in termini di costi orari delle istanze database riservate acquistate da un altro account. Per ulteriori informazioni sulla fatturazione consolidata, consulta Istanze database riservate di Amazon RDS nella Guida per l'utente di Costi e fatturazione AWS .

Flessibilità della configurazione del cluster database Aurora

È possibile utilizzare le istanze database riservate di Aurora con entrambe le configurazioni di cluster database:

- Aurora I/O-Optimized— Paghi solo per l'utilizzo e l'archiviazione dei tuoi cluster DB, senza costi aggiuntivi per le operazioni di lettura e scrittura I/O .
- Aurora Standard— Oltre all'utilizzo e all'archiviazione dei cluster DB, paghi anche una tariffa standard per 1 milione di richieste operative. I/O

Aurora considera automaticamente la differenza di prezzo tra queste configurazioni. Aurora I/O-Optimized consuma il 30% in più di unità normalizzate all'ora rispetto a Aurora Standard.

Per ulteriori informazioni sulle configurazioni dell'archiviazione dei cluster Aurora, consulta Configurazioni dell'archiviazione per i cluster database Amazon Aurora. Per informazioni sui prezzi relativi alle configurazioni dell'archiviazione dei cluster Aurora, consulta la pagina dei prezzi di Amazon Aurora.

Istanze database riservate con dimensioni flessibili

Quando acquisti un'istanza database riservata, devi specificare la classe di istanza, ad esempio db.r5.large. Per maggiori informazioni sulle classi di istanza database, consulta Classi di istanze database Amazon Aurora.

Se hai un'istanza database e devi dimensionarla per aumentarne la capacità, l'istanza database riservata viene applicata automaticamente all'istanza database ridimensionata. Ciò significa che le istanze database riservate vengono applicate automaticamente a tutte le dimensioni di classi di istanze database. Le istanze DB riservate con dimensioni flessibili sono disponibili per le istanze DB con lo stesso motore di database. Regione AWS Le istanze database riservate con dimensioni flessibili sono scalabili solo nel loro tipo di classe istanza. Ad esempio, un'istanza DB riservata per un db.r6i.large può essere applicata a un db.r6i.xlarge, ma non a un db.r6id.large o db.r7g.large, perché db.r6id.large e db.r7g.large sono tipi di classi di istanza diversi.

Le istanze database riservate con dimensioni flessibili sono disponibili per i motori di database Aurora seguenti:

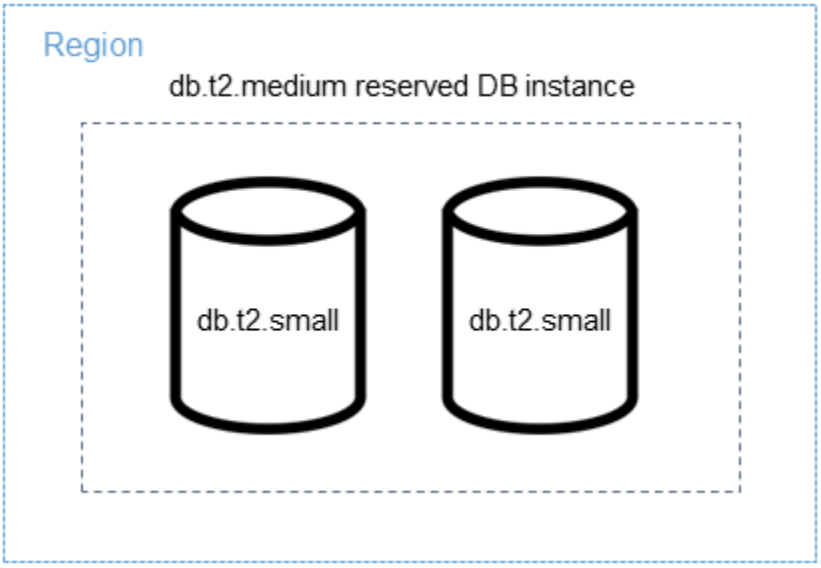
- Aurora MySQL
- Aurora PostgreSQL

È possibile confrontare l'utilizzo per le diverse dimensioni di istanze database riservate usando unità normalizzate all'ora. Ad esempio, un'unità di utilizzo in due istanze database db.r3.large equivale a 8 unità normalizzate di utilizzo all'ora in un'istanza db.r3.small. La tabella seguente mostra il numero di unità normalizzate all'ora per ogni dimensione di istanza database.

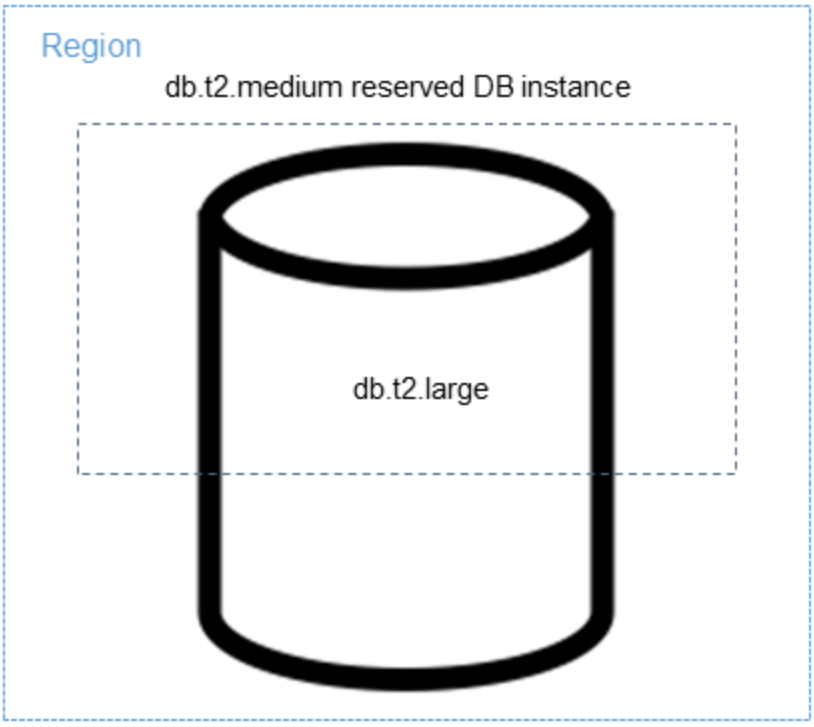
Dimensioni istanza	Unità normalizzate all'ora per un'istanza database, Aurora Standard	Unità normalizzate all'ora per un'istanza database, Aurora I/O-Optimized	Unità normalizzate all'ora per tre istanze database (istanza di scrittura e due istanze di lettura), Aurora Standard	Unità normalizzate all'ora per tre istanze database (istanza di scrittura e due istanze di lettura), Aurora I/O-Optimized
small	1	1.3	3	3.9
medium	2	2.6	6	7.8

Dimensioni istanza	Unità normalizzate all'ora per un'istanza database, Aurora Standard	Unità normalizzate all'ora per un'istanza database, Aurora I/O-Optimized	Unità normalizzate all'ora per tre istanze database (istanza di scrittura e due istanze di lettura), Aurora Standard	Unità normalizzate all'ora per tre istanze database (istanza di scrittura e due istanze di lettura), Aurora I/O-Optimized
large	4	5.2	12	15.6
xlarge	8	10.4	24	31,2
2xlarge	16	20,8	48	62,4
4xlarge	32	41,6	96	124,8
8xlarge	64	83,2	192	249,6
12xlarge	96	124,8	288	374,4
16xlarge	128	166,4	384	499,2
24xlarge	192	249,6	576	748,8
32xlarge	256	332,8	768	998,4

Supponiamo ad esempio che acquisti un'istanza database riservata `db.t2.medium` e che siano presenti due istanze database `db.t2.small` in esecuzione nel tuo account nella stessa Regione AWS. In questo caso, il vantaggio di fatturazione viene applicato per intero a entrambe le istanze.



In alternativa, se nel tuo account è in esecuzione un'`db.t2.large`istanza nello stesso account Regione AWS, il vantaggio di fatturazione viene applicato al 50 percento dell'utilizzo dell'istanza DB.



> **Note**
>
> Consigliamo di utilizzare le classi di istanza database T solo per i server di sviluppo e test o altri server non di produzione. Per ulteriori informazioni sulle classi di istanza T, consulta Tipi di classi di istanza database.

Esempi di fatturazione di istanze database riservate Aurora

Gli esempi seguenti illustrano i prezzi delle istanze database riservate per i cluster database Aurora che utilizzano le configurazioni di cluster database sia Aurora Standard che Aurora I/O-Optimized.

Esempio di utilizzo di Aurora Standard

Il prezzo di un'istanza database riservata non prevede uno sconto per i costi associati all'archiviazione, ai backup e all'I/O. L'esempio seguente illustra il costo totale mensile di un'istanza database riservata:

- Una classe di istanza database Single-AZ db.r5.large riservata in Aurora MySQL nella regione Stati Uniti orientali (Virginia settentrionale) al costo di 0,19 USD all'ora o di 138,70 USD al mese
- Storage Aurora al costo di 0,10 USD per GiB al mese (per questo esempio, supponiamo 45,60 USD al mese)
- Aurora I/O al costo di 0,20 USD per 1 milione di richieste (supponiamo 20 USD al mese per questo esempio)
- Storage di backup Aurora a 0,021 USD per GiB al mese (per questo esempio, supponiamo 30 USD al mese)

Sommando tutte queste opzioni (138,70 USD + 45,60 USD + 20 USD + 30 USD) all'istanza database riservata, il costo mensile totale è di 234,30 USD.

Se hai scelto di usare un'istanza database on demand anziché un'istanza database riservata, una classe di istanza database Single-AZ db.r5.large di Aurora MySQL nella regione Stati Uniti orientali (Virginia settentrionale) costa 0,29 USD all'ora o 217,50 USD al mese. Quindi, per un'istanza database on demand, sommando tutte queste opzioni (217,50 USD + 45,60 USD + 20 USD+ 30 USD), il costo mensile totale è di 313,10 USD. Risparmi quasi $79 al mese utilizzando l'istanza database riservata.

Esempio di utilizzo di un cluster database Aurora Standard con due istanze di lettura

Per utilizzare le istanze riservate per i cluster database Aurora, è sufficiente acquistare un'istanza riservata per ogni istanza database del cluster.

Estendendo il primo esempio, si dispone di un cluster database Aurora MySQL con un'istanza database di scrittura e due repliche Aurora, per un totale di tre istanze database nel cluster. Le due repliche Aurora non comportano costi aggiuntivi di archiviazione o backup. Se acquisti tre istanze DB riservate Aurora MySQL db.r5.large, il costo sarà di 234,30 USD (per l'istanza DB Writer) + 2 * (138,70 $+ 20$ per Aurora Replica), per un totale di 551,70 USD al mese. I/O

Il costo on-demand corrispondente per un cluster Aurora MySQL DB con un'istanza DB writer e due repliche Aurora è di 313,10$ + 2 * (217,50$ + 20$ per istanza) per un totale di 788,10 USD I/O al mese. Risparmi quasi $ 236,40 al mese utilizzando l'istanza database riservata.

Esempio di utilizzo di Aurora I/O-Optimized

È possibile riutilizzare le istanze database Aurora Standard riservate esistenti con Aurora I/O-Optimized. Per sfruttare tutti i vantaggi degli sconti sulle istanze riservate con Aurora I/O-Optimized, è possibile acquistare il 30% di istanze riservate aggiuntive simili alle attuali istanze riservate.

La tabella seguente mostra alcuni esempi di come stimare le istanze riservate aggiuntive durante l'utilizzo di Aurora I/O-Optimized. Se le istanze riservate richieste sono una frazione, è possibile sfruttare la flessibilità delle dimensioni disponibile con le istanze riservate per ottenere un numero intero. In questi esempi, "attuale" si riferisce alle istanze Aurora Standard riservate attualmente disponibili. Le istanze riservate aggiuntive sono il numero di istanze Aurora Standard riservate che è necessario acquistare per mantenere gli attuali sconti sulle istanze riservate durante l'utilizzo di Aurora I/O-Optimized.

DB instance class (Classe istanza database)	Istanze Aurora Standard riservate attuali	Istanze riservate richieste per Aurora I/O-Optimized	Istanze riservate aggiuntive necessarie	Istanze riservate aggiuntive necessarie, utilizzando la flessibilità delle dimensioni
db.r6g.large	10	10 * 1,3 = 13	3 * db.r6g.large	3 * db.r6g.large

DB instance class (Classe istanza database)	Istanze Aurora Standard riservate attuali	Istanze riservate richieste per Aurora I/O-Optimized	Istanze riservate aggiuntive necessarie	Istanze riservate aggiuntive necessarie, utilizzando la flessibilità delle dimensioni
db.r6g.4xlarge	20	20 * 1,3 = 26	6 * db.r6g.4xlarge	6 * db.r6g.4xlarge
db.r6g.12xlarge	5	5 * 1,3 = 6,5	1,5 * db.r6g.12xlarge	Uno per ogni istanza db.r6g.12xlarge, r6g.4xlarge e r6g.2xlarge (0,5 * db.r6g.12xlarge = 1 * db.r6g.4xlarge + 1 * db.r6g.2xlarge)
db.r6i.24xlarge	15	15 * 1,3 = 19,5	4,5 * db.r6i.24xlarge	4 * db.r6i.24xlarge + 1 * db.r6i.12xlarge (0,5 * db.r6i.24xlarge = 1 * db.r6i.12xlarge)

Esempio di utilizzo di un cluster database Aurora I/O-Optimized con due istanze di lettura

È presente un cluster database Aurora MySQL con un'istanza database di scrittura e due repliche Aurora, per un totale di tre istanze database nel cluster. Usano la configurazione del cluster database Aurora I/O-Optimized. Per utilizzare istanze database riservate per questo cluster, è necessario acquistare quattro istanze database riservate della stessa classe di istanze database. Tre istanze database che utilizzano Aurora I/O-Optimized consumano 3,9 unità normalizzate all'ora,

rispetto alle 3 unità normalizzate all'ora per tre istanze database che utilizzano Aurora Standard. Tuttavia I/O , si risparmiano i costi mensili per ogni istanza DB.

> **Note**
>
> I prezzi citati in questo esempio sono prezzi di esempio e potrebbero non corrispondere ai prezzi effettivi. Per informazioni sui prezzi di Aurora, consulta Prezzi di Amazon Aurora.

Eliminazione di un'istanza database riservata

Un'istanza database riservata può essere prenotata con un impegno per un periodo di un anno o di tre anni. Non è possibile annullare un'istanza database riservata. È comunque possibile eliminare un'istanza database coperta da uno sconto per istanza database riservata. Il processo di eliminazione di un'istanza database coperta da uno sconto per istanza database riservata è uguale a quello per l'eliminazione di qualsiasi altra istanza database.

I costi iniziali vengono fatturati indipendentemente dal fatto che si utilizzi le risorse.

Se elimini un'istanza database coperta da uno sconto per istanza database riservata, puoi avviare un'altra istanza database con specifiche compatibili. In questo caso, continuare a usufruire della tariffa scontata durante il periodo della prenotazione (un anno o tre anni).

Acquisto di istanze database riservate per Amazon Aurora

Puoi utilizzare l'API Console di gestione AWSAWS CLI, the e RDS per lavorare con istanze DB riservate.

Console

È possibile utilizzarla Console di gestione AWS per lavorare con istanze DB riservate, come illustrato nelle seguenti procedure.

Per ottenere informazioni sui prezzi e sulle offerte disponibili per le istanze database riservate

1. Accedi a Console di gestione AWS e apri la console Amazon RDS all'indirizzo https://console.aws.amazon.com/rds/.
2. Nel riquadro di navigazione scegliere Reserved instances (Istanze riservate).
3. Scegliere Purchase Reserved DB Instance (Acquista istanza database riservata).
4. Per Product description (Descrizione prodotto) scegliere il motore di database e il tipo di licenza.

5. Per DB instance class (Classe istanza database), scegliere la classe di istanza database.
6. In Opzione di implementazione, specifica se desideri un'implementazione Single-AZ o Multi-AZ per le istanze database.

> **Note**
>
> Per le istanze Amazon Aurora riservate, l'opzione di implementazione è sempre impostata su Istanza database Single-AZ. Tuttavia, quando crei un cluster di database Aurora, l'opzione di implementazione predefinita è Crea una replica o una sessione di lettura Aurora in una zona di disponibilità diversa (Multi-AZ).
> È necessario acquistare un'istanza database riservata per ogni istanza che si intende utilizzare, incluse le repliche Aurora. Pertanto, per le implementazioni multi-AZ in Aurora, è necessario acquistare altre istanze database riservate.

7. In Periodo, scegli per quanto tempo riservare l'istanza database.
8. Per Offering type (Tipo di offerta), scegliere il tipo di offerta.

   Dopo aver selezionato il tipo di offerta, vengono visualizzate le informazioni sui prezzi.

> **Important**
>
> Scegliere Cancel (Annulla) per evitare di acquistare un'istanza database riservata con il conseguente addebito dei relativi costi.

Dopo avere ottenuto le informazioni sulle offerte disponibili per le istanze database riservate, puoi basarti su tali informazioni per acquistare un'offerta, come illustrato nella procedura seguente.

Per acquistare un'istanza database riservata

1. Accedi a Console di gestione AWS e apri la console Amazon RDS all'indirizzo https://console.aws.amazon.com/rds/.
2. Nel riquadro di navigazione scegliere Reserved instances (Istanze riservate).
3.

> **Important**
>
> Prima di procedere, verifica di trovarti nella Regione AWS appropriata. Le istanze database riservate sono specifiche della Regione e non possono essere trasferite tra

Regioni. Assicurati di acquistare l'istanza riservata nella Regione appropriata tramite il selettore di Regione nell'angolo in alto a destra della console.

4. Scegliere Purchase reserved DB instance (Acquista istanza database riservata).
5. Per Product description (Descrizione prodotto) scegliere il motore di database e il tipo di licenza.
6. Per DB instance class (Classe istanza database), scegliere la classe di istanza database.
7. In Implementazione Multi-AZ, specifica se desideri un'implementazione Single-AZ o Multi-AZ per le istanze database.

> **Note**
>
> Per le istanze Amazon Aurora riservate, l'opzione di implementazione è sempre impostata su Istanza database Single-AZ. Quando si crea un cluster di database Amazon Aurora dall'istanza database riservata, il cluster di database viene creato automaticamente come Multi-AZ. Assicurati acquistare un'istanza database riservata per ogni istanza database che intendi utilizzare, incluse le repliche Aurora.

8. Per Term (Periodo), scegliere per quanto tempo prenotare l'istanza database.
9. Per Offering type (Tipo di offerta), scegliere il tipo di offerta.

   Dopo aver selezionato il tipo di offerta, vengono visualizzate le informazioni sui prezzi.
   Acquisto di un'istanza database riservata nella console
10. (Facoltativo) È possibile assegnare un identificatore alle istanze database riservate acquistate, per tenerne traccia. Per Reserved Id (ID istanza riservata), digitare un identificatore per l'istanza database riservata.
11. Seleziona Invia.

    L'istanza database riservata viene acquistata, quindi visualizzata nell'elenco Reserved instances (Istanze riservate).

Dopo avere acquistato le istanze database riservate, puoi ottenere informazioni sulle tue istanze database riservate come illustrato nella procedura seguente.

Per ottenere informazioni sulle istanze DB riservate per il tuo account AWS

1. Accedi a Console di gestione AWS e apri la console Amazon RDS all'indirizzo https://console.aws.amazon.com/rds/.

2. Nel riquadro di navigazione scegliere Reserved Instances (Istanze riservate).

   Verranno visualizzate le istanze database riservate per l'account. Per visualizzare informazioni dettagliate su una particolare istanza database riservata, scegli l'istanza nell'elenco. Le informazioni dettagliate su quell'istanza verranno visualizzate nel riquadro dei dettagli nella parte inferiore della console.

AWS CLI

Puoi usarla AWS CLI per lavorare con istanze DB riservate, come mostrato negli esempi seguenti.

Example di recuperare le offerte disponibili per le istanze database riservate

Per ottenere informazioni sulle offerte disponibili di istanze DB riservate, chiamate il AWS CLI comando. `describe-reserved-db-instances-offerings`

```
aws rds describe-reserved-db-instances-offerings
```

Questa chiamata restituisce un output simile al seguente:

```
OFFERING  OfferingId                            Class         Multi-AZ  Duration  Fixed
 Price  Usage Price  Description  Offering Type
OFFERING  438012d3-4052-4cc7-b2e3-8d3372e0e706  db.r3.large   y         1y
 1820.00 USD  0.368 USD    mysql        Partial  Upfront
OFFERING  649fd0c8-cf6d-47a0-bfa6-060f8e75e95f  db.r3.small   n         1y
 227.50 USD  0.046 USD    mysql        Partial  Upfront
OFFERING  123456cd-ab1c-47a0-bfa6-12345667232f  db.r3.small   n         1y
 162.00 USD   0.00 USD    mysql        All      Upfront
    Recurring Charges:   Amount  Currency  Frequency
    Recurring Charges:   0.123   USD       Hourly
OFFERING  123456cd-ab1c-37a0-bfa6-12345667232d  db.r3.large   y         1y
 700.00 USD   0.00 USD    mysql        All      Upfront
    Recurring Charges:   Amount  Currency  Frequency
    Recurring Charges:   1.25    USD       Hourly
OFFERING  123456cd-ab1c-17d0-bfa6-12345667234e  db.r3.xlarge  n         1y
 4242.00 USD   2.42 USD    mysql        No       Upfront
```

Dopo avere ottenuto le informazioni sulle offerte disponibili per le istanze database riservate, puoi basarti su tali informazioni per acquistare un'offerta.

Per acquistare un'istanza DB riservata, utilizza il AWS CLI comando [purchase-reserved-db-instances-offering](#)con i seguenti parametri:

- `--reserved-db-instances-offering-id` – L'ID dell'offerta da acquistare. Per ottenere l'ID dell'offerta, consulta l'esempio precedente.
- `--reserved-db-instance-id` – Puoi assegnare un identificatore alle istanze database riservate acquistate, per tenerne traccia.

Example di acquistare un'istanza database riservata

L'esempio seguente acquista l'offerta di istanze DB riservate con ID *649fd0c8-cf6d-47a0-bfa6-060f8e75e95f* e assegna l'identificatore di. *MyReservation*

Per Linux, macOS o Unix:

```
aws rds purchase-reserved-db-instances-offering \
    --reserved-db-instances-offering-id 649fd0c8-cf6d-47a0-bfa6-060f8e75e95f \
    --reserved-db-instance-id MyReservation
```

Per Windows:

```
aws rds purchase-reserved-db-instances-offering ^
    --reserved-db-instances-offering-id 649fd0c8-cf6d-47a0-bfa6-060f8e75e95f ^
    --reserved-db-instance-id MyReservation
```

Il comando restituisce un output simile al seguente:

```
RESERVATION  ReservationId      Class        Multi-AZ  Start Time
 Duration  Fixed Price  Usage Price  Count  State            Description  Offering Type
RESERVATION  MyReservation      db.r3.small  y         2011-12-19T00:30:23.247Z  1y
    455.00 USD   0.092 USD    1      payment-pending  mysql        Partial  Upfront
```

Dopo avere acquistato le istanze database riservate, puoi ottenere informazioni sulle tue istanze database riservate.

Per ottenere informazioni sulle istanze DB riservate per il tuo AWS account, chiama il AWS CLI comando [describe-reserved-db-instances](#), come mostrato nell'esempio seguente.

Example di ottenere le istanze DB riservate

```
aws rds describe-reserved-db-instances
```

Il comando restituisce un output simile al seguente:

```
RESERVATION  ReservationId     Class        Multi-AZ  Start Time
 Duration  Fixed Price  Usage Price  Count  State    Description  Offering Type
RESERVATION  MyReservation     db.r3.small  y         2011-12-09T23:37:44.720Z  1y
   455.00 USD   0.092 USD    1      retired  mysql        Partial  Upfront
```

API RDS

Puoi utilizzare l'API RDS per lavorare con istanze database riservate:

- Per ottenere informazioni sulle offerte disponibili per le istanze database riservate, chiamare l'operazione API Amazon RDS `DescribeReservedDBInstancesOfferings`.
- Dopo avere ottenuto le informazioni sulle offerte disponibili per le istanze database riservate, puoi basarti su tali informazioni per acquistare un'offerta. Richiama l'operazione API RDS `PurchaseReservedDBInstancesOffering` con i seguenti parametri:
  - `--reserved-db-instances-offering-id` – L'ID dell'offerta da acquistare.
  - `--reserved-db-instance-id` – Puoi assegnare un identificatore alle istanze database riservate acquistate, per tenerne traccia.
- Dopo avere acquistato le istanze database riservate, puoi ottenere informazioni sulle tue istanze database riservate. Richiama l'operazione API RDS `DescribeReservedDBInstances`.

Visualizzazione della fatturazione per le istanze database riservate per Amazon Aurora

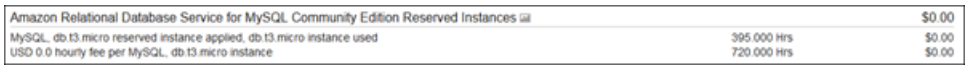
Puoi visualizzare la fatturazione per le istanze database riservate nel pannello di controllo di fatturazione nella Console di gestione AWS.

Per visualizzare la fatturazione di istanze database riservate

1. Accedi alla Console di gestione AWS.
2. Dal menu account in alto a destra, scegliere Billing Dashboard (Pannello di controllo di fatturazione).
3. Scegliere Dettagli di fatturazione nell'angolo in alto a destra del pannello di controllo.

4. In Costi di servizio AWS, espandere Servizio di database relazionale.
5. Espandi la Regione AWS in cui si trovano le istanze database riservate, ad esempio Stati Uniti occidentali (Oregon).

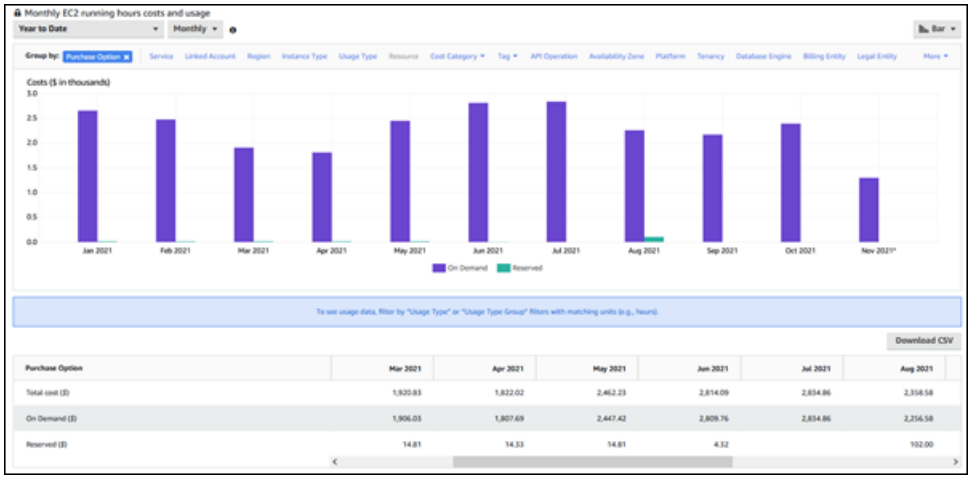
   Le istanze database riservate e i relativi addebiti orari per il mese corrente sono mostrati sotto Amazon Relational Database Service per ***Motore del database*** Istanze riservate.



   L'istanza database riservata in questo esempio è stata acquistata con pagamento anticipato, quindi non ci sono costi orari.
6. Scegliere l'icona Cost Explorer (grafico a barre) accanto alla voce Istanze riservate.

   Il Cost Explorer visualizza il grafico Costi e utilizzo delle ore di esecuzione mensili di EC2.
7. Cancellare il filtro Usage Type Group (Gruppo tipi di utilizzo) a destra del grafico.
8. Scegliere il periodo di tempo e l'unità di tempo per la quale si desidera esaminare i costi di utilizzo.

   L'esempio seguente mostra i costi di utilizzo per istanze database on-demand e riservate per l'anno in corso per mese.



   I costi delle istanza database riservate da gennaio a giugno 2021 sono oneri mensili per un'istanza con parziale pagamento anticipato, mentre il costo nell'agosto 2021 è un addebito una tantum per un'istanza con pagamento anticipato completo.

   Lo sconto dell'istanza riservata per l'istanza con parziale pagamento anticipato è scaduto nel giugno 2021, ma l'istanza database non è stata eliminata. Dopo la data di scadenza, è stata semplicemente addebitata la tariffa on-demand.

Configurazione dell'ambiente per Amazon Aurora

Prima di usare Amazon Aurora per la prima volta, è necessario completare le seguenti operazioni:

Argomenti

- Registrati per un Account AWS
- Crea un utente con accesso amministrativo
- Concessione dell'accesso programmatico
- Determinazione dei requisiti
- Fornitura dell'accesso al cluster di database nel VPC creando un gruppo di sicurezza

Se ne possiedi già uno Account AWS, conosci i requisiti di Aurora e preferisci utilizzare le impostazioni predefinite per i gruppi di sicurezza IAM e VPC, passa subito a. Nozioni di base su Amazon Aurora

Registrati per un Account AWS

Se non ne hai uno Account AWS, completa i seguenti passaggi per crearne uno.

Per iscriverti a un Account AWS

1. Apri la https://portal.aws.amazon.com/billing/registrazione.
2. Segui le istruzioni online.

   Nel corso della procedura di registrazione riceverai una telefonata o un messaggio di testo e ti verrà chiesto di inserire un codice di verifica attraverso la tastiera del telefono.

   Quando ti iscrivi a un Account AWS, Utente root dell'account AWSviene creato un. L'utente root dispone dell'accesso a tutte le risorse e tutti i Servizi AWS nell'account. Come best practice di sicurezza, assegna l'accesso amministrativo a un utente e utilizza solo l'utente root per eseguire le attività che richiedono l'accesso di un utente root.

AWS ti invia un'email di conferma dopo il completamento della procedura di registrazione. In qualsiasi momento, puoi visualizzare l'attività corrente del tuo account e gestirlo accedendo a https://aws.amazon.com/e scegliendo Il mio account.

Crea un utente con accesso amministrativo

Dopo esserti registrato Account AWS, proteggi Utente root dell'account AWS AWS IAM Identity Center, abilita e crea un utente amministrativo in modo da non utilizzare l'utente root per le attività quotidiane.

Proteggi i tuoi Utente root dell'account AWS

1. Accedi Console di gestione AWScome proprietario dell'account scegliendo Utente root e inserendo il tuo indirizzo Account AWS email. Nella pagina successiva, inserisci la password.

   Per informazioni sull'accesso utilizzando un utente root, consulta la pagina Signing in as the root user della Guida per l'utente di Accedi ad AWS .
2. Abilita l'autenticazione a più fattori (MFA) per l'utente root.

   Per istruzioni, consulta Abilitare un dispositivo MFA virtuale per l'utente Account AWS root (console) nella Guida per l'utente IAM.

Crea un utente con accesso amministrativo

1. Abilita Centro identità IAM.

   Per istruzioni, consulta Abilitazione di AWS IAM Identity Center nella Guida per l'utente di AWS IAM Identity Center .
2. In IAM Identity Center, assegna l'accesso amministrativo a un utente.

   Per un tutorial sull'utilizzo di IAM Identity Center directory come fonte di identità, consulta Configurare l'accesso utente con l'impostazione predefinita IAM Identity Center directory nella Guida per l'AWS IAM Identity Center utente.

Accesso come utente amministratore

- Per accedere con l'utente IAM Identity Center, utilizza l'URL di accesso che è stato inviato al tuo indirizzo e-mail quando hai creato l'utente IAM Identity Center.

  Per informazioni sull'accesso utilizzando un utente IAM Identity Center, consulta AWS Accedere al portale di accesso nella Guida per l'Accedi ad AWS utente.

Assegna l'accesso a ulteriori utenti

1. In IAM Identity Center, crea un set di autorizzazioni conforme alla best practice dell'applicazione di autorizzazioni con il privilegio minimo.

   Segui le istruzioni riportate nella pagina Creazione di un set di autorizzazioni nella Guida per l'utente di AWS IAM Identity Center .
2. Assegna al gruppo prima gli utenti e poi l'accesso con autenticazione unica (Single Sign-On).

   Per istruzioni, consulta Aggiungere gruppi nella Guida per l'utente di AWS IAM Identity Center .

Concessione dell'accesso programmatico

Gli utenti hanno bisogno di un accesso programmatico se vogliono interagire con l' AWS Console di gestione AWS esterno di. Il modo per concedere l'accesso programmatico dipende dal tipo di utente che accede. AWS

Per fornire agli utenti l'accesso programmatico, scegli una delle seguenti opzioni.

Quale utente necessita dell'accesso programmatico?	Per	Come
IAM	(Consigliato) Utilizza le credenziali della console come credenziali temporanee per firmare le richieste programmatiche a,, o. AWS CLI AWS SDKs AWS APIs	Segui le istruzioni per l'interfaccia che desideri utilizzare. • Per la AWS CLI, consulta Login for AWS local development nella Guida per l'AWS Command Line Interface utente. • Per AWS SDKs, consulta Login for AWS local development nella AWS SDKs and Tools Reference Guide.
Identità della forza lavoro	Utilizza credenziali temporanee per firmare le richieste	Segui le istruzioni per l'interfaccia che desideri utilizzare.

Quale utente necessita dell'accesso programmatico?	Per	Come
(Utenti gestiti nel centro identità IAM)	programmatiche a AWS CLI, AWS SDKs, o. AWS APIs	• Per la AWS CLI, vedere Configurazione dell'uso AWS IAM Identity Center nella AWS CLI Guida per l'utente.AWS Command Line Interface • Per AWS SDKs gli strumenti e AWS APIs, consulta l'autenticazione di IAM Identity Center nella Guida di riferimento AWS SDKs and Tools.
IAM	Utilizza credenziali temporane e per firmare le richieste programmatiche a AWS CLI, AWS SDKs, o. AWS APIs	Seguendo le istruzioni riportate in Utilizzo delle credenziali temporanee con le AWS risorse nella Guida per l'utente IAM.

Quale utente necessita dell'accesso programmatico?	Per	Come
IAM	(Non consigliato) Utilizza credenziali a lungo termine per firmare richieste programmatiche a AWS CLI,, AWS SDKs o. AWS APIs	Segui le istruzioni per l'interfaccia che desideri utilizzare. • Per la AWS CLI, consulta Autenticazione tramite credenziali utente IAM nella Guida per l'utente.AWS Command Line Interface • Per gli strumenti AWS SDKs e gli strumenti, consulta Autenticazione tramite credenziali a lungo termine nella Guida di riferimento agli strumenti e agli AWS SDKs strumenti. • Per AWS APIs, consulta la sezione Gestione delle chiavi di accesso per gli utenti IAM nella Guida per l'utente IAM.

Determinazione dei requisiti

Il cluster di database rappresenta l'elemento di base di Aurora. Una o più istanze database possono appartenere a un cluster di database. Un cluster di database fornisce un indirizzo di rete denominato endpoint del cluster. Le applicazioni si connettono all'endpoint del cluster esposto dal cluster di database quando devono accedere ai database creati in quel cluster di database. Le informazioni specificate al momento della creazione del cluster di database controllano elementi di configurazione come memoria, motore di database e versione, configurazione di rete, sicurezza e periodi di manutenzione.

Prima di creare un cluster DB e un gruppo di sicurezza, devi conoscere le necessità del cluster DB e della rete. Ecco alcune cose importanti da considerare:

- Requisiti delle risorse – Quali sono i requisiti di memoria e del processore per l'applicazione o il servizio? Utilizzerai queste impostazioni nella definizione della classe di istanza database da usare durante la creazione del cluster di database. Per specifiche sulle classi di istanza database, consulta Classi di istanze database Amazon Aurora.
- VPC, sottorete e gruppo di sicurezza– Il cluster DB si trova in un Virtual Private Cloud (VPC). È necessario configurare le regole del gruppo di sicurezza per la connessione a un cluster di database. L'elenco seguente descrive le regole per ogni opzione VPC:
  - VPC predefinito: se il tuo AWS account ha un VPC predefinito nella regione AWS , quel VPC è configurato per supportare i cluster DB. Se specifichi il VPC predefinito quando crei il cluster di database:
    - Assicurati di creare un gruppo di sicurezza VPC che autorizza le connessioni dall'applicazione o dal servizio al cluster DB Aurora. Utilizza l'opzione Security Group sulla console VPC o AWS CLI per creare gruppi di sicurezza VPC. Per informazioni, consulta Fase 3: creazione di un gruppo di sicurezza VPC.
    - Devi specificare il gruppo di sottoreti del database predefinito. Se questo è il primo cluster DB creato nella AWS regione, Amazon RDS creerà il gruppo di sottoreti DB predefinito al momento della creazione del cluster DB.
  - VPC definito dall'utente: se vuoi specificare un VPC definito dall'utente quando crei un cluster di database:
    - Assicurati di creare un gruppo di sicurezza VPC che autorizza le connessioni dall'applicazione o dal servizio al cluster DB Aurora. Utilizza l'opzione Security Group sulla console VPC o AWS CLI per creare gruppi di sicurezza VPC. Per informazioni, consulta Fase 3: creazione di un gruppo di sicurezza VPC.
    - Il VPC deve soddisfare certi requisiti per ospitare cluster di database, come avere almeno due sottoreti, ognuna in una zona di disponibilità separata. Per informazioni, consulta VPC Amazon e Amazon Aurora.
    - Devi specificare un gruppo di sottoreti del database che definisce quali sottoreti in quel VPC possono essere utilizzate dal cluster di database. Per informazioni, consulta la sezione relativa al gruppo di sottoreti del database in Uso di un cluster database in un VPC.
- Elevata disponibilità: hai bisogno di supporto per il failover? In Aurora, un'implementazione Multi-AZ crea un'istanza primaria e repliche di Aurora. Puoi configurare l'istanza primaria e le repliche Aurora in modo che si trovino in zone di disponibilità diverse per il supporto per failover. Consigliamo implementazioni Multi-AZ per carichi di lavoro di produzione per mantenere alta disponibilità. Per scopi di sviluppo e test, puoi utilizzare una non-Multi-AZ distribuzione. Per ulteriori informazioni, consulta Elevata disponibilità di Amazon Aurora.

- Policy IAM: il tuo AWS account dispone di politiche che concedono le autorizzazioni necessarie per eseguire le operazioni di Amazon RDS? Se ti connetti AWS tramite credenziali IAM, il tuo account IAM deve disporre di policy IAM che concedano le autorizzazioni necessarie per eseguire le operazioni di Amazon RDS. Per ulteriori informazioni, consulta Gestione accessi e identità per Amazon Aurora.
- Porte aperte: su quale TCP/IP porta sarà in ascolto il tuo database? Il firewall in alcune aziende può bloccare le connessioni alla porta predefinita del motore di database. Se il firewall dell'azienda blocca la porta predefinita, scegliere un'altra porta per il nuovo cluster database. Tieni presente che dopo avere creato un cluster di database in ascolto su una determinata porta, puoi cambiare la porta modificando il cluster di database.
- AWS Regione: in quale AWS regione vuoi inserire il tuo database? Se il database si trova in prossimità dell'applicazione o del servizio Web, la latenza di rete potrebbe ridursi. Per ulteriori informazioni, consulta Regioni e zone di disponibilità.

Quando disponi delle informazioni necessarie per creare il gruppo di sicurezza e il cluster di database, procedi alla fase successiva.

Fornitura dell'accesso al cluster di database nel VPC creando un gruppo di sicurezza

Il cluster DB verrà creato in un VPC. I gruppi di sicurezza forniscono l'accesso al cluster di database nel VPC. Fungono da firewall per il cluster di database associato, controllando il traffico sia in entrata che in uscita a livello di cluster. Per impostazione predefinita, i cluster di database vengono creati con un firewall e un gruppo di sicurezza predefinito che impedisce l'accesso al cluster di database. Pertanto, a un gruppo di sicurezza è necessario aggiungere regole che consentano di connettersi al cluster di database. Utilizza le informazione di rete e di configurazione specificate nella fase precedente per creare regole che consentano l'accesso al cluster di database.

Ad esempio, se hai un'applicazione che accederà a un database nel tuo cluster di database in un VPC, devi aggiungere una regola TCP personalizzata che specifichi l'intervallo di porte e gli indirizzi IP che l'applicazione userà per accedere al database. Se hai un'applicazione su un' EC2 istanza Amazon, puoi utilizzare il gruppo di sicurezza VPC che hai configurato per l'istanza Amazon EC2.

Puoi configurare la connettività tra un' EC2 istanza Amazon e un cluster DB quando crei il cluster DB. Per ulteriori informazioni, consulta Configurazione della connettività di rete automatica con un'istanza EC2.

> **Tip**
>
> Puoi configurare automaticamente la connettività di rete tra un' EC2 istanza Amazon e un cluster DB quando crei il cluster DB. Per ulteriori informazioni, consulta Configurazione della connettività di rete automatica con un'istanza EC2.

Per informazioni su come connettere le risorse in Amazon Lightsail ai cluster DB, consulta Connettere le risorse Lightsail all'utilizzo del peering VPC. Servizi AWS

Per ulteriori informazioni sulla creazione di un VPC da utilizzare con Aurora, consulta Tutorial: Creazione di un Amazon VPC da utilizzare con un cluster database (solo IPv4). Per informazioni sugli scenari comuni per l'accesso a un'istanza database, consult Scenari per accedere a un cluster database in un VPC.

Per creare un gruppo di sicurezza VPC

1. Accedi a Console di gestione AWS e apri la console Amazon VPC all'indirizzo https://console.aws.amazon.com /vpc.

   > **Note**
   >
   > Assicurati di essere nella console VPC, non nella console RDS.

2. Nell'angolo in alto a destra di Console di gestione AWS, scegli la AWS regione in cui desideri creare il gruppo di sicurezza VPC e il cluster DB. Nell'elenco delle risorse Amazon VPC per quella regione AWS , dovresti vedere almeno un VPC e diverse sottoreti. In caso contrario, non disponi di un VPC predefinito in quella AWS regione.
3. Fare clic su Security Groups (Gruppi di sicurezza) nel pannello di navigazione.
4. Scegliere Create Security Group (Crea gruppo di sicurezza).

   Viene visualizzata la pagina Create security group (Crea gruppo di sicurezza).
5. In Basic details (Dettagli di base), immettere il Security group name (Nome del gruppo di sicurezza) e la Description (Descrizione). Per VPC, seleziona il VPC nel quale desideri creare il cluster DB.
6. Per Inbound rules (Regole in entrata), scegli Add rule (Aggiungi regola).

   a. Per Type (Tipo), scegliere Custom TCP (TCP personalizzato).

b. Per Port Range (Intervallo porte), digita il valore della porta da utilizzare per il cluster DB.

c. Per Source (Origine), seleziona il nome del gruppo di sicurezza o digita l'intervallo di indirizzi IP (valore CIDR) da dove accedi al cluster DB. Se scegli My IP (Il mio IP), questo consente l'accesso al cluster DB dall'indirizzo IP rilevato nel browser.

7. Se occorre aggiungere altri indirizzi IP o intervalli di porta diversi, scegliere Add rule (Aggiungi regola) e immettere le informazioni relative alla regola.
8. (Facoltativo) In Outbound Rules (Regole in uscita), aggiungi regole per il traffico in uscita. Come impostazione predefinita, tutto il traffico in uscita è permesso.
9. Scegliere Create Security Group (Crea gruppo di sicurezza).

Si può utilizzare il gruppo di sicurezza VPC appena creato come gruppo di sicurezza per il cluster di database al momento della creazione.

> **Note**
>
> Se utilizzi un VPC predefinito, viene creato un gruppo di sottoreti predefinito che include tutte le sottoreti VPC. Quando si crea un cluster di database, è possibile selezionare il VPC predefinito e utilizzare default (predefinito) per DB Subnet Group (Gruppo di sottoreti del database).

Quando hai completato i requisiti di configurazione, puoi creare un cluster DB utilizzando i requisiti e il gruppo di sicurezza seguendo le istruzioni in Creazione di un cluster database Amazon Aurora. Per informazioni su come iniziare con la creazione di un cluster DB che utilizza un motore DB specifico, consulta Nozioni di base su Amazon Aurora.

Nozioni di base su Amazon Aurora

In questa sezione viene descritto come creare e connettersi ad un cluster di database Aurora usando Amazon RDS.

Le procedure seguenti sono dei tutorial che offrono le nozioni di base su Aurora. Le sezioni successive presentano procedure e concetti Aurora più avanzati, ad esempio i vari tipi di endpoint e le modalità di dimensionamento dei cluster Aurora.

> **Important**
>
> Devi completare le attività indicate nella sezione Configurazione dell'ambiente per Amazon Aurora prima di poter creare o connettersi ad un cluster database .

Argomenti

- Creazione e connessione a un cluster di database Aurora MySQL
- Creazione e connessione di un cluster di database Aurora PostgreSQL
- Tutorial: creazione di un server Web e un cluster database Amazon Aurora

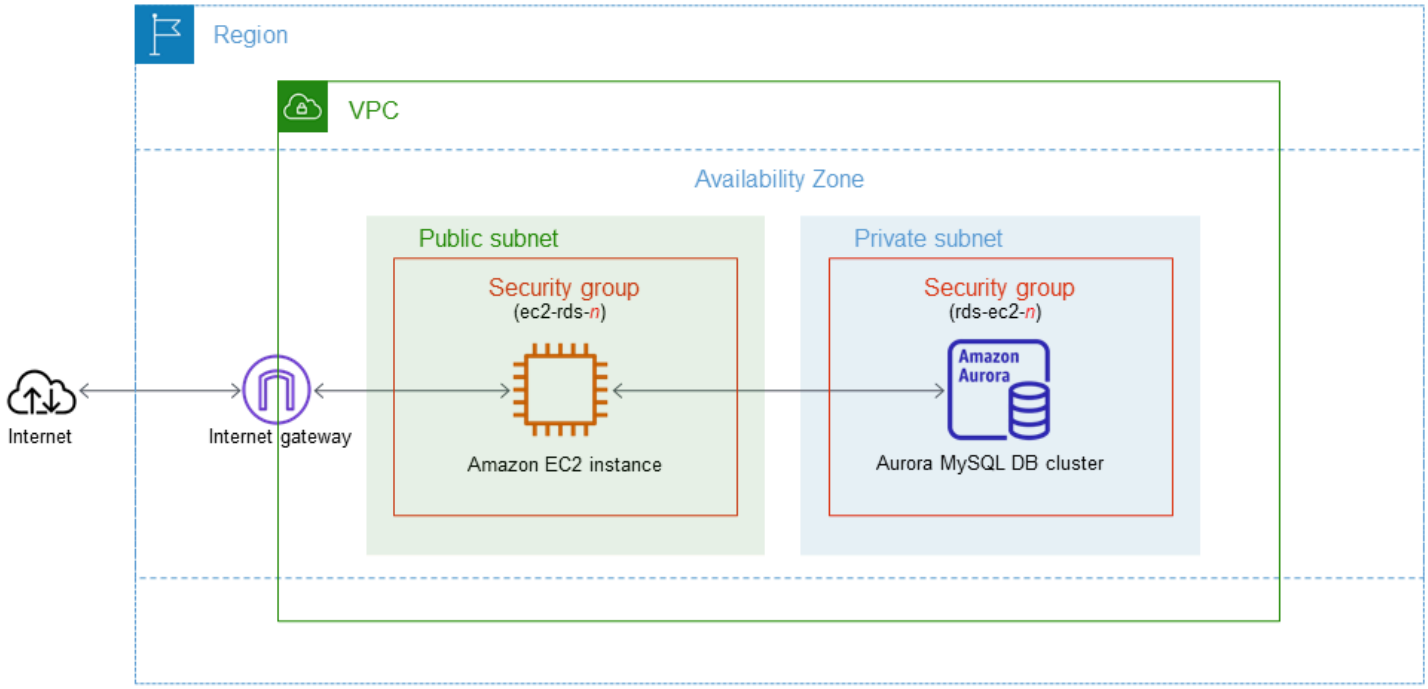
Creazione e connessione a un cluster di database Aurora MySQL

Questo tutorial crea un' EC2 istanza e un cluster Aurora MySQL DB. Il tutorial mostra come accedere al cluster DB dall' EC2 istanza utilizzando un client MySQL standard. Come best practice, questo tutorial spiega come creare un cluster di database privato in un cloud privato virtuale (VPC). Nella maggior parte dei casi, altre risorse nello stesso VPC, come EC2 le istanze, possono accedere al cluster DB, ma le risorse esterne al VPC non possono accedervi.

Dopo aver completato il tutorial, è presente una sottorete pubblica e una privata in ogni zona di disponibilità del VPC. In una zona di disponibilità, l' EC2 istanza si trova nella sottorete pubblica e l'istanza DB si trova nella sottorete privata.

> **Important**
>
> La creazione di un AWS account è gratuita. Tuttavia, completando questo tutorial, potresti incorrere in costi per le AWS risorse che utilizzi. È possibile eliminare queste risorse dopo aver completato l'esercitazione se non sono più necessarie.

Il seguente diagramma illustra la configurazione al completamento del tutorial.



Questo tutorial consente di creare le risorse utilizzando uno dei seguenti metodi:

1. Usa - e Console di gestione AWS Fase 1: Creare un' EC2istanza Fase 2: creazione di un cluster di database Aurora MySQL
2. Utilizzare CloudFormation per creare l'istanza e EC2 l'istanza del database - (Facoltativo) Crea VPC, EC2 istanza e cluster Aurora MySQL utilizzando CloudFormation

Il primo metodo utilizza Creazione semplice per creare un cluster Aurora MySQL DB privato con la Console di gestione AWS. Si specifica solo il tipo di motore di database, la dimensione dell'istanza database e l'identificatore del cluster di database. Easy create (Creazione rapida) utilizza l'impostazione predefinita per altre opzioni di configurazione.

Quando invece si utilizza la Creazione standard, si specificano più opzioni di configurazione al momento della creazione di un cluster di database. Queste opzioni includono impostazioni per la disponibilità, la sicurezza, i backup e la manutenzione. Per creare un cluster di database pubblico, è necessario utilizzare la Creazione standard. Per informazioni, consulta the section called “Creazione di un cluster di database”.

Argomenti

- Prerequisiti
- Fase 1: Creare un' EC2istanza
- Fase 2: creazione di un cluster di database Aurora MySQL
- (Facoltativo) Crea VPC, EC2 istanza e cluster Aurora MySQL utilizzando CloudFormation
- Fase 3: connessione a un cluster di database Aurora MySQL
- Passaggio 4: Eliminare l' EC2 istanza e il cluster DB
- (Facoltativo) Elimina l' EC2 istanza e il cluster DB creati con CloudFormation
- (Facoltativo) Connessione del cluster database a una funzione Lambda

Prerequisiti

Prima di iniziare, completa le fasi descritte in questa sezione:

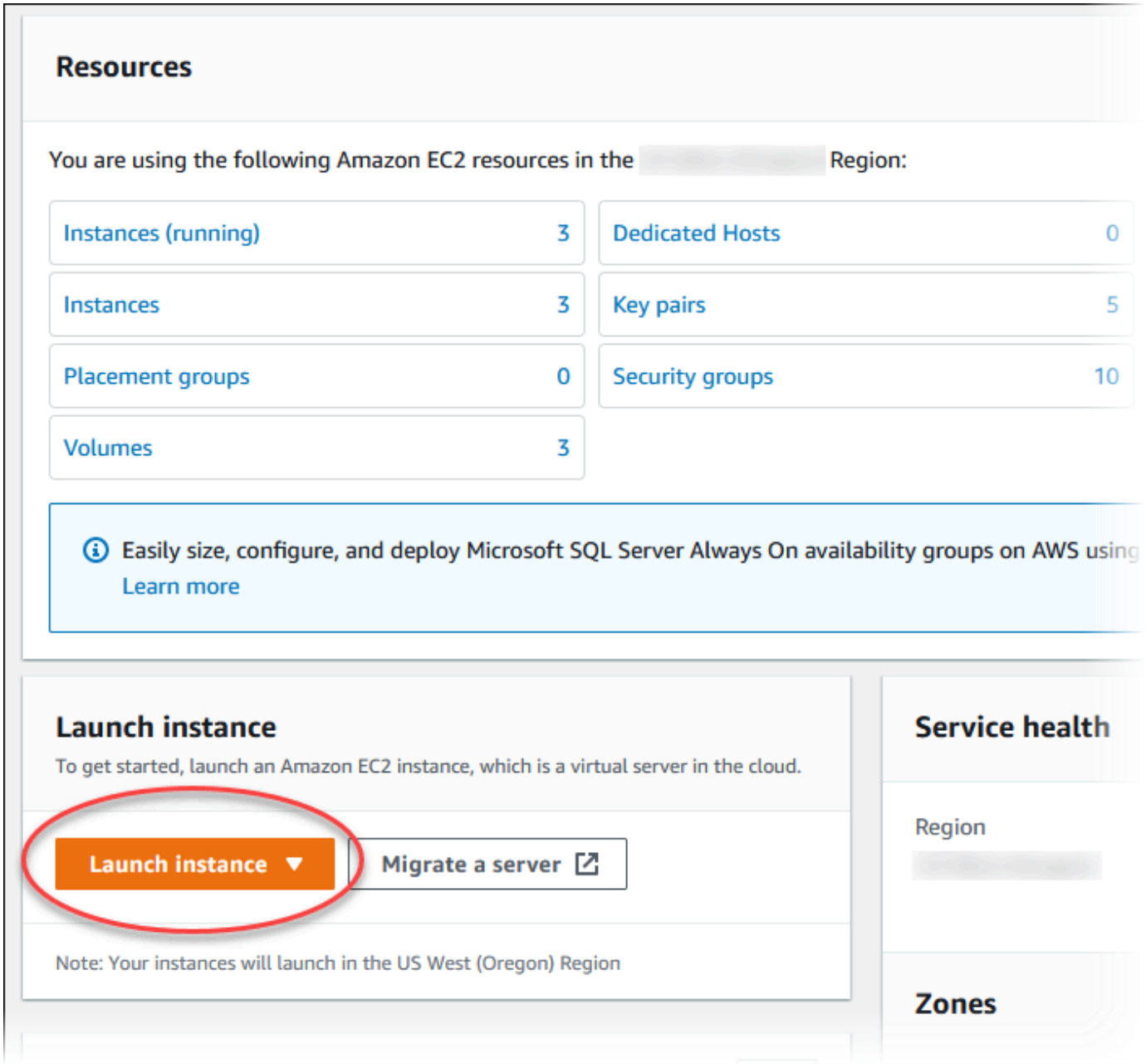
- Registrati per un Account AWS
- Crea un utente con accesso amministrativo

Fase 1: Creare un' EC2istanza

Crea un' EC2 istanza Amazon da utilizzare per connetterti al tuo database.

Per creare un' EC2 istanza

1. Accedi a Console di gestione AWS e apri la EC2 console Amazon all'indirizzo https://console.aws.amazon.com/ec2/.
2. Nell'angolo in alto a destra di Console di gestione AWS, scegli l'istanza Regione AWS in cui desideri creare l'istanza. EC2
3. Scegli EC2 Dashboard, quindi scegli Launch instance, come mostrato nell'immagine seguente.

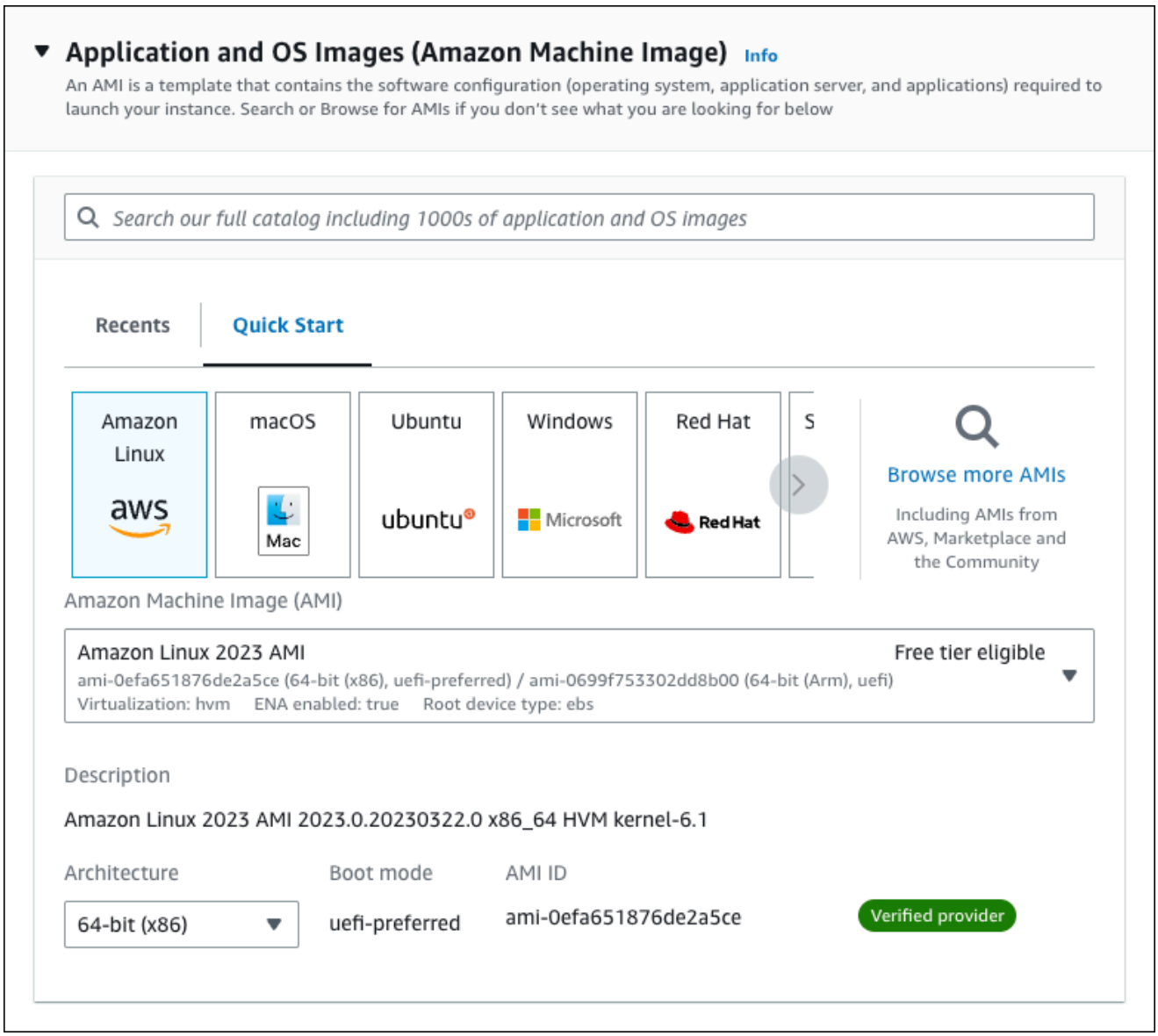


Viene visualizzata la pagina Avvia un'istanza.

4. Scegli le seguenti impostazioni nella pagina Avvia un'istanza.

   a. Nell'area Name and tags (Nome e tag), in Name (Nome) inserisci **`ec2-database-connect`**.

   b. In Immagini applicazione e sistema operativo (Amazon Machine Image), scegli Amazon Linux, quindi AMI Amazon Linux 2023. Mantieni le selezioni predefinite per le altre opzioni.



c. In Instance type (Tipo di istanza), scegli t2.micro.

d. In Key pair (login) (Coppia di chiavi (login), per Key pair name (Nome della coppia di chiavi), scegli una coppia di chiavi esistente. Per creare una nuova coppia di chiavi per l' EC2 istanza Amazon, scegli Crea nuova coppia di chiavi e usa la finestra Crea key pair per crearla.

   Per ulteriori informazioni sulla creazione di una nuova coppia di chiavi, consulta Create a key pair nella Amazon EC2 User Guide.

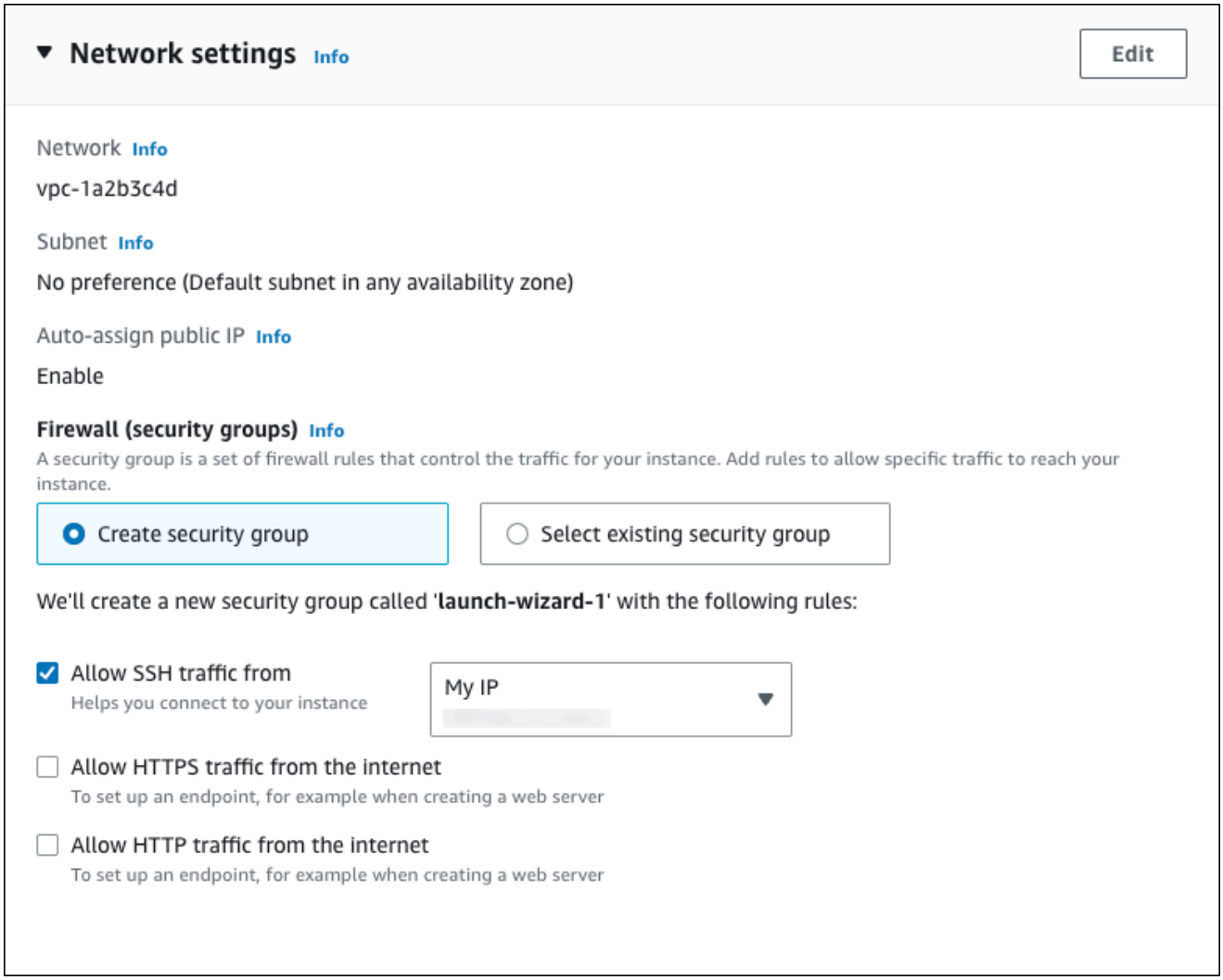
e. Per Consenti il traffico SSH nelle impostazioni di rete, scegli l'origine delle connessioni SSH all' EC2istanza.

È possibile scegliere My IP (Il mio IP) se l'indirizzo IP visualizzato è corretto per le connessioni SSH. Altrimenti, puoi determinare l'indirizzo IP da utilizzare per connetterti alle EC2 istanze nel tuo VPC utilizzando Secure Shell (SSH). Per determinare il tuo indirizzo IP pubblico, in un'altra finestra o scheda del browser, puoi utilizzare il servizio all'indirizzo. https://checkip.amazonaws.com Un esempio di indirizzo IP è 192.0.2.1/32.

In molti casi, è possibile eseguire la connessione tramite un fornitore di servizi Internet (ISP) o con la protezione di un firewall senza un indirizzo IP statico. In tal caso, accertati di determinare l'intervallo di indirizzi IP utilizzati dai computer client.

> **Warning**
>
> Se si utilizza `0.0.0.0/0` l'accesso SSH, si consente a tutti gli indirizzi IP di accedere alle EC2 istanze pubbliche tramite SSH. Questo approccio è accettabile per un breve periodo di tempo in un ambiente di test, ma non è sicuro per gli ambienti di produzione. In produzione, autorizzate solo un indirizzo IP specifico o un intervallo di indirizzi per accedere alle istanze tramite EC2 SSH.

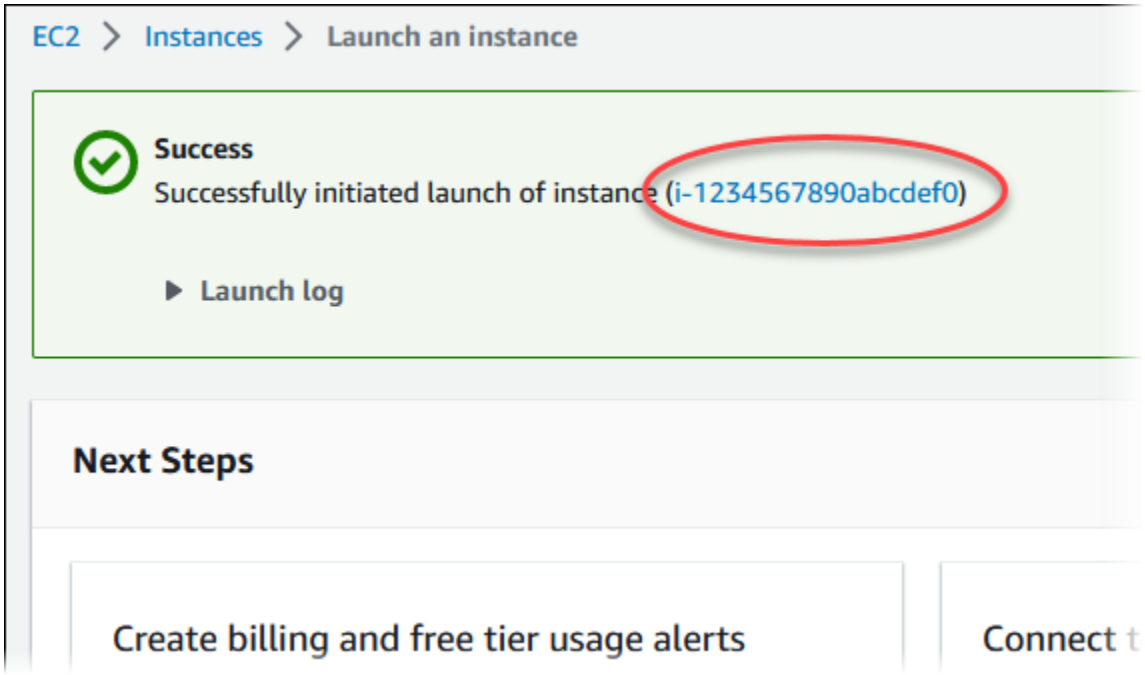
L'immagine seguente mostra un esempio della sezione Impostazioni di rete.



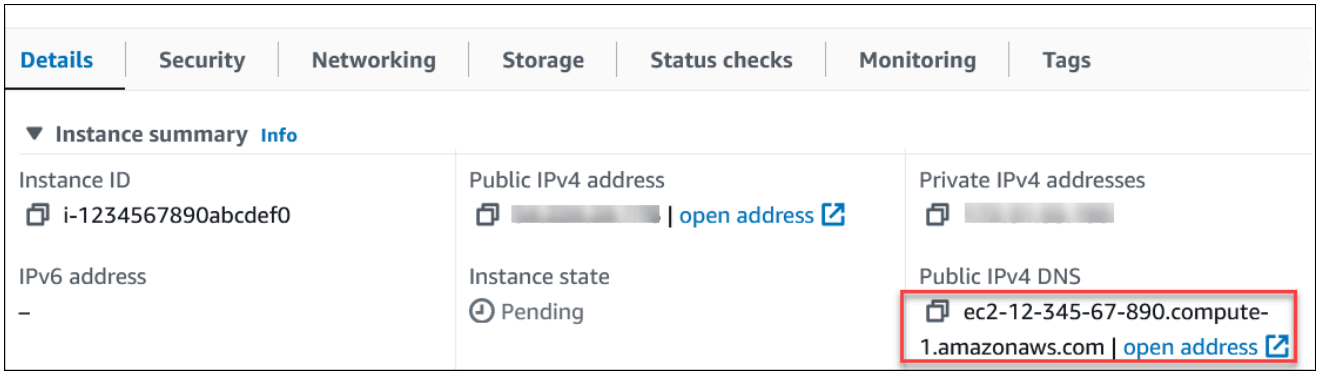
f. Lascia i valori predefiniti per le sezioni rimanenti.

g. Esamina un riepilogo della configurazione dell' EC2 istanza nel pannello Riepilogo e, quando sei pronto, scegli Launch instance.

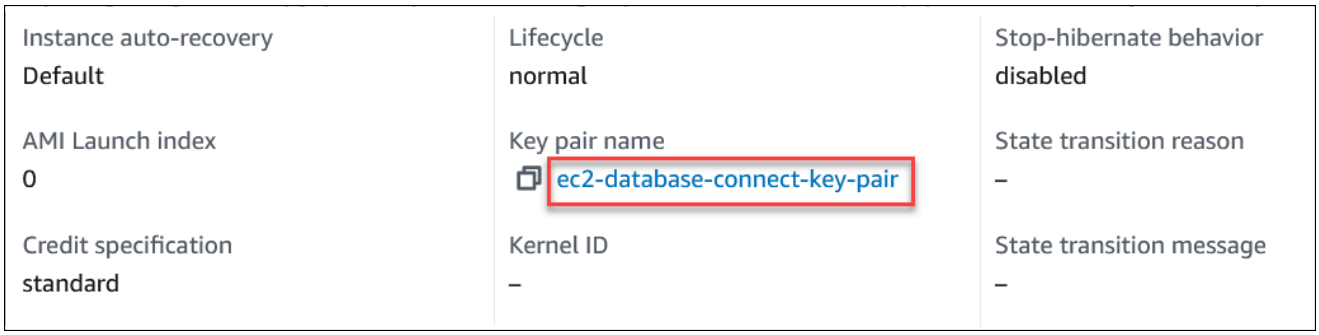
5. Nella pagina Launch Status, prendi nota dell'identificatore della tua nuova EC2 istanza, ad esempio:`i-1234567890abcdef0`.



6. Scegli l'identificatore dell' EC2 istanza per aprire l'elenco delle EC2 istanze, quindi seleziona l'istanza. EC2
7. Nella scheda Dettagli, annota i seguenti valori, necessari quando ti connetti tramite SSH:
   a. Nel riepilogo dell'istanza, annota il valore per Public IPv4 DNS.



   b. In Dettagli istanza, annota il valore visualizzato in Nome coppia di chiavi.



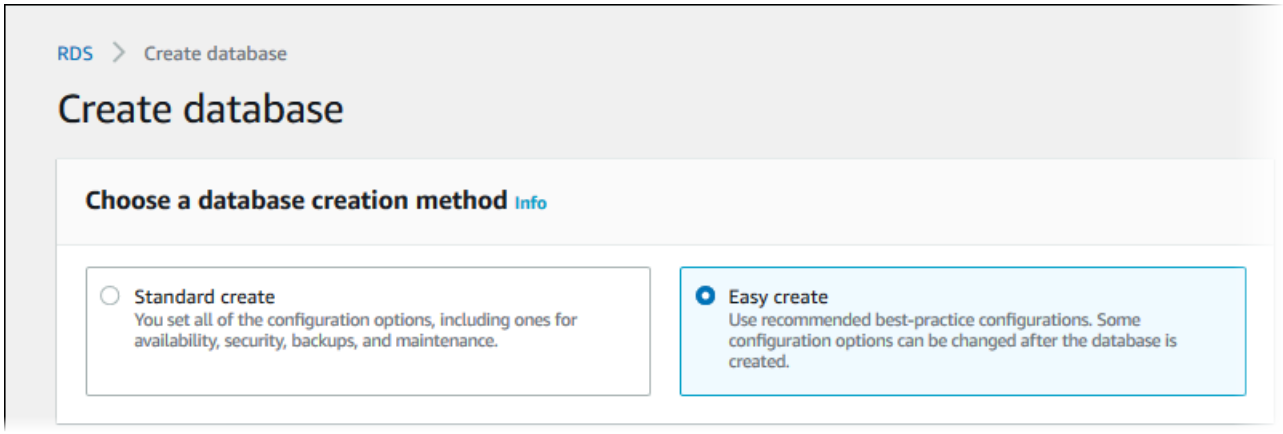
8. Attendi che lo stato dell' EC2 istanza abbia lo stato In esecuzione prima di continuare.

Fase 2: creazione di un cluster di database Aurora MySQL

In questo esempio, utilizzi la Creazione semplice per creare un cluster di database Aurora MySQL con una classe di istanza database db.r6g.large.

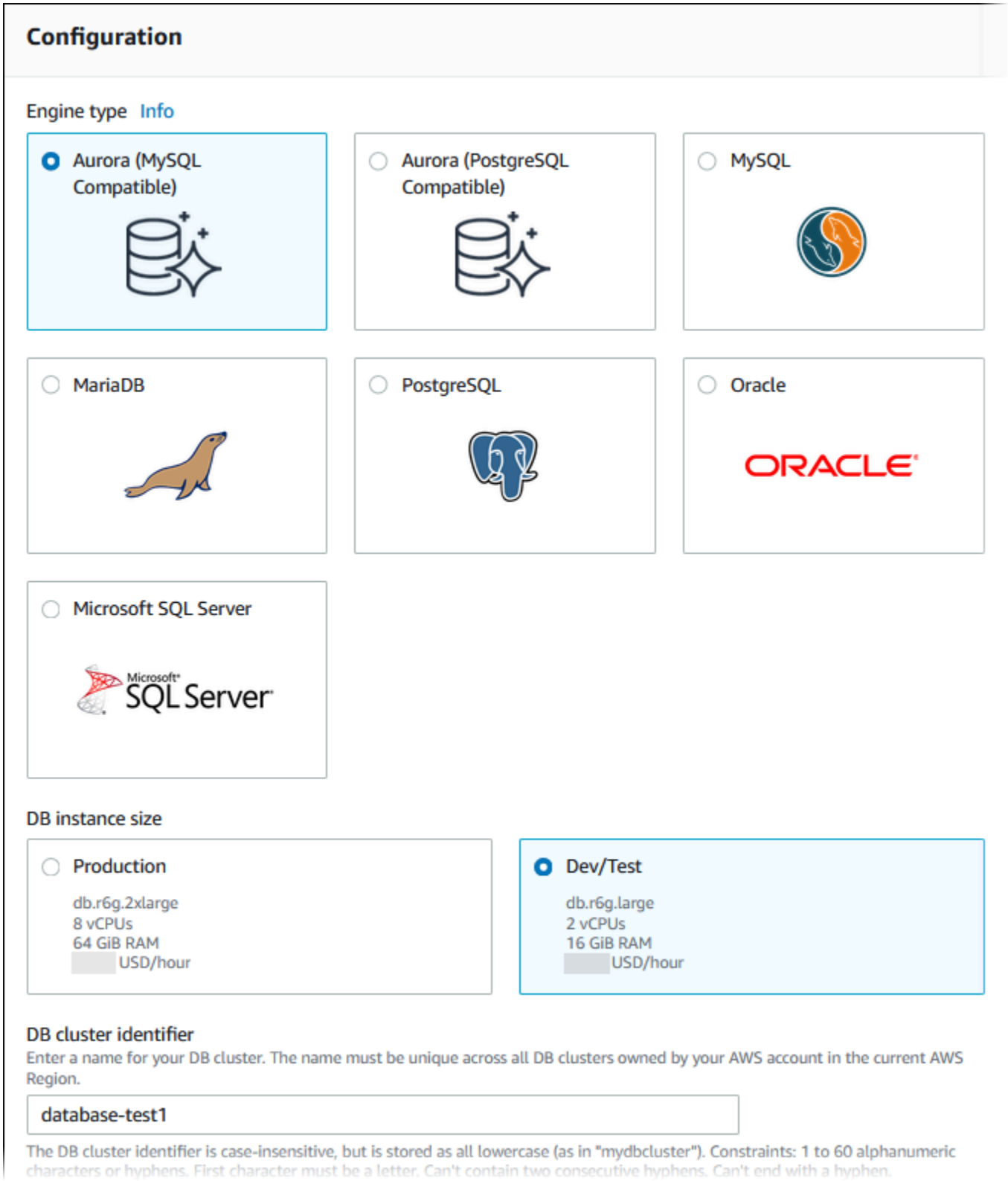
Per creare un cluster di database Aurora MySQL con Creazione semplice

1. Accedi a Console di gestione AWS e apri la console Amazon RDS all'indirizzo https://console.aws.amazon.com/rds/.
2. Nell'angolo in alto a destra della console Amazon RDS, scegli il cluster di database Regione AWS in cui desideri creare il cluster DB.
3. Nel riquadro di navigazione, scegliere Databases (Database).
4. Scegliere Create database (Crea database) e verificare che l'opzione Easy Create (Creazione rapida) sia selezionata.



5. In Configurazione scegli Aurora (compatibile con MySQL) per Tipo di motore.
6. Per DB instance size (Dimensione istanza database), seleziona Dev/Test.
7. Per Identificatore cluster di database, immetti **`database-test1`**.

   La pagina Create database (Crea database) la pagina dovrebbe apparire simile alla seguente immagine.

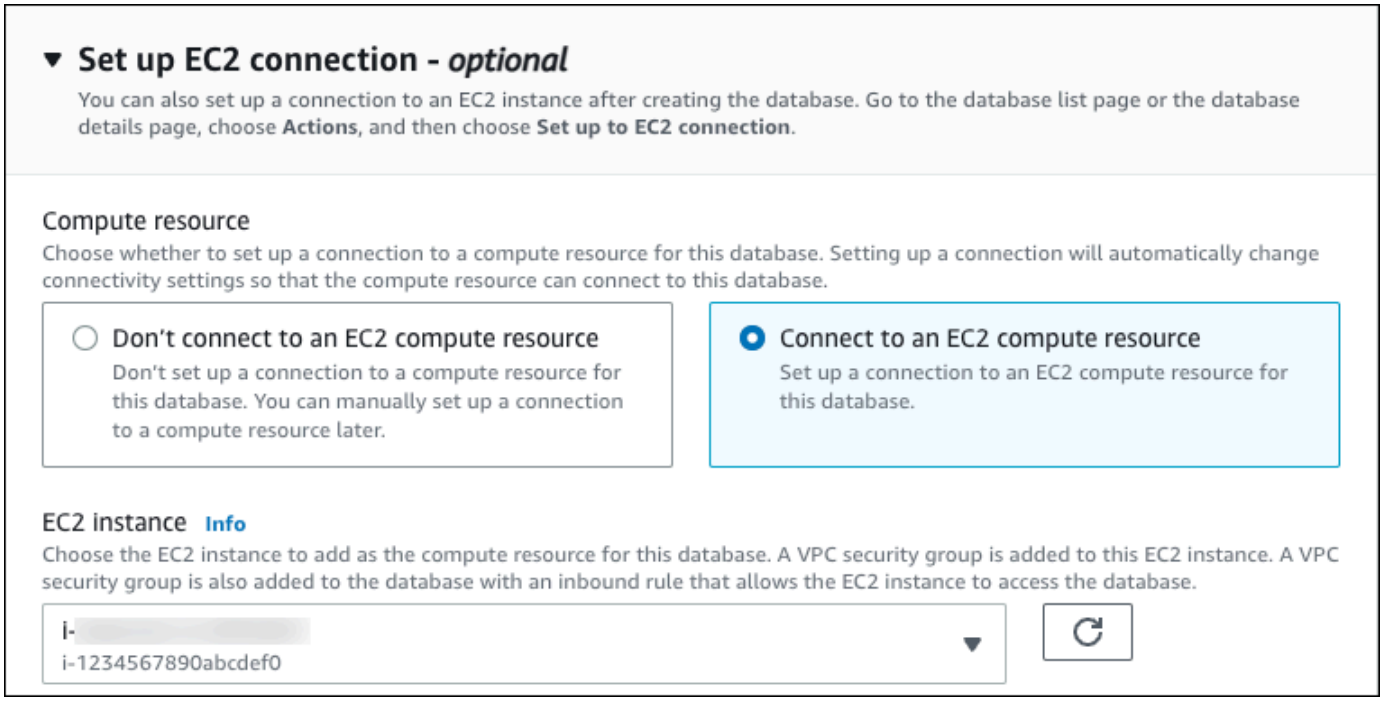


8. Per Nome utente master, inserisci un nome per l'utente master o lascia il nome predefinito.
9. Per utilizzare una password master generata automaticamente per il cluster di database, seleziona Genera automaticamente una password.

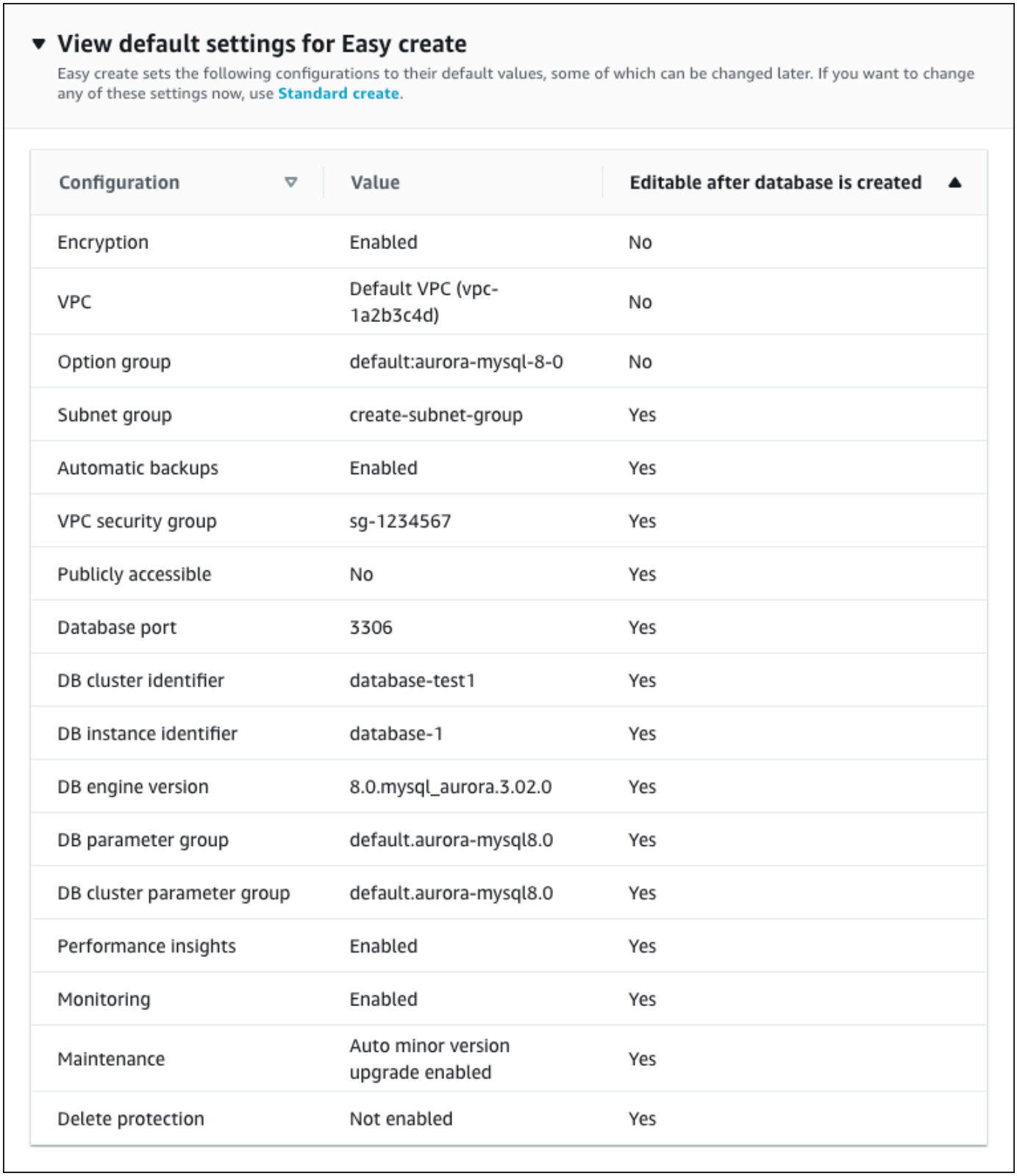
Per inserire la password master, deseleziona la casella Genera automaticamente una password e inserisci la stessa password in Password master e Conferma password.

10. Per configurare una connessione con l' EC2 istanza creata in precedenza, apri Configura EC2 connessione (opzionale).

    Seleziona Connetti a una risorsa di EC2 calcolo. Scegli l'EC2 istanza che hai creato in precedenza.



11. Apri Visualizza le impostazioni predefinite per la creazione Semplice.



View default settings for Easy create

Easy create sets the following configurations to their default values, some of which can be changed later. If you want to change any of these settings now, use **Standard create**.

Configuration	Value	Editable after database is created
Encryption	Enabled	No
VPC	Default VPC (vpc-1a2b3c4d)	No
Option group	default:aurora-mysql-8-0	No
Subnet group	create-subnet-group	Yes
Automatic backups	Enabled	Yes
VPC security group	sg-1234567	Yes
Publicly accessible	No	Yes
Database port	3306	Yes
DB cluster identifier	database-test1	Yes
DB instance identifier	database-1	Yes
DB engine version	8.0.mysql_aurora.3.02.0	Yes
DB parameter group	default.aurora-mysql8.0	Yes
DB cluster parameter group	default.aurora-mysql8.0	Yes
Performance insights	Enabled	Yes
Monitoring	Enabled	Yes
Maintenance	Auto minor version upgrade enabled	Yes
Delete protection	Not enabled	Yes

Puoi esaminare le impostazioni predefinite utilizzate con Easy create (Creazione rapida). La colonna Modificabile dopo la creazione del database mostra le opzioni che puoi modificare dopo aver creato il database.

- Se un'impostazione contiene No in quella colonna e desideri cambiarla, puoi utilizzare la Creazione standard per creare il cluster di database.
- Se un'impostazione contiene Sì in quella colonna e desideri cambiarla, puoi utilizzare la Creazione standard per creare il cluster di database o modificare il cluster di database dopo averlo creato per cambiare l'impostazione.

12. Scegliere Crea database.

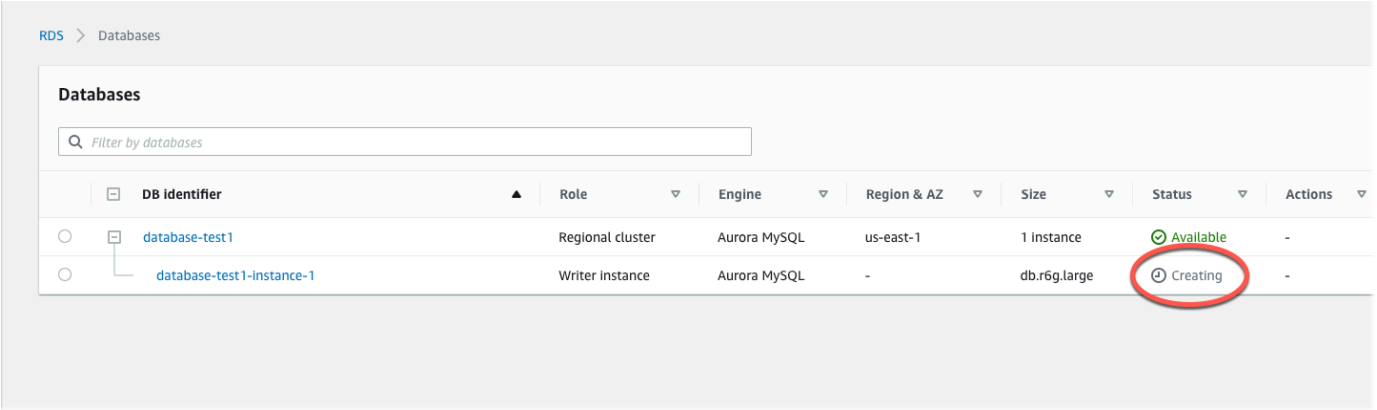
    Per vedere nome utente e password per il cluster di database, seleziona Visualizza i dettagli delle credenziali.

    Per connetterti al cluster di database come utente master, utilizza il nome utente e la password visualizzati.

    > **Important**
    >
    > Non potrai visualizzare di nuovo la password dell'utente principale. Se non la registri, potresti doverla modificare.
    > Se devi modificare la password dell'utente principale dopo che il cluster database è disponibile, puoi modificare il cluster database per eseguire tale operazione. Per ulteriori informazioni sulla modifica di un cluster di database, consultare Modifica di un cluster database Amazon Aurora.

13. Nell'elenco Database seleziona il nome del nuovo cluster di database Aurora MySQL per visualizzarne i dettagli.

    L'istanza di scrittura ha lo stato Creazione in corso fino a quando il cluster di database non è pronto per essere utilizzato.



Quando lo stato dell'istanza di scrittura cambia in Disponibile, puoi connetterti al cluster di database. A seconda della classe di istanza database e della quantità di storage, prima che il nuovo cluster di database sia disponibile possono trascorrere fino a 20 minuti.

(Facoltativo) Crea VPC, EC2 istanza e cluster Aurora MySQL utilizzando CloudFormation

Invece di usare la console per creare il VPC, l' EC2 istanza e il cluster Aurora MySQL DB, puoi usarla CloudFormation per fornire risorse trattando l'infrastruttura come codice. AWS Per aiutarti a organizzare AWS le tue risorse in unità più piccole e più gestibili, puoi utilizzare la funzionalità nested stack. CloudFormation Per ulteriori informazioni, consulta Creare uno stack sulla CloudFormation console e Lavorare con gli stack annidati.

> **Important**
>
> CloudFormation è gratuito, ma le risorse che CloudFormation crea sono attive. Il costo di utilizzo standard per queste risorse sarà addebitato finché non vengono terminate. Per ulteriori informazioni, consulta Prezzi di Amazon Aurora.

Per creare le tue risorse utilizzando la CloudFormation console, completa i seguenti passaggi:

- Passaggio 1: scarica il CloudFormation modello
- Passaggio 2: configura le tue risorse utilizzando CloudFormation

Scarica il CloudFormation modello

Un CloudFormation modello è un file di testo JSON o YAML che contiene le informazioni di configurazione sulle risorse che desideri creare nello stack. Questo modello crea automaticamente anche un VPC e un host bastione insieme al cluster Aurora.

Per scaricare il file modello, apri il seguente link, Modello Aurora MySQL. CloudFormation

Nella pagina Github, fare clic sul pulsante Scarica file non elaborato per salvare il file YAML.

Configura le tue risorse usando CloudFormation

> **Note**
>
> Prima di iniziare questo processo, assicurati di avere una coppia di chiavi per un' EC2 istanza nel tuo Account AWS. Per ulteriori informazioni, consulta coppie di EC2 chiavi Amazon e istanze Linux.

Quando utilizzi il CloudFormation modello, devi selezionare i parametri corretti per assicurarti che le risorse vengano create correttamente. Segui la procedura riportata di seguito:

1. Accedi Console di gestione AWS e apri la CloudFormation console all'indirizzo https://console.aws.amazon.com/cloudformazione.
2. Scegli Crea stack.
3. Nella sezione in cui specificare il modello, seleziona Carica un file modello dal computer, quindi Avanti.
4. Nella pagina Specifica i dettagli dello stack, impostare i seguenti parametri:
   a. Imposta il nome dello stack su Stack. AurMy SQLTest
   b. In Parametri, impostare Zone di disponibilità selezionando due zone di disponibilità.
   c. Nella configurazione Linux Bastion Host, in Key Name, seleziona una coppia di chiavi per accedere alla tua EC2 istanza.
   d. Nelle impostazioni di Configurazione host bastione di Linux, imposta Intervallo IP consentito sul tuo indirizzo IP. Per connetterti alle EC2 istanze nel tuo VPC utilizzando Secure Shell (SSH), determina il tuo indirizzo IP pubblico utilizzando il servizio all'indirizzo. https://checkip.amazonaws.com Un esempio di indirizzo IP è 192.0.2.1/32.

> **Warning**
>
> Se utilizzi `0.0.0.0/0` l'accesso SSH, consenti a tutti gli indirizzi IP di accedere alle tue istanze pubbliche EC2 tramite SSH. Questo approccio è accettabile per un breve periodo di tempo in un ambiente di test, ma non è sicuro per gli ambienti di produzione. In produzione, autorizzate solo un indirizzo IP specifico o un intervallo di indirizzi per accedere alle istanze tramite EC2 SSH.

e. In Configurazione generale del database, impostare la Classe di istanza del database su db.r6g.large.

f. Impostare il Nome del database su **`database-test1`**.

g. Per Nome utente principale del database, specifica il nome dell'utente master.

h. Imposta Gestisci password utente master di database con Secrets Manager su `false` per questo tutorial.

i. Per Database password, imposta una password di tua scelta. Tenere a mente questa password per altri passaggi del tutorial.

j. Impostare Implementazione Multi-AZ su `false`.

k. Lasciare i valori predefiniti per tutte le altre impostazioni. Fai clic su Next per continuare.

5. Nella pagina Configura opzioni dello stack, lasciare tutte le opzioni predefinite. Fai clic su Next per continuare.
6. Nella pagina Rivedi stack, seleziona Invia dopo aver verificato le opzioni del database e dell'host bastione di Linux.

Una volta completato il processo di creazione dello stack, visualizza gli stack con nomi BastionStacke AMSNS per annotare le informazioni necessarie per connetterti al database. Per ulteriori informazioni, consulta Visualizzazione dei dati e delle risorse CloudFormation dello stack su. Console di gestione AWS

Fase 3: connessione a un cluster di database Aurora MySQL

È possibile utilizzare qualsiasi applicazione client SQL standard per la connessione al cluster di database. In questo esempio, ti connetti a un cluster di database Aurora MySQL utilizzando il client della linea di comando mysql.

Per connettersi al cluster di database Aurora MySQL

1. Individua l'endpoint (nome DNS) e il numero di porta per l'istanza di scrittura del cluster di database.

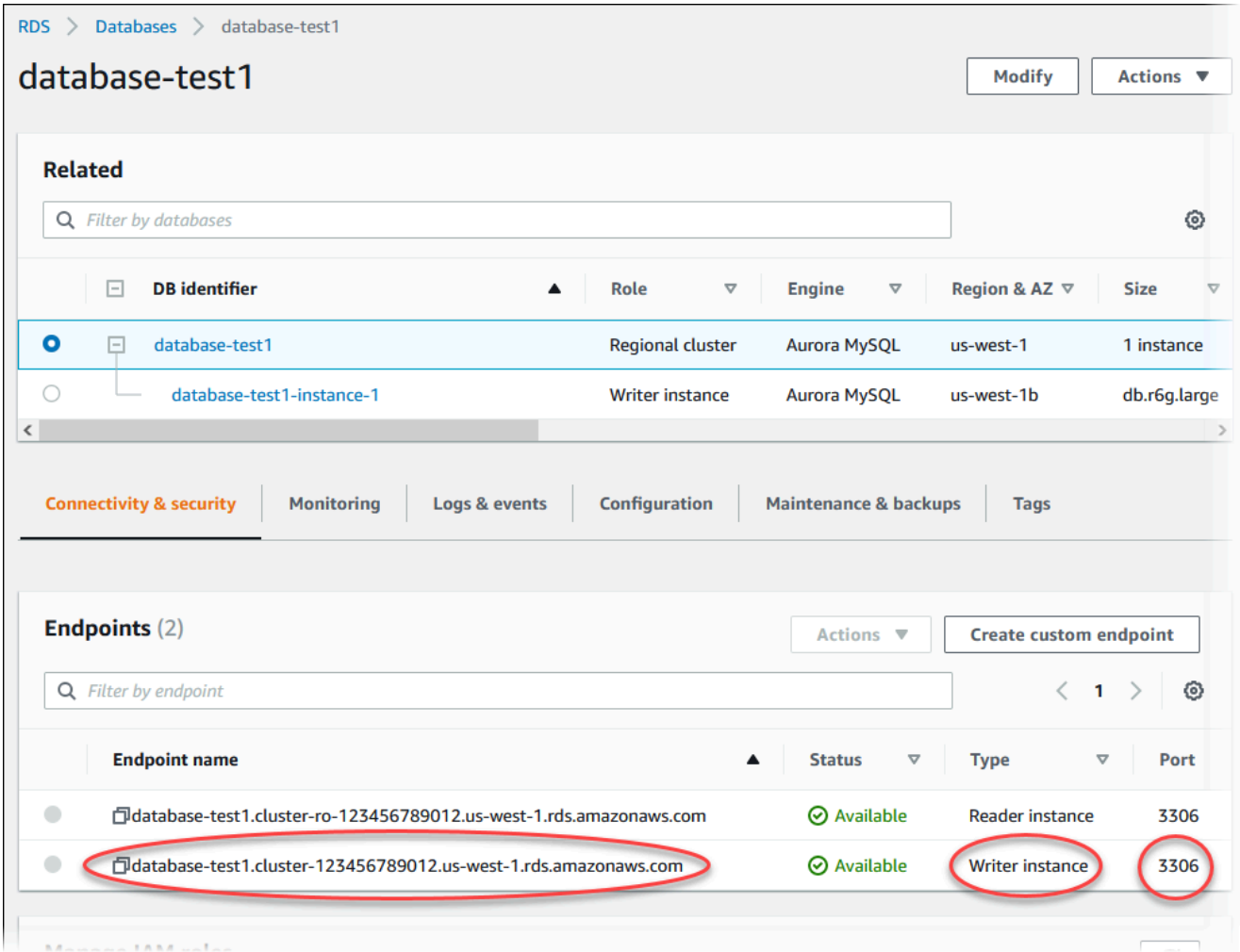
   a. Accedi a Console di gestione AWS e apri la console Amazon RDS all'indirizzo https://console.aws.amazon.com/rds/.

   b. Nell'angolo superiore destro della console Amazon RDS, scegli la Regione AWS del cluster di database.

   c. Nel pannello di navigazione, seleziona Database.

   d. Seleziona il nome di un cluster di database Aurora MySQL per visualizzarne i dettagli.

   e. Nella scheda Connettività e sicurezza, copia l'endpoint dell'istanza di scrittura. Annotare anche il numero di porta. L'endpoint e il numero di porta sono necessari per la connessione al cluster di database.



2. Connettiti all' EC2 istanza che hai creato in precedenza seguendo i passaggi in Connetti alla tua istanza Linux nella Amazon EC2 User Guide.

   Ti consigliamo di connetterti alla tua EC2 istanza tramite SSH. Se l'utilità client SSH è installata su Windows, Linux o Mac, puoi connetterti all'istanza utilizzando il comando nel seguente formato:

   ```
   ssh -i location_of_pem_file ec2-user@ec2-instance-public-dns-name
   ```

   Ad esempio, supponiamo che `ec2-database-connect-key-pair.pem` sia archiviato `/dir1` su Linux e che il IPv4 DNS pubblico dell'istanza lo sia. EC2 `ec2-12-345-678-90.compute-1.amazonaws.com` Quindi, il comando SSH sarà simile al seguente:

   ```
   ssh -i /dir1/ec2-database-connect-key-pair.pem ec2-
   user@ec2-12-345-678-90.compute-1.amazonaws.com
   ```

3. Ottieni le correzioni di bug e gli aggiornamenti di sicurezza più recenti aggiornando il software sulla tua istanza. EC2 A tale scopo, utilizzare il comando seguente.

   > **Note**
   >
   > L'opzione `-y` installa gli aggiornamenti senza chiedere conferma. Per esaminare gli aggiornamenti prima di installarli, omettere questa opzione.

   ```
   sudo dnf update -y
   ```

4. Per installare il client della linea di comando mysql da MariaDB su Amazon Linux 2023, esegui il comando seguente:

   ```
   sudo dnf install mariadb105
   ```

5. Connessione a un cluster di database Aurora MySQL. Ad esempio, immetti il comando seguente: Questa azione consente di connetterti al cluster di database Aurora MySQL utilizzando il client MySQL.

Sostituisci l'endpoint dell'istanza di scrittura per *endpoint* e sostituisci il nome utente master utilizzato per *admin*. Devi fornire la password master utilizzata quando viene richiesta una password.

```
mysql -h endpoint -P 3306 -u admin -p
```

Dopo aver immesso la password per l'utente, l'output dovrebbe essere analogo a quanto mostrato di seguito.

```
Welcome to the MariaDB monitor.  Commands end with ; or \g.
Your MySQL connection id is 217
Server version: 8.0.23 Source distribution

Copyright (c) 2000, 2018, Oracle, MariaDB Corporation Ab and others.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

MySQL [(none)]>
```

Per ulteriori informazioni sulla connessione a un cluster di database Aurora MySQL, consulta Connessione a un cluster di database Amazon Aurora MySQL. In caso di mancata connessione al cluster database, consulta Impossibile connettersi all'istanza database di Amazon RDS.

Per motivi di sicurezza, la best practice è utilizzare connessioni crittografate. Utilizzare una connessione MySQL non crittografata solo quando il client e il server sono nello stesso VPC e la rete è attendibile. Per ulteriori informazioni sull'uso di connessioni crittografate, consulta Connessione ad Aurora MySQL con SSL.

6. Eseguire comandi SQL.

   Ad esempio, il seguente comando SQL mostra la data e l'ora correnti:

   ```
   SELECT CURRENT_TIMESTAMP;
   ```

Passaggio 4: Eliminare l' EC2 istanza e il cluster DB

Dopo esserti connesso ed esplorato l'istanza di EC2 esempio e il cluster DB che hai creato, eliminali in modo che non ti vengano più addebitati costi.

Se in passato CloudFormation creavi risorse, salta questo passaggio e vai al passaggio successivo.

Per eliminare l'istanza EC2

1. Accedi a Console di gestione AWS e apri la EC2 console Amazon all'indirizzo https://console.aws.amazon.com/ec2/.
2. Nel riquadro di navigazione, scegliere Instances (Istanze).
3. Seleziona l' EC2 istanza e scegli Stato dell'istanza, Termina istanza.
4. Quando viene richiesta la conferma, seleziona Interrompi.

Per ulteriori informazioni sull'eliminazione di un' EC2 istanza, consulta Terminate your istance nella Amazon EC2 User Guide.

Per eliminare un cluster di database

1. Accedi a Console di gestione AWS e apri la console Amazon RDS all'indirizzo https://console.aws.amazon.com/rds/.
2. Scegliere Databases (Database), quindi selezionare l'istanza database associata con il cluster di database.
3. In Actions (Azioni), selezionare Delete (Elimina).
4. Deseleziona Creare uno snapshot finale?
5. Completa la conferma e scegli Elimina.

Dopo aver eliminato tutte le istanze database associate a un cluster di database, il cluster di database viene eliminato automaticamente.

(Facoltativo) Elimina l' EC2 istanza e il cluster DB creati con CloudFormation

Se in passato CloudFormation creavi risorse, elimina lo CloudFormation stack dopo esserti connesso ed esplorato l'istanza di EC2 esempio e il cluster DB, in modo che non ti vengano più addebitati costi.

Per eliminare le risorse CloudFormation

1. Apri la CloudFormation console.

2. Nella pagina Stacks della CloudFormation console, seleziona lo stack principale (lo stack senza il nome VPCStack BastionStack o AMSNS).
3. Scegli Elimina.
4. Quando viene richiesta la conferma, seleziona Elimina stack.

Per ulteriori informazioni sull'eliminazione di uno stack in CloudFormation, consulta Eliminazione di uno stack sulla console nella Guida per l'utente. CloudFormation AWS CloudFormation

(Facoltativo) Connessione del cluster database a una funzione Lambda

È anche possibile connettere il cluster database Aurora MySQL a una risorsa di calcolo serverless Lambda. Le funzioni Lambda consentono di eseguire il codice senza il provisioning o la gestione dell'infrastruttura. Una funzione Lambda consente inoltre di rispondere automaticamente alle richieste di esecuzione del codice su qualsiasi scala, da una dozzina di eventi al giorno a centinaia al secondo. Per ulteriori informazioni, consulta Connessione automatica di una funzione Lambda e di un cluster database Aurora.

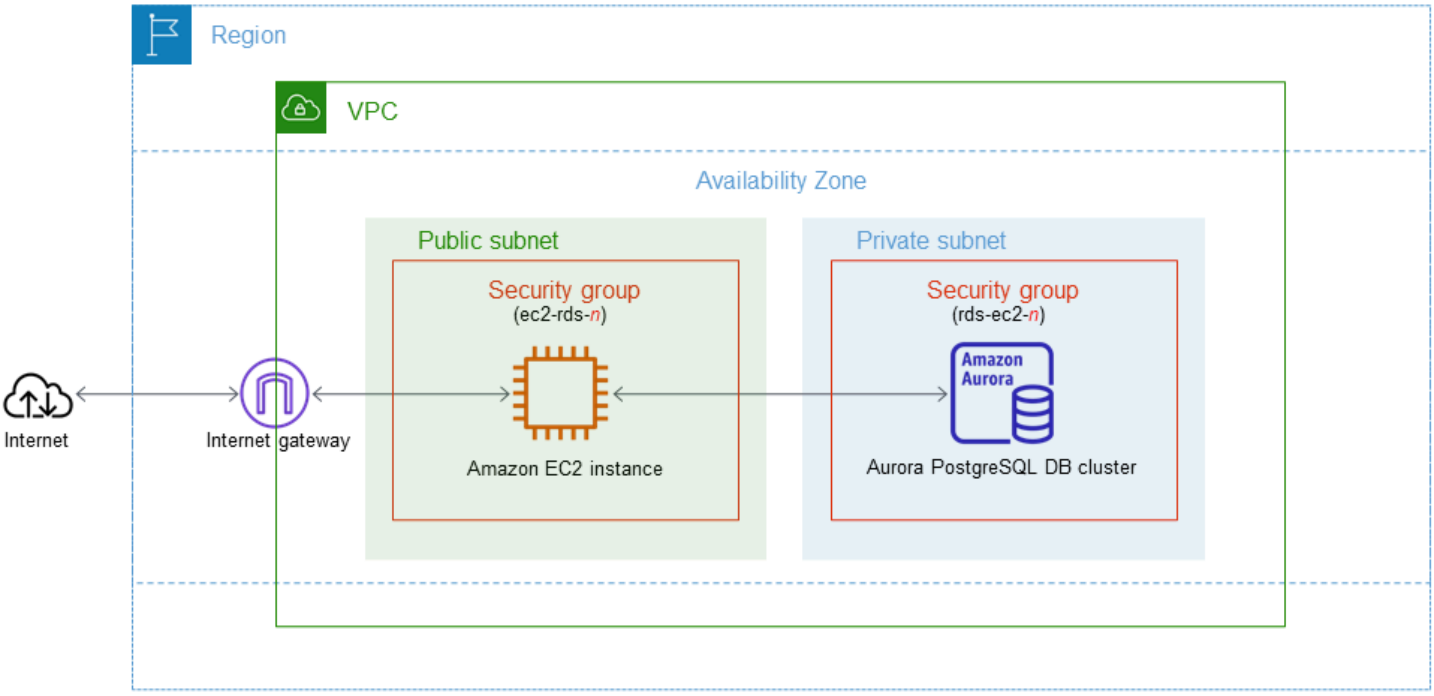
Creazione e connessione di un cluster di database Aurora PostgreSQL

Questo tutorial crea un' EC2 istanza e un cluster DB Aurora PostgreSQL. Il tutorial mostra come accedere al cluster DB dall' EC2 istanza utilizzando un client PostgreSQL standard. Come best practice, questo tutorial spiega come creare un cluster di database privato in un cloud privato virtuale (VPC). Nella maggior parte dei casi, altre risorse nello stesso VPC, come EC2 le istanze, possono accedere al cluster DB, ma le risorse esterne al VPC non possono accedervi.

Dopo aver completato il tutorial, è presente una sottorete pubblica e una privata in ogni zona di disponibilità del VPC. In una zona di disponibilità, l' EC2 istanza si trova nella sottorete pubblica e l'istanza DB si trova nella sottorete privata.

> **Important**
>
> Non è previsto alcun costo per la creazione di un AWS account. Tuttavia, completando questo tutorial, potresti incorrere in costi per le AWS risorse che utilizzi. È possibile eliminare queste risorse dopo aver completato l'esercitazione se non sono più necessarie.

Il seguente diagramma illustra la configurazione al completamento del tutorial.



Questo tutorial consente di creare le risorse utilizzando uno dei seguenti metodi:

1. Usa - e Console di gestione AWS Fase 1: Creare un' EC2 istanza Fase 2: creazione di un cluster di database Aurora PostgreSQL
2. Utilizzare CloudFormation per creare l'istanza e EC2 l'istanza del database - (Facoltativo) Crea VPC, EC2 istanza e cluster Aurora PostgreSQL utilizzando CloudFormation

Il primo metodo utilizza Creazione semplice per creare un cluster di database Aurora PostgreSQL privato con la Console di gestione AWS. Si specifica solo il tipo di motore di database, la dimensione dell'istanza database e l'identificatore del cluster di database. Easy create (Creazione rapida) utilizza l'impostazione predefinita per altre opzioni di configurazione.

Quando invece si utilizza la Creazione standard, si specificano più opzioni di configurazione al momento della creazione di un cluster di database. Queste opzioni includono impostazioni per la disponibilità, la sicurezza, i backup e la manutenzione. Per creare un cluster di database pubblico, è necessario utilizzare la Creazione standard. Per informazioni, consulta the section called "Creazione di un cluster di database".

Argomenti

- Prerequisiti

- Fase 1: Creare un' EC2 istanza
- Fase 2: creazione di un cluster di database Aurora PostgreSQL
- (Facoltativo) Crea VPC, EC2 istanza e cluster Aurora PostgreSQL utilizzando CloudFormation
- Fase 3: connessione a un cluster di database Aurora PostgreSQL
- Passaggio 4: Eliminare l' EC2istanza e il cluster DB
- (Facoltativo) Elimina l' EC2 istanza e il cluster DB creati con CloudFormation
- (Facoltativo) Connessione del cluster database a una funzione Lambda

Prerequisiti

Prima di iniziare, completa le fasi descritte in questa sezione:

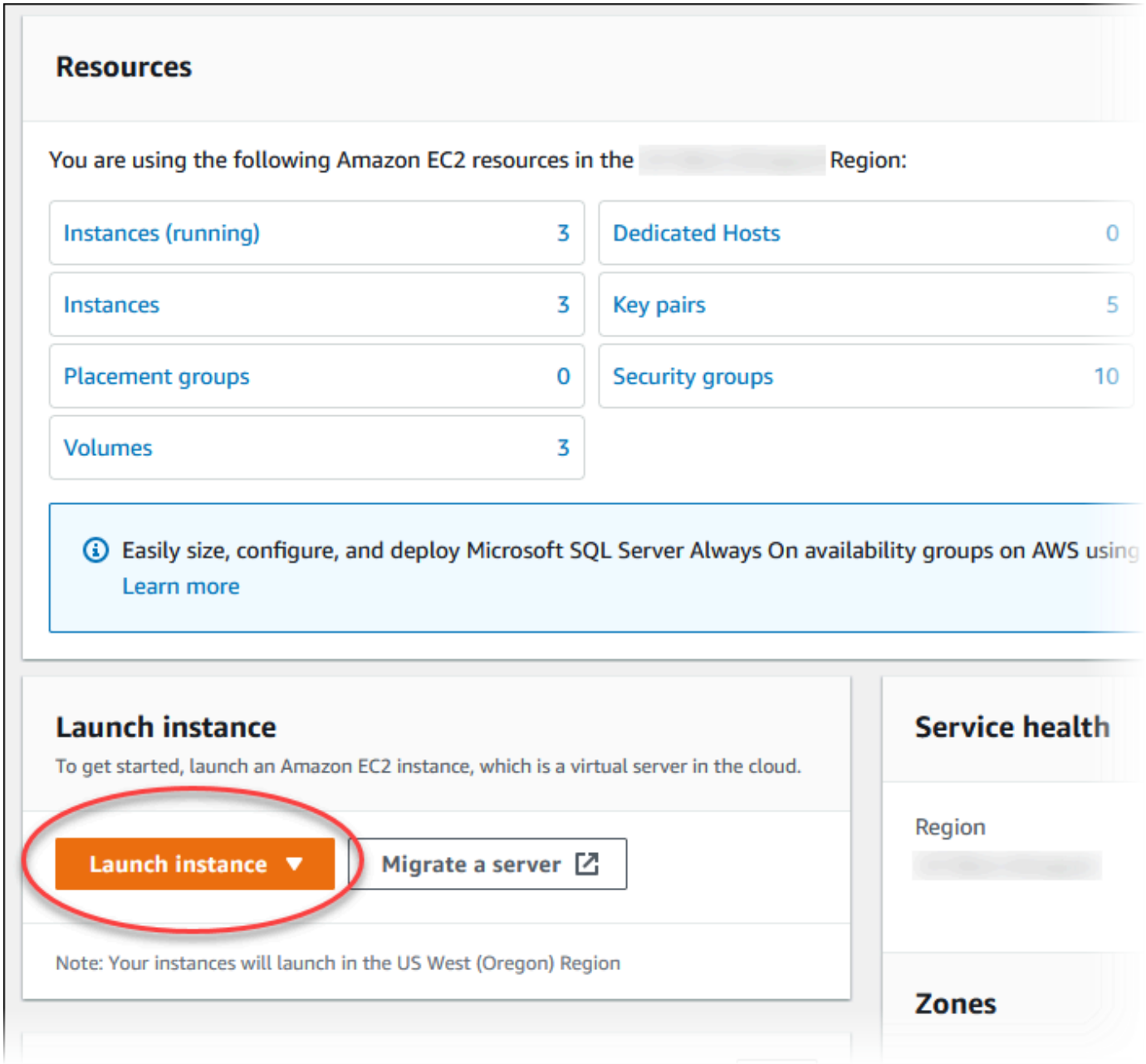
- Registrati per un Account AWS
- Crea un utente con accesso amministrativo

Fase 1: Creare un' EC2 istanza

Crea un' EC2 istanza Amazon da utilizzare per connetterti al tuo database.

Per creare un' EC2 istanza

1. Accedi a Console di gestione AWS e apri la EC2 console Amazon all'indirizzo https://console.aws.amazon.com/ec2/.
2. Nell'angolo in alto a destra di Console di gestione AWS, scegli l'istanza Regione AWS in cui desideri creare l'istanza. EC2
3. Scegli EC2 Dashboard, quindi scegli Launch instance, come mostrato nell'immagine seguente.

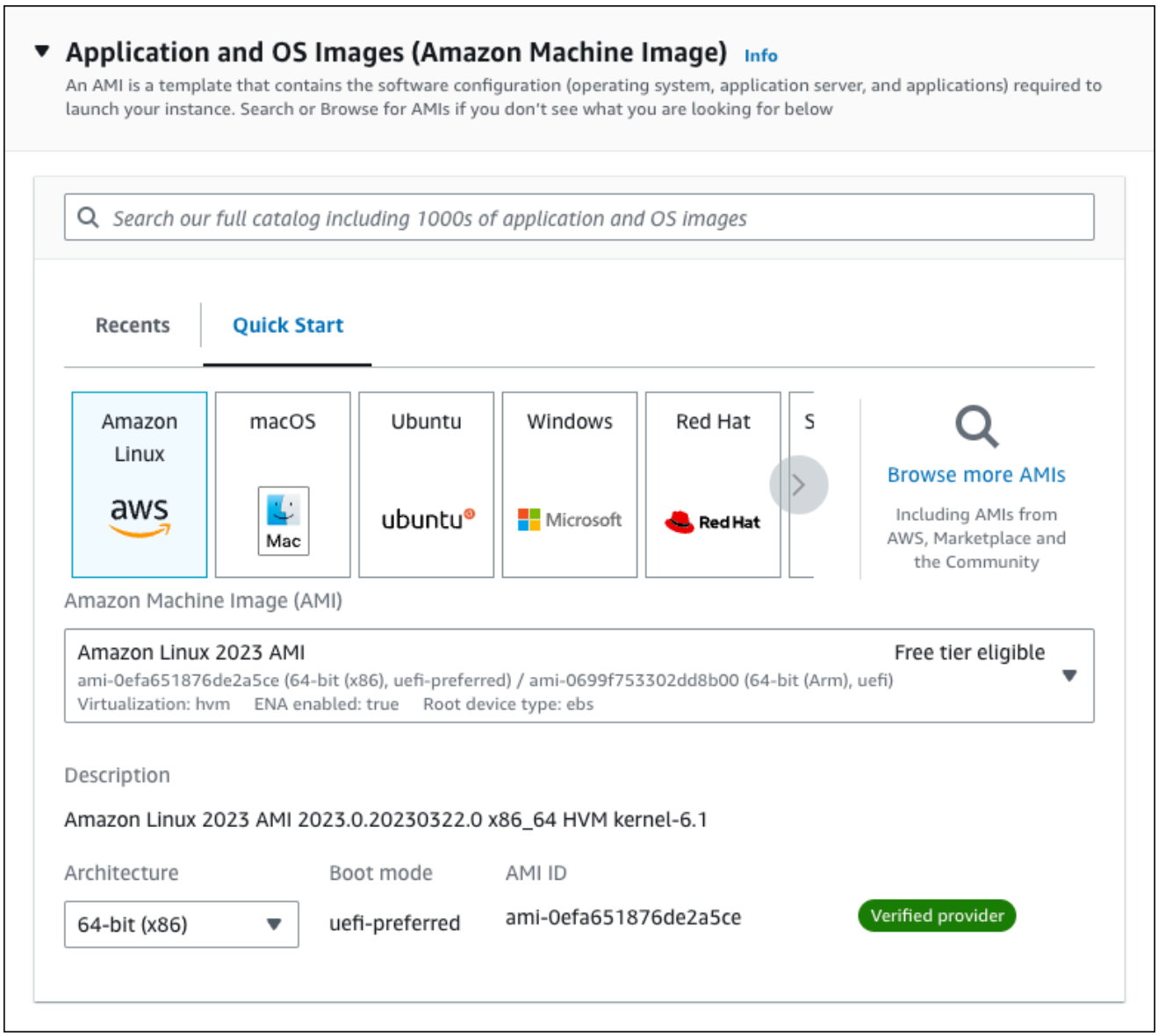


Viene visualizzata la pagina Avvia un'istanza.

4. Scegli le seguenti impostazioni nella pagina Avvia un'istanza.

   a. Nell'area Name and tags (Nome e tag), in Name (Nome) inserisci **`ec2-database-connect`**.

   b. In Immagini applicazione e sistema operativo (Amazon Machine Image), scegli Amazon Linux, quindi AMI Amazon Linux 2023. Mantieni le selezioni predefinite per le altre opzioni.



c. In Instance type (Tipo di istanza), scegli t2.micro.

d. In Key pair (login) (Coppia di chiavi (login), per Key pair name (Nome della coppia di chiavi), scegli una coppia di chiavi esistente. Per creare una nuova coppia di chiavi per l' EC2 istanza Amazon, scegli Crea nuova coppia di chiavi, quindi utilizza la finestra Crea key pair per crearla.

   Per ulteriori informazioni sulla creazione di una nuova coppia di chiavi, consulta Create a key pair nella Amazon EC2 User Guide.

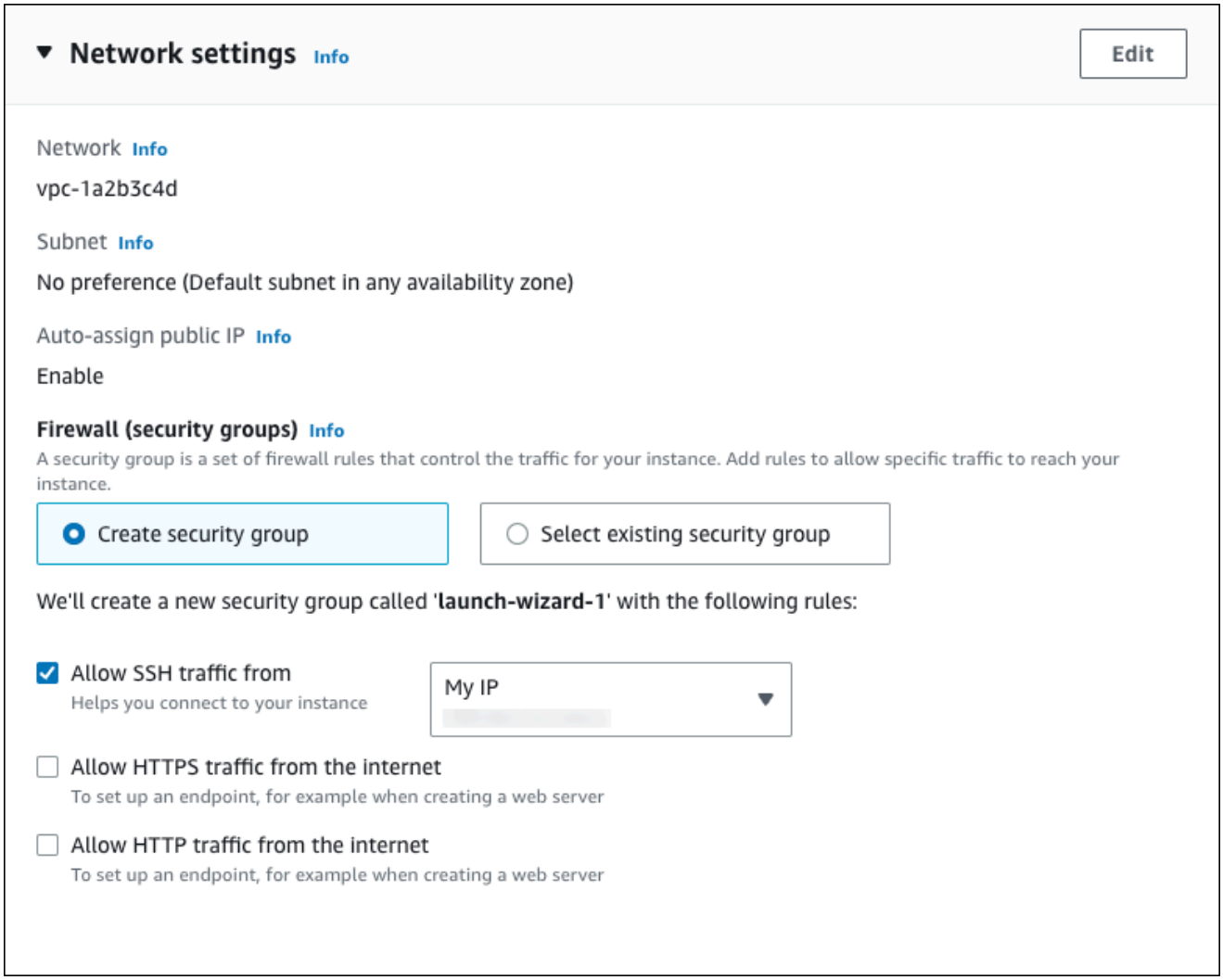
e. Per Consenti il traffico SSH nelle impostazioni di rete, scegli l'origine delle connessioni SSH all' EC2istanza.

È possibile scegliere My IP (Il mio IP) se l'indirizzo IP visualizzato è corretto per le connessioni SSH. Altrimenti, puoi determinare l'indirizzo IP da utilizzare per connetterti alle EC2 istanze nel tuo VPC utilizzando Secure Shell (SSH). Per determinare il tuo indirizzo IP pubblico, in un'altra finestra o scheda del browser, puoi utilizzare il servizio all'indirizzo. https://checkip.amazonaws.com Un esempio di indirizzo IP è 192.0.2.1/32.

In molti casi, è possibile eseguire la connessione tramite un fornitore di servizi Internet (ISP) o con la protezione di un firewall senza un indirizzo IP statico. In tal caso, accertati di determinare l'intervallo di indirizzi IP utilizzati dai computer client.

> **Warning**
>
> Se si utilizza `0.0.0.0/0` l'accesso SSH, si consente a tutti gli indirizzi IP di accedere alle EC2 istanze pubbliche tramite SSH. Questo approccio è accettabile per un breve periodo di tempo in un ambiente di test, ma non è sicuro per gli ambienti di produzione. In produzione, autorizzate solo un indirizzo IP specifico o un intervallo di indirizzi per accedere alle istanze tramite EC2 SSH.

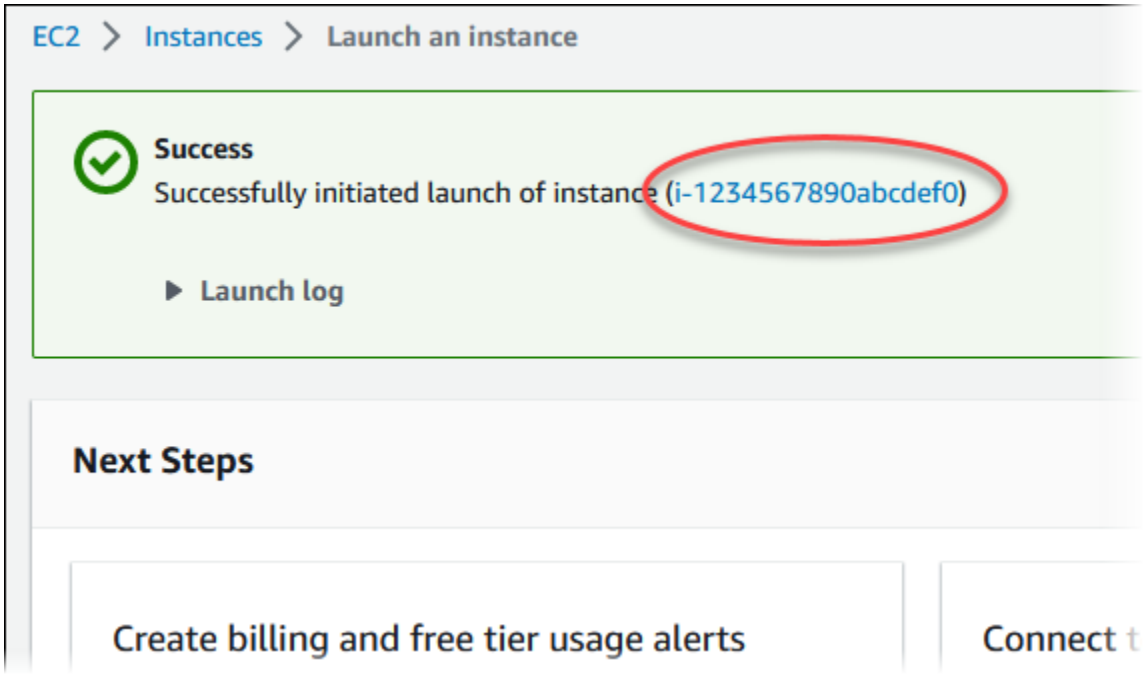
L'immagine seguente mostra un esempio della sezione Impostazioni di rete.



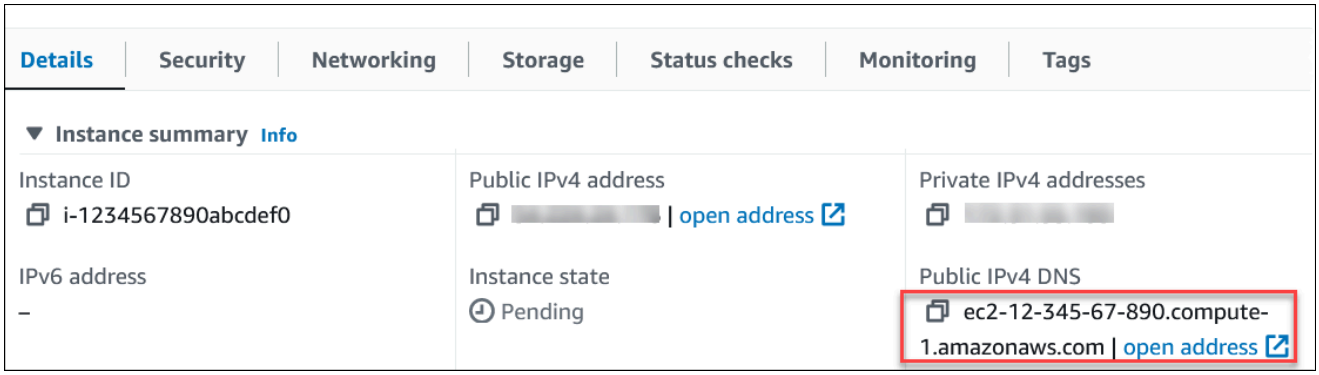
f. Lascia i valori predefiniti per le sezioni rimanenti.

g. Esamina un riepilogo della configurazione dell' EC2 istanza nel pannello Riepilogo e, quando sei pronto, scegli Launch instance.

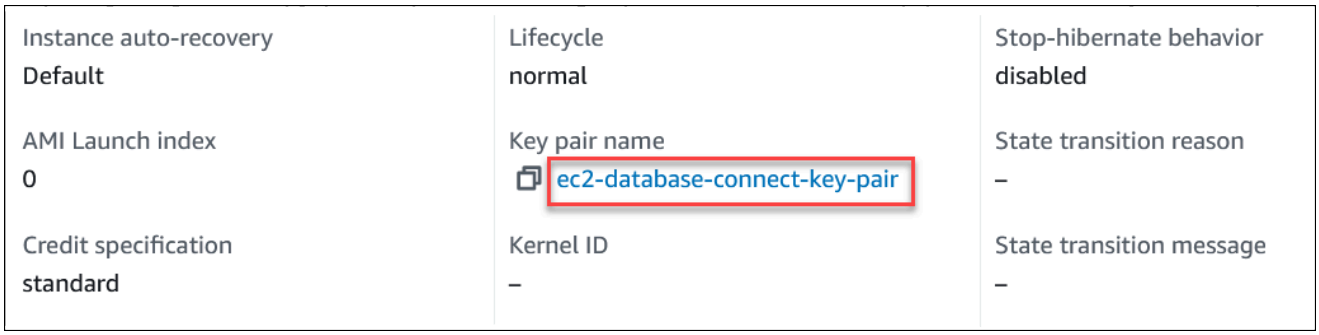
5. Nella pagina Launch Status, prendi nota dell'identificatore della tua nuova EC2 istanza, ad esempio:`i-1234567890abcdef0`.



6. Scegli l'identificatore dell' EC2 istanza per aprire l'elenco delle EC2 istanze, quindi seleziona l'istanza. EC2
7. Nella scheda Dettagli, annota i seguenti valori, necessari quando ti connetti tramite SSH:
   a. Nel riepilogo dell'istanza, annota il valore per Public IPv4 DNS.



   b. In Dettagli istanza, annota il valore visualizzato in Nome coppia di chiavi.



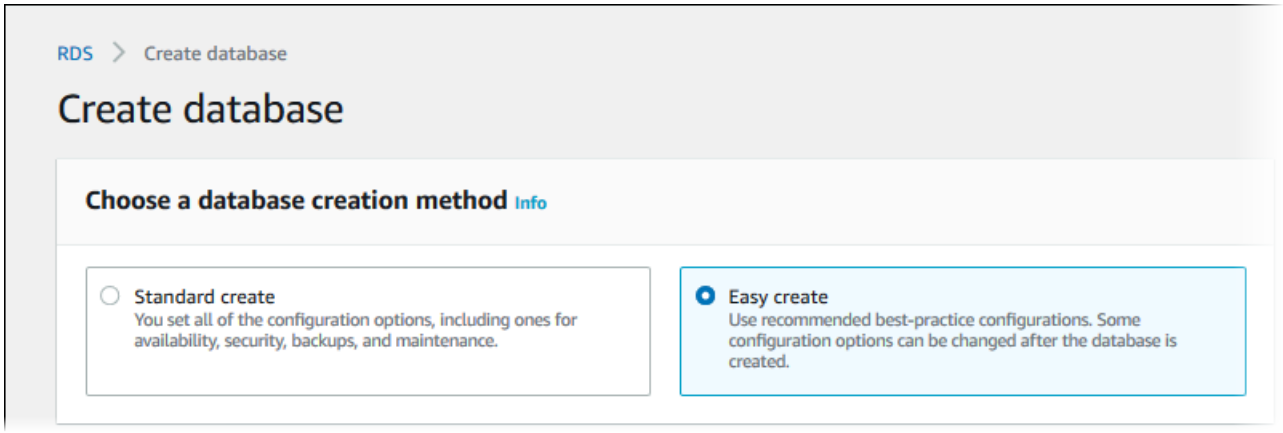
8. Attendi che lo stato dell' EC2 istanza abbia lo stato In esecuzione prima di continuare.

Fase 2: creazione di un cluster di database Aurora PostgreSQL

In questo esempio, utilizzi la Creazione semplice per creare un cluster di database Aurora PostgreSQL con una classe di istanza database db.t4g.large.

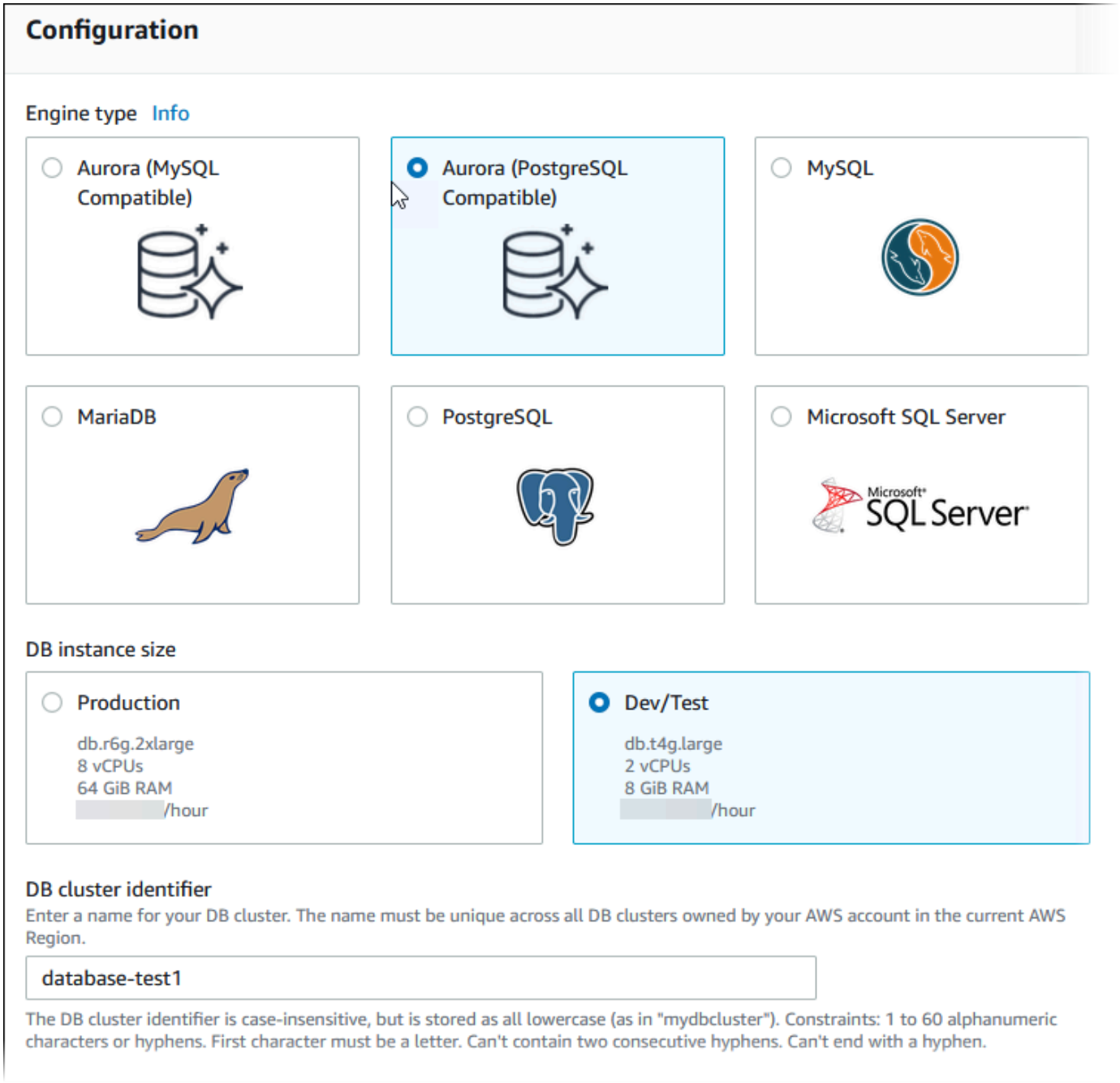
Per creare un cluster di database Aurora PostgreSQL con Creazione semplice

1. Accedi a Console di gestione AWS e apri la console Amazon RDS all'indirizzo https://console.aws.amazon.com/rds/.
2. Nell'angolo in alto a destra della console Amazon RDS, scegli la Regione AWS in cui desideri creare il cluster di database.
3. Nel pannello di navigazione, seleziona Database.
4. Scegli Crea database e verifica che l'opzione Creazione semplice sia selezionata.



5. In Configurazione scegli Aurora (compatibile con PostgreSQL) per Tipo di motore.
6. Per DB instance size (Dimensione istanza database), seleziona Dev/Test.
7. Per Identificatore cluster di database, immetti **`database-test1`**.

   La pagina Create database (Crea database) la pagina dovrebbe apparire simile alla seguente immagine.

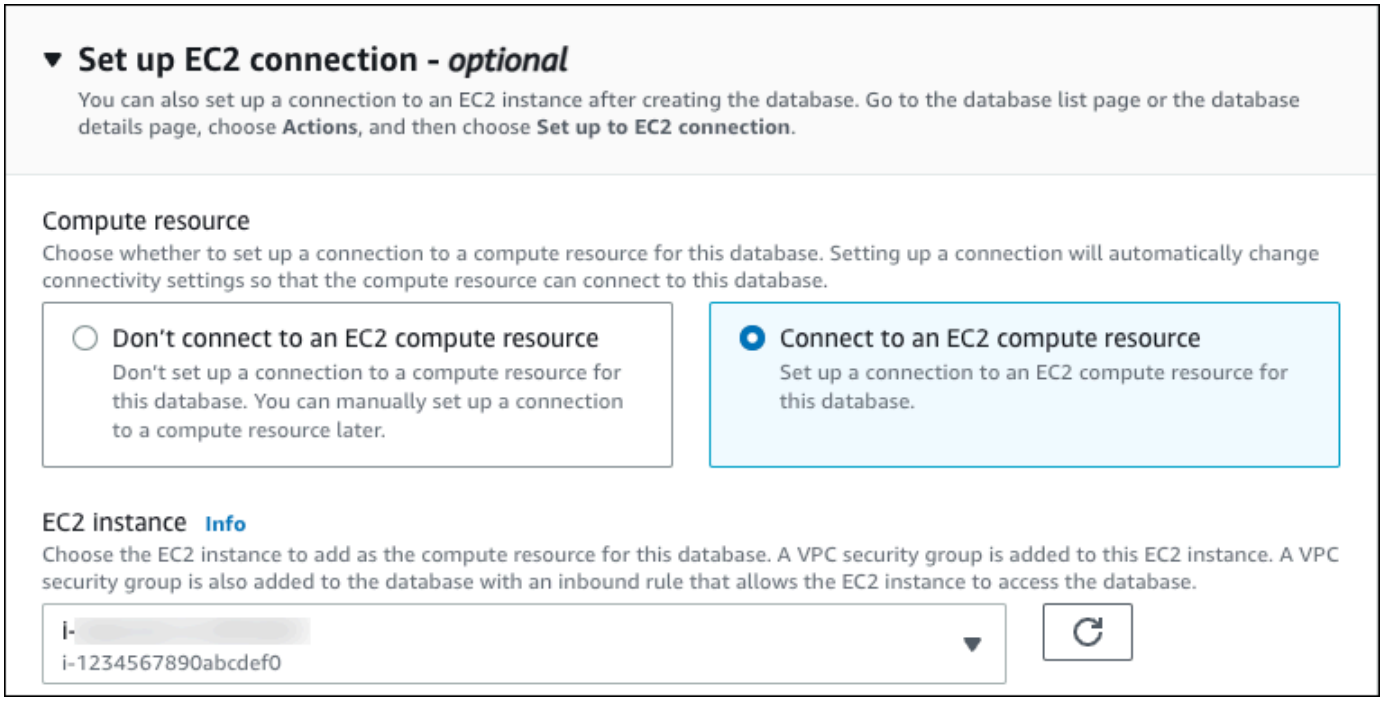


8. In Nome utente master, inserisci un nome per l'utente o lascia invariato il nome predefinito (**`postgres`**).
9. Per utilizzare una password master generata automaticamente per il cluster di database, seleziona Genera automaticamente una password.

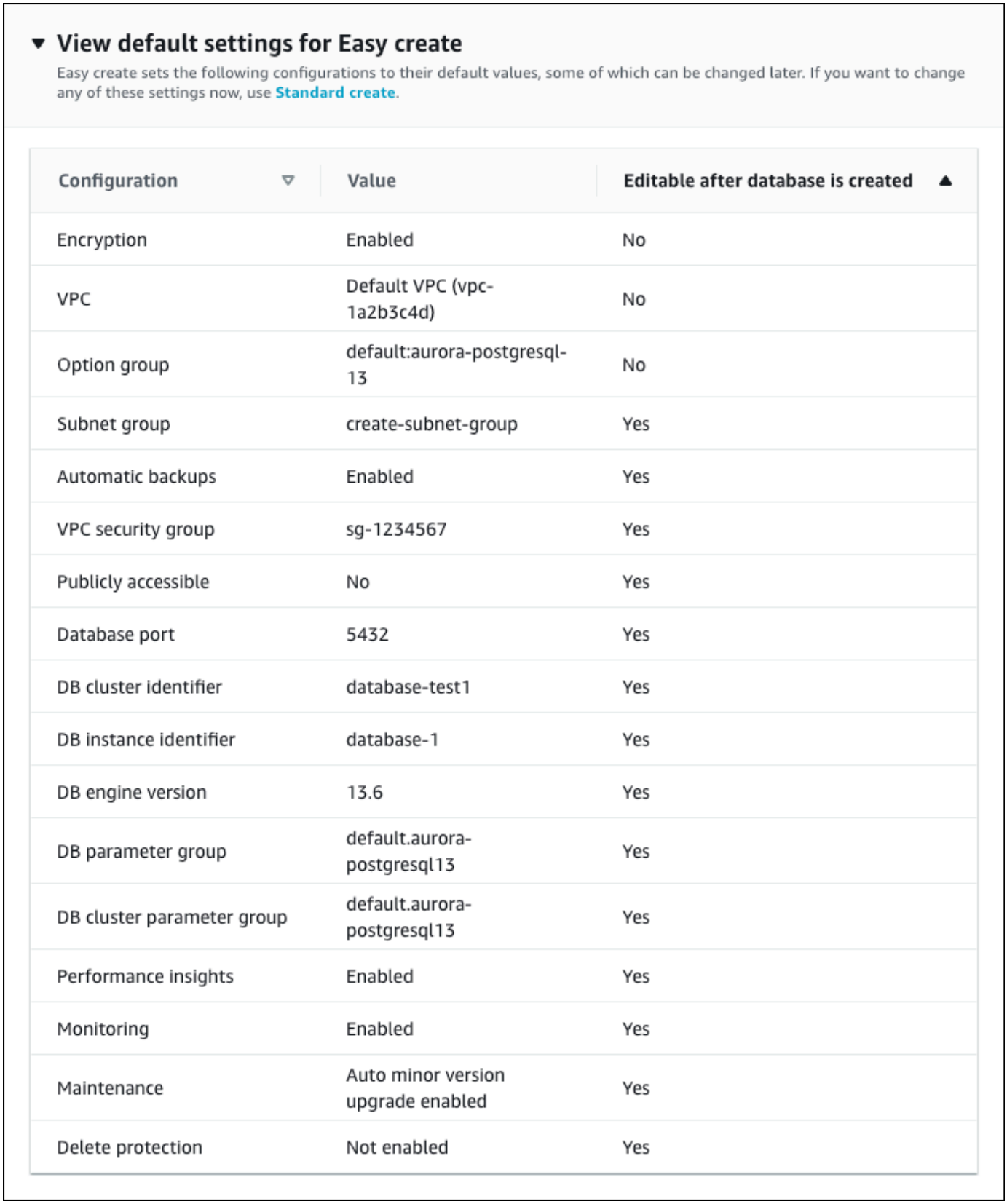
   Per inserire la password master, deseleziona la casella Genera automaticamente una password e inserisci la stessa password in Password master e Conferma password.

10. Per configurare una connessione con l' EC2 istanza creata in precedenza, apri EC2 Configura connessione (opzionale).

    Seleziona Connetti a una risorsa di EC2 calcolo. Scegli l'EC2 istanza che hai creato in precedenza.



11. Apri Visualizza le impostazioni predefinite per la creazione Semplice.



View default settings for Easy create

Easy create sets the following configurations to their default values, some of which can be changed later. If you want to change any of these settings now, use **Standard create**.

Configuration	Value	Editable after database is created
Encryption	Enabled	No
VPC	Default VPC (vpc-1a2b3c4d)	No
Option group	default:aurora-postgresql-13	No
Subnet group	create-subnet-group	Yes
Automatic backups	Enabled	Yes
VPC security group	sg-1234567	Yes
Publicly accessible	No	Yes
Database port	5432	Yes
DB cluster identifier	database-test1	Yes
DB instance identifier	database-1	Yes
DB engine version	13.6	Yes
DB parameter group	default.aurora-postgresql13	Yes
DB cluster parameter group	default.aurora-postgresql13	Yes
Performance insights	Enabled	Yes
Monitoring	Enabled	Yes
Maintenance	Auto minor version upgrade enabled	Yes
Delete protection	Not enabled	Yes

Puoi esaminare le impostazioni predefinite utilizzate con Easy create (Creazione rapida). La colonna Modificabile dopo la creazione del database mostra le opzioni che puoi modificare dopo aver creato il database.

- Se un'impostazione contiene No in quella colonna e desideri cambiarla, puoi utilizzare la Creazione standard per creare il cluster di database.
- Se un'impostazione contiene Sì in quella colonna e desideri cambiarla, puoi utilizzare la Creazione standard per creare il cluster di database o modificare il cluster di database dopo averlo creato per cambiare l'impostazione.

12. Scegliere Crea database.

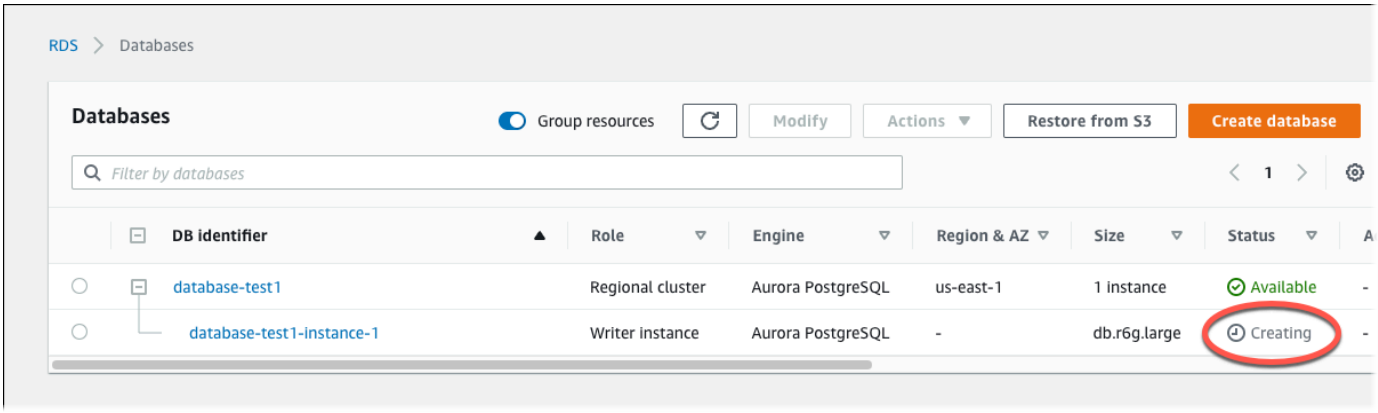
    Per vedere nome utente e password per il cluster di database, seleziona Visualizza i dettagli delle credenziali.

    Per connetterti al cluster di database come utente master, utilizza il nome utente e la password visualizzati.

    > **Important**
    >
    > Non potrai visualizzare di nuovo la password dell'utente principale. Se non la registri, potresti doverla modificare.
    > Se devi modificare la password dell'utente principale dopo che il cluster database è disponibile, puoi modificare il cluster database per eseguire tale operazione. Per ulteriori informazioni sulla modifica di un cluster di database, consultare Modifica di un cluster database Amazon Aurora.

13. Nell'elenco Database seleziona il nome del nuovo cluster di database Aurora PostgreSQL per visualizzarne i dettagli.

    L'istanza di scrittura ha lo stato Creazione in corso fino a quando il cluster di database non è pronto per essere utilizzato.



Quando lo stato dell'istanza di scrittura cambia in Disponibile, puoi connetterti al cluster di database. A seconda della classe di istanza database e della quantità di storage, prima che il nuovo cluster di database sia disponibile possono trascorrere fino a 20 minuti.

(Facoltativo) Crea VPC, EC2 istanza e cluster Aurora PostgreSQL utilizzando CloudFormation

Invece di utilizzare la console per creare il VPC, l' EC2 istanza e il cluster Aurora PostgreSQL DB, è possibile utilizzarla per il provisioning delle risorse trattando l'infrastruttura come codice. CloudFormation AWS Per aiutarti a organizzare AWS le tue risorse in unità più piccole e più gestibili, puoi utilizzare la funzionalità nested stack. CloudFormation Per ulteriori informazioni, consulta Creare uno stack sulla CloudFormation console e Lavorare con gli stack annidati.

> **Important**
>
> CloudFormation è gratuito, ma le risorse che CloudFormation crea sono attive. Il costo di utilizzo standard per queste risorse sarà addebitato finché non vengono terminate. Per ulteriori informazioni, consulta Prezzi di Amazon Aurora.

Per creare le tue risorse utilizzando la CloudFormation console, completa i seguenti passaggi:

- Passaggio 1: scarica il CloudFormation modello
- Passaggio 2: configura le tue risorse utilizzando CloudFormation

Scarica il CloudFormation modello

Un CloudFormation modello è un file di testo JSON o YAML che contiene le informazioni di configurazione sulle risorse che desideri creare nello stack. Questo modello crea automaticamente anche un VPC e un host bastione insieme al cluster Aurora.

Per scaricare il file modello, apri il seguente link, modello Aurora PostgreSQL. CloudFormation

Nella pagina Github, fare clic sul pulsante Scarica file non elaborato per salvare il file YAML.

Configura le tue risorse utilizzando CloudFormation

> **Note**
>
> Prima di iniziare questo processo, assicurati di avere una coppia di chiavi per un' EC2 istanza nel tuo Account AWS. Per ulteriori informazioni, consulta coppie di EC2 chiavi Amazon e istanze Linux.

Quando utilizzi il CloudFormation modello, devi selezionare i parametri corretti per assicurarti che le risorse vengano create correttamente. Segui la procedura riportata di seguito:

1. Accedi Console di gestione AWS e apri la CloudFormation console all'indirizzo https://console.aws.amazon.com/cloudformazione.
2. Scegli Crea stack.
3. Nella sezione in cui specificare il modello, seleziona Carica un file modello dal computer, quindi Avanti.
4. Nella pagina Specifica i dettagli dello stack, impostare i seguenti parametri:
   a. Imposta il nome dello stack su Stack. AurPostgre SQLTest
   b. In Parametri, impostare Zone di disponibilità selezionando due zone di disponibilità.
   c. Nella configurazione Linux Bastion Host, in Key Name, seleziona una coppia di chiavi per accedere alla tua EC2 istanza.
   d. Nelle impostazioni di Configurazione host bastione di Linux, imposta Intervallo IP consentito sul tuo indirizzo IP. Per connetterti alle EC2 istanze nel tuo VPC utilizzando Secure Shell (SSH), determina il tuo indirizzo IP pubblico utilizzando il servizio all'indirizzo. https://checkip.amazonaws.com Un esempio di indirizzo IP è 192.0.2.1/32.

> **Warning**
>
> Se utilizzi `0.0.0.0/0` l'accesso SSH, consenti a tutti gli indirizzi IP di accedere alle tue istanze pubbliche EC2 tramite SSH. Questo approccio è accettabile per un breve periodo di tempo in un ambiente di test, ma non è sicuro per gli ambienti di produzione. In produzione, autorizzate solo un indirizzo IP specifico o un intervallo di indirizzi per accedere alle istanze tramite EC2 SSH.

e. In Configurazione generale del database, impostare la Classe di istanza del database su db.t4g.large.

f. Imposta Database name su **`database-test1`**.

g. Per Nome utente principale del database, specifica il nome dell'utente master.

h. Imposta Gestisci password utente master di database con Secrets Manager su `false` per questo tutorial.

i. Per Database password, imposta una password di tua scelta. Tenere a mente questa password per altri passaggi del tutorial.

j. Impostare Implementazione Multi-AZ su `false`.

k. Lasciare i valori predefiniti per tutte le altre impostazioni. Fai clic su Next per continuare.

5. Nella pagina Configura opzioni dello stack, lasciare tutte le opzioni predefinite. Fai clic su Next per continuare.
6. Nella pagina Rivedi stack, seleziona Invia dopo aver verificato le opzioni del database e dell'host bastione di Linux.

Una volta completato il processo di creazione dello stack, visualizza gli stack con nomi BastionStacke APGNS per annotare le informazioni necessarie per connetterti al database. Per ulteriori informazioni, vedere Visualizzazione dei dati e delle risorse CloudFormation dello stack su. Console di gestione AWS

Fase 3: connessione a un cluster di database Aurora PostgreSQL

È possibile utilizzare qualsiasi applicazione client PostgreSQL standard per la connessione al cluster di database. In questo esempio, ti connetti a un cluster di database Aurora PostgreSQL utilizzando il client della linea di comando psql.

Per effettuare la connessione a un cluster di database Aurora PostgreSQL

1. Individua l'endpoint (nome DNS) e il numero di porta per l'istanza di scrittura del cluster di database.

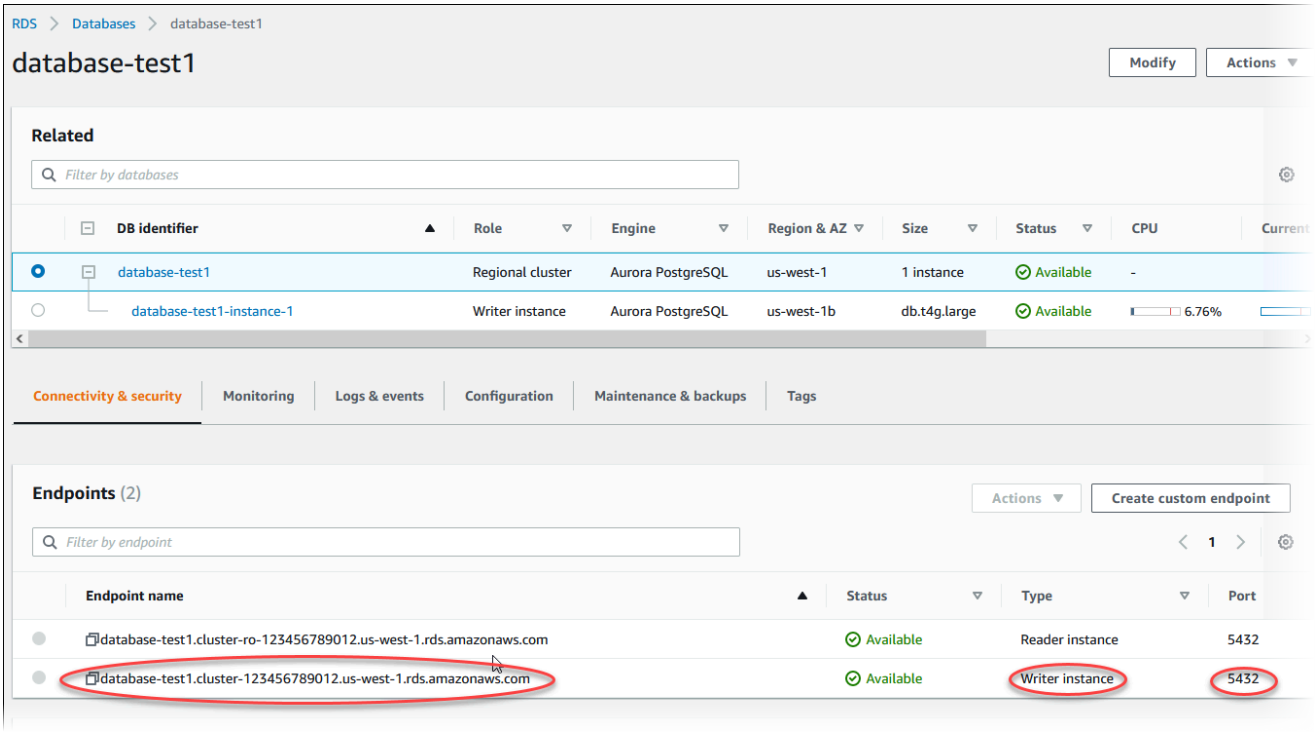
   a. Accedi a Console di gestione AWS e apri la console Amazon RDS all'indirizzo https://console.aws.amazon.com/rds/.

   b. Nell'angolo in alto a destra della console Amazon RDS, scegli Regione AWS per il cluster DB.

   c. Nel pannello di navigazione, seleziona Database.

   d. Scegli il nome del cluster di database Aurora PostgreSQL per visualizzarne i dettagli.

   e. Nella scheda Connettività e sicurezza, copia l'endpoint dell'istanza di scrittura. Annotare anche il numero di porta. L'endpoint e il numero di porta sono necessari per la connessione al cluster di database.



2. Connettiti all' EC2 istanza che hai creato in precedenza seguendo i passaggi in Connetti alla tua istanza Linux nella Amazon EC2 User Guide.

   Ti consigliamo di connetterti alla tua EC2 istanza tramite SSH. Se l'utilità client SSH è installata su Windows, Linux o Mac, puoi connetterti all'istanza utilizzando il comando nel seguente formato:

```
ssh -i location_of_pem_file ec2-user@ec2-instance-public-dns-name
```

Ad esempio, supponiamo che `ec2-database-connect-key-pair.pem` sia archiviata `/dir1` su Linux e che il IPv4 DNS pubblico dell' EC2 istanza lo sia. `ec2-12-345-678-90.compute-1.amazonaws.com` Il comando SSH sarà simile al seguente:

```
ssh -i /dir1/ec2-database-connect-key-pair.pem ec2-
user@ec2-12-345-678-90.compute-1.amazonaws.com
```

3. Ottieni le correzioni di bug e gli aggiornamenti di sicurezza più recenti aggiornando il software sulla tua EC2 istanza. A tale scopo, utilizzare il comando seguente.

> **Note**
>
> L'opzione `-y` installa gli aggiornamenti senza chiedere conferma. Per esaminare gli aggiornamenti prima di installarli, omettere questa opzione.

```
sudo dnf update -y
```

4. Installa il client della linea di comando psql da PostgreSQL su Amazon Linux 2023 utilizzando il seguente comando:

```
sudo dnf install postgresql15
```

5. Effettua la connessione al cluster di database Aurora PostgreSQL. Ad esempio, immetti il comando seguente: Questa azione ti consente di effettuare la connessione al cluster di database Aurora PostgreSQL utilizzando il client psql.

   Sostituisci l'endpoint dell'istanza di scrittura per *`endpoint`*, sostituisci il nome database `--dbname` a cui vuoi connetterti per *`postgres`* e sostituisci il nome utente master utilizzato per *`postgres`*. Devi fornire la password master utilizzata quando viene richiesta una password.

   ```
   psql --host=endpoint --port=5432 --dbname=postgres --username=postgres
   ```

   Dopo aver immesso la password per l'utente, l'output dovrebbe essere analogo a quanto mostrato di seguito.

```
psql (14.3, server 14.6)
SSL connection (protocol: TLSv1.2, cipher: ECDHE-RSA-AES256-GCM-SHA384, bits: 256,
 compression: off)
Type "help" for help.

postgres=>
```

Per ulteriori informazioni sulla connessione a un cluster di database Aurora PostgreSQL, consulta Connessione a un cluster di database Amazon Aurora PostgreSQL. In caso di mancata connessione al cluster database, consulta Impossibile connettersi all'istanza database di Amazon RDS.

Per motivi di sicurezza, la best practice è utilizzare connessioni crittografate. Utilizza una connessione PostgreSQL non crittografata solo quando il client e il server sono nello stesso VPC e la rete è attendibile. Per ulteriori informazioni sull'uso di connessioni crittografate, consulta Sicurezza dei dati Aurora PostgreSQL con SSL/TLS.

6. Eseguire comandi SQL.

   Ad esempio, il seguente comando SQL mostra la data e l'ora correnti:

   ```
   SELECT CURRENT_TIMESTAMP;
   ```

Passaggio 4: Eliminare l' EC2istanza e il cluster DB

Dopo esserti connesso ed esplorato l'istanza di EC2 esempio e il cluster DB che hai creato, eliminali in modo che non ti vengano più addebitati costi.

Se in passato CloudFormation creavi risorse, salta questo passaggio e vai al passaggio successivo.

Per eliminare l'istanza EC2

1. Accedi a Console di gestione AWS e apri la EC2 console Amazon all'indirizzo https://console.aws.amazon.com/ec2/.
2. Nel riquadro di navigazione, scegliere Instances (Istanze).
3. Seleziona l' EC2 istanza e scegli Stato dell'istanza, Termina istanza.
4. Quando viene richiesta la conferma, seleziona Interrompi.

Per ulteriori informazioni sull'eliminazione di un' EC2 istanza, consulta Terminate your istance nella Amazon EC2 User Guide.

Per eliminare un cluster database

1. Accedi a Console di gestione AWS e apri la console Amazon RDS all'indirizzo https://console.aws.amazon.com/rds/.
2. Scegliere Databases (Database), quindi selezionare l'istanza database associata con il cluster di database.
3. In Actions (Azioni), selezionare Delete (Elimina).
4. Scegliere Delete (Elimina).

   Dopo aver eliminato tutte le istanze database associate a un cluster di database, il cluster di database viene eliminato automaticamente.

(Facoltativo) Elimina l' EC2 istanza e il cluster DB creati con CloudFormation

Se in passato CloudFormation creavi risorse, elimina lo CloudFormation stack dopo esserti connesso ed esplorato l'istanza di EC2 esempio e il cluster DB, in modo che non ti vengano più addebitati costi.

Per eliminare le risorse CloudFormation

1. Apri la CloudFormation console.
2. Nella pagina Stacks della CloudFormation console, seleziona lo stack principale (lo stack senza il nome VPCStack BastionStack o APGNS).
3. Scegli Elimina.
4. Quando viene richiesta la conferma, seleziona Elimina stack.

Per ulteriori informazioni sull'eliminazione di uno stack in CloudFormation, vedere Eliminazione di uno stack sulla console nella Guida per l'utente. CloudFormation AWS CloudFormation

(Facoltativo) Connessione del cluster database a una funzione Lambda

È anche possibile connettere il cluster di database Aurora PostgreSQL a una risorsa di calcolo serverless Lambda. Le funzioni Lambda consentono di eseguire il codice senza il provisioning o la gestione dell'infrastruttura. Una funzione Lambda consente inoltre di rispondere automaticamente alle

richieste di esecuzione del codice su qualsiasi scala, da una dozzina di eventi al giorno a centinaia al secondo. Per ulteriori informazioni, consulta Connessione automatica di una funzione Lambda e di un cluster database Aurora.

Tutorial: creazione di un server Web e un cluster database Amazon Aurora

Questo tutorial descrive come installare un server Web Apache con PHP e creare un database MariaDB, MySQL o PostgreSQL. Il server Web viene eseguito in un'istanza Amazon EC2 utilizzando Amazon Linux 2023 e puoi scegliere tra un cluster database Aurora MySQL o Aurora PostgreSQL. Sia l'istanza Amazon EC2 che l'Cluster DB vengono eseguite in un virtual private cloud (VPC) basato sul servizio Amazon VPC.

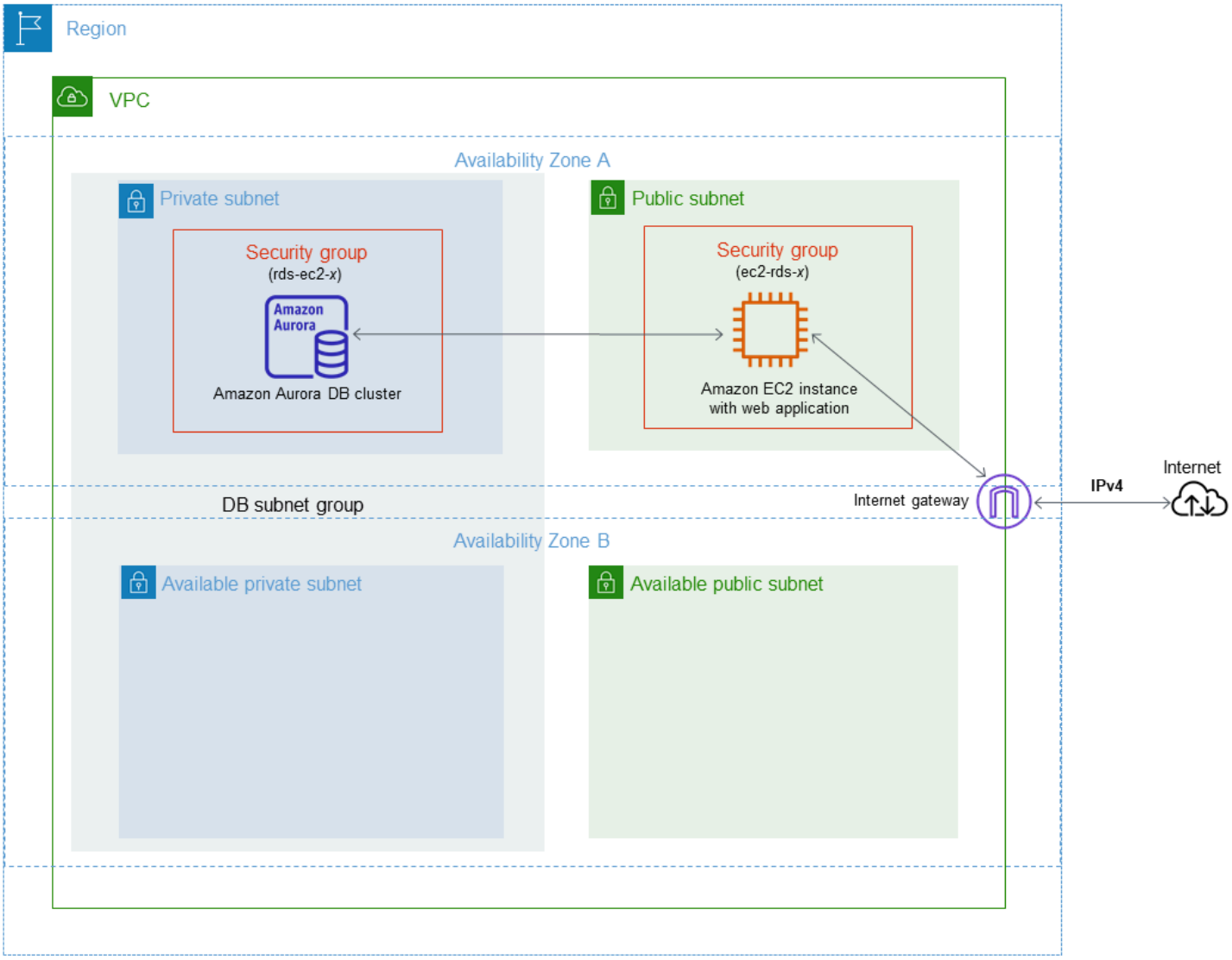
> **Important**
>
> Non vi sono costi aggiuntivi per la creazione di un account AWS. Tuttavia, completando l'esercitazione, potresti incorrere in costi per le risorse AWS utilizzate. È possibile eliminare queste risorse dopo aver completato l'esercitazione se non sono più necessarie.

> **Note**
>
> Questo tutorial funziona con Amazon Linux 2023 e potrebbe non funzionare per altre versioni di Linux.

Nel tutorial che segue, viene creata un'istanza EC2 che utilizza VPC, sottoreti e gruppo di sicurezza predefiniti per Account AWS. In questo tutorial viene illustrato come creare il cluster database e configurare automaticamente la connettività con l'istanza EC2 creata. Nel tutorial viene quindi mostrato come installare il server Web sull'istanza EC2. Il server Web viene connesso al cluster database nel VPC utilizzando l'endpoint del cluster di scrittura del database.

1. Avvio di un'istanza EC2 per la connessione al cluster di database
2. Creazione di un cluster di database Amazon Aurora
3. Installazione di un server Web nell'istanza EC2

Il seguente diagramma illustra la configurazione al completamento del tutorial.



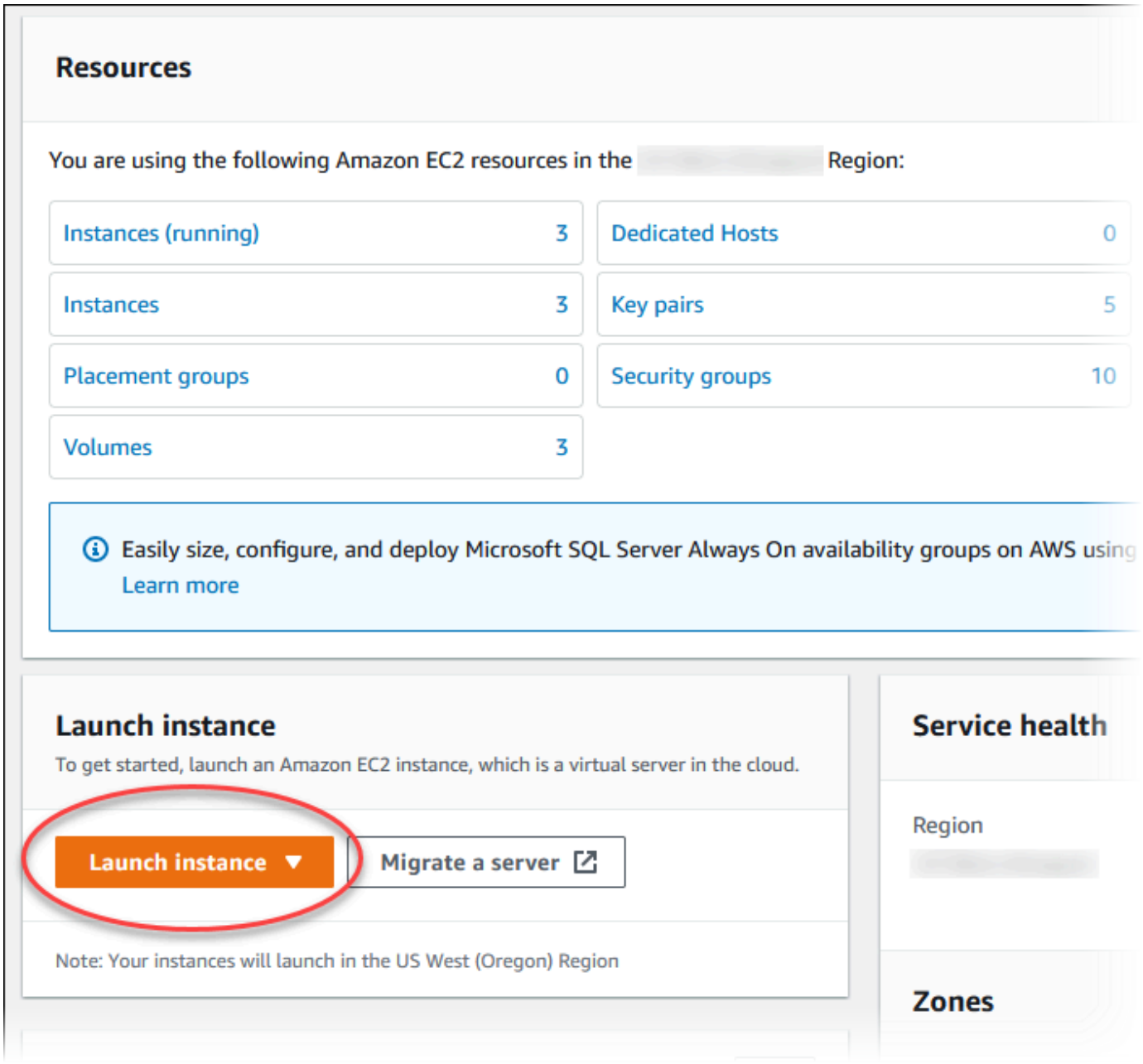
> **Note**
>
> Dopo aver completato il tutorial, è presente una sottorete pubblica e una privata in ogni zona di disponibilità del VPC. Questo tutorial utilizza il VPC predefinito per Account AWS e configura automaticamente la connettività tra l'istanza EC2 e il cluster database. Se preferisci invece configurare un nuovo VPC per questo scenario, completa le attività in Tutorial: Creazione di un Amazon VPC da utilizzare con un cluster database (solo IPv4).

Avvio di un'istanza EC2 per la connessione al cluster di database

Crea un'istanza Amazon EC2 nella sottorete pubblica del VPC.

Per avviare un'istanza EC2

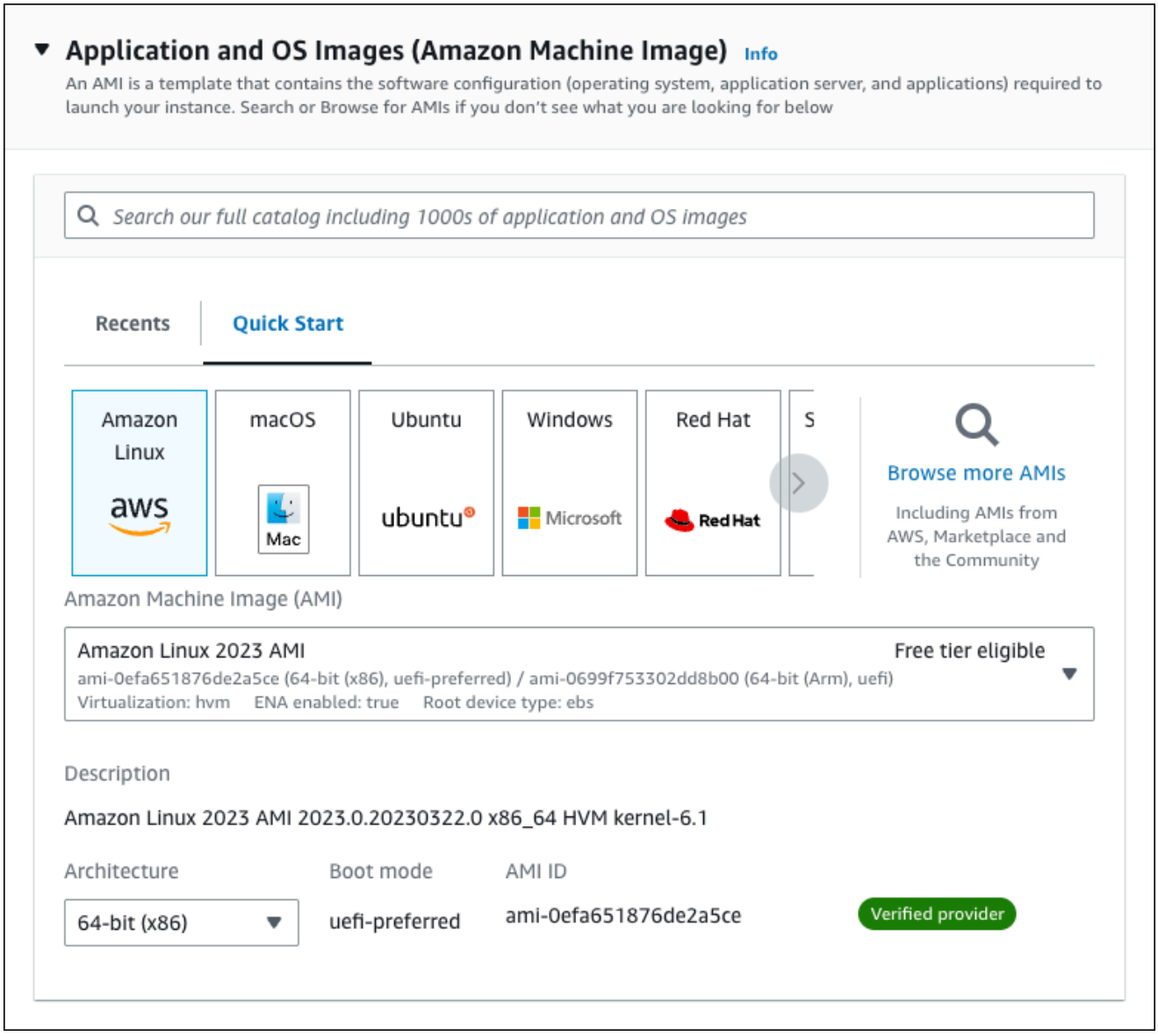
1. Accedi a Console di gestione AWS e apri la console Amazon EC2 all'indirizzo https://console.aws.amazon.com/ec2/.
2. Nell'angolo in alto a destra della Console di gestione AWS, scegli la Regione AWS in cui si desidera creare l'istanza EC2.
3. Selezionare Pannello di controllo EC2, quindi Avvia istanza, come visualizzato di seguito.



4. Scegli le seguenti impostazioni nella pagina Avvia un'istanza.

a. Nell'area Name and tags (Nome e tag), in Name (Nome) inserisci **`tutorial-ec2-instance-web-server`**.

b. In Immagini applicazione e sistema operativo (Amazon Machine Image), scegli Amazon Linux, quindi AMI Amazon Linux 2023. Manteni i valori predefiniti per le altre opzioni.



c. In Instance type (Tipo di istanza), scegli t2.micro.

d. In Key pair (login) (Coppia di chiavi (login), per Key pair name (Nome della coppia di chiavi), scegli una coppia di chiavi esistente. Per creare una nuova coppia di chiavi per l'istanza Amazon EC2, scegli Create new key pair (Crea nuova coppia di chiavi) e quindi utilizza la finestra Create key pair (Crea coppia di chiavi) per crearla.

Per ulteriori informazioni sulla creazione di una nuova coppia di chiavi, consulta Creazione di una coppia di chiavi nella Guida per l'utente di Amazon EC2.

e. In Network settings (Impostazioni di rete), imposta questi valori e conserva le impostazioni predefinite degli altri valori:

- In Allow SSH traffic from (Consenti traffico SSH da), scegli l'origine delle connessioni SSH all'istanza EC2.

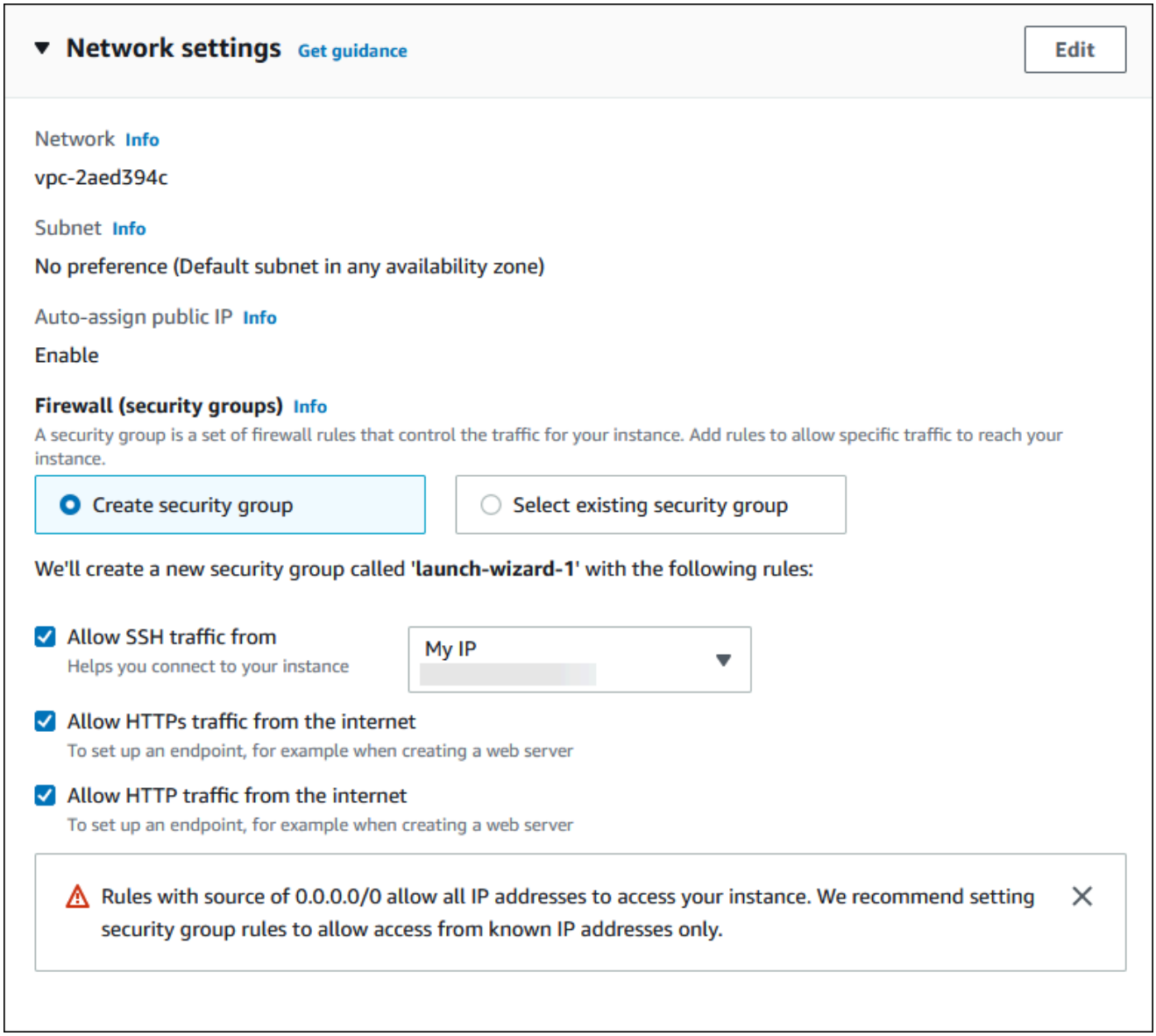
  È possibile scegliere My IP (Il mio IP) se l'indirizzo IP visualizzato è corretto per le connessioni SSH.

  In caso contrario, è possibile determinare l'indirizzo IP da utilizzare per connettersi alle istanze EC2 nel VPC utilizzando Secure Shell (SSH). Per determinare l'indirizzo IP pubblico, in una finestra o una scheda del browser diversa, è possibile utilizzare il servizio all'indirizzo https://checkip.amazonaws.com. Un esempio di indirizzo IP è `203.0.113.25/32`.

  In molti casi, è possibile eseguire la connessione tramite un fornitore di servizi Internet (ISP) o con la protezione di un firewall senza un indirizzo IP statico. In tal caso, accertati di determinare l'intervallo di indirizzi IP utilizzati dai computer client.

  > **Warning**
  >
  > Se utilizzi `0.0.0.0/0` per l'accesso SSH, consenti a tutti gli indirizzi IP di accedere alle istanze pubbliche utilizzando SSH. Questo approccio è accettabile per un breve periodo di tempo in un ambiente di test, ma non è sicuro per gli ambienti di produzione. In produzione, autorizza solo un determinato indirizzo IP o un intervallo di indirizzi per accedere alle istanze utilizzando SSH.

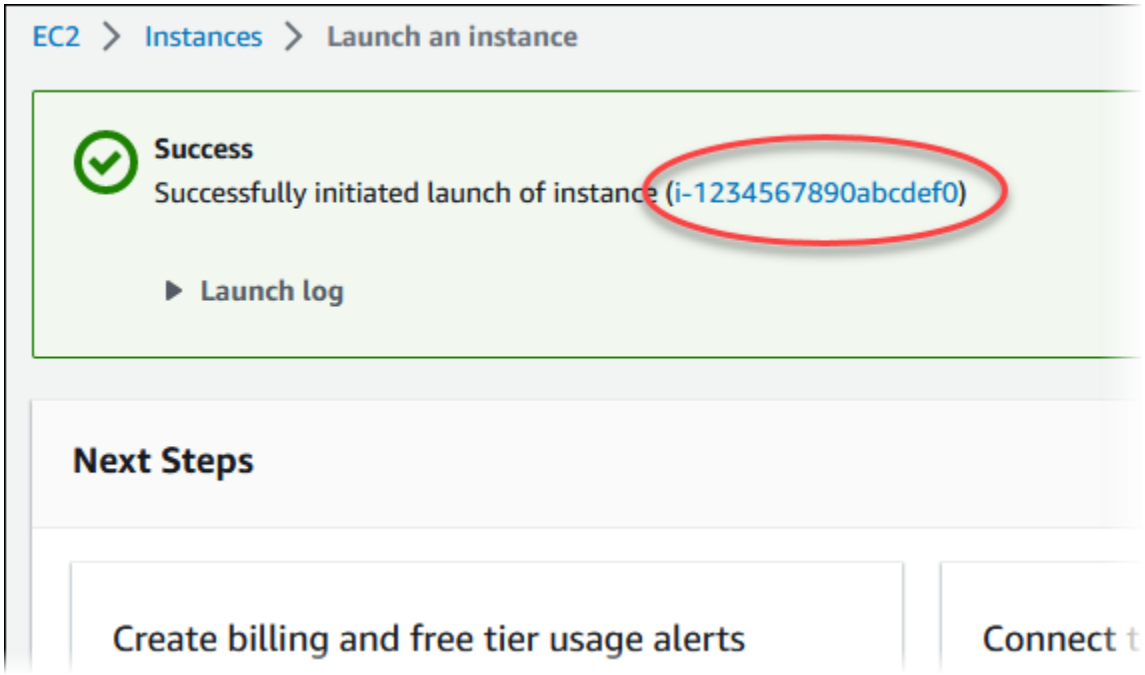
- Attiva Allow HTTPs traffic from the internet (Autorizzare il traffico HTTPS da Internet).
- Attiva Allow HTTP traffic from the internet (Autorizzare il traffico HTTP da Internet).



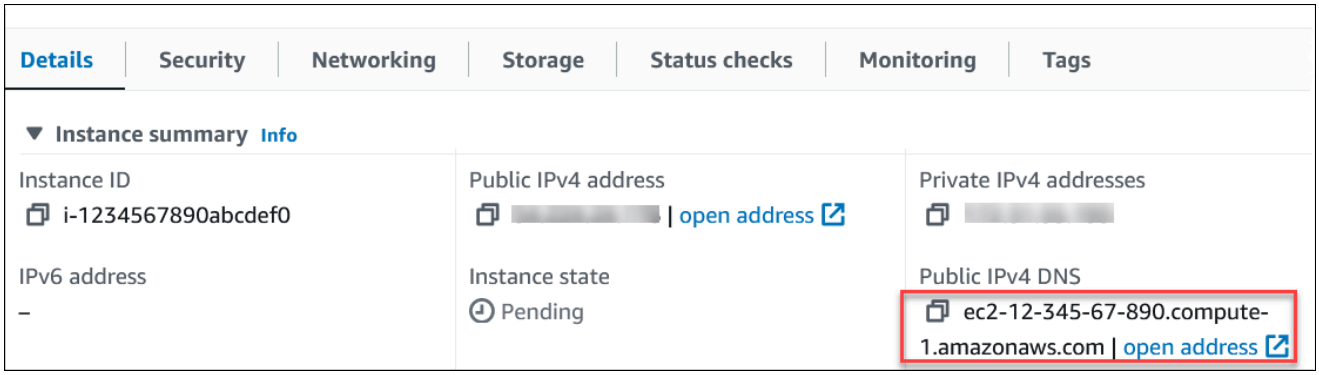
f. Lascia i valori predefiniti per le sezioni rimanenti.

g. Analizza un riepilogo della configurazione dell'istanza nel pannello Summary (Riepilogo) e, quando è tutto pronto, scegli Launch instance (Avvia istanza).

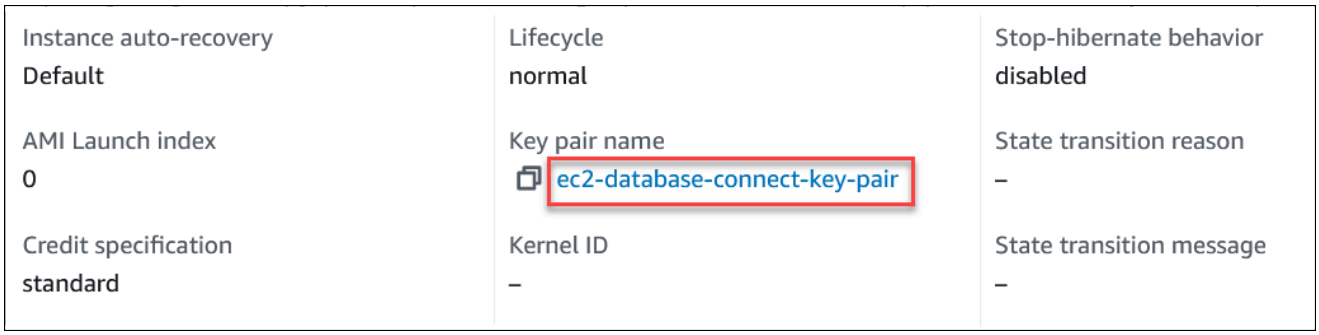
5. Nella pagina Stato avvio prendi nota dell'identificatore per la nuova istanza EC2, ad esempio: `i-1234567890abcdef0`.



6. Scegli l'identificatore dell'istanza EC2 per aprire l'elenco delle istanze EC2, quindi seleziona l'istanza EC2.
7. Nella scheda Dettagli, annota i seguenti valori, necessari quando ti connetti tramite SSH:
   a. In Riepilogo istanza, annota il valore visualizzato in DNS IPv4 pubblico.



   b. In Dettagli istanza, annota il valore visualizzato in Nome coppia di chiavi.



8. Attendi che Instance state (Stato istanza) per l'istanza sia Running (In esecuzione) prima di continuare.
9. Completo Creazione di un cluster di database Amazon Aurora.

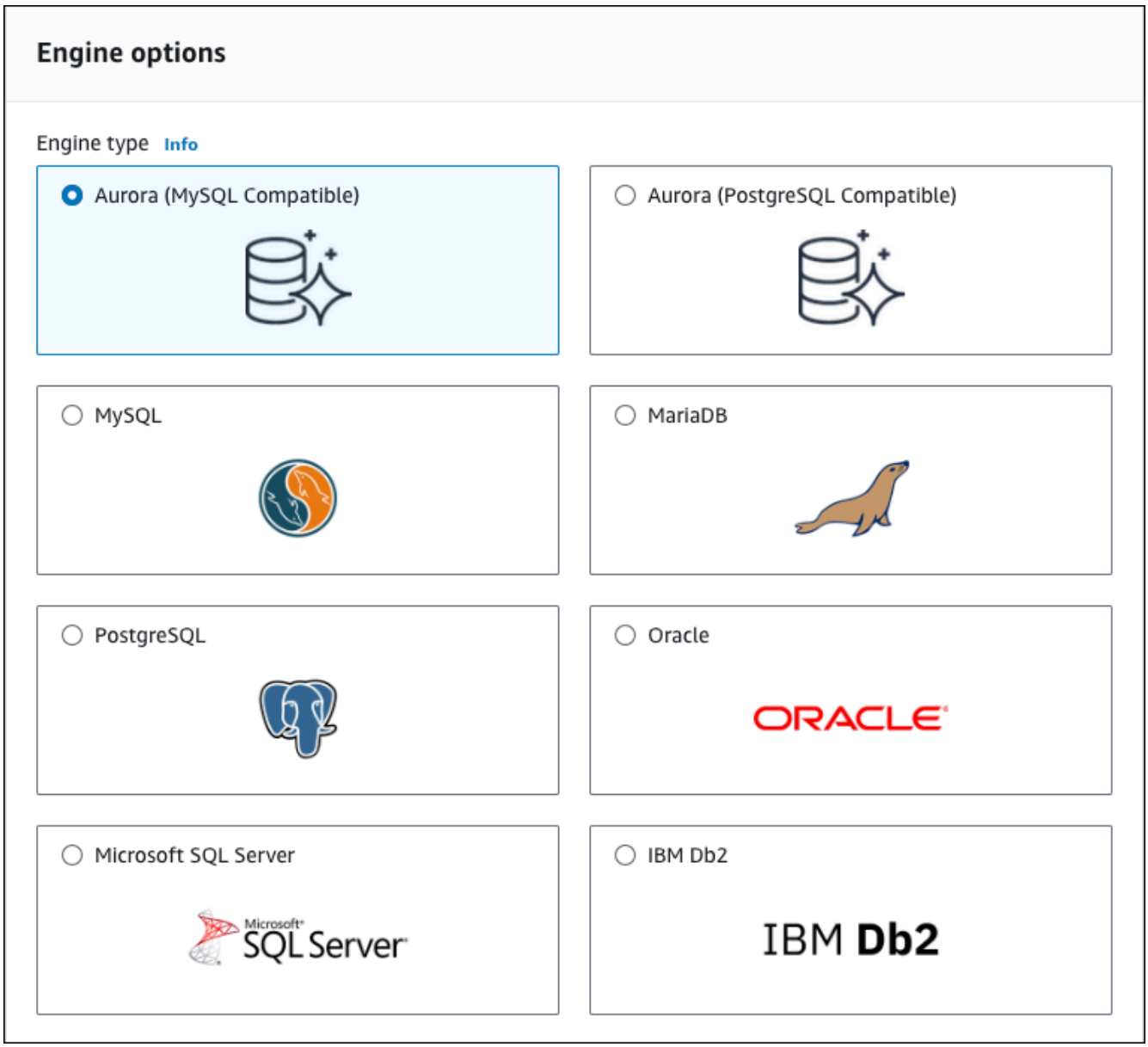
Creazione di un cluster di database Amazon Aurora

Crea un cluster database Amazon Aurora MySQL o Aurora PostgreSQL per conservare i dati utilizzati da un'applicazione web.

Aurora MySQL

Creazione di un cluster di database Aurora MySQL

1. Accedi alla Console di gestione AWS e apri la console Amazon RDS all'indirizzo https://console.aws.amazon.com/rds/.
2. Nell'angolo in alto a destra della Console di gestione AWS, assicurati che la Regione AWS sia identica a quella in cui hai creato l'istanza EC2.
3. Nel riquadro di navigazione, scegliere Databases (Database).
4. Scegliere Create database (Crea database).
5. Nella pagina Crea database scegli Creazione standard.
6. In Opzioni motore scegli Aurora (compatibile con MySQL).

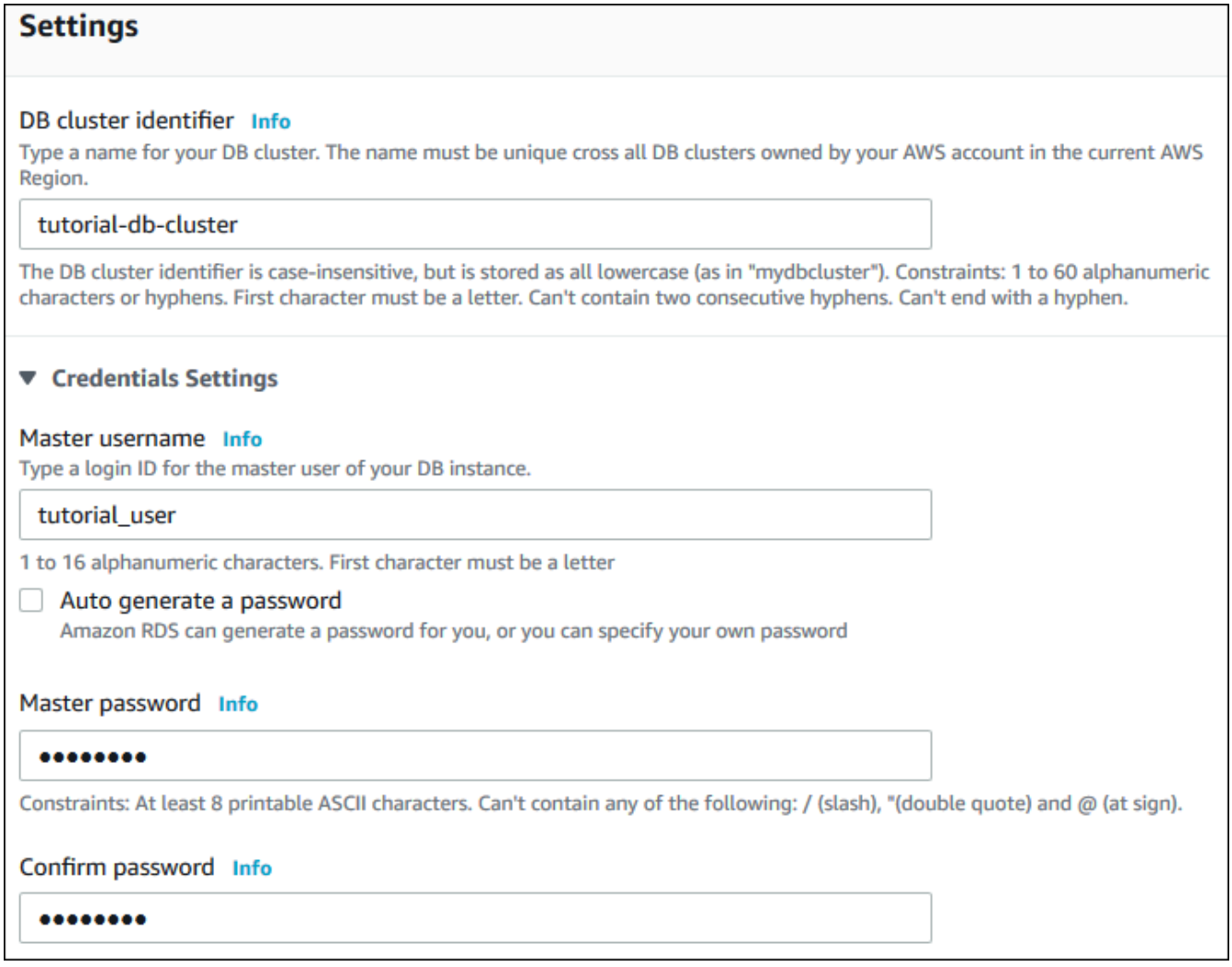


Mantenere i valori predefiniti per la versione e le altre opzioni del motore.

7. Nella sezione Templates (Modelli), selezionare Dev/Test.

8. Nella sezione Settings (Impostazioni) impostare questi valori:

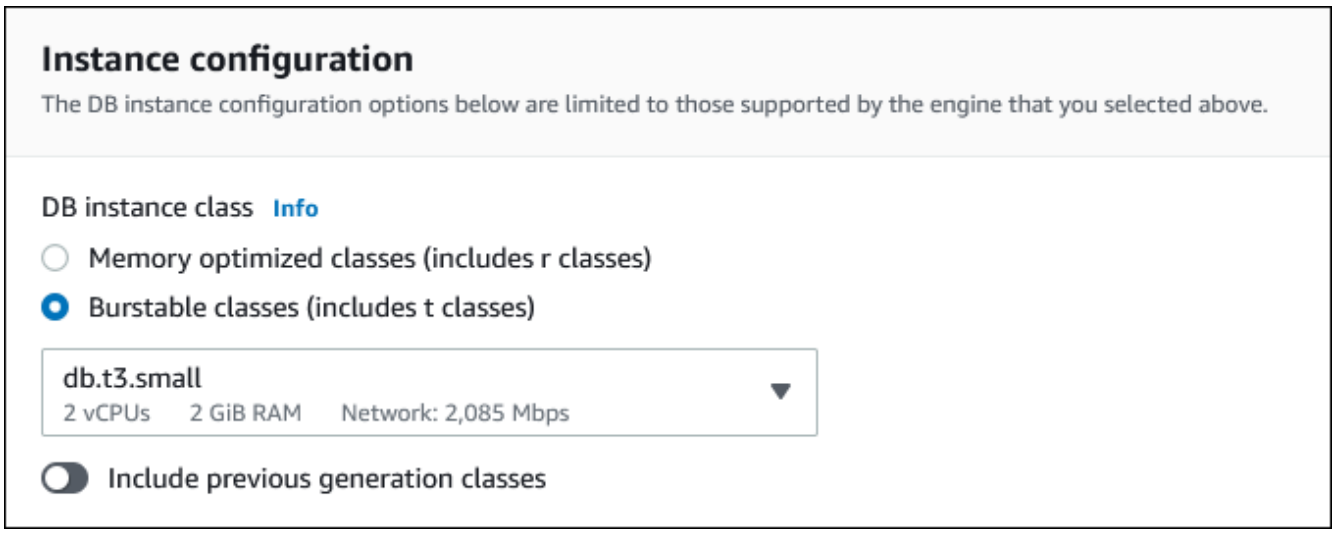
   - DB cluster identifier (Identificatore cluster DB): digita **`tutorial-db-cluster`**.
   - Master username (Nome utente master): digita **`tutorial_user`**.
   - Auto generate a password (Genera automaticamente una password): lascia l'opzione disattivata.
   - Master password (Password master): scegli una password.
   - Confirm password (Conferma la password): –digitare di nuovo la password.



9. Nella sezione Instance configuration (Configurazione dell'istanza), imposta i seguenti valori:

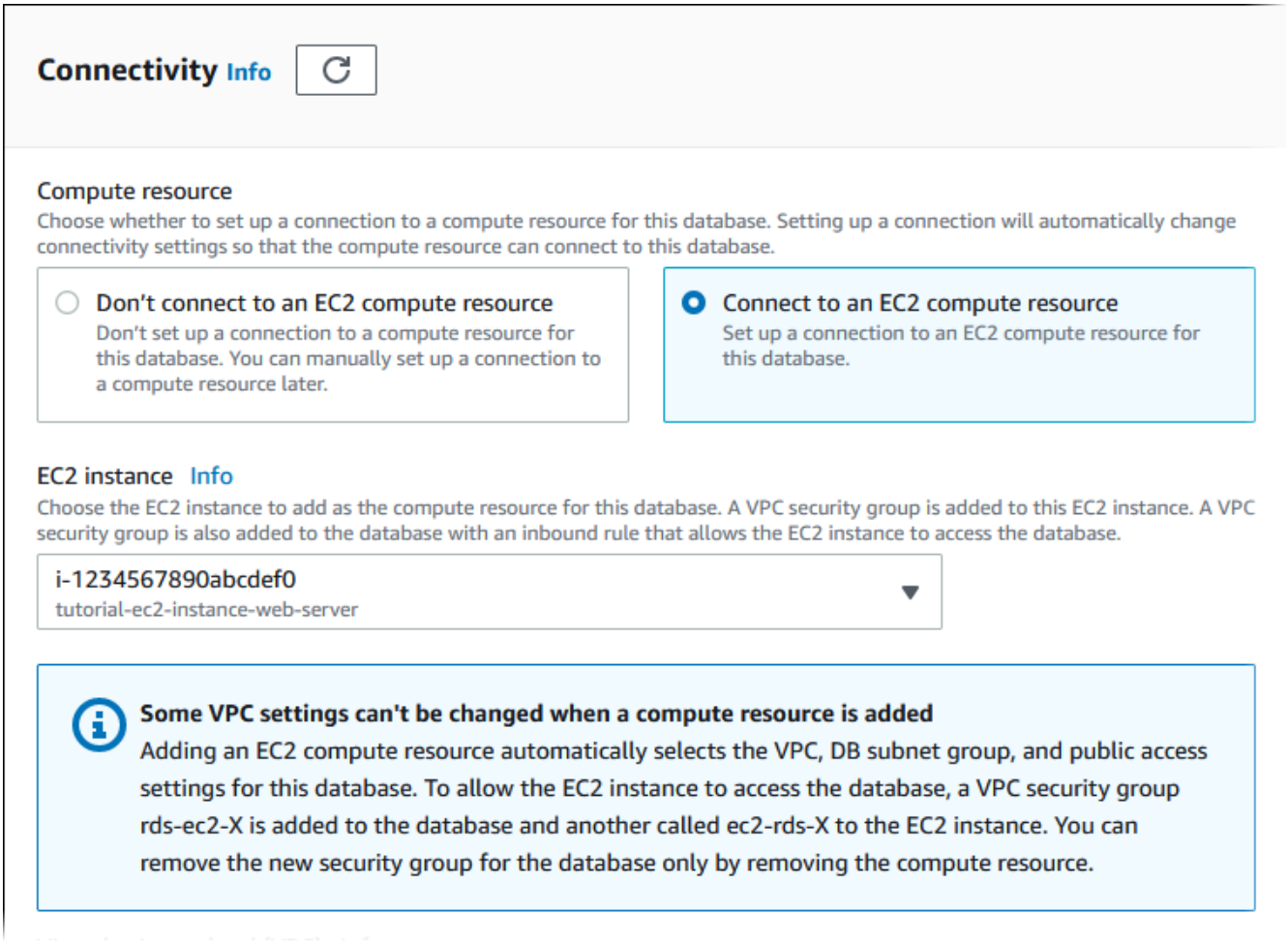
   - Classi espandibili (include le classi t)
   - db.t3.small o db.t3.medium

> **Note**
>
> Consigliamo di utilizzare le classi di istanza database T solo per i server di sviluppo e test o altri server non di produzione. Per ulteriori informazioni sulle classi di istanza T, consulta Tipi di classi di istanza database.



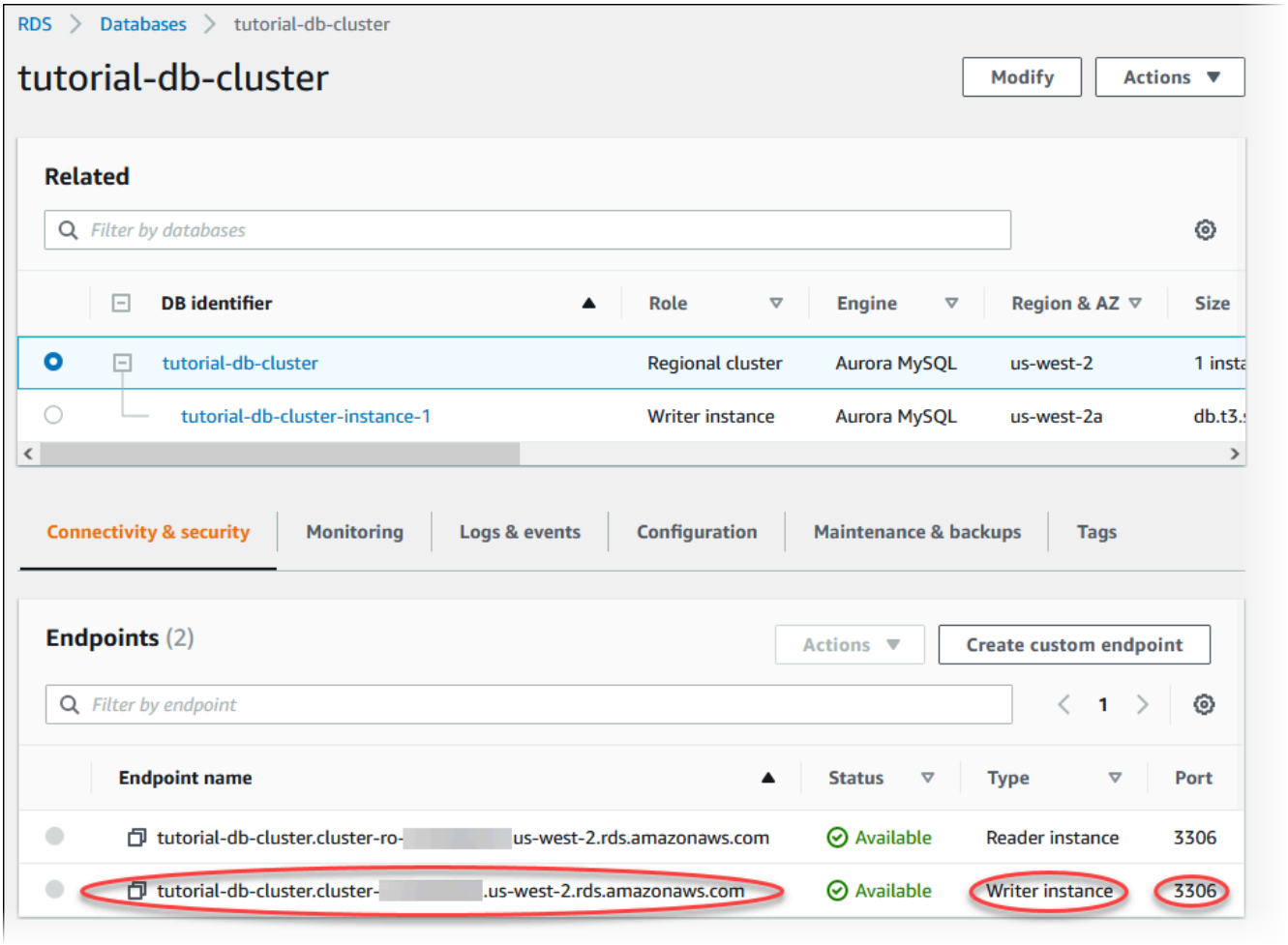
10. Nella sezione Availability and durability (Disponibilità e durata), utilizza i valori predefiniti.
11. Nella sezione Connectivity (Connettività), imposta i seguenti valori e lascia gli altri valori come predefiniti:

    - In Compute resource (Risorse di calcolo), seleziona Connect to an EC2 compute resource (Connetti a una risorsa di calcolo EC2).
    - In EC2 instance (Istanza EC2), scegli l'istanza EC2 creata in precedenza, ad esempio tutorial-ec2-instance-web-server.



12. Aprire la sezione Additional configuration (Configurazione aggiuntiva) e specificare **`sample`** per Initial database name (Nome database iniziale). Lasciare le impostazioni predefinite per le altre opzioni.
13. Per creare il cluster Aurora MySQL DB, scegliere Crea database.

    Il nuovo cluster di database apparirà nell'elenco Database con lo stato Creazione in corso.
14. Attendere che lo Stato del nuovo cluster del database appaia come Disponibile. Quindi scegliere il nome del cluster del database per visualizzarne i dettagli.
15. Nella sezione Connettività e sicurezza, visualizzare Endpoint e Porta dell'istanza di scrittura del database.



Prendere nota dell'endpoint e della porta dell'istanza database writer. Queste informazioni verranno utilizzate per effettuare la connessione del server Web al cluster DB.

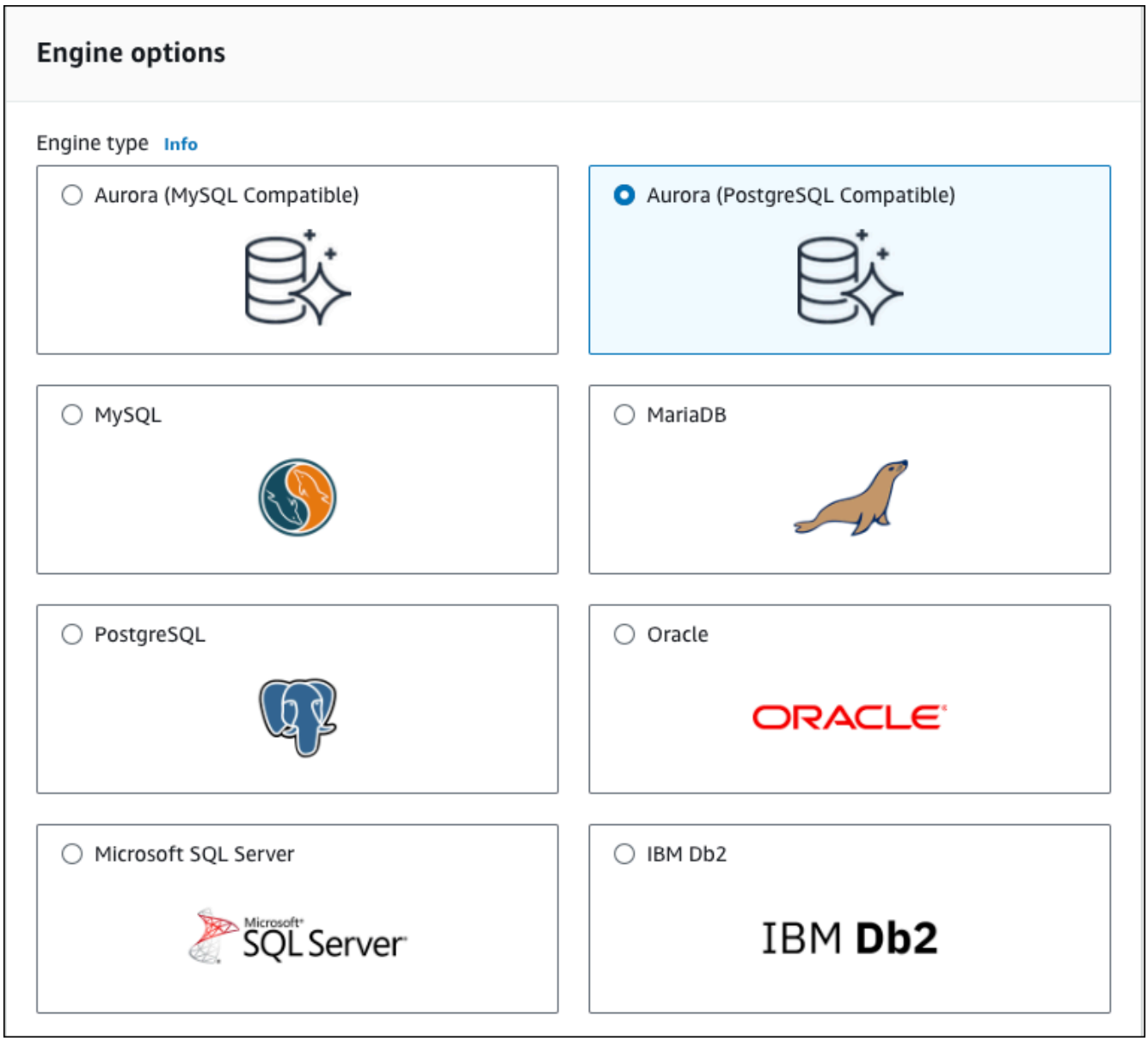
16. Completo Installazione di un server Web nell'istanza EC2.

Aurora PostgreSQL

Creazione di un cluster di database Aurora PostgreSQL

1. Accedi alla Console di gestione AWS e apri la console Amazon RDS all'indirizzo https://console.aws.amazon.com/rds/.
2. Nell'angolo in alto a destra della Console di gestione AWS, assicurati che la Regione AWS sia identica a quella in cui hai creato l'istanza EC2.
3. Nel riquadro di navigazione, scegliere Databases (Database).
4. Scegliere Create database (Crea database).

5. Nella pagina Crea database scegli Creazione standard.
6. In Opzioni motore scegli Aurora (compatibile con PostgreSQL).

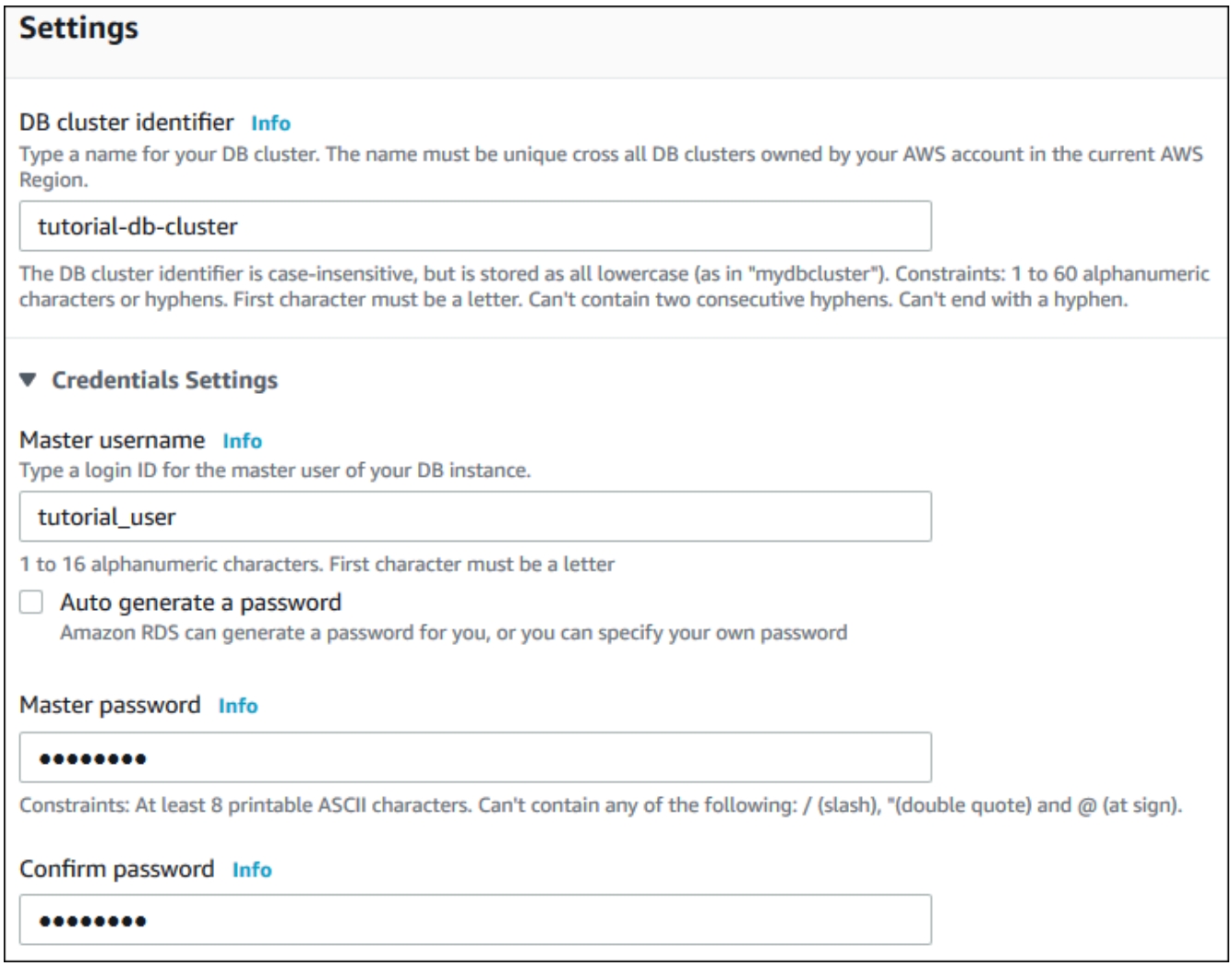


Mantenere i valori predefiniti per la versione e le altre opzioni del motore.

7. Nella sezione Templates (Modelli), selezionare Dev/Test.

8. Nella sezione Settings (Impostazioni) impostare questi valori:

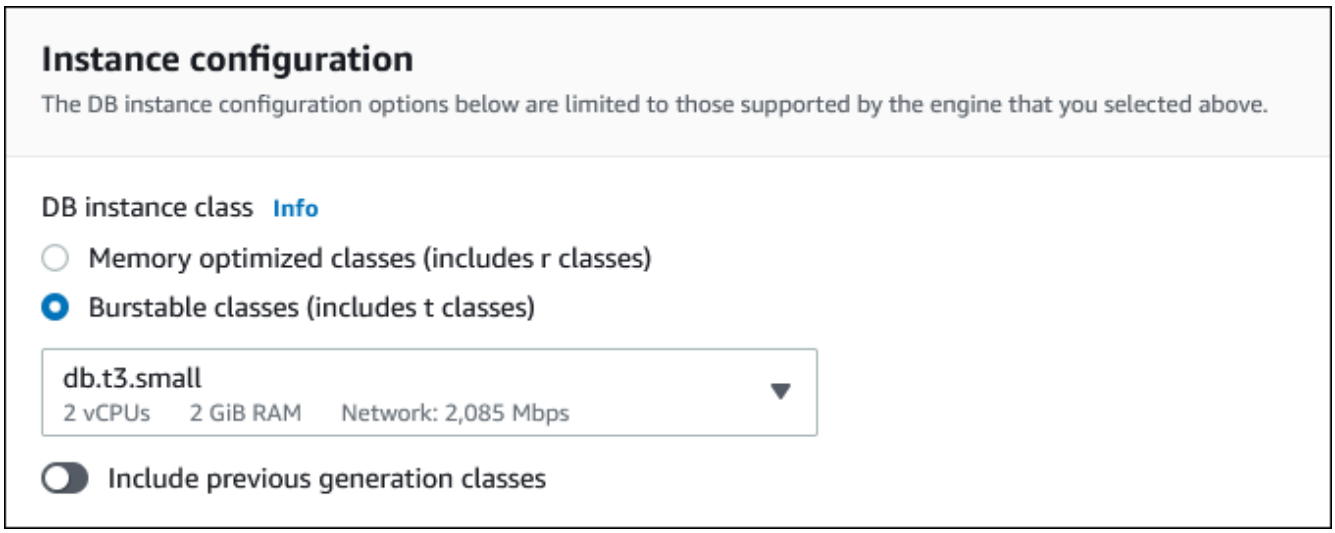
   - DB cluster identifier (Identificatore cluster DB): digita **`tutorial-db-cluster`**.
   - Master username (Nome utente master): digita **`tutorial_user`**.
   - Auto generate a password (Genera automaticamente una password): lascia l'opzione disattivata.
   - Master password (Password master): scegli una password.
   - Confirm password (Conferma la password): –digitare di nuovo la password.



9. Nella sezione Instance configuration (Configurazione dell'istanza), imposta i seguenti valori:

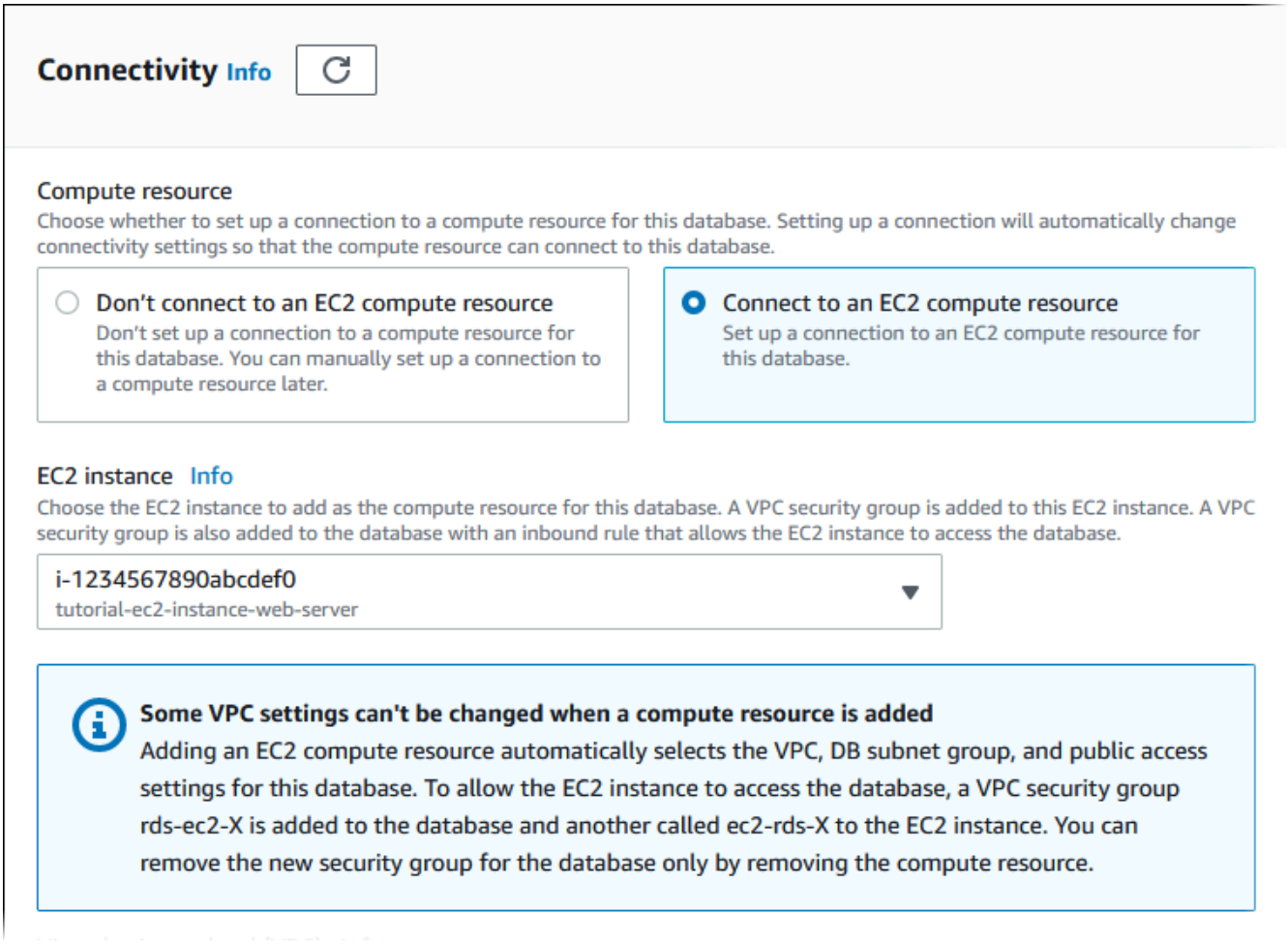
   - Classi espandibili (include le classi t)
   - db.t3.small o db.t3.medium

> **Note**
>
> Consigliamo di utilizzare le classi di istanza database T solo per i server di sviluppo e test o altri server non di produzione. Per ulteriori informazioni sulle classi di istanza T, consulta Tipi di classi di istanza database.



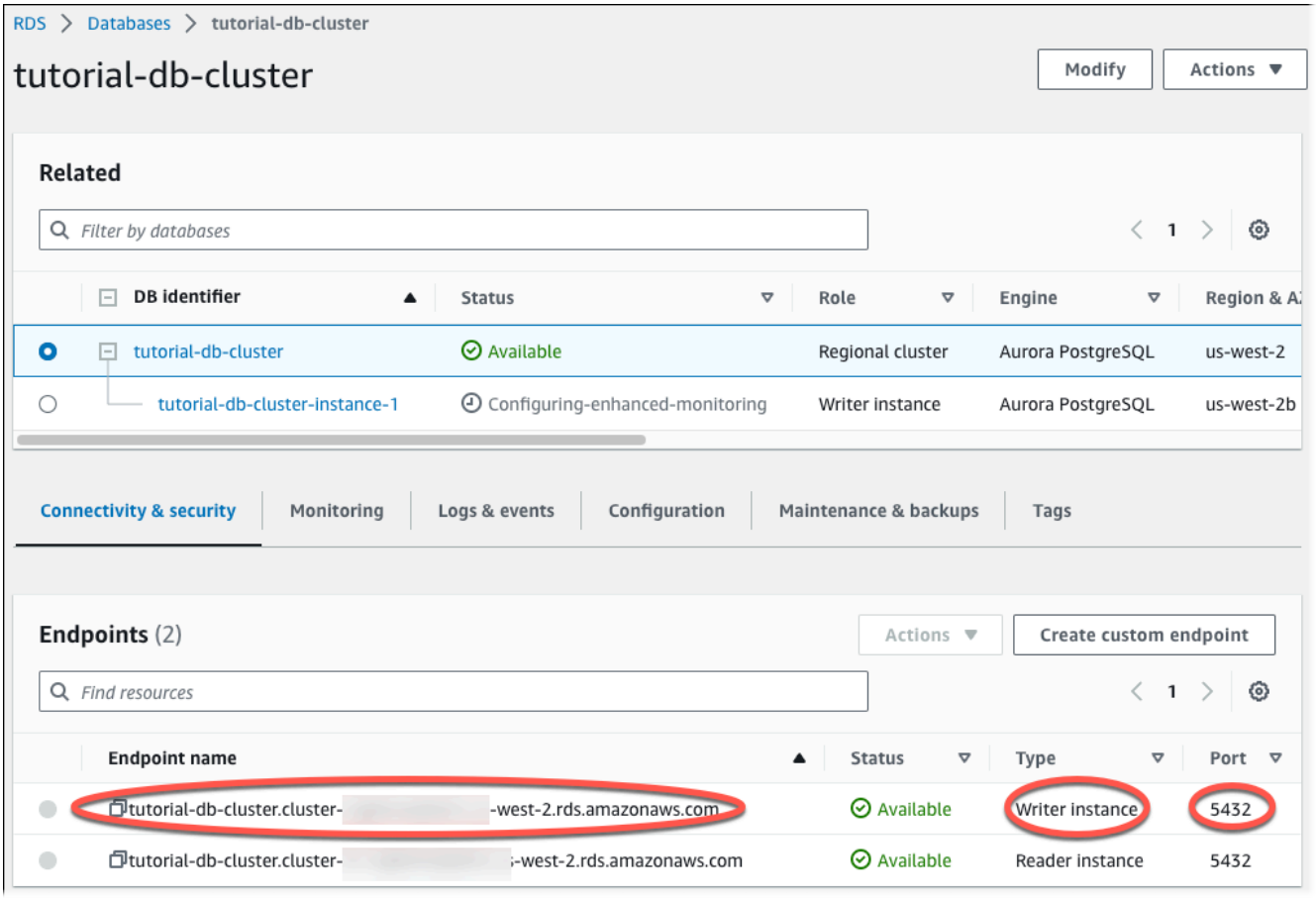
10. Nella sezione Availability and durability (Disponibilità e durata), utilizza i valori predefiniti.
11. Nella sezione Connectivity (Connettività), imposta i seguenti valori e lascia gli altri valori come predefiniti:

    - In Compute resource (Risorse di calcolo), seleziona Connect to an EC2 compute resource (Connetti a una risorsa di calcolo EC2).
    - In EC2 instance (Istanza EC2), scegli l'istanza EC2 creata in precedenza, ad esempio tutorial-ec2-instance-web-server.



12. Aprire la sezione Additional configuration (Configurazione aggiuntiva) e specificare **`sample`** per Initial database name (Nome database iniziale). Lasciare le impostazioni predefinite per le altre opzioni.
13. Per creare il cluster database Aurora MySQL, scegli Crea database.

    Il nuovo cluster di database apparirà nell'elenco Database con lo stato Creazione in corso.
14. Attendere che lo Stato del nuovo cluster del database appaia come Disponibile. Quindi scegliere il nome del cluster del database per visualizzarne i dettagli.
15. Nella sezione Connettività e sicurezza, visualizzare Endpoint e Porta dell'istanza di scrittura del database.



Prendere nota dell'endpoint e della porta dell'istanza database writer. Queste informazioni verranno utilizzate per effettuare la connessione del server Web al cluster DB.

16. Completo Installazione di un server Web nell'istanza EC2.

Installazione di un server Web nell'istanza EC2

Installa un server Web in un'istanza EC2 creata in Avvio di un'istanza EC2 per la connessione al cluster di database. Il server web si connette al cluster di database Amazon Aurora creato in Creazione di un cluster di database Amazon Aurora.

Installazione di un server Web Apache con PHP e MariaDB

Esegui la connessione all'istanza EC2 e installa il server Web Apache.

Per effettuare la connessione all'istanza EC2 e installare il server Web Apache con PHP.

1. Eseguire la connessione all'istanza EC2 creata in precedenza seguendo la procedura riportata in Connect to your Linux instance nella Guida per l'utente di Amazon EC2.

Ti consigliamo di connetterti all'istanza EC2 tramite SSH. Se l'utilità client SSH è installata su Windows, Linux o Mac, puoi connetterti all'istanza utilizzando il comando nel seguente formato:

```
ssh -i location_of_pem_file ec2-user@ec2-instance-public-dns-name
```

Ad esempio, supponi che `ec2-database-connect-key-pair.pem` sia archiviato in `/dir1` su Linux e che il DNS IPv4 pubblico per l'istanza EC2 sia `ec2-12-345-678-90.compute-1.amazonaws.com`. Il comando SSH sarà simile al seguente:

```
ssh -i /dir1/ec2-database-connect-key-pair.pem ec2-
user@ec2-12-345-678-90.compute-1.amazonaws.com
```

2. Ottieni le ultime correzioni di bug e gli aggiornamenti di sicurezza aggiornando il software sulla tua istanza EC2. A questo scopo, eseguire il comando seguente.

> **Note**
>
> L'opzione `-y` installa gli aggiornamenti senza chiedere conferma. Per esaminare gli aggiornamenti prima di installarli, omettere questa opzione.

```
sudo dnf update -y
```

3. Al completamento degli aggiornamenti, installa il server Web Apache, PHP e il software MariaDB utilizzando i comandi seguenti. Con questo comando vengono installati contemporaneamente più pacchetti software e dipendenze correlate.

   MariaDB & MySQL

   ```
   sudo dnf install -y httpd php php-mysqli mariadb105
   ```

   PostgreSQL

   ```
   sudo dnf install -y httpd php php-pgsql postgresql15
   ```

Se si verifica un errore, è possibile che l'istanza non sia stata lanciata con un'AMI Amazon Linux 2023. Puoi invece utilizzare l'AMI Amazon Linux 2. È possibile visualizzare la versione di Amazon Linux con il comando seguente.

```
cat /etc/system-release
```

Per ulteriori informazioni, consulta la pagina relativa all'aggiornamento del software dell'istanza.

4. Avviare il server Web con il comando visualizzato di seguito.

```
sudo systemctl start httpd
```

È possibile verificare che il server Web sia installato e avviato correttamente. A tale scopo, immettere il nome DNS (Domain Name System) pubblico dell'istanza EC2 nella barra degli indirizzi di un browser Web, ad esempio: `http://ec2-42-8-168-21.us-west-1.compute.amazonaws.com`. Se il server Web è in esecuzione, verrà visualizzata la pagina di test di Apache.

Se la pagina di test Apache non viene visualizzata, controllare le regole in entrata per il gruppo di sicurezza VPC creato in Tutorial: Creazione di un Amazon VPC da utilizzare con un cluster database (solo IPv4). Assicurati che le regole in entrata ne includano una che consenta l'accesso HTTP (porta 80) per l'indirizzo IP utilizzato per connettersi al server Web.

> **Note**
>
> La pagina di test di Apache viene visualizzata solo quando la directory principale dei documenti è vuota, `/var/www/html`. Dopo aver aggiunto contenuti alla directory root dei documenti, i contenuti vengono visualizzati all'indirizzo DNS pubblico dell'istanza EC2. Prima di questo punto, vengono visualizzati nella pagina di test di Apache.

5. Configurare il server Web affinché si avvii a ogni avvio del sistema tramite il comando `systemctl`.

```
sudo systemctl enable httpd
```

Per permettere a ec2-user di gestire file nella directory principale predefinita del server Web Apache, è necessario modificare la proprietà e le autorizzazioni della directory /var/www. Sono disponibili molti modi per completare questa attività. In questo tutorial, aggiungi l'utente ec2-user al gruppo apache per assegnare la proprietà del gruppo apache della directory /var/www e assegnare autorizzazioni di scrittura al gruppo.

Per impostare le autorizzazioni dei file sul server Web Apache

1. Aggiungere l'utente ec2-user al gruppo apache.

   ```
   sudo usermod -a -G apache ec2-user
   ```

2. Per aggiornare le autorizzazioni e includere il nuovo gruppo apache, eseguire la disconnessione.

   ```
   exit
   ```

3. Effettuare nuovamente l'accesso e verificare che il gruppo apache esista mediante il comando groups.

   ```
   groups
   ```

   L'output avrà un aspetto simile al seguente:

   ```
   ec2-user adm wheel apache systemd-journal
   ```

4. Cambiare la proprietà del gruppo della directory /var/www e dei suoi contenuti sul gruppo apache.

   ```
   sudo chown -R ec2-user:apache /var/www
   ```

5. Cambiare le autorizzazioni della directory /var/www e delle sue sottodirectory per aggiungere le autorizzazioni di scrittura di gruppo e impostare l'ID di gruppo per le sottodirectory create in futuro.

   ```
   sudo chmod 2775 /var/www
   find /var/www -type d -exec sudo chmod 2775 {} \;
   ```

6. Cambiare le autorizzazioni in modo ricorsivo per i file nella directory /var/www e nelle sue sottodirectory per aggiungere le autorizzazioni di scrittura di gruppo.

```
find /var/www -type f -exec sudo chmod 0664 {} \;
```

Ora, `ec2-user` (e qualsiasi membro futuro del gruppo `apache`) può aggiungere, eliminare e modificare i file nella root del documento di Apache. Questo consente di aggiungere contenuti, ad esempio un sito Web statico o un'applicazione PHP.

> **Note**
>
> Un server Web che esegue il protocollo HTTP non offre alcuna sicurezza di trasporto per i dati inviati e ricevuti. Quando ti connetti a un server HTTP tramite un browser Web, la molte informazioni sono visibili a persone non autorizzate in qualsiasi punto del percorso di rete. Queste informazioni includono gli URL visitati, il contenuto delle pagine Web ricevute e i contenuti (incluse le password) di eventuali moduli HTML.
> La best practice per la protezione del tuo server Web prevede l'installazione del supporto per HTTPS (HTTP Secure). Questo protocollo protegge i dati con la crittografia SSL/TLS. Per ulteriori informazioni, consulta Esercitazione: Configurare SSL/TLS con l'AMI di Amazon Linux nella Guida per l'utente di Amazon EC2 .

Connessione del server web Apache al cluster di database

Successivamente, aggiungere contenuti al server Web Apache che effettua la connessione all'Amazon Aurora Cluster DB.

Per aggiungere contenuti al server Web Apache che effettua la connessione all'Cluster DB.

1. Mentre è ancora in corso la connessione all'istanza EC2, modificare la directory in `/var/www` e creare una nuova sottodirectory denominata `inc`.

   ```
   cd /var/www
   mkdir inc
   cd inc
   ```

2. Creare un nuovo file nella directory `inc` denominato `dbinfo.inc` e poi modificarlo con nano (o un altro editor a scelta).

   ```
   >dbinfo.inc
   ```

```
nano dbinfo.inc
```

3. Aggiungi i seguenti contenuti al file `dbinfo.inc`. Qui, *`db_instance_endpoint`* è l'endpoint dell'istanza di scrittura del cluster database, senza la porta, per il cluster database.

> **Note**
>
> Si consiglia di inserire le informazioni relative al nome utente e alla password in una cartella che non fa parte della directory principale del documento per il server Web. In questo modo si riduce la possibilità che le informazioni di sicurezza vengano esposte. Assicurati di modificare `master password` in una password adatta per la tua applicazione.

```
<?php

define('DB_SERVER', 'db_cluster_writer_endpoint');
define('DB_USERNAME', 'tutorial_user');
define('DB_PASSWORD', 'master password');
define('DB_DATABASE', 'sample');
?>
```

4. Salvare e chiudere il file `dbinfo.inc`. Se stai usando nano, salva e chiudi il file usando Ctrl+S e Ctrl+X.
5. Cambiare la directory in `/var/www/html`.

```
cd /var/www/html
```

6. Creare un nuovo file nella directory `html` denominato `SamplePage.php` e poi modificarlo con nano (o un altro editor a scelta).

```
>SamplePage.php
nano SamplePage.php
```

7. Aggiungere i seguenti contenuti al file `SamplePage.php`:

   MariaDB & MySQL

```
<?php include "../inc/dbinfo.inc"; ?>
<html>
```

```
<body>
<h1>Sample page</h1>
<?php

  /* Connect to MySQL and select the database. */
  $connection = mysqli_connect(DB_SERVER, DB_USERNAME, DB_PASSWORD);

  if (mysqli_connect_errno()) echo "Failed to connect to MySQL: " .
 mysqli_connect_error();

  $database = mysqli_select_db($connection, DB_DATABASE);

  /* Ensure that the EMPLOYEES table exists. */
  VerifyEmployeesTable($connection, DB_DATABASE);

  /* If input fields are populated, add a row to the EMPLOYEES table. */
  $employee_name = htmlentities($_POST['NAME']);
  $employee_address = htmlentities($_POST['ADDRESS']);

  if (strlen($employee_name) || strlen($employee_address)) {
    AddEmployee($connection, $employee_name, $employee_address);
  }
?>

<!-- Input form -->
<form action="<?PHP echo $_SERVER['SCRIPT_NAME'] ?>" method="POST">
  <table border="0">
    <tr>
      <td>NAME</td>
      <td>ADDRESS</td>
    </tr>
    <tr>
      <td>
        <input type="text" name="NAME" maxlength="45" size="30" />
      </td>
      <td>
        <input type="text" name="ADDRESS" maxlength="90" size="60" />
      </td>
      <td>
        <input type="submit" value="Add Data" />
      </td>
    </tr>
  </table>
</form>
```

```
<!-- Display table data. -->
<table border="1" cellpadding="2" cellspacing="2">
  <tr>
    <td>ID</td>
    <td>NAME</td>
    <td>ADDRESS</td>
  </tr>

<?php

$result = mysqli_query($connection, "SELECT * FROM EMPLOYEES");

while($query_data = mysqli_fetch_row($result)) {
  echo "<tr>";
  echo "<td>",$query_data[0], "</td>",
       "<td>",$query_data[1], "</td>",
       "<td>",$query_data[2], "</td>";
  echo "</tr>";
}
?>

</table>

<!-- Clean up. -->
<?php

  mysqli_free_result($result);
  mysqli_close($connection);

?>

</body>
</html>


<?php

/* Add an employee to the table. */
function AddEmployee($connection, $name, $address) {
   $n = mysqli_real_escape_string($connection, $name);
   $a = mysqli_real_escape_string($connection, $address);

   $query = "INSERT INTO EMPLOYEES (NAME, ADDRESS) VALUES ('$n', '$a');";
```

```
   if(!mysqli_query($connection, $query)) echo("<p>Error adding employee data.</
p>");
}

/* Check whether the table exists and, if not, create it. */
function VerifyEmployeesTable($connection, $dbName) {
  if(!TableExists("EMPLOYEES", $connection, $dbName))
  {
     $query = "CREATE TABLE EMPLOYEES (
         ID int(11) UNSIGNED AUTO_INCREMENT PRIMARY KEY,
         NAME VARCHAR(45),
         ADDRESS VARCHAR(90)
       )";

     if(!mysqli_query($connection, $query)) echo("<p>Error creating table.</
p>");
  }
}

/* Check for the existence of a table. */
function TableExists($tableName, $connection, $dbName) {
  $t = mysqli_real_escape_string($connection, $tableName);
  $d = mysqli_real_escape_string($connection, $dbName);

  $checktable = mysqli_query($connection,
      "SELECT TABLE_NAME FROM information_schema.TABLES WHERE TABLE_NAME = '$t'
 AND TABLE_SCHEMA = '$d'");

  if(mysqli_num_rows($checktable) > 0) return true;

  return false;
}
?>
```

PostgreSQL

```
<?php include "../inc/dbinfo.inc"; ?>

<html>
<body>
<h1>Sample page</h1>
```

```
<?php

/* Connect to PostgreSQL and select the database. */
$constring = "host=" . DB_SERVER . " dbname=" . DB_DATABASE . " user=" .
 DB_USERNAME . " password=" . DB_PASSWORD ;
$connection = pg_connect($constring);

if (!$connection){
 echo "Failed to connect to PostgreSQL";
 exit;
}

/* Ensure that the EMPLOYEES table exists. */
VerifyEmployeesTable($connection, DB_DATABASE);

/* If input fields are populated, add a row to the EMPLOYEES table. */
$employee_name = htmlentities($_POST['NAME']);
$employee_address = htmlentities($_POST['ADDRESS']);

if (strlen($employee_name) || strlen($employee_address)) {
  AddEmployee($connection, $employee_name, $employee_address);
}

?>

<!-- Input form -->
<form action="<?PHP echo $_SERVER['SCRIPT_NAME'] ?>" method="POST">
  <table border="0">
    <tr>
      <td>NAME</td>
      <td>ADDRESS</td>
    </tr>
    <tr>
      <td>
    <input type="text" name="NAME" maxlength="45" size="30" />
      </td>
      <td>
    <input type="text" name="ADDRESS" maxlength="90" size="60" />
      </td>
      <td>
    <input type="submit" value="Add Data" />
      </td>
    </tr>
  </table>
```

```
</form>
<!-- Display table data. -->
<table border="1" cellpadding="2" cellspacing="2">
  <tr>
    <td>ID</td>
    <td>NAME</td>
    <td>ADDRESS</td>
  </tr>

<?php

$result = pg_query($connection, "SELECT * FROM EMPLOYEES");

while($query_data = pg_fetch_row($result)) {
  echo "<tr>";
  echo "<td>",$query_data[0], "</td>",
       "<td>",$query_data[1], "</td>",
       "<td>",$query_data[2], "</td>";
  echo "</tr>";
}
?>
</table>

<!-- Clean up. -->
<?php

  pg_free_result($result);
  pg_close($connection);
?>
</body>
</html>


<?php

/* Add an employee to the table. */
function AddEmployee($connection, $name, $address) {
   $n = pg_escape_string($name);
   $a = pg_escape_string($address);
   echo "Forming Query";
   $query = "INSERT INTO EMPLOYEES (NAME, ADDRESS) VALUES ('$n', '$a');";

   if(!pg_query($connection, $query)) echo("<p>Error adding employee data.</
p>");
```

```
}

/* Check whether the table exists and, if not, create it. */
function VerifyEmployeesTable($connection, $dbName) {
  if(!TableExists("EMPLOYEES", $connection, $dbName))
  {
     $query = "CREATE TABLE EMPLOYEES (
         ID serial PRIMARY KEY,
         NAME VARCHAR(45),
         ADDRESS VARCHAR(90)
       )";

     if(!pg_query($connection, $query)) echo("<p>Error creating table.</p>");
  }
}
/* Check for the existence of a table. */
function TableExists($tableName, $connection, $dbName) {
  $t = strtolower(pg_escape_string($tableName)); //table name is case sensitive
  $d = pg_escape_string($dbName); //schema is 'public' instead of 'sample' db
 name so not using that

  $query = "SELECT TABLE_NAME FROM information_schema.TABLES WHERE TABLE_NAME =
 '$t';";
  $checktable = pg_query($connection, $query);

  if (pg_num_rows($checktable) >0) return true;
  return false;

}
?>
```

8. Salvare e chiudere il file `SamplePage.php`.
9. Verificare che il server Web effettui correttamente la connessione all'cluster DB aprendo un browser web e navigando fino a `http://`*`EC2 instance endpoint`*`/SamplePage.php`, ad esempio: `http://ec2-12-345-67-890.us-west-2.compute.amazonaws.com/SamplePage.php`.

È possibile utilizzare `SamplePage.php` per aggiungere dati all'cluster DB. I dati aggiunti verranno visualizzati nella pagina. Per verificare che i dati siano stati inseriti nella tabella, installa il client MySQL nell'istanza Amazon EC2. Esegui quindi la connessione al cluster di database ed esegui la query sulla tabella.

Per ulteriori informazioni sulla connessione a un cluster DB, consultare Connessione a un cluster database Amazon Aurora.

Per assicurarsi che l'il cluster DB sia il più possibile sicura, verificare che le fonti esterne al VPC non possano connettersi all'Cluster DB.

Dopo aver terminato il test del server Web e del database, è necessario eliminare l'istanza il DB e l'istanza Amazon EC2.

- Per eliminare un cluster di database, segui le istruzioni in Eliminazione di cluster e istanze database di Aurora. Non è necessario creare uno snapshot finale.
- Per terminare un'istanza Amazon EC2, segui le istruzioni riportate in Termina istanza nella Guida per l'utente di Amazon EC2.

Tutorial di Amazon Aurora e codice di esempio

La AWS documentazione include diversi tutorial che ti guidano attraverso i casi d'uso più comuni di . Molti di questi tutorial mostrano come usare Amazon con altri servizi. AWS Inoltre, è possibile accedere al codice di esempio in .

> **Note**
>
> Puoi trovare altri tutorial nel Blog di AWS Database. Per ulteriori informazioni sulla formazione, consulta AWS Training and Certification.

Argomenti

- Tutorial in questa guida
- Tutorial in altre guide AWS
- Tutorial e codice di esempio in GitHub
- AWS Ricettario del database
- AWS portale di contenuti per workshop e laboratori per PostgreSQL
- AWS portale di contenuti per workshop e laboratori per MySQL
- Utilizzo di questo servizio con un AWS SDK

Tutorial in questa guida

I seguenti tutorial mostrano come eseguire le attività comuni con Amazon Aurora:

- Tutorial: Creazione di un Amazon VPC da utilizzare con un cluster database (solo IPv4)

  Scopri come includere un cluster di database in un cloud privato virtuale (VPC) basato sul servizio Amazon VPC. In questo caso, il VPC condivide i dati con un server Web in esecuzione su un' EC2 istanza Amazon nello stesso VPC.
- Tutorial: Creazione di un VPC per l'utilizzo con un cluster database (modalità dual-stack)

  Scopri come includere un cluster di database in un cloud privato virtuale (VPC) basato sul servizio Amazon VPC. In questo caso, il VPC condivide i dati con EC2 un'istanza Amazon nello stesso

VPC. In questo tutorial crei il VPC per questo scenario che funziona con un database in esecuzione in modalità dual-stack.

- Tutorial: creazione di un server Web e un cluster database Amazon Aurora

  Questo tutorial consente di installare un server Web Apache con PHP e creare un database MySQL. Il server Web viene eseguito su un' EC2 istanza Amazon che utilizza Amazon Linux e il database MySQL è . Sia l' EC2 istanza Amazon che il cluster di DB vengono eseguiti in un Amazon VPC.

- Tutorial: Ripristino di un cluster di database Amazon Aurora da uno snapshot del cluster di database

  Scopri come ripristinare un cluster di database da uno snapshot di cluster di database.

- Tutorial: utilizza i tag per specificare i cluster DB Aurora da arrestare

  Scopri come utilizzare i tag per specificare i cluster di database di Aurora da arrestare.

- Tutorial: Registrazione delle modifiche dello stato di un'istanza database tramite Amazon EventBridge

  Scopri come registrare una modifica dello stato di un'istanza DB utilizzando Amazon EventBridge e AWS Lambda.

Tutorial in altre guide AWS

I seguenti tutorial in altre AWS guide mostrano come eseguire attività comuni con Amazon :

> **Note**
>
> Alcuni tutorial utilizzano istanze database Amazon RDS, ma possono essere adattate per utilizzare cluster di database Aurora.

- Tutorial: Aurora Serverless nella Guida per gli sviluppatori di AWS AppSync

  Scopri come utilizzare per AWS AppSync fornire una fonte di dati per l'esecuzione di comandi SQL su cluster Aurora Serverless DB con l'API Data abilitata. È possibile utilizzare i resolver di AWS AppSync per eseguire le istruzioni SQL sull'API dati tramite query GraphQL, mutazioni e sottoscrizioni.

- Tutorial: Rotazione di un segreto per un AWS database nella guida per l'utente Gestione dei segreti AWS

  Scopri come creare un segreto per un AWS database e configurare il segreto in modo che ruoti in base a una pianificazione. Attivi una rotazione manualmente e confermi che la nuova versione del segreto continua a fornire l'accesso.
- Tutorial ed esempi nella Guida per gli sviluppatori di AWS Elastic Beanstalk

  Scopri come distribuire applicazioni che utilizzano database Amazon RDS con. AWS Elastic Beanstalk
- Utilizzo dei dati da un database Amazon RDS per creare un'origine dati Amazon ML nella Amazon Machine Learning Developer Guide

  Scopri come creare un oggetto dell'origine dati Amazon Machine Learning (Amazon ML) dai dati memorizzati in un'istanza database MySQL.
- Abilitazione manuale dell'accesso a un'istanza Amazon RDS in un VPC nella Guida per l'utente di Amazon Quick Suite

  Scopri come abilitare l'accesso Quick Suite a un'istanza database Amazon RDS in un VPC.

Tutorial e codice di esempio in GitHub

I seguenti tutorial e codice di esempio GitHub mostrano come eseguire attività comuni con Amazon :

- Creazione di un libreria di prestiti Aurora Serverless v2

  Scopri come creare un'applicazione libreria di prestiti in cui i clienti possono prendere in prestito e restituire libri. L'esempio utilizza e. Aurora Serverless v2 AWS SDK per Python (Boto3)
- Creazione di un'applicazione di tracciamento degli articoli Amazon Aurora con un REST API Spring che esegue query dei dati Aurora Serverless v2 tramite SDK per Java 2.x

  Scopri come creare una REST API Spring che esegue query sui dati di Aurora Serverless v2. È destinato all'utilizzo da parte di un'applicazione React che usa SDK per Java 2.x.
- Creazione di un'applicazione di tracciamento degli articoli di Amazon Aurora che interroga i dati utilizzando Aurora Serverless v2AWS SDK per PHP

  Scopri come creare un'applicazione che utilizza `RdsDataClient` dell'API dati e Aurora Serverless v2 per monitorare e creare report degli elementi di lavoro. L'esempio utilizza. AWS SDK per PHP

- Creazione di un'applicazione di tracciamento degli articoli Amazon Aurora che esegue query dei dati Aurora Serverless v2 tramite AWS SDK per Python (Boto3)

  Scopri come creare un'applicazione che utilizza `RdsDataClient` dell'API dati e Aurora Serverless v2 per monitorare e creare report degli elementi di lavoro. L'esempio utilizza AWS SDK per Python (Boto3).

AWS Ricettario del database

Il AWS DB Cookbook è una guida completa ai database che insegna come creare, implementare e gestire soluzioni di database ad alte prestazioni ed economiche su. AWS Step-by-stepi tutorial ti guidano nella creazione di applicazioni pronte per la produzione e nella distribuzione delle app con modelli. CloudFormation Apprenderai AWS i servizi essenziali costruendo infrastrutture, implementando reti, sviluppando architetture serverless, gestendo database e integrando l'intelligenza artificiale generativa. Scopri le AWS migliori pratiche che ti aiutano a creare soluzioni sicure e scalabili ottimizzando al contempo i costi. Che tu sia un principiante AWS o un professionista esperto, AWS DB Cookbook ti aiuta a sviluppare competenze per risolvere le sfide comuni dei database e implementare soluzioni pronte per l'uso aziendale. Il ricettario include le seguenti sezioni:

- Guida introduttiva alle AWS applicazioni DB: scopri le AWS nozioni fondamentali come configurare il tuo account e l'ambiente Jupyter Notebook.
- Nozioni di base sui database: esplora i concetti essenziali del database e confronta i servizi di database per scegliere la soluzione AWS giusta per i tuoi carichi di lavoro.
- App web serverless con Amazon Aurora: crea end-to-end un'applicazione di vendita al dettaglio con Amazon Aurora PostgreSQL che gestisce inventario, ordini e dati dei clienti.
- Monitoring and observability: configura il monitoraggio delle prestazioni e gli avvisi per identificare potenziali problemi del database prima che abbiano un impatto sulle applicazioni.
- Scaling with Amazon Aurora: apprendi come creare implementazioni resilienti multi-Regione con Aurora DSQL e come aumentare verticalmente i database per una maggiore potenza di elaborazione oppure orizzontalmente su più istanze per una maggiore capacità.
- Optimization performance and cost: ottimizza le prestazioni del database e riduci i costi con strategie di ottimizzazione comprovate.
- Passa a database AWS creati appositamente: crea un'infrastruttura sicura e affidabile in grado di scalare le tue soluzioni di intelligenza artificiale generativa e le applicazioni basate sui dati dal prototipo all'implementazione aziendale.

- GenAI applications with RAG: crea un sistema di ricerca intelligente per documenti assicurativi e sanitari che utilizza generazione potenziata da recupero dati (RAG) per fornire risultati accurati e sensibili al contesto.

AWS portale di contenuti per workshop e laboratori per PostgreSQL

La seguente raccolta di workshop e altri contenuti pratici ti aiuta a comprendere le caratteristiche e le funzionalità di Amazon Aurora PostgreSQL:

- Creazione di un nuovo cluster Aurora manualmente

  Informazioni su come creare manualmente un cluster Amazon Aurora PostgreSQL.

- Configurazione di Cloud9 e inizializzazione del database

  Informazioni su come configurare Cloud9 e inizializzare il database PostgreSQL.

- Clonazione rapida

  Informazioni su come creare un clone rapido Aurora.

- Gestione del piano di query

  Informazioni su come controllare i piani di esecuzione per una serie di istruzioni tramite l gestione dei piani di query.

- Gestione della cache del cluster

  Informazioni sulla funzionalità di gestione della cache del cluster in Aurora PostgreSQL.

- Streaming delle attività di database

  Informazioni su come monitorare e controllare l'attività del database con questa funzione.

- Uso di Approfondimenti sulle prestazioni

  Informazioni su come monitorare e ottimizzare la tua istanza DB tramite Approfondimenti sulle prestazioni.

- Monitoraggio delle prestazioni con strumenti RDS

  Scopri come utilizzare AWS gli strumenti Postgres (Cloudwatch, Enhanced Monitoring, Slow Query Logs, Performance Insights, PostgreSQL Catalog Views) per comprendere i problemi di prestazioni e identificare modi per migliorare le prestazioni del tuo database.

- Repliche di lettura con dimensionamento automatico

  Informazioni su come funziona il dimensionamento automatico delle repliche di lettura Aurora utilizzando uno script per la generazione di carichi.

- Test della tolleranza ai guasti

  Informazioni su come un cluster di database può gestire un errore.

- Database globale Aurora

  Informazioni su Database globale Amazon Aurora.

- Utilizzo del machine learning

  Informazioni sul machine learning Aurora.

- Aurora Serverless v2

  Informazioni su Aurora Serverless v2.

- Trusted Language Extensions per Aurora PostgreSQL

  Informazioni su come creare estensioni ad alte prestazioni che funzionano in sicurezza su Aurora PostgreSQL.

AWS portale di contenuti per workshop e laboratori per MySQL

La seguente raccolta di workshop e altri contenuti pratici ti aiuta a comprendere le caratteristiche e le funzionalità di Amazon Aurora MySQL:

- Creazione di un cluster Aurora

  Informazioni su come creare manualmente un cluster Amazon Aurora MySQL.

- Creazione di un ambiente IDE basato su Cloud9 per connettersi al database

  Informazioni su come configurare Cloud9 e inizializzare il database MySQL.

- Clonazione rapida

  Informazioni su come creare un clone rapido Aurora.

- Esecuzione del backtrack di un cluster

  Informazioni su come eseguire il backtrack di un cluster di database.

- Uso di Approfondimenti sulle prestazioni

  Informazioni su come monitorare e ottimizzare la tua istanza DB tramite Approfondimenti sulle prestazioni.
- Monitoraggio delle prestazioni con strumenti RDS

  Scopri come utilizzare AWS gli strumenti SQL per comprendere i problemi di prestazioni e identificare modi per migliorare le prestazioni del tuo database.
- Analisi delle prestazioni delle query

  Informazioni su come risolvere i problemi relativi alle prestazioni SQL utilizzando diversi strumenti.
- Repliche di lettura con dimensionamento automatico

  Informazioni su come funzionano le repliche di lettura con dimensionamento automatico.
- Test della tolleranza ai guasti

  Informazioni sulle funzionalità relative al disponibilità elevata e tolleranza ai guasti in Aurora MySQL.
- Database globale Aurora

  Informazioni su Database globale Amazon Aurora.
- Aurora Serverless v2

  Informazioni su Aurora Serverless v2.
- Utilizzo del machine learning

  Informazioni sul machine learning Aurora.

Utilizzo di questo servizio con un AWS SDK

AWS i kit di sviluppo software (SDKs) sono disponibili per molti linguaggi di programmazione più diffusi. Ogni SDK fornisce un'API, esempi di codice, e documentazione che facilitano agli sviluppatori la creazione di applicazioni nel loro linguaggio preferito.

Documentazione sugli SDK	Esempi di codice
AWS SDK per C++	AWS SDK per C++ esempi di codice

Documentazione sugli SDK	Esempi di codice
AWS CLI	AWS CLI esempi di codice
AWS SDK per Go	AWS SDK per Go esempi di codice
AWS SDK per Java	AWS SDK per Java esempi di codice
AWS SDK per JavaScript	AWS SDK per JavaScript esempi di codice
AWS SDK per Kotlin	AWS SDK per Kotlin esempi di codice
AWS SDK per .NET	AWS SDK per .NET esempi di codice
AWS SDK per PHP	AWS SDK per PHP esempi di codice
AWS Strumenti per PowerShell	AWS Strumenti per PowerShell esempi di codice
AWS SDK per Python (Boto3)	AWS SDK per Python (Boto3) esempi di codice
AWS SDK per Ruby	AWS SDK per Ruby esempi di codice
AWS SDK per Rust	AWS SDK per Rust esempi di codice
AWS SDK per SAP ABAP	AWS SDK per SAP ABAP esempi di codice
AWS SDK per Swift	AWS SDK per Swift esempi di codice

Per esempi specifici del servizio, consulta Esempi di codice per l'utilizzo di Aurora AWS SDKs.

> **Esempio di disponibilità**
>
> Non riesci a trovare quello che ti serve? Richiedi un esempio di codice utilizzando il link Provide feedback (Fornisci un feedback) nella parte inferiore di questa pagina.

Accesso programmatico ad Amazon Aurora

Amazon RDS offre i seguenti strumenti per gestire le risorse Amazon Aurora in modo programmatico.

AWS Command Line Interface (AWS CLI)

È possibile creare e gestire le risorse Amazon Aurora utilizzando AWS CLI nella shell a riga di comando. AWS CLI fornisce accesso diretto alle API per i servizi AWS, come Amazon RDS. Per la sintassi e gli esempi dei comandi per Amazon RDS, consulta rds nel Riferimento ai comandi della AWS CLI.

AWS CloudFormation

Con questo strumento AWS Infrastructure as Code (IaC), è possibile creare modelli che descrivono tutte le risorse Amazon Aurora desiderate, e AWS CloudFormation ne esegue automaticamente il provisioning e la configurazione. Per ulteriori informazioni, consulta the section called “Creazione di risorse con AWS CloudFormation”.

Software Development Kit (SDK) AWS

AWS fornisce SDK per molte tecnologie e linguaggi di programmazione popolari. Consentono di chiamare più facilmente i servizi AWS dalle applicazioni in quel linguaggio o tecnologia. Per ulteriori informazioni su questi SDK, consulta Strumenti per lo sviluppo e la gestione di applicazioni su AWS.

API Amazon RDS

Questa API è l’interfaccia a livello di protocollo per Amazon RDS. Quando si utilizza questa API, è necessario formattare correttamente ogni richiesta HTTPS e aggiungere una firma digitale valida a ogni richiesta. Per ulteriori informazioni, consulta Documentazione di riferimento dell'API Amazon RDS.

Console-to-Code

Con questo strumento, è possibile generare codice per le azioni eseguite nella console Amazon RDS e utilizzare tale codice in altri strumenti come AWS CloudFormation. Per ulteriori informazioni, consulta the section called “Console-to-Code”.

API dati in RDS

API dati in RDS (API dati) fornisce l’accesso programmatico ai dati nei database Amazon Aurora Serverless. Con l’API dati è possibile utilizzare un endpoint HTTP sicuro per eseguire le istruzioni

SQL senza gestire le connessioni. Per ulteriori informazioni, consulta Utilizzo dell'API dati di Amazon RDS.

Utilizzo di Console-to-Code per generare il codice per le azioni della console Amazon Aurora.

La console fornisce un percorso guidato per creare risorse e testare prototipi. Se vuoi creare le stesse risorse su larga scala, avrai bisogno del codice di automazione. Console-to-Code è una funzionalità di Amazon Q Developer che può essere utile per iniziare a usare il codice di automazione. Console-to-Code registra le operazioni della console, inclusi i valori predefiniti e i valori dei parametri che fornisci. Utilizza quindi l'IA generativa per suggerire il codice nella lingua e nel formato preferiti per le azioni che scegli. Poiché il flusso di lavoro della console assicura che i valori dei parametri specificati siano validi insieme, il codice generato utilizzando Console-to-Code ha valori di parametro compatibili. Puoi usare il codice come punto di partenza, personalizzandolo in seguito al fine di renderlo pronto per la produzione per il tuo caso d'uso specifico.

Ad esempio, con Console-to-Code, puoi registrare la creazione di un cluster di database Aurora e scegliere di generare codice in formato JSON AWS CloudFormation. Quindi, puoi copiare quel codice e personalizzarlo per utilizzarlo nel tuo modello AWS CloudFormation.

Al momento Console-to-Code può generare infrastructure-as-code (IaC) nelle seguenti lingue e formati:

- Java CDK
- Python CDK
- TypeScript CDK
- JSON CloudFormation
- YAML CloudFormation

Per ulteriori informazioni e istruzioni su come utilizzare Console-to-Code, consultare Automazione dei servizi AWS con Amazon Q Developer Console-to-Code nella Guida per l'utente Amazon Q Developer.

Configurazione di un cluster di database Amazon Aurora

Questa sezione illustra come configurare il cluster di database Aurora. Prima di creare un cluster di database Aurora, scegli la classe di istanze database che eseguirà il cluster di database. Scegli anche la regione AWS dove verrà eseguito il cluster DB. Quindi, crea il cluster di database. Se hai dei dati al di fuori di Aurora, puoi eseguirne la migrazione in un cluster di database Aurora.

Argomenti

- Creazione di un cluster database Amazon Aurora
- Creazione di risorse Amazon Aurora con AWS CloudFormation
- Connessione a un cluster database Amazon Aurora
- Gruppi di parametri per Amazon Aurora
- Migrazione di dati a un cluster di database Amazon Aurora
- Creazione di una cache Amazon ElastiCache utilizzando le impostazioni del cluster di database Aurora
- Migrazione automatica dei database EC2 ad Amazon Aurora utilizzando AWS Database Migration Service
- Tutorial: creazione di un cluster di database MySQL con un gruppo di parametri personalizzati

Creazione di un cluster database Amazon Aurora

Un cluster di database Amazon Aurora è costituito da un'istanza database, compatibile con MySQL o PostgreSQL, e un volume cluster che conserva i dati per il cluster di database, di cui viene eseguita la copia su tre zone di disponibilità some singolo volume virtuale. Per impostazione predefinita, un cluster di database Aurora contiene un'istanza database primaria che esegue letture e scritture e, facoltativamente, fino a 15 repliche di Aurora (istanze database di scrittura). Per ulteriori informazioni sui cluster di database Aurora, consulta Cluster database Amazon Aurora.

Aurora dispone di due tipi principali di cluster database:

- Aurora con provisioning: la classe di istanza database per le istanze di scrittura e lettura viene scelta in base al carico di lavoro previsto. Per ulteriori informazioni, consulta Classi di istanze database Amazon Aurora. Aurora con provisioning dispone di diverse opzioni, inclusi i database globali Aurora. Per ulteriori informazioni, consulta Utilizzo di Database globale Amazon Aurora.
- Aurora Serverless – Aurora Serverless v2 è una configurazione on demand di dimensionamento automatico per Aurora. La capacità viene regolata automaticamente in base alle necessità dall'applicazione. L'addebito viene calcolato in base alle risorse utilizzate dal cluster database. Questa automazione è particolarmente utile per ambienti con carichi di lavoro estremamente variabili e imprevedibili. Per ulteriori informazioni, consulta Uso di Aurora Serverless v2.

Di seguito, troverai informazioni su come creare un cluster di database Aurora. Per iniziare, consulta innanzitutto Prerequisiti per i cluster di database.

Per istruzioni sulla connessione al cluster database Aurora, consulta Connessione a un cluster database Amazon Aurora.

Indice

- Prerequisiti per i cluster di database
  - Configurazione della rete per il cluster database
    - Configurazione della connettività di rete automatica con un'istanza EC2
    - Configurazione manuale della rete
  - Prerequisiti aggiuntivi
- Creazione di un cluster di database
  - Creazione di un’istanza database (di scrittura) primaria
- Impostazioni per cluster di database Aurora

- Impostazioni che non si applicano ad Amazon Aurora per i cluster di database
- Impostazioni che non si applicano alle istanze database Amazon Aurora

Prerequisiti per i cluster di database

> **Important**
>
> Prima di poter creare un cluster di database Aurora, devi completare le attività descritte in Configurazione dell'ambiente per Amazon Aurora.

Di seguito sono indicati i prerequisiti da completare prima di creare un cluster database.

Argomenti

- Configurazione della rete per il cluster database
- Prerequisiti aggiuntivi

Configurazione della rete per il cluster database

Un cluster database Amazon Aurora può essere creato solo in un cloud privato virtuale (VPC) basato sul servizio Amazon VPC, in una Regione AWS che disponga di almeno due zone di disponibilità. Il gruppo di sottoreti di database scelto per il cluster di database deve coprire almeno due zone di disponibilità. Questa configurazione assicura che nel cluster di database sia sempre disponibile almeno un'istanza database per il failover, nell'improbabile caso di errore in una zona.

Se necessario, puoi configurare la connettività tra il nuovo cluster database e un'istanza EC2 nello stesso VPC durante la creazione del cluster database. Per connetterti al cluster database da risorse diverse dalle istanze EC2 nello stesso VPC, puoi configurare le connessioni di rete manualmente.

Argomenti

- Configurazione della connettività di rete automatica con un'istanza EC2
- Configurazione manuale della rete

Configurazione della connettività di rete automatica con un'istanza EC2

Durante la creazione di un cluster database Aurora, puoi usare la Console di gestione AWS per configurare la connettività tra un'istanza Amazon EC2 e il nuovo cluster database. In questo caso,

RDS configura automaticamente il VPC e le impostazioni di rete. Il cluster di database viene creato nello stesso VPC dell'istanza EC2, per consentire all'istanza EC2 di accedere al cluster di database.

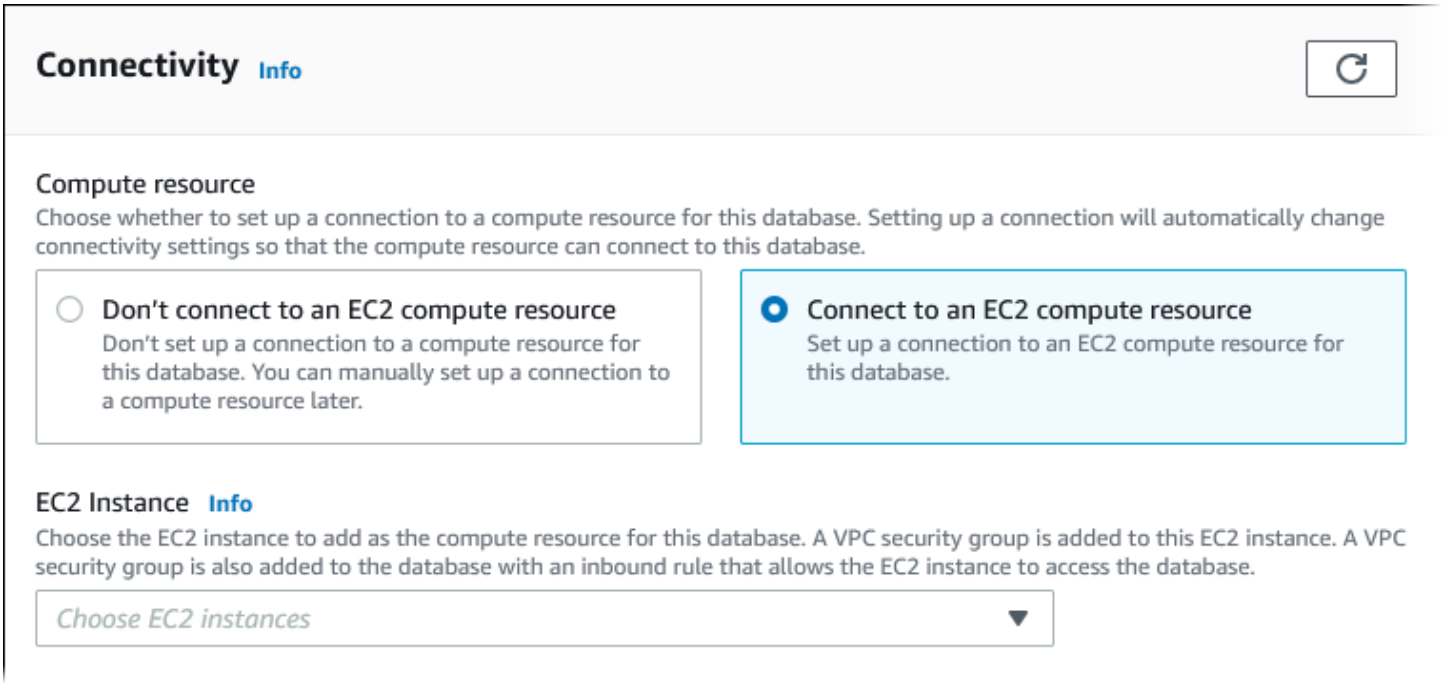
Di seguito sono riportati i requisiti per connettere un'istanza EC2 al cluster di database:

- L'istanza EC2 deve esistere nella Regione AWS prima di creare il cluster di database.

  Se nella Regione AWS non esistono istanze EC2, la console fornisce un collegamento per crearne una.
- Attualmente, il cluster database non può essere un cluster database Aurora Serverless o parte di un database globale Aurora.
- L'utente che sta creando l'istanza database deve disporre delle autorizzazioni per eseguire le seguenti operazioni:
  - `ec2:AssociateRouteTable`
  - `ec2:AuthorizeSecurityGroupEgress`
  - `ec2:AuthorizeSecurityGroupIngress`
  - `ec2:CreateRouteTable`
  - `ec2:CreateSubnet`
  - `ec2:CreateSecurityGroup`
  - `ec2:DescribeInstances`
  - `ec2:DescribeNetworkInterfaces`
  - `ec2:DescribeRouteTables`
  - `ec2:DescribeSecurityGroups`
  - `ec2:DescribeSubnets`
  - `ec2:ModifyNetworkInterfaceAttribute`
  - `ec2:RevokeSecurityGroupEgress`

L'utilizzo di questa opzione crea un cluster database privato. Il cluster di database utilizza un gruppo di sottoreti DB con solo sottoreti private per limitare l'accesso alle risorse all'interno del VPC.

Per connettere un'istanza EC2 al cluster di database, scegli Connect to an EC2 compute resource (Connetti a una risorsa di calcolo EC2) nella sezione Connectivity (Connettività) della pagina Create database (Crea database).



Se scegli Connect to an EC2 compute resource (Connetti a una risorsa di calcolo EC2), RDS imposta automaticamente le seguenti opzioni. Queste impostazioni non possono essere modificate a meno che non si scelga di non configurare la connettività con un'istanza EC2 selezionando Don't connect to an EC2 compute resource (Non connetterti a una risorsa di calcolo EC2).

Opzione Console	Impostazione automatica
Tipo di rete	RDS imposta il tipo di rete su IPv4. Attualmente, la modalità dual-stack non è supportata quando si configura una connessio ne tra un'istanza EC2 e il cluster database.
Virtual Private Cloud (VPC)	RDS imposta il VPC su quello associato all'istanza EC2.
DB subnet group (Gruppo di sottoreti DB)	RDS richiede un gruppo di sottoreti database con una sottorete privata nella stessa zona di disponibilità dell'istanza EC2. Se esiste un gruppo di sottoreti database che soddisfa questo requisito, RDS lo utilizza. Per impostazione predefinita, questa opzione è impostata su Automatic setup (Configurazione automatica). Quando si sceglie Automatic setup (Configurazione automatica) e non esiste un gruppo di sottoreti database che soddisfi questo

Opzione Console	Impostazione automatica
	requisito, viene eseguita la seguente procedura. RDS utilizza tre sottoreti private disponibili in tre zone di disponibilità di cui una è la stessa dell'istanza EC2. Se una sottorete privata non è disponibile in una zona di disponibilità, RDS crea una sottorete privata nella zona di disponibilità. Quindi RDS crea il gruppo di sottoreti database. Quando è disponibile una sottorete privata, RDS utilizza la tabella di instradamento associata alla sottorete e aggiunge tutte le sottoreti create a questa tabella di instradamento. Quando una sottorete privata non è disponibile, RDS crea una tabella di instradamento senza accesso al gateway Internet e aggiunge le sottoreti create alla tabella di instradamento. RDS consente inoltre di utilizzare i gruppi di sottoreti database esistenti. Seleziona Choose existing (Scegli esistente) se desideri utilizzare un gruppo di sottoreti database esistente.
Accesso pubblico	RDS sceglie No in modo che il cluster di database non sia accessibile pubblicamente. Per motivi di sicurezza, come best practice si consiglia di mantenere il database privato e accertarsi che non sia accessibi le da Internet.

Opzione Console	Impostazione automatica
VPC security group (firewall) (Gruppo di sicurezza VPC (firewall)	RDS crea un nuovo gruppo di sicurezza associato al cluster di database. Il gruppo di sicurezza è denominato `rds-ec2-`*n*, dove *n* è un numero. Questo gruppo di sicurezza include una regola in entrata con il gruppo di sicurezza VPC EC2 (firewall) come origine. Questo gruppo di sicurezza associato al cluster di database consente all'istanza EC2 di accedere al cluster. RDS crea, inoltre, un nuovo gruppo di sicurezza associato all'istanza EC2. Il gruppo di sicurezza è denominato `ec2-rds-`*n*, dove *n* è un numero. Questo gruppo di sicurezza include una regola in uscita con il gruppo di sicurezza VPC del cluster di database come origine. Questo gruppo di sicurezza consente all'istanza EC2 di inviare traffico al cluster di database. Puoi aggiungere un nuovo gruppo di sicurezza aggiuntivo scegliendo Create nuovo (Crea nuovo) e digitando il nome del nuovo gruppo di sicurezza. Puoi aggiungere gruppi di sicurezza esistenti scegliendo Choose existing (Scegli esistente) e selezionando i gruppi di sicurezza da aggiungere.

Opzione Console	Impostazione automatica
Zona di disponibilità	Se in Availability & durability (Durabilità e disponibilità) non si crea alcuna replica di Aurora durante la creazione del cluster database (Implementazione Single-AZ), RDS sceglie la zona di disponibilità dell'istanza EC2. Quando si crea una replica Aurora durante la creazione del cluster database (implementazione multi-AZ), RDS sceglie la zona di disponibilità dell'istanza EC2 per un'istanza database nel cluster database. RDS sceglie casualmente una zona di disponibilità diversa per l'altra istanza database nel cluster database. L'istanza database primaria o la replica di Aurora vengono create nella stessa zona di disponibilità dell'istanza EC2. Esiste la possibilità di costi tra zone di disponibilità se l'istanza database primaria e l'istanza EC2 si trovano in zone di disponibilità diverse.

Per ulteriori informazioni su queste impostazioni, consultare Impostazioni per cluster di database Aurora.

Se dopo la creazione del cluster database le impostazioni vengono modificate, le modifiche potrebbero influire sulla connessione tra l'istanza EC2 e il cluster di database.

Configurazione manuale della rete

Per connetterti al cluster database da risorse diverse dalle istanze EC2 nello stesso VPC, puoi configurare le connessioni di rete manualmente. Se utilizzi la Console di gestione AWS per creare il cluster database, puoi configurare Amazon RDS per creare automaticamente un VPC. Altrimenti, puoi utilizzare un VPC esistente o crearne uno nuovo per il tuo cluster di database Aurora. Qualunque sia l'approccio scelto, il VPC deve disporre di almeno una sottorete in ciascuna di almeno due zone di disponibilità affinché tu possa utilizzarlo con un cluster di database Amazon Aurora.

Per impostazione predefinita, Amazon RDS crea automaticamente l'istanza database primaria e la replica di Aurora nelle zone di disponibilità. Per scegliere una zona di disponibilità specifica, l'impostazione di implementazione Multi-AZ Availability & durability (Disponibilità e durata) in Don't create an Aurora Replica (Non creare una replica di Aurora) deve essere modificata. Questo espone un'impostazione Availability Zone (Zona di disponibilità) che consente di scegliere tra le zone di

disponibilità nel VPC. Tuttavia, è consigliabile mantenere le impostazioni predefinite e consentire ad Amazon RDS di creare un'implementazione Multi-AZ e scegliere automaticamente le zone di disponibilità. In questo modo, il cluster di database Aurora viene creato con funzionalità di failover veloce e alta disponibilità che sono due dei principali vantaggi di Aurora.

Se non si dispone di un VPC predefinito o uno non è stato creato, è possibile che un VPC venga creato automaticamente da Amazon RDS durante la creazione di un cluster database mediante la console. In caso contrario, devi completare le attività seguenti:

- Creare un VPC con almeno una sottorete in ciascuna di almeno due zone di disponibilità nella Regione AWS in cui desideri implementare il cluster database. Per ulteriori informazioni, consultare Uso di un cluster database in un VPC e Tutorial: Creazione di un Amazon VPC da utilizzare con un cluster database (solo IPv4).
- Specificare un gruppo di sicurezza VPC che autorizzi le connessioni al proprio cluster di database . Per ulteriori informazioni, consultare Fornitura dell'accesso al cluster di database nel VPC creando un gruppo di sicurezza e Controllo dell'accesso con i gruppi di sicurezza.
- Specifica di un gruppo di sottoreti del database RDS che definisca almeno due sottoreti nel VPC che possono essere utilizzate dal cluster di database . Per ulteriori informazioni, consulta Utilizzo di gruppi di sottoreti database.

Per informazioni sui VPC, consulta VPC Amazon e Amazon Aurora. Per un tutorial di configurazione della rete per un cluster database privato, consulta Tutorial: Creazione di un Amazon VPC da utilizzare con un cluster database (solo IPv4).

Se desideri connetterti a una risorsa che non si trova nello stesso VPC del cluster Aurora DB, consulta gli scenari appropriati descritti in Scenari per accedere a un cluster database in un VPC.

Prerequisiti aggiuntivi

Prima di creare il cluster database, considera i seguenti prerequisiti aggiuntivi:

- Se ti stai collegando a AWS tramite le credenziali AWS Identity and Access Management (IAM), il tuo account AWS dovrà disporre di policy IAM che concedono le autorizzazioni necessarie per eseguire le operazioni Amazon RDS. Per ulteriori informazioni, consulta Gestione accessi e identità per Amazon Aurora.

Se utilizzi IAM per accedere alla console Amazon RDS, devi innanzitutto accedere alla Console di gestione AWS con le tue credenziali utente. Quindi, passa alla console Amazon RDS all'indirizzo https://console.aws.amazon.com/rds/.

- Se desideri personalizzare i parametri di configurazione per il cluster di database, devi specificare un gruppo di parametri del cluster di database e un gruppo di parametri database con le impostazioni dei parametri richieste. Per informazioni sulla creazione o la modifica di un gruppo di parametri del cluster di database o un gruppo di parametri database, consulta Gruppi di parametri per Amazon Aurora.
- Determina il numero di porta TCP/IP da specificare per il cluster di database. I firewall presso alcune aziende bloccano le connessioni alle porte predefinite (3306 per MySQL, 5432 per PostgreSQL) per Aurora. Se il firewall della tua azienda blocca la porta predefinita, scegli un'altra porta per il cluster di database. Tutte le istanze in un cluster di database utilizzano la stessa porta.
- Se la versione principale del motore per il database ha raggiunto la data di fine del supporto standard per RDS, è necessario utilizzare l'opzione CLI con supporto esteso o il parametro API RDS. Per ulteriori informazioni, consulta Supporto esteso per RDS in Impostazioni per cluster di database Aurora.

Creazione di un cluster di database

Puoi creare un cluster di database Aurora utilizzando la Console di gestione AWS, la AWS CLI o l'API RDS.

Console

Puoi creare un cluster di database utilizzando la Console di gestione AWS con l'opzione Creazione rapida abilitata o non abilitata. Con l'opzione Easy create (Creazione rapida) attivata, specifichi solo il tipo di motore del database, la dimensione dell'istanza di database e l'identificatore dell'istanza di database. Easy create (Creazione rapida) utilizza l'impostazione predefinita per altre opzioni di configurazione. Con l'opzione Easy create (Creazione rapida) disattivata, specifichi più opzioni di configurazione quando crei un database, comprese quelle relative a disponibilità, sicurezza, backup e manutenzione.

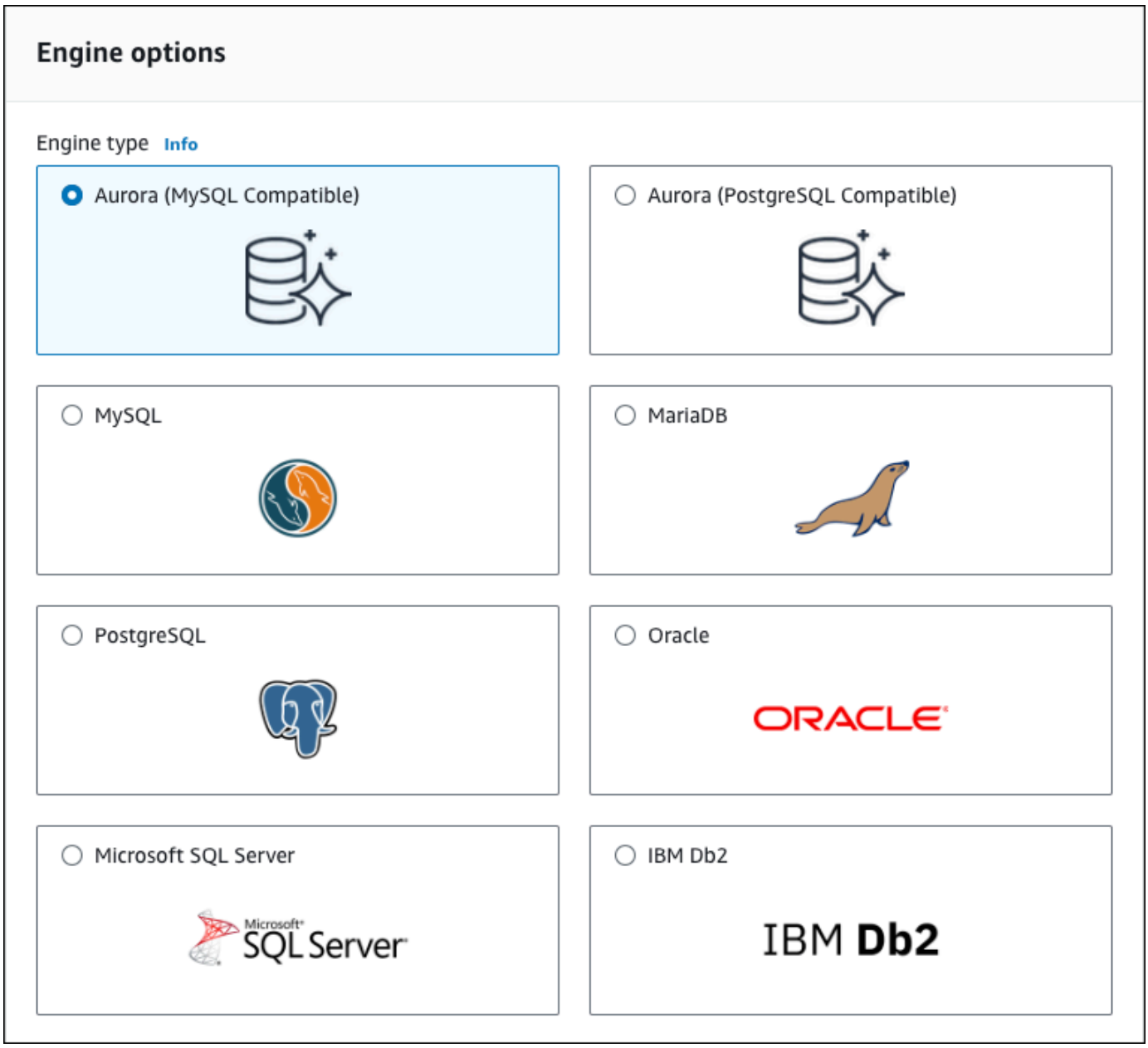
> **Note**
>
> Per questo esempio, Standard create (Creazione standard) è attivato e Easy create (Creazione rapida) non è abilitata. Per informazioni sulla creazione di un cluster di database con l'opzione Creazione rapida abilitata, consulta Nozioni di base su Amazon Aurora.

Per creare un cluster di database Aurora tramite la console

1. Accedi alla Console di gestione AWS e apri la console Amazon RDS all'indirizzo https://console.aws.amazon.com/rds/.
2. Nell'angolo in alto a destra della Console di gestione AWS, scegli la regione AWS in cui si desidera creare il cluster DB.

   Aurora non è disponibile in tutte le regioni AWS. Per un elenco delle regioni AWS in cui è disponibile Aurora, consulta Disponibilità nelle regioni.
3. Nel riquadro di navigazione, scegliere Databases (Database).
4. Scegliere Create database (Crea database).
5. Per Scegli un metodo di creazione del database seleziona Creazione standard.
6. Per Tipo di motore scegli una delle seguenti opzioni:

   - Aurora (compatibile con MySQL)
   - Aurora (compatibile con PostgreSQL)



7. Scegli la Versione del motore.

   Per ulteriori informazioni, consulta Versioni di Amazon Aurora. Puoi utilizzare i filtri per scegliere versioni compatibili con le funzionalità che desideri, ad esempio Aurora Serverless v2. Per ulteriori informazioni, consulta Uso di Aurora Serverless v2.

8. In Templates (Modelli), selezionare il modello che corrisponde al proprio caso d'uso.

9. Per inserire la password principale, procedere come segue:

   a. Nella sezione Impostazioni espandi Impostazioni delle credenziali.

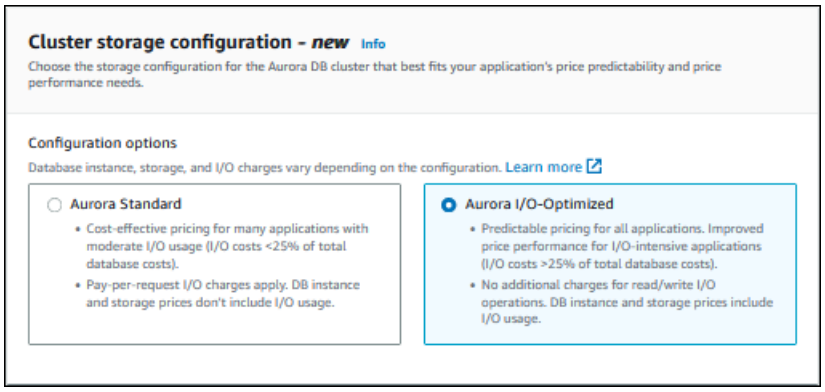
b. Deselezionare la casella di controllo Auto generate a password (Genera automaticamente una password).

c. (Opzionale) Modificare il valore Master username (Nome utente principale) e inserire la stessa password in Master password (Password principale) e Confirm password (Conferma password).

Per impostazione predefinita, la nuova istanza database utilizza una password generata automaticamente per l'utente principale.

10. Nella sezione Connettività in Gruppo di sicurezza VPC (firewall), se selezioni Crea nuovo, viene creato un gruppo di sicurezza VPC con una regola in entrata che consente all'indirizzo IP del computer locale di accedere al database.

11. In Configurazione archiviazione cluster, scegliere Aurora I/O-Optimized o Aurora Standard. Per ulteriori informazioni, consulta Configurazioni dell'archiviazione per i cluster database Amazon Aurora.



12. (Facoltativo) Configura una connessione a una risorsa di calcolo per questo cluster di database.

Puoi configurare la connettività tra un'istanza Amazon EC2 e il nuovo cluster di database durante la creazione del cluster di database. Per ulteriori informazioni, consulta Configurazione della connettività di rete automatica con un'istanza EC2.

13. Per le restanti sezioni, specifica le impostazioni del cluster di database. Per informazioni su ciascuna impostazione, consulta Impostazioni per cluster di database Aurora.

14. Scegliere Create database (Crea database).

Se scegli di utilizzare una password generata in modo automatico, il pulsante View credential details (Vedi dettagli delle credenziali) appare nella paginaDatabases.

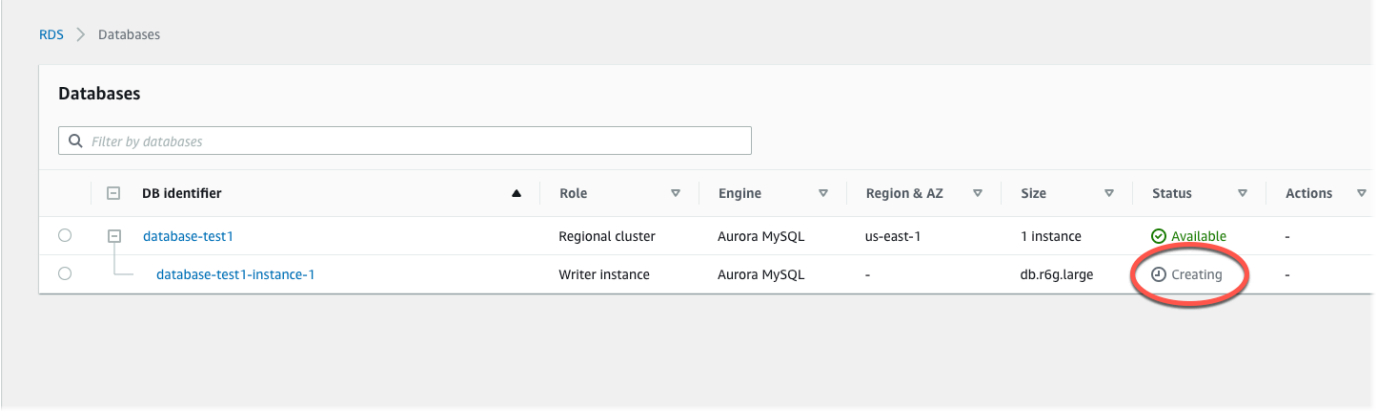
Per vedere nome utente e password per il cluster di database, seleziona View credential details (Vedi dettagli delle credenziali).

Per connetterti all'istanza database come utente principale, utilizza il nome utente e la password visualizzati.

> **Important**
>
> Non potrai visualizzare di nuovo la password dell'utente principale. Se non la registri, potresti doverla modificare. Se devi modificare la password dell'utente principale dopo che l'istanza database è disponibile, puoi modificare l'istanza database per eseguire tale operazione. Per ulteriori informazioni sulla modifica di un'istanza database , consulta Modifica di un cluster database Amazon Aurora.

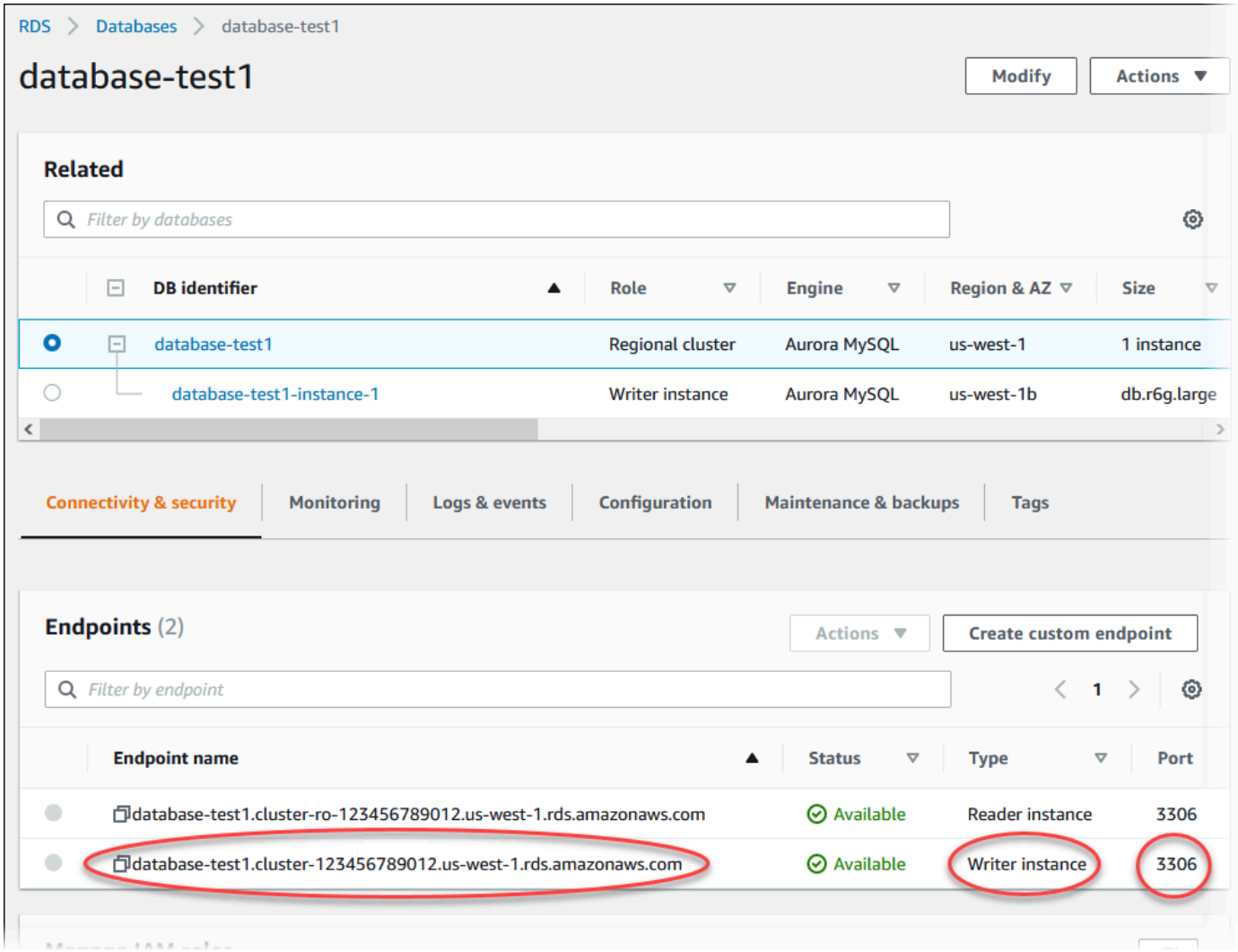
15. Per Databases, seleziona il nome del nuovo cluster di database Aurora.

Nella console RDS vengono visualizzati i dettagli per il nuovo cluster di database. Lo stato del cluster database e della sua istanza database sarà creating (in creazione) fino a quando sarà pronto per essere impiegato.



Quando lo stato cambia in available (disponibile) è possibile connettersi al cluster di database. A seconda della classe di istanza database e della quantità di storage, prima che il nuovo cluster di database sia disponibile possono trascorrere fino a 20 minuti.

Per visualizzare il cluster appena creato, scegli Databases (Database) dal riquadro di navigazione nella console Amazon RDS. Quindi scegli il cluster di database per mostrare i dettagli del cluster di database. Per ulteriori informazioni, consulta Visualizzazione di un cluster di database Amazon Aurora.



Nella scheda Connectivity & security (Connettività e sicurezza), prendi nota della porta e dell'endpoint dell'istanza database di scrittura. Utilizzare l'endpoint e la porta del cluster nelle stringhe di connessione JDBC e ODBC per un'applicazione che esegue operazioni di scrittura o lettura.

AWS CLI

> **Note**
>
> Per poter creare un cluster di database Aurora tramite la AWS CLI, è necessario che vengano soddisfatti i prerequisiti necessari, come la creazione di un VPC e di un gruppo di sottoreti del database RDS. Per ulteriori informazioni, consulta Prerequisiti per i cluster di database.

Puoi usare la AWS CLI per creare un cluster database Aurora MySQL o Aurora PostgreSQL.

Come creare un cluster database Aurora MySQL tramite la AWS CLI

Quando si crea un'istanza database o un cluster di database Aurora MySQL compatibile con 8.0 o 5.7, è necessario specificare `aurora-mysql` per l'opzione `--engine`.

Completa questa procedura:

1. Identifica il gruppo di sottoreti del database e l'ID del gruppo di sicurezza VPC per il nuovo cluster database, quindi emetti il comando create-db-cluster della AWS CLI per creare il cluster database Aurora MySQL.

   Il comando seguente crea ad esempio un nuovo cluster di database –compatibile con MySQL 8.0 denominato `sample-cluster`. Il cluster utilizza la versione del motore predefinita e il tipo di archiviazione Aurora I/O-Optimized.

   Per Linux, macOS o Unix:

   ```
   aws rds create-db-cluster --db-cluster-identifier sample-cluster \
       --engine aurora-mysql --engine-version 8.0 \
       --storage-type aurora-iopt1 \
       --master-username user-name --manage-master-user-password \
       --db-subnet-group-name mysubnetgroup --vpc-security-group-ids sg-c7e5b0d2
   ```

   Per Windows:

   ```
   aws rds create-db-cluster --db-cluster-identifier sample-cluster ^
       --engine aurora-mysql --engine-version 8.0 ^
       --storage-type aurora-iopt1 ^
       --master-username user-name --manage-master-user-password ^
       --db-subnet-group-name mysubnetgroup --vpc-security-group-ids sg-c7e5b0d2
   ```

   Il comando seguente crea un nuovo cluster di database –compatibile con MySQL 5.7 denominato `sample-cluster`. Il cluster utilizza la versione del motore predefinita e il tipo di archiviazione Aurora Standard.

   Per Linux, macOS o Unix:

   ```
   aws rds create-db-cluster --db-cluster-identifier sample-cluster  \
       --engine aurora-mysql --engine-version 5.7 \
   ```

```
    --storage-type aurora \
    --master-username user-name --manage-master-user-password \
    --db-subnet-group-name mysubnetgroup --vpc-security-group-ids sg-c7e5b0d2
```

Per Windows:

```
aws rds create-db-cluster --db-cluster-identifier sample-cluster sample-cluster  ^
    --engine aurora-mysql --engine-version 5.7 ^
    --storage-type aurora ^
    --master-username user-name --manage-master-user-password ^
    --db-subnet-group-name mysubnetgroup --vpc-security-group-ids sg-c7e5b0d2
```

2. Se si utilizza la console per creare un cluster di database, Amazon RDS crea automaticamente l'istanza principale (writer) per il cluster di database. Se si utilizza AWS CLI per creare un cluster di database, è necessario creare in modo esplicito l'istanza principale per il cluster di database. L'istanza primaria è la prima istanza creata in un cluster di database. Se non crei l'istanza database primaria, lo stato degli endpoint del cluster database rimane impostato su `Creating`.

   Chiamare il comando AWS CLI create-db-instance per creare l'istanza principale per il cluster di database. Includere il nome del cluster di database come valore dell'opzione `--db-cluster-identifier`.

   > **Note**
   >
   > Non è possibile impostare l'opzione `--storage-type` per le istanze database. È possibile impostarla solo per i cluster database.

   Il comando seguente crea ad esempio una nuova istanza database –compatibile con MySQL 5.7 o con MySQL 8.0 denominata `sample-instance`.

   Per Linux, macOS o Unix:

   ```
   aws rds create-db-instance --db-instance-identifier sample-instance \
        --db-cluster-identifier sample-cluster --engine aurora-mysql --db-instance-
   class db.r5.large
   ```

   Per Windows:

```
aws rds create-db-instance --db-instance-identifier sample-instance ^
     --db-cluster-identifier sample-cluster --engine aurora-mysql --db-instance-
class db.r5.large
```

Come creare un cluster database Aurora PostgreSQL tramite la AWS CLI

1. Identifica il gruppo di sottoreti del database e l'ID del gruppo di sicurezza VPC per il nuovo cluster database, quindi emetti il comando create-db-cluster della AWS CLI per creare il cluster database Aurora PostgreSQL.

   Il comando seguente crea ad esempio un nuovo cluster di database denominato `sample-cluster`. Il cluster utilizza la versione del motore predefinita e il tipo di archiviazione Aurora I/O-Optimized.

   Per Linux, macOS o Unix:

   ```
   aws rds create-db-cluster --db-cluster-identifier sample-cluster \
       --engine aurora-postgresql \
       --storage-type aurora-iopt1 \
       --master-username user-name --manage-master-user-password \
       --db-subnet-group-name mysubnetgroup --vpc-security-group-ids sg-c7e5b0d2
   ```

   Per Windows:

   ```
   aws rds create-db-cluster --db-cluster-identifier sample-cluster ^
       --engine aurora-postgresql ^
       --storage-type aurora-iopt1 ^
       --master-username user-name --manage-master-user-password ^
       --db-subnet-group-name mysubnetgroup --vpc-security-group-ids sg-c7e5b0d2
   ```

2. Se si utilizza la console per creare un cluster di database, Amazon RDS crea automaticamente l'istanza principale (writer) per il cluster di database. Se si utilizza AWS CLI per creare un cluster di database, è necessario creare in modo esplicito l'istanza principale per il cluster di database. L'istanza primaria è la prima istanza creata in un cluster di database. Se non crei l'istanza database primaria, lo stato degli endpoint del cluster database rimane impostato su `Creating`.

Chiamare il comando AWS CLI create-db-instance per creare l'istanza principale per il cluster di database. Includere il nome del cluster di database come valore dell'opzione `--db-cluster-identifier`.

Per Linux, macOS o Unix:

```
aws rds create-db-instance --db-instance-identifier sample-instance \
     --db-cluster-identifier sample-cluster --engine aurora-postgresql --db-
instance-class db.r5.large
```

Per Windows:

```
aws rds create-db-instance --db-instance-identifier sample-instance ^
     --db-cluster-identifier sample-cluster --engine aurora-postgresql --db-
instance-class db.r5.large
```

In questi esempi è specificata l'opzione `--manage-master-user-password` per generare la password dell'utente master e gestirla in Secrets Manager. Per ulteriori informazioni, consulta Gestione delle password con Amazon Aurora e Gestione dei segreti AWS. In alternativa, puoi utilizzare l'opzione `--master-password` per specificare e gestire personalmente la password.

API RDS

> **Note**
>
> Per poter creare un cluster di database Aurora tramite la AWS CLI, è necessario che vengano soddisfatti i prerequisiti necessari, come la creazione di un VPC e di un gruppo di sottoreti del database RDS. Per ulteriori informazioni, consulta Prerequisiti per i cluster di database.

Identificare il gruppo di sottoreti del database e l'ID del gruppo di sicurezza VPC per il nuovo cluster di database, quindi chiamare l'operazione CreateDBCluster per creare il cluster di database.

Quando si crea un'istanza database o un cluster database Aurora MySQL versione 2 o 3, specificare `aurora-mysql` per il parametro `Engine`.

Quando si crea un'istanza database o un cluster di database Aurora PostgreSQL, specificare `aurora-postgresql` per il parametro `Engine`.

Se si utilizza la console per creare un cluster di database, Amazon RDS crea automaticamente l'istanza principale (writer) per il cluster di database. Se si utilizza l'API RDS per creare un cluster database, è necessario creare in modo esplicito l'istanza principale per il cluster database utilizzando CreateDBInstance. L'istanza primaria è la prima istanza creata in un cluster di database. Se non crei l'istanza database primaria, lo stato degli endpoint del cluster database rimane impostato su `Creating`.

Creazione di un'istanza database (di scrittura) primaria

Se si utilizza la Console di gestione AWS per creare un cluster di database, Amazon RDS crea automaticamente l'istanza primaria (di scrittura) per il cluster di database. Se si utilizza l'interfaccia AWS CLI o l'API RDS per creare un cluster di database, è necessario creare in modo esplicito l'istanza primaria per il cluster di database. L'istanza primaria è la prima istanza creata in un cluster di database. Se non crei l'istanza database primaria, lo stato degli endpoint del cluster database rimane impostato su `Creating`.

Per ulteriori informazioni, consulta Creazione di un cluster di database.

> **Note**
>
> Se hai un cluster di database senza un'istanza database di scrittura, denominata anche cluster headless, non puoi utilizzare la console per creare un'istanza di scrittura. È necessario utilizzare l'interfaccia AWS CLI o l'API RDS.

L'esempio seguente utilizza il comando create-db-instance dell'interfaccia AWS CLI per creare un'istanza di scrittura per un cluster di database Aurora PostgreSQL denominato `headless-test`.

```
aws rds create-db-instance \
    --db-instance-identifier no-longer-headless \
    --db-cluster-identifier headless-test \
    --engine aurora-postgresql \
    --db-instance-class db.t4g.medium
```

Impostazioni per cluster di database Aurora

La tabella seguente contiene informazioni dettagliate sulle impostazioni disponibili quando crei un cluster di database Aurora.

Impostazione della console	Descrizione impostazione	Opzione CLI e parametro API di RDS
Auto minor version upgrade (Aggiornamento automatico della versione secondaria)	Selezionare Enable auto minor version upgrade (Abilita aggiornamenti automatici versioni secondarie) affinché il cluster database Aurora riceva automaticamente gli aggiornamenti secondari preferiti della versione nel motore di database non appena diventano disponibili. L'impostazione Auto Minor Version Upgrade (Aggiornamento automatico minore della versione) si applica ai cluster di database Aurora PostgreSQL e Aurora MySQL. Per ulteriori informazioni sugli aggiornamenti del motore per Aurora PostgreSQL, consultare Aggiornamenti del motore di database Amazon Aurora PostgreSQL per 02/06/2020. Per ulteriori informazioni sugli aggiornamenti del motore per Aurora MySQL, consultare Aggiornamenti del motore del	Imposta questo valore per ogni istanza database nel cluster Aurora. Se un'istanza database del cluster ha questa impostazione disattivata, il cluster non viene aggiornato automaticamente. Tramite AWS CLI, esegui `create-db-instance` e imposta l'opzione `--auto-minor-version-upgrade|--no-auto-minor-version-upgrade`. Tramite l'API RDS, chiama `CreateDBInstance` e imposta il parametro `AutoMinorVersionUpgrade`.

Impostazione della console	Descrizione impostazione	Opzione CLI e parametro API di RDS
	database per Amazon Aurora MySQL.	
AWS KMS key	Disponibile solo se Encryption (Crittografia) è impostato su Enable encryption (Abilita crittografia). Scegliere la AWS KMS key da utilizzare per crittografare questo cluster DB. Per ulteriori informazioni, consulta Crittografia delle risorse Amazon Aurora.	Tramite AWS CLI, esegui `create-db-cluster` e imposta l'opzione `--kms-key-id`. Tramite l'API RDS, chiama `CreateDBCluster` e imposta il parametro `KmsKeyId`.
Backtrack	Si applica solo ad Aurora MySQL. Selezionare Enable Backtrack (Abilita backtrack) o Disable Backtrack (Disabilita backtrack) per abilitare o disabilitare il backtrack, rispettivamente. Utilizzando il backtrack, sarà possibile riportare il cluster di database a un momento specifico, senza creare un nuovo cluster di database. Questa opzione è disabilitata per impostazione predefinita. Se si abilita il backtrack, sarà necessario anche specificare l'intervallo di tempo per il quale si desidera eseguire il backtrack del cluster di database (finestra target di backtrack). Per ulteriori informazioni, consulta Backtrack di un cluster database Aurora.	Tramite AWS CLI, esegui `create-db-cluster` e imposta l'opzione `--backtrack-window`. Tramite l'API RDS, chiama `CreateDBCluster` e imposta il parametro `BacktrackWindow`.

Impostazione della console	Descrizione impostazione	Opzione CLI e parametro API di RDS
Autorità di certificazione	L'autorità di certificazione (CA) per il certificato del server utilizzato dalle istanze database nel cluster database. Per ulteriori informazioni, consulta Utilizzo SSL/TLS per crittografare una connessione a un'.	Tramite AWS CLI, esegui `create-db-instance` e imposta l'opzione `--ca-certificate-identifier`. Tramite l'API RDS, chiama `CreateDBInstance` e imposta il parametro `CACertificateIdentifier`.
Configurazione dell'archiviazione del cluster	Tipo di archiviazione per il cluster database: Aurora I/O-Optimized o Aurora Standard. Per ulteriori informazioni, consulta Configurazioni dell'archiviazione per i cluster database Amazon Aurora.	Tramite AWS CLI, esegui `create-db-cluster` e imposta l'opzione `--storage-type`. Tramite l'API RDS, chiama `CreateDBCluster` e imposta il parametro `StorageType`.
Copy tags to snapshots (Copia tag in snapshot)	Selezionare questa opzione per copiare i tag dell'istanza database in una snapshot database quando si crea una snapshot. Per ulteriori informazioni, consulta Applicazione di tag alle risorse Amazon Aurora e Amazon RDS.	Tramite AWS CLI, esegui `create-db-cluster` e imposta l'opzione `--copy-tags-to-snapshot | --no-copy-tags-to-snapshot`. Tramite l'API RDS, chiama `CreateDBCluster` e imposta il parametro `CopyTagsToSnapshot`.

Impostazione della console	Descrizione impostazione	Opzione CLI e parametro API di RDS
Database authentication (Autenticazione del database)	L'autenticazione del database da utilizzare. Per MySQL: • Scegliere Password authentication (Autenticazione tramite password) per autenticare gli utenti di database solo con le password del database. • Seleziona Autenticazione database tramite password e IAM per autenticare gli utenti del database con password del database e credenziali utente tramite utenti e ruoli IAM. Per ulteriori informazioni, consulta Autenticazione del database IAM . Per PostgreSQL: • Scegli IAM database authentication (Autenticazione database IAM) per autenticare gli utenti del database con password del database e credenziali utente tramite utenti e ruoli. Per ulteriori informazioni, consulta Autenticazione del database IAM . • Seleziona Autenticazione Kerberos per autenticare le password del database e le credenziali utente utilizzando	Per utilizzare l'autenticazione database IAM con AWS CLI, esegui `create-db-cluster` e imposta l'opzione `--enable-iam-database-authentication | --no-enable-iam-database-authentication` . Per utilizzare l'autenticazione database IAM con l'API RDS, chiama `CreateDBCluster` e imposta il parametro `EnableIAMDatabaseAuthentication` . Per utilizzare l'autenticazione Kerberos con AWS CLI, esegui `create-db-cluster` e imposta le opzioni `--domain` e `--domain-iam-role-name` . Per utilizzare l'autenticazione Kerberos con l'API RDS, chiama `CreateDBCluster` e imposta i parametri `Domain` e `DomainIAMRoleName` .

Impostazione della console	Descrizione impostazione	Opzione CLI e parametro API di RDS
	l'autenticazione Kerberos. Per ulteriori informazioni, consulta Utilizzo di Autenticazione Kerberos con Aurora PostgreSQL.	
Database port (Porta del database)	Specifica la porta per applicazioni e utilità da utilizzare per accedere al database. Per impostazione predefinita, i cluster di database Aurora MySQL sono impostati sulla porta MySQL predefinita 3306 e i cluster di database Aurora PostgreSQL sulla porta PostgreSQL 5432. I firewall presso alcune aziende bloccano le connessioni di tali porte predefinite. Se il firewall dell'azienda blocca la porta predefinita, scegliere un'altra porta per il nuovo cluster di database.	Tramite AWS CLI, esegui `create-db-cluster` e imposta l'opzione `--port`. Tramite l'API RDS, chiama `CreateDBCluster` e imposta il parametro `Port`.

Impostazione della console	Descrizione impostazione	Opzione CLI e parametro API di RDS
DB Cluster Identifier (Identificatore cluster DB)	Immettere un nome per il cluster database che sia univoco per l'account nella regione AWS selezionata. Questo identific atore viene utilizzato nell'indirizzo dell'endpoint del cluster per il cluster database. Per informazioni sull'endpoint del cluster, consulta Connessioni degli endpoint Amazon Aurora. L'identificatore del cluster DB prevede i seguenti vincoli: • Deve contenere da 1 a 63 caratteri alfanumerici o trattini. • Il primo carattere deve essere una lettera. • Non può terminare con un trattino o contenere due trattini consecutivi. • Deve essere univoco per tutti i cluster DB per ogni account AWS e in ogni regione AWS.	Tramite AWS CLI, esegui `create-db-cluster` e imposta l'opzione `--db-cluster-identifier` . Tramite l'API RDS, chiama `CreateDBCluster` e imposta il parametro `DBCluster Identifier` .

Impostazione della console	Descrizione impostazione	Opzione CLI e parametro API di RDS
DB cluster parameter group (Gruppo di parametri del cluster database)	Scegliere gruppo di parametri del cluster database. Puoi utilizzare il gruppo di parametri del cluster database predefinito di Aurora oppure puoi creare il tuo gruppo di parametri. Per ulteriori informazioni sui gruppi di parametri del cluster di database, consulta Gruppi di parametri per Amazon Aurora.	Tramite AWS CLI, esegui `create-db-cluster` e imposta l'opzione `--db-cluster-parameter-group-name` . Tramite l'API RDS, chiama `CreateDBCluster` e imposta il parametro `DBClusterParameterGroupName` .
DB instance class (Classe istanza database)	Si applica solo al tipo di Provision ed Capacity (Capacità assegnata): Scegliere una classe di istanza database che definisca i requisiti di elaborazione e di memoria per ogni istanza nel cluster di database. Per altre informazioni sulle classi di istanza database, consulta Classi di istanze database Amazon Aurora.	Imposta questo valore per ogni istanza database nel cluster Aurora. Tramite AWS CLI, esegui `create-db-instance` e imposta l'opzione `--db-instance-class` . Tramite l'API RDS, chiama `CreateDBInstance` e imposta il parametro `DBInstanceClass` .

Impostazione della console	Descrizione impostazione	Opzione CLI e parametro API di RDS
DB parameter group (Gruppo di parametri database)	Scegliere un gruppo di parametri . Puoi utilizzare il gruppo di parametri predefinito di Aurora oppure puoi creare il tuo gruppo di parametri personalizzato. Per ulteriori informazioni sui gruppi di parametri, consultare Gruppi di parametri per Amazon Aurora.	Imposta questo valore per ogni istanza database nel cluster Aurora. Tramite AWS CLI, esegui `create-db-instance` e imposta l'opzione `--db-parameter-group-name` . Tramite l'API RDS, chiama `CreateDBInstance` e imposta il parametro `DBParameterGroupName` .
DB subnet group (Gruppo di sottoreti DB)	Il gruppo di sottoreti database da utilizzare per il cluster di database. Seleziona Choose existing (Scegli esistente) per utilizzare un gruppo di sottoreti database esistente. Quindi scegli il gruppo di sottoreti richiesto dall'elenco a discesa Existing DB subnet groups (Gruppi di sottoreti DB esistenti). Scegli Automatic setup (Configurazione automatica) per consentire a RDS di selezionare un gruppo di sottoreti database compatibile. Se non ne esiste uno, RDS crea un nuovo gruppo di sottoreti per il cluster. Per ulteriori informazioni, consulta Prerequisiti per i cluster di database.	Tramite AWS CLI, esegui `create-db-cluster` e imposta l'opzione `--db-subnet-group-name` . Tramite l'API RDS, chiama `CreateDBCluster` e imposta il parametro `DBSubnetGroupName` .

Impostazione della console	Descrizione impostazione	Opzione CLI e parametro API di RDS
Enable deletion protection (Abilita protezione da eliminazione)	Selezionare Enable deletion protection (Abilita protezione da eliminazione) per impedire l'eliminazione del cluster di database. Se si crea un cluster di database di produzione con la console, la protezione da eliminazione è abilitata per impostazione predefinita.	Tramite AWS CLI, esegui `create-db-cluster` e imposta l'opzione `--deletion-protection | --no-deletion-protection`. Tramite l'API RDS, chiama `CreateDBCluster` e imposta il parametro `DeletionProtection`.
Abilita crittografia	Scegliere `Enable encryption` per abilitare la crittografia a riposo per questo cluster di database. Per ulteriori informazioni, consulta Crittografia delle risorse Amazon Aurora.	Tramite AWS CLI, esegui `create-db-cluster` e imposta l'opzione `--storage-encrypted | --no-storage-encrypted`. Tramite l'API RDS, chiama `CreateDBCluster` e imposta il parametro `StorageEncrypted`.

Impostazione della console	Descrizione impostazione	Opzione CLI e parametro API di RDS
Enable Enhanced Monitoring (Abilita il monitoraggio avanzato)	Scegliere Enable enhanced monitoring (Abilita monitoraggio avanzato) per abilitare la raccolta di parametri in tempo reale per il sistema operativo su cui viene eseguito il cluster DB. Per ulteriori informazioni, consulta Monitoraggio dei parametri del sistema operativo con il monitoraggio avanzato.	Impostare questi valori per ogni istanza database nel cluster Aurora. Tramite AWS CLI, esegui `create-db-instance` e imposta le opzioni `--monitoring-interval` e `--monitoring-role-arn`. Tramite l'API RDS, chiama `CreateDBInstance` e imposta i parametri `MonitoringInterval` e `MonitoringRoleArn`.
Abilita l'API dati in RDS	Scegli Abilita l'API dati in RDS per abilitare l'API dati in RDS (API dati). Con l'API dati è possibile utilizzare un endpoint HTTP sicuro per eseguire le istruzioni SQL senza gestire le connessioni. Per ulteriori informazioni, consulta Utilizzo dell'API dati di Amazon RDS.	Tramite AWS CLI, esegui `create-db-cluster` e imposta l'opzione `--enable-http-endpoint | --no-enable-http-endpoint`. Tramite l'API RDS, chiama `CreateDBCluster` e imposta il parametro `EnableHttpEndpoint`.
Tipo di motore	Scegli il motore di database da utilizzare per il cluster di database.	Tramite AWS CLI, esegui `create-db-cluster` e imposta l'opzione `--engine`. Tramite l'API RDS, chiama `CreateDBCluster` e imposta il parametro `Engine`.

Impostazione della console	Descrizione impostazione	Opzione CLI e parametro API di RDS
Versione del motore	Si applica solo al tipo di Provision ed Capacity (Capacità assegnata): Selezionare il numero di versione del motore del database.	Tramite AWS CLI, esegui `create-db-cluster` e imposta l'opzione `--engine-version` . Tramite l'API RDS, chiama `CreateDBCluster` e imposta il parametro `EngineVersion` .
Failover priority (Priorità failover)	Scegliere una priorità di failover per l'istanza. Se non specifich i alcun valore, l'impostazione predefinita è tier-1. Questa priorità determina l'ordine di promozione delle repliche di Aurora durante il recupero da un errore dell'istanza principale. Per ulteriori informazi oni, consulta Tolleranza ai guasti di un cluster DB Aurora.	Imposta questo valore per ogni istanza database nel cluster Aurora. Tramite AWS CLI, esegui `create-db-instance` e imposta l'opzione `--promoti on-tier` . Tramite l'API RDS, chiama `CreateDBInstance` e imposta il parametro `PromotionTier` .

Impostazione della console	Descrizione impostazione	Opzione CLI e parametro API di RDS
Initial database name (Nome del database iniziale)	Immettere un nome per il database predefinito. Se non si specifica un nome per il cluster database Aurora MySQL, Amazon RDS non crea un database nel cluster database che si sta creando. Se non si fornisce un nome per un cluster database Aurora PostgreSQL, Amazon RDS crea un database denominato `postgres`. Per Aurora MySQL, il nome del database predefinito presenta questi vincoli: • Deve contenere da 1 a 64 caratteri alfanumerici. • Non può essere una parola riservata del motore di database. Per Aurora PostgreSQL, il nome del database predefinito presenta questi vincoli: • Deve contenere da 1 a 63 caratteri alfanumerici. • Deve iniziare con una lettera. I caratteri successivi possono essere lettere, trattini bassi o cifre (da 0 a 9). • Non può essere una parola riservata del motore di database.	Tramite AWS CLI, esegui `create-db-cluster` e imposta l'opzione `--database-name`. Tramite l'API RDS, chiama `CreateDBCluster` e imposta il parametro `DatabaseName`.

Impostazione della console	Descrizione impostazione	Opzione CLI e parametro API di RDS
	Per creare database aggiuntivi, eseguire la connessione al cluster di database e utilizzare il comando SQL CREATE DATABASE. Per ulteriori informazioni sulla connessione al cluster di database, consulta Connessione a un cluster database Amazon Aurora.	
Log exports (Esportazioni log)	Nella sezione Log exports (Esportazioni log), scegliere i log che si vuole iniziare a pubblicare in Amazon CloudWatch Logs. Per ulteriori informazioni relative alla pubblicazione di log Aurora MySQL in CloudWatch Logs, consulta Pubblicazione di log Amazon Aurora MySQL in Amazon CloudWatch Logs. Per ulteriori informazioni relative alla pubblicazione di log Aurora MySQL in CloudWatch Logs, consulta Pubblicazione di log Aurora PostgreSQL in Amazon CloudWatch Logs.	Tramite AWS CLI, esegui `create-db-cluster` e imposta l'opzione `--enable-cloudwatch-logs-exports`. Tramite l'API RDS, chiama `CreateDBCluster` e imposta il parametro `EnableCloudwatchLogsExports`.

Impostazione della console	Descrizione impostazione	Opzione CLI e parametro API di RDS
Maintenance window (Finestra di manutenzione)	Selezionare Select window (Seleziona finestra) e specifica l'intervallo temporale settimanale durante il quale può avvenire la manutenzione del sistema. Oppure, selezionare No preference (Nessuna preferenza) per consentire a Amazon RDS di assegnare un periodo casuale.	Tramite AWS CLI, esegui `create-db-cluster` e imposta l'opzione `--preferred-maintenance-window`. Tramite l'API RDS, chiama `CreateDBCluster` e imposta il parametro `PreferredMaintenanceWindow`.
Manage master credentials in Gestione dei segreti AWS (Gestione credenziali master in Gestione dei segreti AWS)	Seleziona Manage master credentials in Gestione dei segreti AWS (Gestione credenziali master in Gestione dei segreti AWS) per gestire la password dell'utente master in un segreto di Secrets Manager. Facoltativamente, scegli la chiave KMS da utilizzare per proteggere il segreto. Scegliere tra le chiavi KMS presenti nell'account o inserire la chiave da un altro account. Per ulteriori informazioni, consulta Gestione delle password con Amazon Aurora e Gestione dei segreti AWS.	Tramite AWS CLI, esegui `create-db-cluster` e imposta le opzioni `--manage-master-user-password | --no-manage-master-user-password` e `--master-user-secret-kms-key-id`. Tramite l'API RDS, chiama `CreateDBCluster` e imposta i parametri `ManageMasterUserPassword` e `MasterUserSecretKmsKeyId`.

Impostazione della console	Descrizione impostazione	Opzione CLI e parametro API di RDS
Master password (Password master)	Immettere una password per accedere al cluster DB: • Per Aurora MySQL, la password deve contenere da 8 a 41 caratteri ASCII stampabili. • Per Aurora PostgreSQL, deve contenere da 8 a 99 caratteri ASCII stampabili. • Non può contenere /, ", @ o uno spazio.	Tramite AWS CLI, esegui create-db-cluster e imposta l'opzione --master-user-password . Tramite l'API RDS, chiama CreateDBCluster e imposta il parametro MasterUserPassword .
Master username (Nome utente master)	Immettere un nome da utilizzare come nome utente master per accedere al cluster DB: • Per Aurora MySQL, il nome deve contenere da 1 a 16 caratteri alfanumerici. • Per Aurora PostgreSQL deve contenere da 1 a 63 caratteri alfanumerici. • Il primo carattere deve essere una lettera. • Il nome non può essere una parola riservata del motore di database. Dopo la creazione del cluster database, il nome utente master non può essere modificato.	Tramite AWS CLI, esegui create-db-cluster e imposta l'opzione --master-username . Tramite l'API RDS, chiama CreateDBCluster e imposta il parametro MasterUsername .

Impostazione della console	Descrizione impostazione	Opzione CLI e parametro API di RDS
Multi-AZ deployment (Implementazione Multi-AZ)	Si applica solo al tipo di Provisioned Capacity (Capacità assegnata): Determinare se si intende creare repliche di Aurora in altre zone di disponibilità ai fini del supporto per il failover. Se si seleziona Create Replica in Different Zone (Crea replica in zona diversa), Amazon RDS crea una replica di Aurora nel cluster database in una zona di disponibilità differente rispetto all'istanza principale del cluster di database. Per altre informazioni sulle zone di disponibilità multiple, consulta Regioni e zone di disponibilità.	Tramite AWS CLI, esegui `create-db-cluster` e imposta l'opzione `--availability-zones`. Tramite l'API RDS, chiama `CreateDBCluster` e imposta il parametro `AvailabilityZones`.

Impostazione della console	Descrizione impostazione	Opzione CLI e parametro API di RDS
Tipo di rete	I protocolli di indirizzo IP supportati dal cluster database. IPv4 per specificare che le risorse possono comunicare con il cluster database solo tramite il protocollo di indirizzamento IPv4. Modalità dual-stack per specificare che le risorse possono comunicare con il cluster database su IPv4, IPv6 o entrambi. Utilizza la modalità dual-stack se le risorse devono comunicare con il cluster database sul protocollo di indirizzamento IPv6. Per utilizzare la modalità dual-stack, assicurarsi che almeno due sottoreti si estendano su due zone di disponibilità che supportano entrambi i protocolli di rete IPv4 e IPv6. Inoltre, assicurarsi di associare un blocco CIDR IPv6 alle sottoreti del gruppo di sottoreti DB specificato. Per ulteriori informazioni, consulta Assegnazione di indirizzi IP in Amazon Aurora.	Tramite AWS CLI, esegui `create-db-cluster` e imposta l'opzione `-network-type` . Tramite l'API RDS, chiama `CreateDBCluster` e imposta il parametro `NetworkType` .

Impostazione della console	Descrizione impostazione	Opzione CLI e parametro API di RDS
Accesso pubblico	Scegli Publicly accessible (Accessibile pubblicamente) per assegnare al cluster DB un indirizzo IP pubblico oppure scegli Not publicly accessible (Non accessibile pubblicamente). Le istanze nel cluster database possono essere una combinazione di istanze database pubbliche e private. Per ulteriori informazioni su come non consentire l'accesso pubblico per le istanze, consulta Nascondere cluster database in un VPC da Internet. Per connettersi a un'istanza DB dall'esterno Amazon VPC, l'istanza DB deve essere accessibile pubblicamente, l'accesso deve essere concesso utilizzando le regole in ingresso del gruppo di sicurezza dell'istanza DB e devono essere soddisfatti altri requisiti. Per ulteriori informazioni, consulta Impossibile connettersi all'istanza database di Amazon RDS. Se l'istanza database non è accessibile pubblicamente, puoi anche utilizzare una connessio ne AWS Site-to-Site VPN o una connessione Direct Connect per accedervi da una rete privata. Per	Imposta questo valore per ogni istanza database nel cluster Aurora. Tramite AWS CLI, esegui `create-db-instance` e imposta l'opzione `--publicly-accessible | --no-publicly-accessible` . Tramite l'API RDS, chiama `CreateDBInstance` e imposta il parametro `PubliclyAccessible` .

Impostazione della console	Descrizione impostazione	Opzione CLI e parametro API di RDS
	ulteriori informazioni, consulta Riservatezza del traffico Internet.	
Supporto esteso per RDS	Seleziona Abilita il supporto esteso RDS per consentire alle versioni principali del motore supportate di continuare a funzionare oltre la data di fine del supporto standard per Aurora. Quando crei un cluster di database, Amazon Aurora utilizza per impostazione predefinita il Supporto esteso per RDS. Disabilita questa impostazione per impedire la creazione di un nuovo cluster di database dopo la data di fine del supporto standard Aurora e per evitare addebiti per il Supporto esteso per RDS. I cluster di database esistenti non incorrono in addebiti fino alla data di inizio del Supporto esteso per RDS. Per ulteriori informazioni, consulta Supporto esteso di Amazon RDS con Amazon Aurora.	Tramite AWS CLI, esegui `create-db-cluster` e imposta l'opzione `--engine-lifecycle-support` . Tramite l'API RDS, chiama `CreateDBCluster` e imposta il parametro `EngineLifecycleSupport` .

Impostazione della console	Descrizione impostazione	Opzione CLI e parametro API di RDS
Server proxy per RDS	Scegli Create an RDS Proxy (Crea un server proxy per RDS) per creare un proxy per il tuo cluster database. Amazon RDS crea automaticamente per il proxy un ruolo IAM e un segreto in Secrets Manager. Per ulteriori informazioni, consulta Server proxy per Amazon RDS per Aurora.	Non disponibile durante la creazione di un cluster database.
Periodo di conservazione	Selezionare l'intervallo di tempo, da 1 a 35 giorni, per il quale Aurora conserva le copie di backup del database. Le copie di backup possono essere utilizzate per operazioni di ripristino point-in-time (PITR) del database al secondo.	Tramite AWS CLI, esegui `create-db-cluster` e imposta l'opzione `--backup-retention-period`. Tramite l'API RDS, chiama `CreateDBCluster` e imposta il parametro `BackupRetentionPeriod`.
Attiva DevOps Guru	Scegli Turn on DevOps Guru (Attiva DevOps Guru) per attivare Amazon DevOps Guru per il database Aurora. Affinché DevOps Guru per RDS fornisca un'analisi dettagliata delle anomalie delle prestazioni, Performance Insights deve essere attivato. Per ulteriori informazioni, consulta Configurazione di Guru per RDS DevOps.	Puoi attivare DevOps Guru per RDS dalla console RDS, ma non utilizzando l'API RDS o CLI. Per ulteriori informazioni sull'attivazione di DevOps Guru, consulta la Guida per l'utente di Amazon DevOps Guru.

Impostazione della console	Descrizione impostazione	Opzione CLI e parametro API di RDS
Attiva Performance Insights	Scegli Turn on Performance Insights (Attiva Performance Insights) per attivare Amazon RDS Performance Insights. Per ulteriori informazioni, consulta Monitoraggio del carico DB con Performance Insights su Amazon Aurora.	Impostare questi valori per ogni istanza database nel cluster Aurora. Tramite la AWS CLI, esegui `create-db-instance` e imposta le opzioni `--enable-performance-insights` | `--no-enable-performance-insights`, `--performance-insights-kms-key-id` e `--performance-insights-retention-period`. Tramite l'API RDS, chiama `CreateDBInstance` e imposta i parametri `EnablePerformanceInsights`, `PerformanceInsightsKMSKeyId` e `PerformanceInsightsRetentionPeriod`.
Virtual Private Cloud (VPC)	Scegliere il VPC che ospita il cluster DB. Scegliere Create a new VPC (Crea un nuovo VPC) per fare in modo che Amazon RDS crei automaticamente un VPC. Per ulteriori informazioni, consulta Prerequisiti per i cluster di database.	Per AWS CLI e API, specificare gli ID del gruppo di sicurezza VPC.

Impostazione della console	Descrizione impostazione	Opzione CLI e parametro API di RDS
VPC security group (firewall) (Gruppo di sicurezza VPC (firewall))	Scegli Create new (Crea nuovo) per fare in modo che Amazon RDS crei un gruppo di sicurezza VPC. Altrimenti, seleziona Choose existing (Seleziona esistenti) e specifica uno o più gruppi di sicurezza VPC per proteggere l'accesso di rete al cluster di database. Quando scegli Create new (Crea nuovo) nella console RDS, viene creato un nuovo gruppo di sicurezza con una regola in entrata che consente l'accesso all'istanza database dall'indirizzo IP rilevato nel browser. Per ulteriori informazioni, consulta Prerequisiti per i cluster di database.	Tramite AWS CLI, esegui `create-db-cluster` e imposta l'opzione `--vpc-security-group-ids`. Tramite l'API RDS, chiama `CreateDBCluster` e imposta il parametro `VpcSecurityGroupIds`.

Impostazioni che non si applicano ad Amazon Aurora per i cluster di database

Le impostazioni seguenti nel comando della AWS CLI `create-db-cluster` e l'operazione `CreateDBCluster` dell'API RDS non si applicano ai cluster di database Amazon Aurora.

> **Note**
>
> La Console di gestione AWS non mostra queste impostazioni per i cluster di database Aurora.

AWS CLIImpostazione	Impostazione API RDS
`--allocated-storage`	`AllocatedStorage`
`--auto-minor-version-upgrade | --no-auto-minor-version-upgrade`	`AutoMinorVersionUpgrade`
`--db-cluster-instance-class`	`DBClusterInstanceClass`
`--enable-performance-insights | --no-enable-performance-insights`	`EnablePerformanceInsights`
`--iops`	`Iops`
`--monitoring-interval`	`MonitoringInterval`
`--monitoring-role-arn`	`MonitoringRoleArn`
`--option-group-name`	`OptionGroupName`
`--performance-insights-kms-key-id`	`PerformanceInsightsKMSKeyId`
`--performance-insights-retention-period`	`PerformanceInsightsRetentionPeriod`
`--publicly-accessible | --no-publicly-accessible`	`PubliclyAccessible`

Impostazioni che non si applicano alle istanze database Amazon Aurora

Le impostazioni seguenti nel comando della AWS CLI `create-db-instance` e l'operazione `CreateDBInstance` dell'API RDS non si applicano ai cluster di database delle istanze database Amazon Aurora.

> **Note**
>
> La Console di gestione AWS non mostra queste impostazioni per le istanze database Aurora.

AWS CLIImpostazione	Impostazione API RDS
`--allocated-storage`	`AllocatedStorage`
`--availability-zone`	`AvailabilityZone`
`--backup-retention-period`	`BackupRetentionPeriod`
`--backup-target`	`BackupTarget`
`--character-set-name`	`CharacterSetName`
`--character-set-name`	`CharacterSetName`
`--custom-iam-instance-profile`	`CustomIamInstanceProfile`
`--db-security-groups`	`DBSecurityGroups`
`--deletion-protection | --no-deletion-protection`	`DeletionProtection`
`--domain`	`Domain`
`--domain-iam-role-name`	`DomainIAMRoleName`
`--enable-cloudwatch-logs-exports`	`EnableCloudwatchLogsExports`
`--enable-customer-owned-ip | --no-enable-customer-owned-ip`	`EnableCustomerOwnedIp`
`--enable-iam-database-authentication | --no-enable-iam-database-authentication`	`EnableIAMDatabaseAuthentication`
`--engine-version`	`EngineVersion`
`--iops`	`Iops`
`--kms-key-id`	`KmsKeyId`
`--master-username`	`MasterUsername`

AWS CLIImpostazione	Impostazione API RDS
`--master-user-password`	`MasterUserPassword`
`--max-allocated-storage`	`MaxAllocatedStorage`
`--multi-az | --no-multi-az`	`MultiAZ`
`--nchar-character-set-name`	`NcharCharacterSetName`
`--network-type`	`NetworkType`
`--option-group-name`	`OptionGroupName`
`--preferred-backup-window`	`PreferredBackupWindow`
`--processor-features`	`ProcessorFeatures`
`--storage-encrypted | --no-stor age-encrypted`	`StorageEncrypted`
`--storage-type`	`StorageType`
`--tde-credential-arn`	`TdeCredentialArn`
`--tde-credential-password`	`TdeCredentialPassword`
`--timezone`	`Timezone`
`--vpc-security-group-ids`	`VpcSecurityGroupIds`

Creazione di risorse Amazon Aurora con AWS CloudFormation

Amazon Aurora è integrato con AWS CloudFormation, un servizio che ti consente di modellare e configurare le tue risorse AWS in modo da dedicare meno tempo alla creazione e alla gestione delle risorse e dell'infrastruttura. È possibile creare un modello che descrive tutte le AWS risorse desiderate (ad esempio cluster DB e gruppi di parametri del cluster DB), nonché il CloudFormation provisioning e la configurazione di tali risorse per l'utente.

Quando usi CloudFormation, puoi riutilizzare il modello per configurare le risorse Aurora in modo coerente e continuo. Descrivere le risorse una volta e quindi allestisci le stesse risorse più volte in più regioni e account AWS.

Aurora e modelli CloudFormation

I modelli CloudFormation sono file di testo formattati in JSON o YAML. Questi modelli descrivono le risorse di cui intendi effettuare il provisioning negli stack CloudFormation. Se non hai familiarità con JSON o YAML, puoi usare CloudFormationDesigner per iniziare a utilizzare i modelli CloudFormation. Per ulteriori informazioni, consulta Che cos'è CloudFormation Designer? nella Guida per l'utente di AWS CloudFormation.

Aurora supporta la creazione di risorse in CloudFormation. Per ulteriori informazioni, inclusi esempi di modelli JSON e YAML per le risorse, consulta Riferimento dei tipi di risorse RDS nella Guida per l'utente di AWS CloudFormation.

Ulteriori informazioni su CloudFormation

Per ulteriori informazioni su CloudFormation, consulta le seguenti risorse:

- AWS CloudFormation
- Guida per l'utente di AWS CloudFormation
- Riferimento APICloudFormation
- Guida per l'utente dell'interfaccia a riga di comando di AWS CloudFormation

Connessione a un cluster database Amazon Aurora

Puoi eseguire la connessione a un cluster di database Aurora utilizzando gli stessi strumenti che utilizzi per connetterti a un database MySQL o PostgreSQL. Devi specificare una stringa di connessione con qualsiasi script, utility o applicazione che si connette a un'istanza di database MySQL o PostgreSQL. Utilizzi la stessa chiave pubblica per le connessioni Secure Sockets Layer (SSL).

Nella stringa di connessione, in genere utilizzerai le informazioni sulla porta e sull'host dall'endpoint speciali associate al cluster di database. Con questi endpoint, puoi utilizzare gli stessi parametri di connessione indipendentemente dal numero di istanze di database presenti nel cluster. Per attività specializzate come la risoluzione dei problemi, puoi anche utilizzare le informazioni sull'host e sulla porta da un'istanza database specifica nel tuo cluster di database di Aurora.

> **Note**
>
> Per i cluster di database di Aurora Serverless, puoi connetterti all'endpoint del database anziché all'istanza database. Puoi trovare l'endpoint di database per un cluster di database Aurora Serverless nella scheda Connectivity & security (Connettività e sicurezza) della Console di gestione AWS. Per ulteriori informazioni, consulta Utilizzo di Amazon Aurora Serverless v1.

Indipendentemente dal motore di database Aurora e dagli strumenti specifici che utilizzi con il cluster o l'istanza database, l'endpoint deve essere accessibile. Un cluster di database Aurora può essere creato solo in un cloud privato virtuale (VPC) basato sul servizio Amazon VPC. Ciò significa accedi all'endpoint dall'interno o dall'esterno del VPC utilizzando uno dei seguenti approcci.

- Accesso al cluster di database Aurora dall'interno del VPC: abilita l'accesso al cluster di database Aurora tramite il VPC. A tale scopo, modifica le regole in entrata nel gruppo di sicurezza per il VPC in modo da consentire l'accesso al cluster di database Aurora specifico. Per saperne di più, incluso come configurare il tuo VPC per diversi scenari di cluster Aurora DB, consulta Amazon Virtual Private Cloud e Amazon Aurora. VPCs
- Accesso al cluster di database Aurora dall'esterno del VPC: per accedere a un cluster di database Aurora dall'esterno del VPC, utilizza l'indirizzo dell'endpoint pubblico del cluster di database.

Per ulteriori informazioni, consulta Risoluzione dei problemi di connessione in Aurora.

Indice

- Connessione ai cluster di database Aurora con i driver AWS
- Connessione a un cluster di database Amazon Aurora MySQL
  - Utilità di connessione per Aurora MySQL
  - Connessione con Aurora MySQL tramite l'utilità MySQL
  - Connessione ad Aurora MySQL con il driver JDBC per Amazon Web Services (AWS)
  - Connessione ad Aurora MySQL con il driver Python per Amazon Web Services (AWS)
  - Connessione ad Aurora MySQL con il driver ODBC per MySQL per Amazon Web Services (AWS).
  - Connessione ad Aurora MySQL con il wrapper NodeJS avanzato per Amazon Web Services (AWS)
  - Connessione ad Aurora MySQL con SSL
- Connessione a un cluster di database Amazon Aurora PostgreSQL
  - Utilità di connessione per Aurora PostgreSQL
  - Connessione ad Aurora PostgreSQL con il driver JDBC per Amazon Web Services (AWS)
  - Connessione ad Aurora PostgreSQL con il driver Python per Amazon Web Services (AWS)
  - Connessione ad Aurora PostgreSQL con il wrapper NodeJS avanzato per Amazon Web Services (AWS)
- Risoluzione dei problemi di connessione in Aurora

Connessione ai cluster di database Aurora con i driver AWS

La AWS suite di driver è stata progettata per fornire supporto per tempi di switchover e failover più rapidi e l'autenticazione conGestione dei segreti AWS, AWS Identity and Access Management (IAM) e Federated Identity. I AWS driver si basano sul monitoraggio dello stato del cluster DB e sulla conoscenza della topologia del cluster per determinare il nuovo writer. Questo approccio riduce i tempi di switchover e failover a meno di dieci secondi, rispetto alle decine di secondi necessari per i driver open source.

La tabella seguente elenca le funzionalità supportate per ciascuno dei driver. Con l'introduzione di nuove funzionalità di servizio, l'obiettivo della AWS suite di driver è disporre di un supporto integrato per queste funzionalità di servizio.

Funzionalità	AWSDriver JDBC	AWSDriver Python	AWSDriver ODBC per MySQL	AWSWrapper NodeJS avanzato
Supporto per il failover	Sì	Sì	Sì	Sì
Monitoraggio avanzato del failover	Sì	Sì	Sì	Sì
Suddivisione in lettura/scrittura	Sì	Sì	No	Sì
Tracker di connessioni Aurora	Sì	Sì	No	Sì
Connessione ai metadati del driver	Sì	N/D	N/D	N/D
Telemetria	Sì	Sì	No	Sì
Secrets Manager	Sì	Sì	Sì	Sì
Autenticazione IAM	Sì	Sì	Sì	Sì
Identità federata (AD FS)	Sì	Sì	No	Sì
Identità federata (Okta)	Sì	Sì	Sì	Sì
Aurora PostgreSQL Limitless Database	Sì (solo Aurora PostgreSQL)	No	No	Sì (solo Aurora PostgreSQL)

Per ulteriori informazioni sui AWS driver, consulta il driver di lingua corrispondente per il cluster Aurora MySQL o Aurora PostgreSQL DB.

Connessione a un cluster di database Amazon Aurora MySQL

Per autenticarsi sul cluster Aurora MySQL DB, è possibile utilizzare l'autenticazione con nome utente e password MySQL o l'autenticazione del database (IAM). AWS Identity and Access Management Per ulteriori informazioni sull'utilizzo dell'autenticazione tramite nome utente e password MySQL, consulta la pagina relativa a Access Control and Account Management nella documentazione di MySQL. Per ulteriori informazioni sull'utilizzo dell'autenticazione database IAM, consulta Autenticazione del database IAM .

Dopo la connessione a un cluster di database Amazon Aurora con compatibilità MySQL 8.0, puoi eseguire comandi SQL compatibili con MySQL versione 8.0. La versione minima compatibile è MySQL 8.0.23. Per ulteriori informazioni sulla sintassi SQL di MySQL 8.0, consulta il manuale di riferimento di MySQL 8.0. Per informazioni sui limiti relativi a Aurora MySQL 3, consulta Confronto tra Aurora MySQL versione 3 e MySQL 8.0 Community Edition.

Dopo la connessione a un cluster di database Amazon Aurora con compatibilità MySQL 5.7, puoi eseguire comandi SQL compatibili con MySQL versione 5.7. Per ulteriori informazioni sulla sintassi SQL di MySQL 5.7, consulta il manuale di riferimento di MySQL 5.7. Per informazioni sui limiti relativi a Aurora MySQL 5.7, consulta Aurora MySQL versione 2 compatibile con MySQL 5.7.

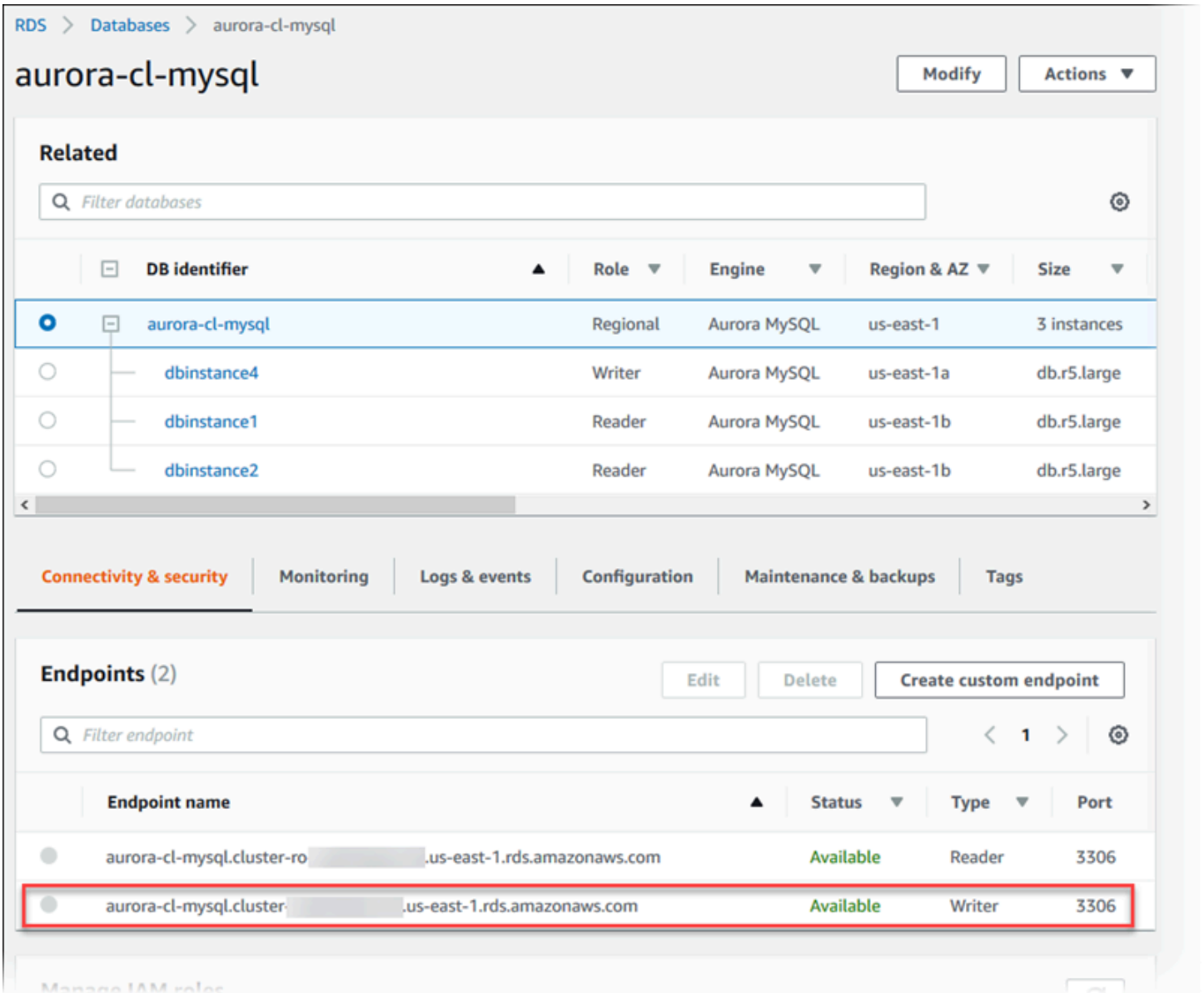
> **Note**
>
> Per una guida dettagliata e utile sulla connessione a un cluster di database Amazon Aurora MySQL, consulta il manuale relativo alla gestione delle connessioni di Aurora.

Nella vista dettagliata del cluster di database, puoi trovare l'endpoint del cluster, utilizzabile nella stringa di connessione MySQL. L'endpoint è costituito da una porta e da un nome di dominio per il cluster di database. Ad esempio, se il valore di un endpoint è `mycluster.cluster-123456789012.us-east-1.rds.amazonaws.com:3306`, specifica i valori seguenti in una stringa di connessione MySQL:

- Per l'host o il nome host, specifica `mycluster.cluster-123456789012.us-east-1.rds.amazonaws.com`
- Per la porta, specifica `3306` o il valore di porta utilizzato alla creazione del cluster di database.

L'endpoint del cluster consente la connessione all'istanza principale del cluster di database. Puoi eseguire operazioni di lettura e scrittura utilizzando l'endpoint del cluster. Il cluster di database può anche avere fino a 15 repliche di Aurora che supportano l'accesso in sola lettura ai dati nel cluster di database. L'istanza principale e ogni replica Aurora hanno un endpoint univoco che è indipendente dall'endpoint del cluster e che ti consente di eseguire la connessione direttamente a una specifica istanza database nel cluster. L'endpoint del cluster punta sempre all'istanza principale. Se l'istanza principale non riesce e viene sostituita, l'endpoint del cluster punta alla nuova istanza principale.

Per visualizzare l'endpoint del cluster (endpoint di scrittura), scegli Databases (Database) nella console Amazon RDS, quindi scegli il nome del cluster di database per visualizzare i dettagli corrispondenti.



Argomenti

- Utilità di connessione per Aurora MySQL
- Connessione con Aurora MySQL tramite l’utilità MySQL
- Connessione ad Aurora MySQL con il driver JDBC per Amazon Web Services (AWS)
- Connessione ad Aurora MySQL con il driver Python per Amazon Web Services (AWS)
- Connessione ad Aurora MySQL con il driver ODBC per MySQL per Amazon Web Services (AWS).
- Connessione ad Aurora MySQL con il wrapper NodeJS avanzato per Amazon Web Services (AWS)
- Connessione ad Aurora MySQL con SSL

Utilità di connessione per Aurora MySQL

Alcune delle utilità di connessione che puoi utilizzare sono elencate di seguito:

- Riga di comando – Puoi eseguire la connessione a un cluster di database Amazon Aurora utilizzando strumenti come l'utilità a riga di comando MySQL. Per ulteriori informazioni sull'utilizzo dell'utilità di MySQL, consulta mysql - Il client della linea di comando di MySQL nella documentazione di MySQL.
- GUI – Puoi utilizzare l'utilità MySQL Workbench per eseguire la connessione utilizzando un'interfaccia utente. Per ulteriori informazioni, consulta la pagina Download MySQL Workbench.
- AWSdriver:
  - Connessione ad Aurora MySQL con il driver JDBC per Amazon Web Services (AWS)
  - Connessione ad Aurora MySQL con il driver Python per Amazon Web Services (AWS)
  - Connessione ad Aurora MySQL con il driver ODBC per MySQL per Amazon Web Services (AWS).
  - Connessione ad Aurora MySQL con il wrapper NodeJS avanzato per Amazon Web Services (AWS)

Connessione con Aurora MySQL tramite l’utilità MySQL

Attenersi alla seguente procedura: Presuppone che il cluster database sia stato configurato in una sottorete privata nel VPC. Ti connetti utilizzando un' EC2 istanza Amazon configurata in base ai tutorial in. Tutorial: creazione di un server Web e un cluster database Amazon Aurora

> **Note**
>
> Questa procedura non richiede l'installazione del server web nel tutorial, ma richiede l'installazione di MariaDB 10.5.

Eseguire la connessione a un cluster database con l'utilità MySQL

1. Accedi all' EC2 istanza che stai utilizzando per connetterti al tuo cluster DB.

   Verrà visualizzato un output simile al seguente.

   ```
   Last login: Thu Jun 23 13:32:52 2022 from xxx.xxx.xxx.xxx

          __|  __|_  )
          _|  (     /   Amazon Linux 2 AMI
         ___|\___|___|

   https://aws.amazon.com/amazon-linux-2/
   [ec2-user@ip-10-0-xxx.xxx ~]$
   ```

2. Digita il comando seguente al prompt dei comandi per eseguire la connessione all'istanza database principale di un cluster database.

   Per il parametro `-h`, sostituisci il nome DNS dell'endpoint per l'istanza principale. Per il parametro `-u`, sostituisci l'ID utente di un account utente di database.

   ```
   mysql -h primary-instance-endpoint.AWS_account.AWS_Region.rds.amazonaws.com -P 3306
    -u database_user -p
   ```

   Ad esempio:

   ```
   mysql -h my-aurora-cluster-instance.c1xy5example.123456789012.eu-
   central-1.rds.amazonaws.com -P 3306 -u admin -p
   ```

3. Immetti la password per l'utente del database.

   Verrà visualizzato un output simile al seguente.

   ```
   Welcome to the MariaDB monitor.  Commands end with ; or \g.
   Your MySQL connection id is 1770
   ```

```
Server version: 8.0.23 Source distribution

Copyright (c) 2000, 2018, Oracle, MariaDB Corporation Ab and others.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

MySQL [(none)]>
```

4. Immetti i comandi SQL.

Connessione ad Aurora MySQL con il driver JDBC per Amazon Web Services (AWS)

Il driver JDBC per Amazon Web Services (AWS) è progettato come wrapper JDBC avanzato. Questo wrapper integra ed estende la funzionalità di un driver JDBC esistente per aiutare le applicazioni a sfruttare le funzionalità dei database con cluster come Aurora MySQL. Il driver è compatibile direttamente con il driver MySQL della community. Connector/J driver and the community MariaDB Connector/J

Per installare il driver AWS JDBC, aggiungi il file.jar del driver AWS JDBC (che si trova nell'applicazione`CLASSPATH`) e mantieni i riferimenti al rispettivo driver della community. Aggiorna il rispettivo prefisso dell'URL di connessione nel modo seguente:

- `jdbc:mysql://` Da a `jdbc:aws-wrapper:mysql://`
- `jdbc:mariadb://` Da a `jdbc:aws-wrapper:mariadb://`

Per ulteriori informazioni sul driver AWS JDBC e istruzioni complete per il suo utilizzo, consulta l'archivio dei driver JDBC di Amazon Web Services (AWS). GitHub

> **Note**
>
> La versione 3.0.3 dell'utilità Connector/J MariaDB elimina il supporto per i cluster Aurora DB, quindi consigliamo vivamente di passare al driver JDBC. AWS

Connessione ad Aurora MySQL con il driver Python per Amazon Web Services (AWS)

Amazon Web Services (AWS) Python Driver è progettato come wrapper Python avanzato. Tale wrapper è complementare al driver open source Psycopg e ne estende le funzionalità. Il AWS Python Driver supporta le versioni Python 3.8 e successive. È possibile installare il pacchetto `aws-`

`advanced-python-wrapper` utilizzando il comando `pip`, insieme ai pacchetti open source `psycopg`.

Per ulteriori informazioni sul driver AWS Python e istruzioni complete per il suo utilizzo, consulta il repository Amazon Web Services () AWS Python Driver. GitHub

Connessione ad Aurora MySQL con il driver ODBC per MySQL per Amazon Web Services (AWS).

Il driver AWS ODBC per MySQL è un driver client progettato per l'elevata disponibilità di Aurora MySQL. Il driver può esistere insieme al driver Connector/ODBC MySQL ed è compatibile con gli stessi flussi di lavoro.

Per ulteriori informazioni sul driver AWS ODBC per MySQL e istruzioni complete per installarlo e utilizzarlo, consulta l'archivio Amazon Web Services AWS () ODBC Driver for MySQL. GitHub

Connessione ad Aurora MySQL con il wrapper NodeJS avanzato per Amazon Web Services (AWS)

L'AWSAdvanced NodeJS Wrapper è complementare ed estende la funzionalità di un driver NodeJS esistente. Aiuta le applicazioni a sfruttare le funzionalità dei database con cluster come Aurora MySQL.

Per ulteriori informazioni su AWS Advanced NodeJS Wrapper e istruzioni complete per il suo utilizzo, consulta il repository Amazon Web Services (AWS) Advanced NodeJS Wrapper. GitHub

Connessione ad Aurora MySQL con SSL

Puoi utilizzare la crittografia SSL sulle connessioni a un'istanza database Aurora MySQL. Per informazioni, consulta Connessioni TLS a cluster di database Aurora MySQL.

Per eseguire una connessione mediante SSL, utilizza l'utilità MySQL come descritto nella procedura successiva. Se utilizzi l'autenticazione database IAM, devi utilizzare una connessione SSL. Per informazioni, consulta Autenticazione del database IAM .

> **Note**
>
> Per eseguire la connessione all'endpoint del cluster mediante SSL, l'utilità di connessione client deve supportare Subject Alternative Names (SAN). Se l'utilità di connessione client non

supporta SAN, puoi eseguire la connessione direttamente alle istanze nel cluster di database Aurora. Per ulteriori informazioni sugli endpoint Aurora, consulta Connessioni degli endpoint Amazon Aurora.

Per eseguire la connessione a un cluster di database con SSL utilizzando l'utilità MySQL

1. Scaricare la chiave pubblica per il certificato di firma Amazon RDS.

   Per ulteriori informazioni sul download dei certificati, consultare Utilizzo SSL/TLS per crittografare una connessione a un'.
2. Digitare il comando seguente a un prompt dei comandi per eseguire la connessione all'istanza principale di un cluster di database con SSL mediante l'utilità MySQL. Per il parametro `-h`, sostituisci il nome DNS dell'endpoint per l'istanza principale. Per il parametro `-u`, sostituisci l'ID utente di un account utente di database. Per il parametro `--ssl-ca`, sostituire il nome file del certificato SSL come appropriato. Digitare la password dell'utente master, quando richiesto.

   ```
   mysql -h mycluster-primary.123456789012.us-east-1.rds.amazonaws.com -u admin_user -p --ssl-ca=[full path]global-bundle.pem --ssl-verify-server-cert
   ```

Verrà visualizzato un output simile al seguente.

```
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 350
Server version: 8.0.26-log MySQL Community Server (GPL)

Type 'help;' or '\h' for help. Type '\c' to clear the buffer.

mysql>
```

Per istruzioni generali sulla costruzione delle stringhe di connessione di RDS for MySQL e sulla ricerca della chiave pubblica per le connessioni SSL, consulta Connessione a un'istanza database che esegue il motore di database di MySQL.

Connessione a un cluster di database Amazon Aurora PostgreSQL

Puoi eseguire la connessione a un'istanza database nel cluster di database Amazon Aurora PostgreSQL utilizzando gli stessi strumenti che utilizzi per connetterti a un database PostgreSQL. A

questo proposito, utilizzi la stessa chiave pubblica per le connessioni Secure Sockets Layer (SSL). Puoi utilizzare le informazioni relative a endpoint e porta dell'istanza principale o le repliche Aurora del cluster di database PostgreSQL di Aurora nella stringa di connessione di qualsiasi script, utilità o applicazione che si connette a un'istanza database PostgreSQL. Nella stringa di connessione, specifica l'indirizzo DNS dell'endpoint dell'istanza principale o della replica Aurora come parametro host. Specifica il numero di porta dell'endpoint come parametro della porta.

Dopo la connessione a un'istanza database nel cluster di database Amazon Aurora PostgreSQL, puoi eseguire qualsiasi comando SQL compatibile con PostgreSQL.

Nella visualizzazione dettagliata del cluster di database Aurora PostgreSQL, puoi trovare il nome lo status, il tipo e il numero di porta dell'endpoint del cluster. Puoi utilizzare il numero di endpoint e di porta nella stringa di connessione PostgreSQL. Ad esempio, se il valore di un endpoint è `mycluster.cluster-123456789012.us-east-1.rds.amazonaws.com`, specifica i valori seguenti in una stringa di connessione PostgreSQL:

- Per l'host o il nome host, specifica `mycluster.cluster-123456789012.us-east-1.rds.amazonaws.com`
- Per la porta, specifica `5432` o il valore di porta utilizzato alla creazione del cluster di database.

L'endpoint del cluster consente la connessione all'istanza principale del cluster di database. Puoi eseguire operazioni di lettura e scrittura utilizzando l'endpoint del cluster. Il cluster di database può anche avere fino a 15 repliche di Aurora che supportano l'accesso in sola lettura ai dati nel cluster di database. Ogni istanza database nel cluster Aurora, ovvero l'istanza principale e ogni replica Aurora, ha un endpoint univoco che è indipendente dall'endpoint del cluster. e che ti consente di eseguire la connessione direttamente a una specifica istanza database nel cluster. L'endpoint del cluster punta sempre all'istanza principale. Se l'istanza principale non riesce e viene sostituita, l'endpoint del cluster punta alla nuova istanza principale.

Per visualizzare l'endpoint del cluster (endpoint di scrittura), scegli Databases (Database) nella console Amazon RDS, quindi scegli il nome del cluster di database per visualizzare i dettagli corrispondenti.
Endpoint di scrittura Aurora PostgreSQL.

Utilità di connessione per Aurora PostgreSQL

Alcune delle utilità di connessione che puoi utilizzare sono elencate di seguito:

- Riga di comando: puoi eseguire la connessione a cluster database Aurora PostgreSQL utilizzando strumenti come psql, il terminale interattivo PostgreSQL. Per ulteriori informazioni sull'utilizzo del terminale interattivo PostgreSQL, consulta psql nella documentazione di PostgreSQL.
- GUI: puoi utilizzare l'utilità pgAdmin per eseguire la connessione a cluster database Aurora PostgreSQL utilizzando un'interfaccia utente. Per ulteriori informazioni, consulta la pagina dei download sul sito Web di pgAdmin.
- AWSdriver:
  - Connessione ad Aurora PostgreSQL con il driver JDBC per Amazon Web Services (AWS)
  - Connessione ad Aurora PostgreSQL con il driver Python per Amazon Web Services (AWS)
  - Connessione ad Aurora PostgreSQL con il wrapper NodeJS avanzato per Amazon Web Services (AWS)

Connessione ad Aurora PostgreSQL con il driver JDBC per Amazon Web Services (AWS)

Il driver JDBC per Amazon Web Services (AWS) è progettato come wrapper JDBC avanzato. Questo wrapper integra ed estende la funzionalità di un driver JDBC esistente per aiutare le applicazioni a sfruttare le funzionalità dei database con cluster come Aurora PostgreSQL. Il driver è immediatamente compatibile con il driver pgJDBC della versione Community.

Per installare il driver AWS JDBC, aggiungi il file.jar del driver AWS JDBC (che si trova nell'applicazione) e mantieni i riferimenti al driver della community pgJDBC. `CLASSPATH` Aggiorna il prefisso dell'URL di connessione da `jdbc:postgresql://` a `jdbc:aws-wrapper:postgresql://`.

Per ulteriori informazioni sul driver AWS JDBC e istruzioni complete per il suo utilizzo, consulta l'archivio dei driver JDBC di Amazon Web Services (AWS). GitHub

Connessione ad Aurora PostgreSQL con il driver Python per Amazon Web Services (AWS)

Amazon Web Services (AWS) Python Driver è progettato come wrapper Python avanzato. Tale wrapper è complementare al driver open source Psycopg e ne estende le funzionalità. Il AWS Python Driver supporta le versioni Python 3.8 e successive. È possibile installare il pacchetto `aws-advanced-python-wrapper` utilizzando il comando `pip`, insieme ai pacchetti open source `psycopg`.

Per ulteriori informazioni sul driver AWS Python e istruzioni complete per il suo utilizzo, consulta il repository Amazon Web Services () AWS Python Driver. GitHub

Connessione ad Aurora PostgreSQL con il wrapper NodeJS avanzato per Amazon Web Services (AWS)

L'AWSAdvanced NodeJS Wrapper è complementare ed estende la funzionalità di un driver NodeJS esistente. Aiuta le applicazioni a sfruttare le funzionalità dei database con cluster come Aurora PostgreSQL.

Per ulteriori informazioni su AWS Advanced NodeJS Wrapper e istruzioni complete per il suo utilizzo, consulta il repository Amazon Web Services (AWS) Advanced NodeJS Wrapper. GitHub

Risoluzione dei problemi di connessione in Aurora

Di seguito, le cause più comuni dei problemi di connessione a un nuovo cluster di database Aurora:

- Il gruppo di sicurezza nel VPC non consente l'accesso: il tuo VPC deve consentire le connessioni dal tuo dispositivo o da un' EC2 istanza Amazon mediante una corretta configurazione del gruppo di sicurezza nel VPC. Per risolvere il problema, modifica le regole del gruppo di sicurezza del VPC in ingresso in modo da consentire le connessioni. Per un esempio, consulta Tutorial: Creazione di un Amazon VPC da utilizzare con un cluster database (solo IPv4).
- La porta è bloccata per via delle regole del firewall – Controlla il valore della porta configurato per il cluster di database Aurora. Se una regola del firewall blocca tale porta, puoi ricreare l'istanza utilizzando una porta diversa.
- La configurazione di IAM è incompleta o non corretta – Se l'istanza database Aurora che hai creato richiede l'utilizzo dell'autenticazione basata su IAM, assicurarti che la configurazione sia corretta. Per ulteriori informazioni, consulta Autenticazione del database IAM .

Per ulteriori informazioni sulla risoluzione dei problemi di connessione al database Aurora, consulta Impossibile connettersi all'istanza database di Amazon RDS.

Gruppi di parametri per Amazon Aurora

Parametri database specificano la modalità di configurazione del database. Ad esempio, i parametri del database possono specificare la quantità di risorse, come la memoria, da allocare a un database.

È possibile gestire la configurazione del database associando le istanze database e i cluster di database Aurora con i gruppi di parametri. Aurora definisce i gruppi di parametri con le impostazioni di default. Puoi definire i gruppi di parametri anche con le impostazioni personalizzate.

Argomenti

- Panoramica dei gruppi di parametri
- Gruppi di parametri del cluster di database per i cluster di database Amazon Aurora
- Gruppi di parametri database per istanze database Amazon Aurora
- Confronto di gruppi di parametri database
- Specifica dei parametri del database

Panoramica dei gruppi di parametri

Un gruppo di parametri cluster database agisce da container per i valori di configurazione del motore che si applicano a ogni istanza database in un cluster database Aurora. Ad esempio, il modello di storage condiviso di Aurora richiede che ogni istanza database in un cluster Aurora utilizzi la stessa impostazione per i parametri come ad esempio `innodb_file_per_table`. Pertanto, i parametri che influenzano il layout di storage fisico sono parte del gruppo di parametri del cluster. Il gruppo di parametri cluster database include anche i valori predefiniti per tutti i parametri a livello di istanza.

Un gruppo di parametri database agisce da container per i valori di configurazione del motore che si applicano a una o più istanze database. I gruppi di parametri di database si applicano alle istanze database in Amazon RDS e Aurora. Queste impostazioni di configurazione si applicano alle proprietà che possono variare tra le istanze database all'interno di un cluster Aurora, ad esempio le dimensioni dei buffer di memoria.

Argomenti

- Gruppi di parametri predefiniti e personalizzati
- Parametri statici e dinamici del cluster database
- Parametri statici e dinamici dell'istanza database

- Parametri del set di caratteri
- Parametri e valori dei parametri supportati

Gruppi di parametri predefiniti e personalizzati

Se decidi di creare un'istanza database senza specificare un gruppo di parametri di database, l'istanza database utilizza un gruppo di parametri predefinito. Allo stesso modo, se crei un cluster di database Aurora senza specificare un gruppo di parametri cluster di database, il cluster di database utilizza un gruppo di parametri cluster di databasedi default. Ogni gruppo di parametri di default contiene le impostazioni predefinite del motore del database e le impostazioni predefinite di sistema di Amazon RDS in base a motore, classe di elaborazione e storage allocato dell'istanza.

Non puoi modificare le impostazioni dei parametri di un gruppo di parametri predefinito. Puoi invece procedere come descritto di seguito:

1. Crea un nuovo set di parametri.
2. Modifica le impostazioni dei parametri desiderati. Non tutti i parametri del motore di database presenti nel gruppo di parametri possono essere modificati.
3. Modificare l'istanza database o il cluster di database per associare il nuovo gruppo di parametri.

   Per informazioni sulla modifica di un cluster database o un'istanza database, consulta Modifica di un cluster database Amazon Aurora.

> **Note**
>
> Se hai modificato l'istanza database per utilizzare un gruppo di parametri personalizzati e avvii l'istanza database, RDS riavvia automaticamente l'istanza database come parte del processo di avvio. Per le istanze Multi-AZ RDS per SQL Server con l'opzione Always On o Mirroring abilitata, è previsto un failover quando l'istanza viene riavviata dopo il processo di avvio.

RDS applica i parametri statici e dinamici modificati in un gruppo di parametri appena associato solo dopo il riavvio dell'istanza database. Tuttavia, se modifichi i parametri dinamici nel gruppo di parametri database associato all'istanza database, tali modifiche vengono applicate immediatamente senza eseguire il riavvio. Per informazioni sulla modifica del gruppo di parametri database, consulta Modifica di un cluster database Amazon Aurora.

Se aggiorni i parametri all'interno di un gruppo di parametri database, le modifiche si applicano a tutte le istanze database associate al gruppo di parametri. Allo stesso modo, se aggiorni i parametri in un gruppo di parametri cluster database Aurora, le modifiche si applicano a tutti i cluster database Aurora associati al gruppo di parametri cluster database.

Per evitare di creare un gruppo di parametri da zero, puoi copiare un gruppo di parametri esistente con il comando AWS CLI copy-db-parameter-group o il comando copy-db-cluster-parameter-group. In alcuni casi la copia di un gruppo di parametri è utile. Ad esempio quando devi includere la maggior parte dei valori e dei parametri personalizzati del gruppo di parametri esistente in un nuovo gruppo di parametri .

Parametri statici e dinamici del cluster database

I parametri di cluster di database sono statici o dinamici. Differiscono nei seguenti modi:

- Quando modifichi un parametro statico e salvi il gruppo di parametri del cluster di database, la modifica del parametro diventa effettiva al riavvio manuale di ogni istanza database sul cluster di database associato. Quando usi la Console di gestione AWS per modificare i valori dei parametri cluster database, utilizza sempre `pending-reboot` per `ApplyMethod`.
- Quando si modifica un parametro dinamico, per impostazione predefinita la modifica del parametro diventa immediatamente effettiva, senza richiedere il riavvio. Quando usi la console, utilizza sempre `immediate` per `ApplyMethod`. Per rinviare la modifica del parametro fino al riavvio delle istanze database in un cluster database associato, utilizza AWS CLI o l'API RDS. Quindi, imposta il valore `ApplyMethod` su `pending-reboot` per la modifica del parametro.

Per ulteriori informazioni sull'utilizzo della AWS CLI per modificare il valore di un parametro, consulta modify-db-cluster-parameter-group. Per ulteriori informazioni sull'utilizzo dell'API RDS per modificare il valore di un parametro, consulta ModifyDBClusterParameterGroup.

Se si modifica il gruppo di parametri cluster database associato a un cluster database, occorre riavviare le istanze database nel cluster database per applicare le modifiche a tutte le istanze database nel cluster database. Per determinare se le istanze database di un cluster di database devono essere riavviate per applicare le modifiche, esegui il seguente comando AWS CLI.

```
aws rds describe-db-clusters --db-cluster-identifier db_cluster_identifier
```

Controlla il valore `DBClusterParameterGroupStatus` per l'istanza database primaria nell'output. Se il valore è `pending-reboot`, riavvia l'istanza database del cluster di database.

Parametri statici e dinamici dell'istanza database

I parametri di istanza database sono statici o dinamici. Di seguito sono riportate le differenze:

- Quando modifichi un parametro statico e salvi il gruppo parametri del database, la modifica del parametro diventa effettiva al riavvio manuale delle istanze database associate. Per i parametri statici, la console utilizza sempre `pending-reboot` per `ApplyMethod`.
- Quando si modifica un parametro dinamico, per impostazione predefinita la modifica del parametro diventa immediatamente effettiva, senza richiedere il riavvio. Quando usi la Console di gestione AWS per modificare i valori dei parametri del cluster di database, utilizza sempre `immediate` per `ApplyMethod` per i parametri dinamici. Per rinviare la modifica del parametro fino al riavvio di un'istanza database associata, utilizza AWS CLI o l'API RDS. Quindi, imposta il valore `ApplyMethod` su `pending-reboot` per la modifica del parametro.

Per ulteriori informazioni sull'utilizzo della AWS CLI per modificare il valore di un parametro, consulta modify-db-parameter-group. Per ulteriori informazioni sull'utilizzo dell'API RDS per modificare il valore di un parametro, consulta ModifyDBParameterGroup.

Se l'istanza database non usa le modifiche più recenti apportate al gruppo di parametri database associato, la console mostra il gruppo di parametri database con lo stato pending-reboot. Questo stato non comporta il riavvio automatico durante la successiva finestra di manutenzione. Per applicare le ultime modifiche del parametro su quella istanza database, riavvia manualmente l'istanza database.

Parametri del set di caratteri

Prima di creare il cluster database imposta tutti i parametri correlati al set di caratteri o alla regola di confronto del database nel gruppo di parametri. prima di creare un database. In questo modo, il database predefinito e i nuovi database usano i valori della regola di confronto e del set di caratteri specificati. Se modifichi parametri di confronto o del set di caratteri, le modifiche dei parametri non vengono applicate a database esistenti.

Per alcuni motori database puoi modificare valori di confronto o del set di caratteri per un database esistente usando il comando `ALTER DATABASE`, ad esempio:

```
ALTER DATABASE database_name CHARACTER SET character_set_name COLLATE collation;
```

Per ulteriori informazioni su come modificare il set di caratteri o i valori di confronto relativi a un database, consulta la documentazione relativa al motore database.

Parametri e valori dei parametri supportati

Per determinare i parametri supportati per il motore del database, vedi i parametri nel gruppo di parametri database e il gruppo di parametri cluster database usato dall'istanza database o dal cluster database. Per ulteriori informazioni, consultare Visualizzazione dei valori dei parametri per un gruppo di parametri del database in Amazon Aurora e Visualizzazione dei valori dei parametri per un gruppo di parametri del cluster di database in Amazon Aurora.

In molti casi è possibile specificare valori interi e booleani per i parametri di database utilizzando espressioni, formule e funzioni. Le funzioni possono includere un'espressione logaritmica matematica. Tuttavia, non tutti i parametri supportano espressioni, formule e funzioni per i valori dei parametri. Per ulteriori informazioni, consulta Specifica dei parametri del database.

Per un Aurora Global Database, puoi specificare impostazioni di configurazione diverse per i singoli cluster Aurora. Assicurati che le impostazioni siano abbastanza simili da produrre un comportamento coerente se promuovi un cluster secondario a cluster primario. Ad esempio, utilizza le stesse impostazioni per fusi orari e set di caratteri su tutti i cluster di un Aurora Global Database.

Un'impostazione errata dei parametri in un gruppo di parametri può avere conseguenze negative impreviste, tra cui il peggioramento delle prestazioni e l'instabilità del sistema. Fai sempre attenzione quando modifichi i parametri database ed esegui il backup dei dati prima di modificare un gruppo di parametri. Prova le modifiche delle impostazioni del gruppo di parametri in un'istanza database o un cluster database di test prima di applicare le modifiche a un'istanza database o un cluster database di produzione.

Gruppi di parametri del cluster di database per i cluster di database Amazon Aurora

I cluster database Amazon Aurora utilizzano gruppi di parametri di cluster di database. Le sezioni seguenti descrivono la configurazione e la gestione dei gruppi di parametri del cluster di database.

Argomenti

- Parametri dell'istanza database e del cluster database di Amazon Aurora
- Creazione di un gruppo di parametri del cluster di database in Amazon Aurora
- Associazione di un gruppo di parametri del cluster di database a un cluster di database in Amazon Aurora

- Modifica dei parametri in un gruppo di parametri del cluster di database in Amazon Aurora
- Reimposta i parametri in un gruppo di parametri del cluster di database in Amazon Aurora
- Copia di un gruppo di parametri del cluster di database in Amazon Aurora
- Generazione di un elenco di gruppi di parametri del cluster di database in Amazon Aurora
- Visualizzazione dei valori dei parametri per un gruppo di parametri del cluster di database in Amazon Aurora
- Eliminazione di un gruppo di parametri del cluster di database in Amazon Aurora

Parametri dell'istanza database e del cluster database di Amazon Aurora

Aurora utilizza un sistema a due livelli di impostazioni di configurazione:

- I parametri in un gruppo di parametri cluster database si applicano a ogni istanza database in un cluster database. I dati vengono memorizzati nel sistema secondario di storage condiviso Aurora. Per questo motivo, tutti i parametri correlati al layout fisico dei dati della tabella devono essere gli stessi per tutte le istanze database in un cluster Aurora. Allo stesso modo, poiché le istanze database Aurora sono connesse dalla replica, tutti i parametri per le impostazioni di replica devono essere identici in un cluster Aurora.
- I parametri in un gruppo di parametri database si applicano a una sola istanza database in un cluster database Aurora. Questi parametri sono correlati ad aspetti quali l'utilizzo di memoria che puoi modificare tra le istanze database nello stesso cluster Aurora. Ad esempio, un cluster spesso contiene istanze database con classi di istanza AWS differenti.

Ogni cluster Aurora è associato a un gruppo di parametri del cluster database. Questo gruppo di parametri assegna i valori di default per ogni valore di configurazione per il motore del database corrispondente. Il gruppo di parametri del cluster include i valori di default per i parametri a livello di cluster e di istanza. Ogni istanza database all'interno di un cluster Aurora Serverless v2 o con provisioning eredita le impostazioni dal gruppo di parametri del cluster di database.

Ogni istanza database è associata anche a un gruppo parametri del database. I valori nel gruppo parametri del database possono sostituire i valori di default del gruppo di parametri del cluster. Ad esempio, se un'istanza in un cluster ha riscontrato problemi, è possibile assegnare a tale istanza un gruppo parametri del database personalizzato. Il gruppo di parametri personalizzati può avere impostazioni specifiche per i parametri relativi al debug o all'ottimizzazione delle prestazioni.

Quando crei un cluster o una nuova istanza database, Aurora assegna gruppi di parametri predefiniti in base al motore database e alla versione specificati. È invece possibile specificare gruppi di parametri personalizzati. Puoi creare autonomamente tali gruppi di parametri e puoi modificare i valori dei parametri. È possibile specificare questi gruppi di parametri personalizzati al momento della creazione. È inoltre possibile modificare un cluster o un'istanza database in un secondo momento per utilizzare un gruppo di parametri personalizzati.

Per le istanze Aurora Serverless v2 e con provisioning, eventuali valori di configurazione che si modificano nel gruppo di parametri del cluster di database sostituiscono i valori di default nel gruppo parametri del database. Se modifichi i valori corrispondenti nel gruppo di parametri di database, tali valori sostituiranno le impostazioni del gruppo di parametri del cluster database.

Ogni impostazione dei parametri di database che modifichi acquisisce la precedenza sui valori del gruppo di parametri del cluster database, anche se modifichi i parametri di configurazione riportandoli ai valori di default. È possibile anche vedere quali parametri vengono sostituiti utilizzando il comando AWS CLI describe-db-parameters o l'operazione API RDS DescribeDBParameters. Il campo `Source` contiene il valore `user` se hai modificato quel parametro. Per ripristinare uno o più parametri in modo che il valore del gruppo di parametri del cluster di database abbia la precedenza, utilizza il comando AWS CLI reset-db-parameter-group o l'operazione API RDS ResetDBParameterGroup.

I parametri dell'istanza database e del cluster database disponibili in Aurora variano in base alla compatibilità del motore del database.

Motore del database	Parametri
Aurora MySQL	Per informazioni, consulta Parametri di configurazione Aurora MySQL. Per i cluster Aurora Serverless, vedi i dettagli aggiuntivi in Uso di gruppi di parametri per Aurora Serverless v2 e Gruppi di parametri per Aurora Serverless v1.
Aurora PostgreSQL	Per informazioni, consulta Amazon Aurora PostgreSQL parametri. Per i cluster Aurora Serverless, vedi i dettagli aggiuntivi in Uso di gruppi di parametri per Aurora Serverless v2 e Gruppi di parametri per Aurora Serverless v1.

> **Note**
>
> I cluster Aurora Serverless v1 hanno solo gruppi di parametri del cluster di database, non gruppi parametri del database. Per i cluster Aurora Serverless v2, apporti tutte le modifiche ai parametri personalizzati nel gruppo di parametri del cluster di database.
> Aurora Serverless v2 utilizza sia gruppi di parametri del cluster di database che gruppi parametri del database. Con Aurora Serverless v2, è possibile modificare quasi tutti i parametri di configurazione. Aurora Serverless v2 sostituisce le impostazioni di alcuni parametri di configurazione relativi alla capacità in modo che il carico di lavoro non venga interrotto quando le istanze Aurora Serverless v2 si riducono.
> Per ulteriori informazioni sulle impostazioni di configurazione di Aurora Serverless e sulle impostazioni che puoi modificare, consulta Uso di gruppi di parametri per Aurora Serverless v2 e Gruppi di parametri per Aurora Serverless v1.

Creazione di un gruppo di parametri del cluster di database in Amazon Aurora

Puoi creare un nuovo gruppo di parametri cluster database tramite la Console di gestione AWS, AWS CLI o l'API di RDS.

Dopo aver creato un gruppo di parametri del cluster database, devi attendere almeno 5 minuti prima di creare un cluster database che utilizza tale gruppo. In questo modo, Amazon RDS può completare l'operazione di creazione del gruppo di parametri prima che tale gruppo venga usato dal nuovo cluster database. Puoi utilizzare la pagina Parameter Groups (Gruppi parametri) nella console Amazon RDS o il comando describe-db-cluster-parameters per verificare l'avvenuta creazione del gruppo di parametri cluster di database.

Le seguenti limitazioni si applicano al nome del gruppo di parametri del·cluster database:

- Il nome deve contenere da 1 a 255 lettere, numeri o trattini.

  I nomi del gruppo di parametri predefiniti possono includere un punto, ad esempio `default.aurora-mysql5.7`. Tuttavia, i nomi del gruppo di parametri personalizzati non possono includere un punto.
- Il primo carattere deve essere una lettera.
- Il nome non può terminare con un trattino o contenere due trattini consecutivi.

Console

Per creare un gruppo di parametri del cluster database

1. Accedi alla Console di gestione AWS e apri la console Amazon RDS all'indirizzo https://console.aws.amazon.com/rds/.
2. Nel pannello di navigazione, scegli Parameter groups (Gruppi di parametri).
3. Scegli Create parameter group (Crea gruppo di parametri).
4. In Nome gruppo di parametri inserisci il nome del nuovo gruppo di parametri del cluster di database.
5. In Descrizione inserisci una descrizione per il nuovo gruppo di parametri del cluster di database.
6. In Tipo di motore, scegli il tuo motore di database.
7. Nell'elenco Famiglia gruppo di parametri seleziona una famiglia del gruppo di parametri del database.
8. Scegli Crea.

AWS CLI

Per creare un gruppo di parametri del cluster di database, utilizza il comando `create-db-cluster-parameter-group` della AWS CLI.

L'esempio seguente crea un gruppo di parametri cluster di database denominato mydbclusterparametergroup per Aurora MySQL versione 5.7 con la descrizione "My new cluster parameter group (Il mio nuovo gruppo di parametri cluster)".

Includi i parametri obbligatori seguenti:

- `--db-cluster-parameter-group-name`
- `--db-parameter-group-family`
- `--description`

Per elencare tutte le famiglie del gruppo di parametri disponibili, usa il comando seguente:

```
aws rds describe-db-engine-versions --query "DBEngineVersions[].DBParameterGroupFamily"
```

> **Note**
>
> L'output contiene duplicati.

Example

Per Linux, macOS o Unix:

```
aws rds create-db-cluster-parameter-group \
    --db-cluster-parameter-group-name mydbclusterparametergroup \
    --db-parameter-group-family aurora-mysql5.7 \
    --description "My new cluster parameter group"
```

Per Windows:

```
aws rds create-db-cluster-parameter-group ^
    --db-cluster-parameter-group-name mydbclusterparametergroup ^
    --db-parameter-group-family aurora-mysql5.7 ^
    --description "My new cluster parameter group"
```

Questo comando genera un output simile al seguente:

```
{
    "DBClusterParameterGroup": {
        "DBClusterParameterGroupName": "mydbclusterparametergroup",
        "DBParameterGroupFamily": "aurora-mysql5.7",
        "Description": "My new cluster parameter group",
        "DBClusterParameterGroupArn": "arn:aws:rds:us-east-1:123456789012:cluster-
pg:mydbclusterparametergroup"
    }
}
```

API RDS

Per creare un gruppo di parametri del cluster di database, utilizza l'operazione `CreateDBClusterParameterGroup` dell'API RDS.

Includi i parametri obbligatori seguenti:

- `DBClusterParameterGroupName`

- `DBParameterGroupFamily`
- `Description`

Associazione di un gruppo di parametri del cluster di database a un cluster di database in Amazon Aurora

Puoi creare gruppi di parametri del cluster database personalizzati con impostazioni personalizzate. Puoi associare un gruppo di parametri del cluster database a un cluster database utilizzando la Console di gestione AWS, la AWS CLI o l'API RDS. Ciò è possibile quando si crea o si modifica un cluster database.

Per ulteriori informazioni sulla creazione di un gruppo di parametri del cluster database, consulta Creazione di un gruppo di parametri del cluster di database in Amazon Aurora. Per ulteriori informazioni sulla creazione di un cluster database, consulta Creazione di un cluster database Amazon Aurora. Per ulteriori informazioni sulla modifica di un cluster database, consulta Modifica di un cluster database Amazon Aurora.

> **Note**
>
> Per Aurora PostgreSQL 15.2, 14.7, 13.10, 12.14 e tutte le 11 versioni, quando modifichi il gruppo di parametri del cluster di database associato a un cluster di database, riavvia ogni istanza di replica per applicare le modifiche.
> Per determinare se l'istanza database primaria di un cluster database deve essere riavviata per applicare le modifiche, emettere il seguente comando AWS CLI:
> `aws rds describe-db-clusters --db-cluster-identifier` *`db_cluster_identifier`*
> Controlla il valore `DBClusterParameterGroupStatus` per l'istanza database primaria nell'output. Se il valore è `pending-reboot`, riavvia l'istanza database primaria del cluster di database.

Console

Per associare un gruppo di parametri del cluster database a un cluster database

1. Accedi alla Console di gestione AWS e apri la console Amazon RDS all'indirizzo https://console.aws.amazon.com/rds/.

2. Nel riquadro di navigazione, scegliere Databases (Database), quindi selezionare il cluster di database che si desidera modificare.
3. Scegliere Modify (Modifica). Viene visualizzata la pagina Modify DB cluster (Modifica cluster di database).
4. Modifica l'impostazione del gruppo di parametri del cluster database .
5. Scegliere Continue (Continua) e controllare il riepilogo delle modifiche.

   La modifica viene applicata immediatamente indipendentemente dall'impostazione Programmazione delle modifiche .
6. Nella pagina di conferma esaminare le modifiche. Se sono corrette, selezionare Modify cluster (Modifica cluster) per salvare le modifiche.

   In alternativa, scegliere Back (Indietro) per cambiare le modifiche o Cancel (Annulla) per annullare le modifiche.

AWS CLI

Per associare un gruppo di parametri del cluster di database a un cluster di database, utilizza il comando `modify-db-cluster` della AWS CLI con le seguenti opzioni:

- `--db-cluster-name`
- `--db-cluster-parameter-group-name`

Nell'esempio seguente il gruppo di parametri database `mydbclpg` viene associato al cluster database `mydbcluster`.

Example

Per Linux, macOS o Unix:

```
aws rds modify-db-cluster \
    --db-cluster-identifier mydbcluster \
    --db-cluster-parameter-group-name mydbclpg
```

Per Windows:

```
aws rds modify-db-cluster ^
    --db-cluster-identifier mydbcluster ^
```

```
    --db-cluster-parameter-group-name mydbclpg
```

API RDS

Per associare un gruppo di parametri del cluster database a un cluster database, utilizza l'operazione `ModifyDBCluster` dell'API RDS con i seguenti parametri:

- `DBClusterIdentifier`
- `DBClusterParameterGroupName`

Modifica dei parametri in un gruppo di parametri del cluster di database in Amazon Aurora

Non puoi modificare i valori di parametri in un gruppo di parametri cluster database creato dal cliente. Non puoi modificare i valori dei parametri in un gruppo di parametri cluster database predefinito. Le modifiche ai parametri in un gruppo di parametri cluster database creato dal cliente vengono applicate a tutti i cluster database associati al gruppo di parametri del cluster database.

Console

Per modificare un gruppo di parametri del cluster database

1. Accedi alla Console di gestione AWS e apri la console Amazon RDS all'indirizzo https://console.aws.amazon.com/rds/.
2. Nel riquadro di navigazione scegliere Parameter groups (Gruppi di parametri).
3. Nell'elenco, selezionare il gruppo di parametri da modificare.
4. Per Parameter group actions (Operazioni del gruppo di parametri), scegliere Edit (Modifica).
5. Modificare i valori dei parametri desiderati. È possibile scorrere i parametri usando i tasti freccia in alto a destra nella finestra di dialogo.

   Non è possibile modificare valori nel gruppo di parametri predefinito.
6. Scegli Save changes (Salva modifiche).
7. Riavvia l'istanza database primaria (in scrittura) nel cluster per applicare le modifiche.
8. Quindi riavvia le istanze database di lettura per applicare le modifiche.

   Se non riavvii le istanze database, un'operazione di failover potrebbe richiedere più tempo del normale.

AWS CLI

Per modificare un gruppo di parametri del cluster di database, utilizza il comando `modify-db-cluster-parameter-group` della AWS CLI con i parametri obbligatori seguenti:

- `--db-cluster-parameter-group-name`
- `--parameters`

L'esempio seguente modifica i valori `server_audit_logging` e `server_audit_logs_upload` nel gruppo di parametri cluster database denominato mydbclusterparametergroup.

Example

Per Linux, macOS o Unix:

```
aws rds modify-db-cluster-parameter-group \
    --db-cluster-parameter-group-name mydbclusterparametergroup \
    --parameters
 "ParameterName=server_audit_logging,ParameterValue=1,ApplyMethod=immediate" \

 "ParameterName=server_audit_logs_upload,ParameterValue=1,ApplyMethod=immediate"
```

Per Windows:

```
aws rds modify-db-cluster-parameter-group ^
    --db-cluster-parameter-group-name mydbclusterparametergroup ^
    --parameters
 "ParameterName=server_audit_logging,ParameterValue=1,ApplyMethod=immediate" ^

 "ParameterName=server_audit_logs_upload,ParameterValue=1,ApplyMethod=immediate"
```

Il comando genera un output simile al seguente:

```
DBCLUSTERPARAMETERGROUP  mydbclusterparametergroup
```

API RDS

Per modificare un gruppo di parametri cluster database, utilizza l'operazione `ModifyDBClusterParameterGroup` dell'API RDS con i parametri obbligatori seguenti:

- `DBClusterParameterGroupName`
- `Parameters`

Reimposta i parametri in un gruppo di parametri del cluster di database in Amazon Aurora

È possibile reimpostare i parametri ai valori predefiniti in un gruppo di parametri di cluster del database creato dal cliente. Le modifiche ai parametri in un gruppo di parametri cluster database creato dal cliente vengono applicate a tutti i cluster database associati al gruppo di parametri cluster database.

> **Note**
>
> In un gruppo di parametri cluster di database predefinito, i parametri vengono sempre impostati sui valori di default.

Console

Per ripristinare i valori predefiniti dei parametri di un gruppo di parametri cluster database

1. Accedi alla Console di gestione AWS e apri la console Amazon RDS all'indirizzo https://console.aws.amazon.com/rds/.
2. Nel riquadro di navigazione scegliere Parameter groups (Gruppi di parametri).
3. Nell'elenco, scegliere il gruppo di parametri.
4. Per Parameter group actions (Operazioni del gruppo di parametri), scegliere Edit (Modifica).
5. Scegliere i parametri per i quali si desidera ripristinare i valori predefiniti. È possibile scorrere i parametri usando i tasti freccia in alto a destra nella finestra di dialogo.

   Non è possibile reimpostare valori in un gruppo di parametri predefinito.
6. Scegli Reimposta , quindi conferma selezionando Ripristina parametri.
7. Riavvia l'istanza database primaria nel cluster di database per applicare le modifiche a tutte le istanze database nel cluster di database.

AWS CLI

Per ripristinare i valori predefiniti dei parametri di un gruppo di parametri del cluster di database, utilizza il comando `reset-db-cluster-parameter-group` della AWS CLI con la seguente opzione obbligatoria: `--db-cluster-parameter-group-name`.

Per reimpostare tutti i parametri nel gruppo di parametri del cluster di database, specificare l'opzione `--reset-all-parameters`. Per reimpostare parametri specifici, specifica l'opzione `--parameters`.

Nell'esempio seguente tutti i parametri del gruppo di parametri del database denominato mydbparametergroup vengono reimpostati sui valori predefiniti.

Example

Per Linux, macOS o Unix:

```
aws rds reset-db-cluster-parameter-group \
    --db-cluster-parameter-group-name mydbparametergroup \
    --reset-all-parameters
```

Per Windows:

```
aws rds reset-db-cluster-parameter-group ^
    --db-cluster-parameter-group-name mydbparametergroup ^
    --reset-all-parameters
```

L'esempio seguente reimposta i valori `server_audit_logging` e `server_audit_logs_upload` nel gruppo di parametri cluster database denominato mydbclusterparametergroup.

Example

Per Linux, macOS o Unix:

```
aws rds reset-db-cluster-parameter-group \
    --db-cluster-parameter-group-name mydbclusterparametergroup \
    --parameters "ParameterName=server_audit_logging,ApplyMethod=immediate" \
                 "ParameterName=server_audit_logs_upload,ApplyMethod=immediate"
```

Per Windows:

```
aws rds reset-db-cluster-parameter-group ^
```

```
    --db-cluster-parameter-group-name mydbclusterparametergroup ^
    --parameters
 "ParameterName=server_audit_logging,ParameterValue=1,ApplyMethod=immediate" ^

 "ParameterName=server_audit_logs_upload,ParameterValue=1,ApplyMethod=immediate"
```

Il comando genera un output simile al seguente:

```
DBClusterParameterGroupName  mydbclusterparametergroup
```

API RDS

Per ripristinare i valori predefiniti dei parametri di un gruppo di parametri del cluster di database, utilizzare il comando RDS API `ResetDBClusterParameterGroup` con il seguente parametro obbligatorio: `DBClusterParameterGroupName`.

Per reimpostare tutti i parametri nel gruppo di parametri del cluster di database, impostare il parametro `ResetAllParameters` su `true`. Per reimpostare parametri specifici, specifica il parametro `Parameters`.

Copia di un gruppo di parametri del cluster di database in Amazon Aurora

Puoi copiare i gruppi di parametri cluster database personalizzati che hai creato. La copia di un gruppo di parametri è una soluzione pratica quando hai già creato un gruppo di parametri cluster database e vuoi includere la maggior parte dei parametri e valori personalizzati dal gruppo in un nuovo gruppo di parametri cluster database. Puoi copiare un gruppo di parametri cluster database utilizzando il comando AWS CLI copy-db-cluster-parameter-group o l'operazione API RDS CopyDBClusterParameterGroup.

Dopo aver copiato un gruppo di parametri del cluster database, devi attendere almeno 5 minuti prima di creare un cluster database che utilizza tale gruppo. In questo modo, Amazon RDS può completare l'operazione di copia del gruppo di parametri prima che tale gruppo venga usato dal nuovo cluster database. Puoi utilizzare la pagina Parameter Groups (Gruppi parametri) nella console Amazon RDS o il comando describe-db-cluster-parameters per verificare l'avvenuta creazione del gruppo di parametri cluster di database.

> **Note**
>
> Non puoi copiare un gruppo di parametri predefinito. Tuttavia, puoi creare un nuovo gruppo di parametri basato su un gruppo di parametri predefinito.

Non è possibile copiare un gruppo di parametri del cluster database in un Account AWS o una Regione AWS diversi.

Console

Per copiare un gruppo di parametri cluster database

1. Accedi alla Console di gestione AWS e apri la console Amazon RDS all'indirizzo https://console.aws.amazon.com/rds/.
2. Nel riquadro di navigazione scegliere Parameter groups (Gruppi di parametri).
3. Nell'elenco, scegliere il gruppo di parametri personalizzato da copiare.
4. Per Parameter group actions (Operazioni del gruppo di parametri), scegliere Copy (Copia).
5. In New DB parameter group identifier (Identificatore nuovo gruppo di parametri database) immettere un nome per il nuovo gruppo di parametri.
6. Nella casella Description (Descrizione), immettere una descrizione per il nuovo gruppo di parametri.
7. Scegliere Copy (Copia).

AWS CLI

Per copiare un gruppo di parametri del cluster di database, utilizza il comando `copy-db-cluster-parameter-group` della AWS CLI con i parametri obbligatori seguenti:

- `--source-db-cluster-parameter-group-identifier`
- `--target-db-cluster-parameter-group-identifier`
- `--target-db-cluster-parameter-group-description`

L'esempio seguente crea un nuovo gruppo di parametri cluster database denominato `mygroup2`, che è una copia del gruppo di parametri cluster database `mygroup1`.

Example

Per Linux, macOS o Unix:

```
aws rds copy-db-cluster-parameter-group \
    --source-db-cluster-parameter-group-identifier mygroup1 \
```

```
    --target-db-cluster-parameter-group-identifier mygroup2 \
    --target-db-cluster-parameter-group-description "DB parameter group 2"
```

Per Windows:

```
aws rds copy-db-cluster-parameter-group ^
    --source-db-cluster-parameter-group-identifier mygroup1 ^
    --target-db-cluster-parameter-group-identifier mygroup2 ^
    --target-db-cluster-parameter-group-description "DB parameter group 2"
```

API RDS

Per copiare un gruppo di parametri cluster database, utilizza l'operazione API RDS `CopyDBClusterParameterGroup` con i parametri obbligatori seguenti:

- `SourceDBClusterParameterGroupIdentifier`
- `TargetDBClusterParameterGroupIdentifier`
- `TargetDBClusterParameterGroupDescription`

Generazione di un elenco di gruppi di parametri del cluster di database in Amazon Aurora

Puoi elencare i gruppi di parametri cluster database creati per il tuo account AWS.

> **Note**
>
> I gruppi di parametri predefiniti vengono creati automaticamente da un modello di parametro predefinito quando crei un cluster database per un motore e una versione di database specifici. Questi gruppi di parametri predefiniti contengono le impostazioni dei parametri preferite e non possono essere modificati. Quando crei un gruppo di parametri personalizzato, puoi modificare le impostazioni dei parametri.

Console

Per elencare tutti i gruppi di parametri del cluster database per un account AWS

1. Accedi alla Console di gestione AWS e apri la console Amazon RDS all'indirizzo https://console.aws.amazon.com/rds/.

2. Nel riquadro di navigazione scegliere Parameter groups (Gruppi di parametri).

   Il gruppo di parametri cluster database viene visualizzato nell'elenco con Type (Tipo) impostato su DB cluster parameter group (Gruppo di parametri cluster database).

AWS CLI

Per elencare tutti i gruppi di parametri del cluster di database per un account AWS, utilizza il comando `describe-db-cluster-parameter-groups` della AWS CLI.

Example

Nell'esempio seguente sono elencati tutti i gruppi di parametri del cluster database disponibili per un account AWS.

```
aws rds describe-db-cluster-parameter-groups
```

L'esempio seguente descrive il gruppo di parametri mydbclusterparametergroup.

Per Linux, macOS o Unix:

```
aws rds describe-db-cluster-parameter-groups \
    --db-cluster-parameter-group-name mydbclusterparametergroup
```

Per Windows:

```
aws rds describe-db-cluster-parameter-groups ^
    --db-cluster-parameter-group-name mydbclusterparametergroup
```

Questo comando restituisce una risposta simile alla seguente:

```
{
    "DBClusterParameterGroups": [
        {
            "DBClusterParameterGroupName": "mydbclusterparametergroup",
            "DBParameterGroupFamily": "aurora-mysql5.7",
            "Description": "My new cluster parameter group",
            "DBClusterParameterGroupArn": "arn:aws:rds:us-east-1:123456789012:cluster-
pg:mydbclusterparametergroup"
        }
```

```
    ]
}
```

API RDS

Per elencare tutti i gruppi di parametri cluster database per un account AWS, utilizza l'operazione `DescribeDBClusterParameterGroups` dell'API RDS.

Visualizzazione dei valori dei parametri per un gruppo di parametri del cluster di database in Amazon Aurora

Puoi ottenere un elenco di tutti i parametri in un gruppo di parametri del cluster database e dei rispettivi valori.

Console

Per visualizzare i valori dei parametri per un gruppo di parametri del cluster database

1. Accedi alla Console di gestione AWS e apri la console Amazon RDS all'indirizzo https://console.aws.amazon.com/rds/.
2. Nel riquadro di navigazione scegliere Parameter groups (Gruppi di parametri).

   Il gruppo di parametri cluster database viene visualizzato nell'elenco con Type (Tipo) impostato su DB cluster parameter group (Gruppo di parametri cluster database).
3. Scegliere il nome del gruppo di parametri cluster di database per visualizzare il relativo elenco di parametri.

AWS CLI

Per visualizzare i valori dei parametri per un gruppo di parametri cluster di database, utilizza il comando `describe-db-cluster-parameters` della AWS CLI con il parametro obbligatorio seguente.

- `--db-cluster-parameter-group-name`

Example

L'esempio seguente elenca i parametri e i valori dei parametri per un gruppo di parametri cluster di database denominato mydbclusterparametergroup, in formato JSON.

Questo comando restituisce una risposta simile alla seguente:

```
aws rds describe-db-cluster-parameters --db-cluster-parameter-group-
name mydbclusterparametergroup
```

```
{
    "Parameters": [
        {
            "ParameterName": "allow-suspicious-udfs",
            "Description": "Controls whether user-defined functions that have only an
 xxx symbol for the main function can be loaded",
            "Source": "engine-default",
            "ApplyType": "static",
            "DataType": "boolean",
            "AllowedValues": "0,1",
            "IsModifiable": false,
            "ApplyMethod": "pending-reboot",
            "SupportedEngineModes": [
                "provisioned"
            ]
        },
        {
            "ParameterName": "aurora_binlog_read_buffer_size",
            "ParameterValue": "5242880",
            "Description": "Read buffer size used by master dump thread when the switch
 aurora_binlog_use_large_read_buffer is ON.",
            "Source": "engine-default",
            "ApplyType": "dynamic",
            "DataType": "integer",
            "AllowedValues": "8192-536870912",
            "IsModifiable": true,
            "ApplyMethod": "pending-reboot",
            "SupportedEngineModes": [
                "provisioned"
            ]
        },

...
```

API RDS

Per visualizzare i valori dei parametri per un gruppo di parametri cluster database, utilizza il comando API RDS `DescribeDBClusterParameters` con il parametro obbligatorio seguente.

- `DBClusterParameterGroupName`

In alcuni casi, i valori consentiti per un parametro non vengono visualizzati. Questi sono sempre parametri in cui l'origine è l'impostazione predefinita del motore di database.

Per visualizzare i valori di questi parametri, puoi eseguire le seguenti istruzioni SQL:

- MySQL:

```
-- Show the value of a particular parameter
mysql$ SHOW VARIABLES LIKE '%parameter_name%';

-- Show the values of all parameters
mysql$ SHOW VARIABLES;
```

- PostgreSQL:

```
-- Show the value of a particular parameter
postgresql=> SHOW parameter_name;

-- Show the values of all parameters
postgresql=> SHOW ALL;
```

Eliminazione di un gruppo di parametri del cluster di database in Amazon Aurora

Puoi eliminare un gruppo di parametri del cluster di database tramite la Console di gestione AWS, AWS CLI o l'API di RDS. Un gruppo di parametri del cluster di database può essere eliminato solo se non è associato a un cluster di database.

Console

Per eliminare gruppi di parametri

1. Accedi alla Console di gestione AWS e apri la console Amazon RDS all'indirizzo https://console.aws.amazon.com/rds/.
2. Nel riquadro di navigazione scegliere Parameter groups (Gruppi di parametri).

   I gruppi di parametri vengono visualizzati in un elenco.
3. Scegli il nome dei gruppi di parametri del cluster di database da eliminare.

4. Scegli Operazioni, quindi Elimina.
5. Controlla i nomi dei gruppi di parametri, quindi scegli Elimina.

AWS CLI

Per eliminare un gruppo di parametri del cluster di database, utilizza il comando `delete-db-cluster-parameter-group` della AWS CLI con il parametro obbligatorio seguente.

- `--db-parameter-group-name`

Example

L'esempio seguente elimina un gruppo di parametri del cluster di database denominato mydbparametergroup.

```
aws rds delete-db-cluster-parameter-group --db-parameter-group-name mydbparametergroup
```

API RDS

Per eliminare un gruppo di parametri del cluster di database, utilizza il comando `DeleteDBClusterParameterGroup` dell'API di RDS con il parametro obbligatorio seguente.

- `DBParameterGroupName`

Gruppi di parametri database per istanze database Amazon Aurora

Le istanze database utilizzano gruppi di parametri database. Le sezioni seguenti descrivono la configurazione e la gestione dei gruppi di parametri dell'istanza database.

Argomenti

- Creazione di un gruppo di parametri database in Amazon Aurora
- Associazione di un gruppo di parametri database a un'istanza database in Amazon Aurora
- Modifica dei parametri in un gruppo di parametri database in Amazon Aurora
- Reimpostazione dei parametri in un gruppo di parametri database sui valori predefiniti in Amazon Aurora
- Copia di un gruppo di parametri database in Amazon Aurora

- Generazione di un elenco di gruppi di parametri del database in Amazon Aurora
- Visualizzazione dei valori dei parametri per un gruppo di parametri del database in Amazon Aurora
- Eliminazione di un gruppo di parametri database in Amazon Aurora

Creazione di un gruppo di parametri database in Amazon Aurora

Puoi creare un nuovo gruppo di parametri DB utilizzando l' Console di gestione AWS AWS CLI, la o l'API RDS.

Le seguenti limitazioni si applicano al nome del gruppo di parametri database:

- Il nome deve contenere da 1 a 255 lettere, numeri o trattini.

  I nomi del gruppo di parametri predefiniti possono includere un punto, ad esempio `default.mysql8.0`. Tuttavia, i nomi del gruppo di parametri personalizzati non possono includere un punto.
- Il primo carattere deve essere una lettera.
- Il nome non può terminare con un trattino o contenere due trattini consecutivi.

Console

Per creare un gruppo di parametri del database

1. Accedi a Console di gestione AWS e apri la console Amazon RDS all'indirizzo https://console.aws.amazon.com/rds/.
2. Nel pannello di navigazione, scegli Parameter groups (Gruppi di parametri).
3. Scegli Create parameter group (Crea gruppo di parametri).
4. In Nome del gruppo di parametri, digita il nome del nuovo gruppo di parametri del cluster di database.
5. In Descrizione, digita una descrizione per il nuovo gruppo di parametri del cluster di database.
6. In Tipo di motore, scegli il tuo motore database.
7. In Famiglia del gruppo di parametri scegli una famiglia del gruppo di parametri database.
8. In Tipo, se applicabile, scegli Gruppo di parametri database.
9. Scegli Create (Crea).

AWS CLI

Per creare un gruppo di parametri DB, usa il AWS CLI `create-db-parameter-group`comando. L'esempio seguente crea un gruppo di parametri database denominato mydbparametergroup per MySQL versione 8.0 con la descrizione "My new parameter group (Il mio nuovo gruppo di parametri)".

Includi i parametri obbligatori seguenti:

- `--db-parameter-group-name`
- `--db-parameter-group-family`
- `--description`

Per elencare tutte le famiglie del gruppo di parametri disponibili, usa il comando seguente:

```
aws rds describe-db-engine-versions --query "DBEngineVersions[].DBParameterGroupFamily"
```

> **Note**
>
> L'output contiene duplicati.

Example

Per Linux, macOS o Unix:

```
aws rds create-db-parameter-group \
    --db-parameter-group-name mydbparametergroup \
    --db-parameter-group-family aurora-mysql5.7 \
    --description "My new parameter group"
```

Per Windows:

```
aws rds create-db-parameter-group ^
    --db-parameter-group-name mydbparametergroup ^
    --db-parameter-group-family aurora-mysql5.7 ^
    --description "My new parameter group"
```

Questo comando genera un output simile al seguente:

```
DBPARAMETERGROUP  mydbparametergroup  aurora-mysql5.7  My new parameter group
```

API RDS

Per creare un gruppo di parametri database, utilizzare l'operazione API RDS `CreateDBParameterGroup`.

Includi i parametri obbligatori seguenti:

- `DBParameterGroupName`
- `DBParameterGroupFamily`
- `Description`

Associazione di un gruppo di parametri database a un'istanza database in Amazon Aurora

Puoi creare i tuoi gruppi di parametri database con impostazioni personalizzate. Puoi associare un gruppo di parametri DB a un'istanza DB utilizzando l' Console di gestione AWS API AWS CLI, the o RDS. Ciò è possibile quando crei o modifichi un'istanza database.

Per ulteriori informazioni sulla creazione di un gruppo di parametri del database, consulta Creazione di un gruppo di parametri database in Amazon Aurora. Per ulteriori informazioni sulla modifica di un'istanza database, consulta Modifica di un'istanza database in un cluster database.

> **Note**
>
> Quando si associa un nuovo gruppo parametri del database a un'istanza database, i parametri statici e dinamici modificati vengono applicati solo dopo il riavvio dell'istanza database. Tuttavia, se modifichi i parametri dinamici nel gruppo di parametri database associato all'istanza database, tali modifiche vengono applicate immediatamente senza eseguire il riavvio.

Console

Per associare un gruppo di parametri del database a un'istanza database

1. Accedi a Console di gestione AWS e apri la console Amazon RDS all'indirizzo https://console.aws.amazon.com/rds/.

2. Nel riquadro di navigazione, scegliere Databases (Database) e selezionare l'istanza database da modificare.
3. Scegliere Modify (Modifica). Viene visualizzata la pagina Modify DB Instance (Modifica istanza database).
4. Modifica l'impostazione del gruppo di parametri database .
5. Scegliere Continue (Continua) e controllare il riepilogo delle modifiche.
6. (Facoltativo) Scegliere Applica immediatamente per applicare immediatamente le modifiche. In alcuni casi, la chiusura di questa opzione può causare un'interruzione.
7. Nella pagina di conferma esaminare le modifiche. Se sono corrette, seleziona Modifica istanza database per salvare le modifiche.

   Oppure scegliere Back (Indietro) per cambiare le modifiche o Cancel (Annulla) per annullare le modifiche.

AWS CLI

Per associare un gruppo di parametri DB a un'istanza DB, usa il AWS CLI `modify-db-instance`comando con le seguenti opzioni:

- `--db-instance-identifier`
- `--db-parameter-group-name`

Nell'esempio seguente il gruppo di parametri database `mydbpg` viene associato all'istanza database `database-1`. Le modifiche vengono applicate immediatamente tramite `--apply-immediately`. Utilizza `--no-apply-immediately` per applicare le modifiche durante la successiva finestra di manutenzione.

Example

Per Linux, macOS o Unix:

```
aws rds modify-db-instance \
    --db-instance-identifier database-1 \
    --db-parameter-group-name mydbpg \
    --apply-immediately
```

Per Windows:

```
aws rds modify-db-instance ^
    --db-instance-identifier database-1 ^
    --db-parameter-group-name mydbpg ^
    --apply-immediately
```

API RDS

Per associare un gruppo parametri del database a un'istanza database, utilizza l'operazione `ModifyDBInstance` dell'API RDS con i seguenti parametri:

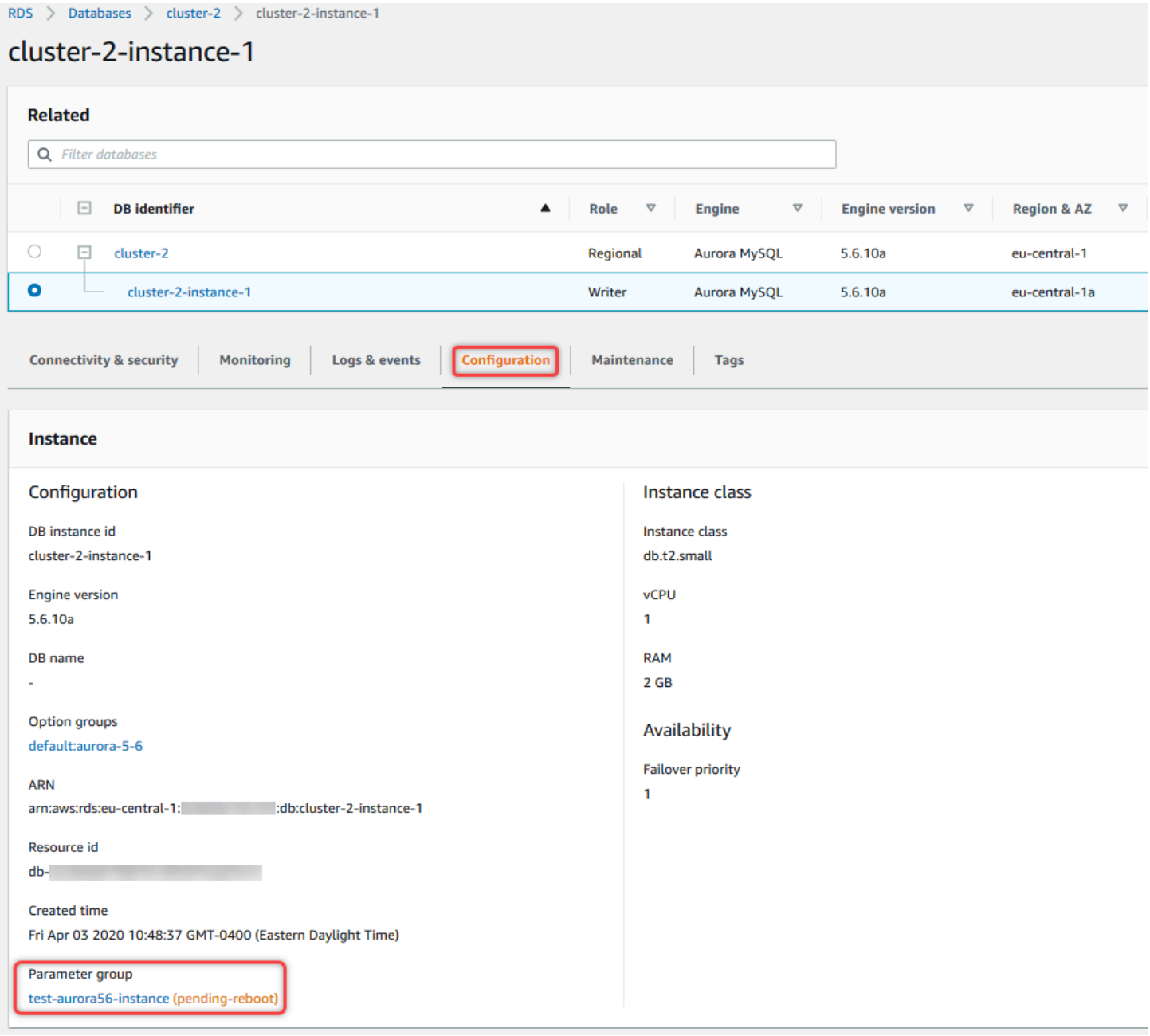
- `DBInstanceName`
- `DBParameterGroupName`

Modifica dei parametri in un gruppo di parametri database in Amazon Aurora

Puoi modificare i valori dei parametri in un gruppo di parametri database creato dal cliente, ma non puoi modificare i valori dei parametri in un gruppo di parametri database predefinito. Le modifiche ai parametri in un gruppo di parametri database creato dal cliente vengono applicate a tutte le istanze database associate al gruppo di parametri database.

Esistono due tipi di parametri: parametri dinamici e parametri statici. Le modifiche ai parametri dinamici vengono applicate all'istanza DB immediatamente senza riavvio. Le modifiche ai parametri statici vengono applicate solo dopo il riavvio dell'istanza DB.

La console RDS mostra lo stato del gruppo di parametri database associato a un'istanza database nella scheda Configuration (Configurazione). Ad esempio, se l'istanza database non usa le modifiche più recenti apportate al gruppo di parametri database associato, la console RDS mostra il gruppo di parametri database con stato pending-reboot (riavvio in attesa). Per applicare le ultime modifiche del parametro su quella istanza database, riavviare manualmente l'istanza database.



Console

Per modificare i parametri in un gruppo di parametri del database

1. Accedi a Console di gestione AWS e apri la console Amazon RDS all'indirizzo https://console.aws.amazon.com/rds/.
2. Nel riquadro di navigazione scegliere Parameter groups (Gruppi di parametri).
3. Nell'elenco, seleziona il nome del gruppo di parametri da modificare.
4. In Azioni del gruppo di parametri), scegli Modifica.

5. Modificare i valori dei parametri desiderati. È possibile scorrere i parametri usando i tasti freccia in alto a destra nella finestra di dialogo.

   Non è possibile modificare valori nel gruppo di parametri predefinito.

6. Scegli Save changes (Salva modifiche).

AWS CLI

Per modificare un gruppo di parametri DB, usa il AWS CLI `modify-db-parameter-group`comando con le seguenti opzioni obbligatorie:

- `--db-parameter-group-name`
- `--parameters`

L'esempio seguente modifica i valori `max_connections` e `max_allowed_packet` nel gruppo di parametri database denominato mydbparametergroup.

Example

Per Linux, macOS o Unix:

```
aws rds modify-db-parameter-group \
    --db-parameter-group-name mydbparametergroup \
    --parameters
 "ParameterName=max_connections,ParameterValue=250,ApplyMethod=immediate" \

 "ParameterName=max_allowed_packet,ParameterValue=1024,ApplyMethod=immediate"
```

Per Windows:

```
aws rds modify-db-parameter-group ^
    --db-parameter-group-name mydbparametergroup ^
    --parameters
 "ParameterName=max_connections,ParameterValue=250,ApplyMethod=immediate" ^

 "ParameterName=max_allowed_packet,ParameterValue=1024,ApplyMethod=immediate"
```

Il comando genera un output simile al seguente:

```
DBPARAMETERGROUP  mydbparametergroup
```

API RDS

Per modificare un gruppo parametri del database, utilizza l'operazione `ModifyDBParameterGroup` dell'API RDS con i seguenti parametri obbligatori:

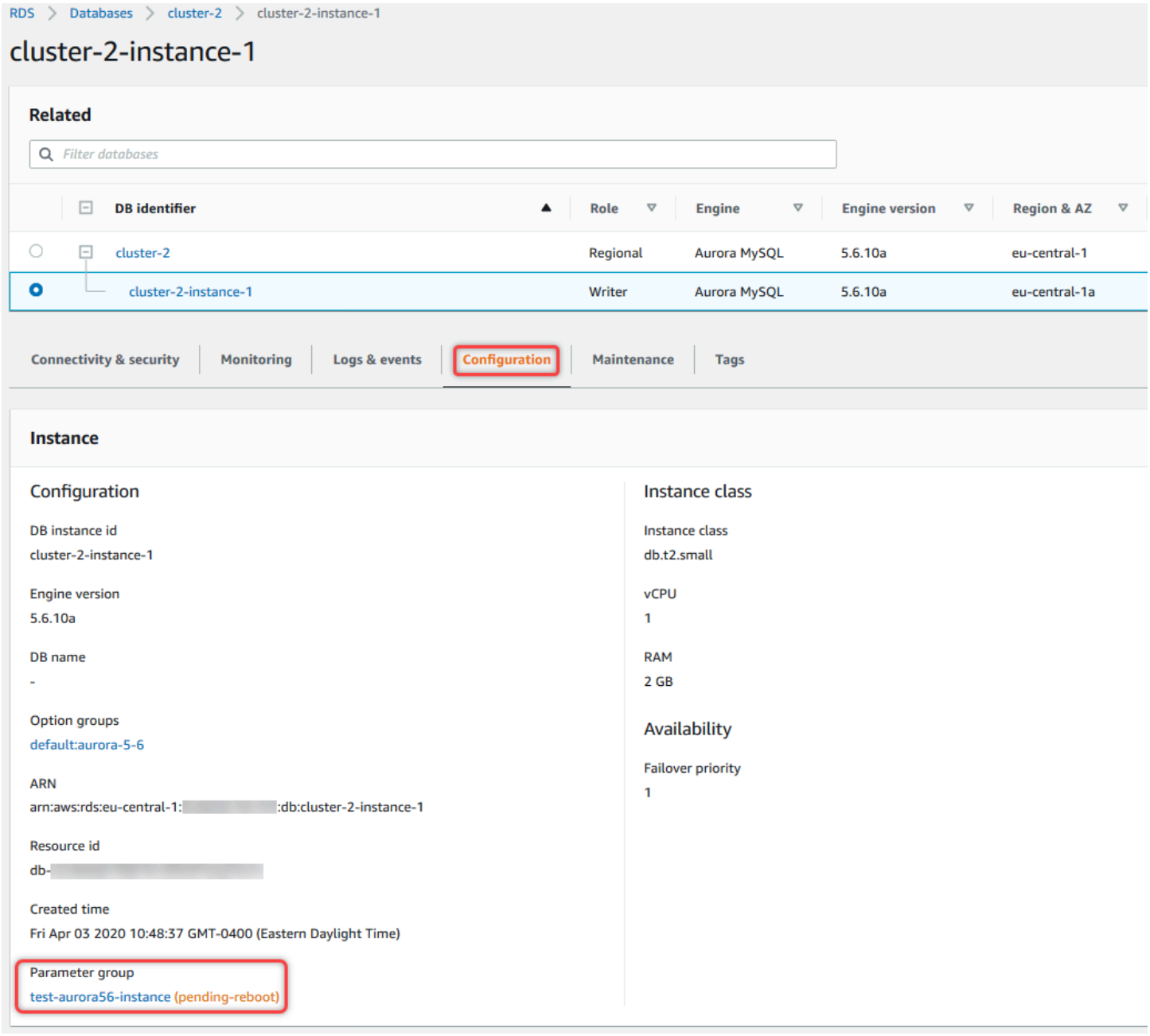
- `DBParameterGroupName`
- `Parameters`

Reimpostazione dei parametri in un gruppo di parametri database sui valori predefiniti in Amazon Aurora

Puoi reimpostare i valori dei parametri in un gruppo di parametri database creato dal cliente sui valori predefiniti. Le modifiche ai parametri in un gruppo di parametri database creato dal cliente vengono applicate a tutte le istanze database associate al gruppo di parametri database.

Quando utilizzi la console, puoi reimpostare specifici parametri sui valori predefiniti. Tuttavia, non è possibile reimpostare altrettanto facilmente tutti i parametri nel gruppo di parametri database contemporaneamente. Quando utilizzi l'API AWS CLI o RDS, puoi ripristinare i valori predefiniti di parametri specifici. Puoi anche reimpostare tutti i parametri del gruppo di parametri database contemporaneamente.

Le modifiche apportate ad alcuni parametri vengono applicate all'istanza database immediatamente senza un riavvio. Le modifiche apportate ad altri parametri vengono applicate solo dopo che l'istanza database viene riavviata. La console RDS mostra lo stato del gruppo di parametri database associato a un'istanza database nella scheda Configuration (Configurazione). Supponi, ad esempio che l'istanza database non utilizzi le modifiche più recenti apportate al gruppo di parametri database associato. In questo caso, la console RDS mostra il gruppo di parametri database con lo stato pending-reboot. Per applicare le ultime modifiche del parametro su quella istanza database, riavvia manualmente l'istanza database.



> **Note**
>
> In un gruppo di parametri database predefinito, i parametri vengono sempre impostati sui valori di default.

Console

Per ripristinare i valori predefiniti dei parametri di un gruppo di parametri database

1. Accedi a Console di gestione AWS e apri la console Amazon RDS all'indirizzo https://console.aws.amazon.com/rds/.
2. Nel riquadro di navigazione scegliere Parameter groups (Gruppi di parametri).
3. Nell'elenco, scegliere il gruppo di parametri.
4. Per Parameter group actions (Operazioni del gruppo di parametri), scegliere Edit (Modifica).
5. Scegliere i parametri per i quali si desidera ripristinare i valori predefiniti. È possibile scorrere i parametri usando i tasti freccia in alto a destra nella finestra di dialogo.

   Non è possibile reimpostare valori in un gruppo di parametri predefinito.
6. Scegli Reimposta , quindi conferma selezionando Ripristina parametri.

AWS CLI

Per reimpostare alcuni o tutti i parametri in un gruppo di parametri DB, usa il AWS CLI `reset-db-parameter-group`comando con la seguente opzione obbligatoria:`--db-parameter-group-name`.

Per reimpostare tutti i parametri nel gruppo di parametri database, specifica l'opzione `--reset-all-parameters`. Per reimpostare parametri specifici, specifica l'opzione `--parameters`.

Nell'esempio seguente tutti i parametri del gruppo di parametri del database denominato mydbparametergroup vengono reimpostati sui valori predefiniti.

Example

Per Linux, macOS o Unix:

```
aws rds reset-db-parameter-group \
    --db-parameter-group-name mydbparametergroup \
    --reset-all-parameters
```

Per Windows:

```
aws rds reset-db-parameter-group ^
    --db-parameter-group-name mydbparametergroup ^
```

```
    --reset-all-parameters
```

Nel seguente esempio le opzioni `max_connections` e `max_allowed_packet` vengono reimpostate sui loro valori predefiniti nel gruppo di parametri database denominato mydbparametergroup.

Example

Per Linux, macOS o Unix:

```
aws rds reset-db-parameter-group \
    --db-parameter-group-name mydbparametergroup \
    --parameters "ParameterName=max_connections,ApplyMethod=immediate" \
                 "ParameterName=max_allowed_packet,ApplyMethod=immediate"
```

Per Windows:

```
aws rds reset-db-parameter-group ^
    --db-parameter-group-name mydbparametergroup ^
    --parameters "ParameterName=max_connections,ApplyMethod=immediate" ^
                 "ParameterName=max_allowed_packet,ApplyMethod=immediate"
```

Il comando genera un output simile al seguente:

```
DBParameterGroupName  mydbparametergroup
```

API RDS

Per ripristinare i valori predefiniti dei parametri di un gruppo di parametri database, utilizza il comando `ResetDBParameterGroup` dell'API RDS con il seguente parametro obbligatorio: `DBParameterGroupName`.

Per reimpostare tutti i parametri nel gruppo di parametri database, imposta il parametro `ResetAllParameters` su `true`. Per reimpostare parametri specifici, specifica il parametro `Parameters`.

Copia di un gruppo di parametri database in Amazon Aurora

Puoi copiare i gruppi di parametri database personalizzati che hai creato. La copia di un gruppo di parametri può essere una soluzione utile. Ad esempio quando crei un gruppo di parametri database

e vuoi includere la maggior parte dei parametri e dei valori personalizzati in un nuovo gruppo di parametri del database. Puoi copiare un gruppo di parametri DB utilizzando. Console di gestione AWSÈ inoltre possibile utilizzare il AWS CLI copy-db-parameter-groupcomando o l'operazione RDS API Copy DBParameter Group.

Dopo aver copiato un gruppo di parametri database, attendi almeno 5 minuti prima di creare la prima istanza database che usa il gruppo di parametri database come predefinito. In questo modo, Amazon RDS può completare l'operazione di copia prima che venga usato il gruppo di parametri. Questo è particolarmente importante per parametri critici durante la creazione del database predefinito per un'istanza database. Un esempio è il set di caratteri per il database predefinito definito dal parametro `character_set_database`. Utilizza l'opzione Parameter Groups della console Amazon RDS o il describe-db-parameterscomando per verificare che il gruppo di parametri DB sia stato creato.

> **Note**
>
> Non puoi copiare un gruppo di parametri predefinito. Tuttavia, puoi creare un nuovo gruppo di parametri basato su un gruppo di parametri predefinito.
> Non puoi copiare un gruppo di parametri DB in un altro Account AWS o Regione AWS.

Console

Per copiare un gruppo di parametri del database

1. Accedi a Console di gestione AWS e apri la console Amazon RDS all'indirizzo https://console.aws.amazon.com/rds/.
2. Nel riquadro di navigazione scegliere Parameter groups (Gruppi di parametri).
3. Nell’elenco, scegliere il gruppo di parametri personalizzato da copiare.
4. Per Parameter group actions (Operazioni del gruppo di parametri), scegliere Copy (Copia).
5. In New DB parameter group identifier (Identificatore nuovo gruppo di parametri database) immettere un nome per il nuovo gruppo di parametri.
6. Nella casella Description (Descrizione), immettere una descrizione per il nuovo gruppo di parametri.
7. Scegliere Copy (Copia).

AWS CLI

Per copiare un gruppo di parametri DB, usa il AWS CLI `copy-db-parameter-group`comando con le seguenti opzioni obbligatorie:

- `--source-db-parameter-group-identifier`
- `--target-db-parameter-group-identifier`
- `--target-db-parameter-group-description`

L'esempio seguente crea un nuovo gruppo di parametri database denominato `mygroup2`, che è una copia del gruppo di parametri database `mygroup1`.

Example

Per Linux, macOS o Unix:

```
aws rds copy-db-parameter-group \
    --source-db-parameter-group-identifier mygroup1 \
    --target-db-parameter-group-identifier mygroup2 \
    --target-db-parameter-group-description "DB parameter group 2"
```

Per Windows:

```
aws rds copy-db-parameter-group ^
    --source-db-parameter-group-identifier mygroup1 ^
    --target-db-parameter-group-identifier mygroup2 ^
    --target-db-parameter-group-description "DB parameter group 2"
```

API RDS

Per copiare un gruppo di parametri del database, utilizza l'operazione API RDS `CopyDBParameterGroup`:

- `SourceDBParameterGroupIdentifier`
- `TargetDBParameterGroupIdentifier`
- `TargetDBParameterGroupDescription`

Generazione di un elenco di gruppi di parametri del database in Amazon Aurora

Puoi elencare i gruppi di parametri DB che hai creato per il tuo AWS account.

> **Note**
>
> I gruppi di parametri predefiniti vengono creati automaticamente da un modello di parametro predefinito quando crei un'istanza database per un motore e una versione di database specifici. Questi gruppi di parametri predefiniti contengono le impostazioni dei parametri preferite e non possono essere modificati. Quando crei un gruppo di parametri personalizzato, puoi modificare le impostazioni dei parametri.

Console

Per elencare tutti i gruppi di parametri DB per un AWS account

1. Accedi a Console di gestione AWS e apri la console Amazon RDS all'indirizzo https://console.aws.amazon.com/rds/.
2. Nel riquadro di navigazione scegliere Parameter groups (Gruppi di parametri).

   I gruppi di parametri database vengono visualizzati in un elenco.

AWS CLI

Per elencare tutti i gruppi di parametri DB per un AWS account, usa il AWS CLI `describe-db-parameter-groups`comando.

Example

L'esempio seguente elenca tutti i gruppi di parametri database disponibili per un account AWS .

```
aws rds describe-db-parameter-groups
```

Questo comando restituisce una risposta simile alla seguente:

```
DBPARAMETERGROUP  default.mysql8.0     mysql8.0  Default parameter group for MySQL8.0
DBPARAMETERGROUP  mydbparametergroup   mysql8.0  My new parameter group
```

L'esempio seguente descrive il gruppo di parametri mydbparamgroup1.

Per Linux, macOS o Unix:

```
aws rds describe-db-parameter-groups \
    --db-parameter-group-name mydbparamgroup1
```

Per Windows:

```
aws rds describe-db-parameter-groups ^
    --db-parameter-group-name mydbparamgroup1
```

Questo comando restituisce una risposta simile alla seguente:

```
DBPARAMETERGROUP  mydbparametergroup1  mysql8.0  My new parameter group
```

API RDS

Per elencare tutti i gruppi di parametri DB per un AWS account, utilizza l'`DescribeDBParameterGroups`operazione API RDS.

Visualizzazione dei valori dei parametri per un gruppo di parametri del database in Amazon Aurora

Puoi ottenere un elenco di tutti i parametri in un gruppo di parametri del database e dei rispettivi valori.

Console

Per visualizzare i valori dei parametri per un gruppo di parametri del database

1. Accedi a Console di gestione AWS e apri la console Amazon RDS all'indirizzo https://console.aws.amazon.com/rds/.
2. Nel riquadro di navigazione scegliere Parameter groups (Gruppi di parametri).

   I gruppi di parametri database vengono visualizzati in un elenco.
3. Scegliere il nome del gruppo di parametri per visualizzarne l'elenco di parametri.

AWS CLI

Per visualizzare i valori dei parametri per un gruppo di parametri DB, usa il AWS CLI `describe-db-parameters`comando con il seguente parametro obbligatorio.

- `--db-parameter-group-name`

Example

L'esempio seguente elenca i parametri e i valori dei parametri per un gruppo di parametri database denominato mydbparametergroup.

```
aws rds describe-db-parameters --db-parameter-group-name mydbparametergroup
```

Questo comando restituisce una risposta simile alla seguente:

```
DBPARAMETER  Parameter Name            Parameter Value  Source           Data Type
 Apply Type  Is Modifiable
DBPARAMETER  allow-suspicious-udfs                      engine-default   boolean
 static      false
DBPARAMETER  auto_increment_increment                   engine-default   integer
 dynamic     true
DBPARAMETER  auto_increment_offset                      engine-default   integer
 dynamic     true
DBPARAMETER  binlog_cache_size         32768            system           integer
 dynamic     true
DBPARAMETER  socket                    /tmp/mysql.sock  system           string
 static      false
```

API RDS

Per visualizzare i valori dei parametri per un gruppo di parametri del database, utilizza il comando RDS API `DescribeDBParameters` con il seguente parametro obbligatorio.

- `DBParameterGroupName`

Eliminazione di un gruppo di parametri database in Amazon Aurora

Puoi eliminare un gruppo di parametri DB utilizzando l'API Console di gestione AWS AWS CLI, o RDS. Un gruppo di parametri è idoneo solo se non è associato a un'istanza database.

Console

Per eliminare un gruppo di parametri database

1. Accedi a Console di gestione AWS e apri la console Amazon RDS all'indirizzo https://console.aws.amazon.com/rds/.
2. Nel riquadro di navigazione scegliere Parameter groups (Gruppi di parametri).

   I gruppi di parametri database vengono visualizzati in un elenco.
3. Sceglie il nome del gruppo di parametri da eliminare.
4. Scegli Operazioni, quindi Elimina.
5. Controlla i nomi dei gruppi di parametri, quindi scegli Elimina.

AWS CLI

Per eliminare un gruppo di parametri DB, usa il AWS CLI `delete-db-parameter-group`comando con il seguente parametro obbligatorio.

- `--db-parameter-group-name`

Example

L'esempio seguente elimina un gruppo di parametri database denominato mydbparametergroup.

```
aws rds delete-db-parameter-group --db-parameter-group-name mydbparametergroup
```

API RDS

Per modificare un gruppo di parametri database, utilizza il comando API RDS `DeleteDBParameterGroup` con il parametro obbligatorio seguente.

- `DBParameterGroupName`

Confronto di gruppi di parametri database

Puoi usare la Console di gestione AWS per visualizzare le differenze tra due gruppi di parametri database.

I gruppi di parametri specificati devono essere entrambi gruppi di parametri database o gruppi di parametri cluster database, anche se il motore e la versione del database sono uguali. Ad esempio, non puoi confrontare un gruppo di parametri database `aurora-mysql8.0` (Aurora MySQL versione 3) e un gruppo di parametri cluster di database `aurora-mysql8.0`.

È possibile confrontare i gruppi di parametri database Aurora MySQL e RDS per MySQL, anche per versioni diverse, ma non è possibile confrontare i gruppi di parametri database Aurora PostgreSQL e RDS per PostgreSQL.

Per confrontare due gruppi di parametri database

1. Accedi alla Console di gestione AWS e apri la console Amazon RDS all'indirizzo https://console.aws.amazon.com/rds/.
2. Nel riquadro di navigazione scegliere Parameter groups (Gruppi di parametri).
3. Nell'elenco, scegliere i due gruppi di parametri da confrontare.

   > **Note**
   >
   > Per confrontare un gruppo di parametri predefinito con un gruppo di parametri personalizzato, scegli prima il gruppo di parametri predefinito nella scheda Predefinito, quindi scegli il gruppo di parametri personalizzato nella scheda Personalizzato.

4. Da Azioni, scegli Confronta.

Specifica dei parametri del database

I tipi di parametri database comprendono:

- Numero intero
- Boolean
- Stringa
- Long
- Doppio
- Timestamp
- Oggetto di altri tipi di dati definiti
- Array di valori di tipo integer, booleano, string, long, double, timestamp o oggetto

È inoltre possibile specificare parametri interi e booleani utilizzando espressioni, formule e funzioni.

Indice

- Formule dei parametri database
  - Variabili di formula dei parametri database
  - Operatori delle formule dei parametri database
- Funzioni dei parametri database
- Espressioni di log di parametri database
- Esempi di valori dei parametri database

Formule dei parametri database

Una formula per un parametro database è un'espressione che restituisce un valore intero o un valore booleano. L'espressione va racchiusa tra parentesi graffe: {}. Puoi utilizzare una formula per un valore di parametro database o come argomento per una funzione di parametro database.

Sintassi

```
{FormulaVariable}
{FormulaVariable*Integer}
{FormulaVariable*Integer/Integer}
{FormulaVariable/Integer}
```

Variabili di formula dei parametri database

Ogni variabile di formula restituisce un valore intero o booleano. I nomi delle variabili fanno distinzione tra maiuscole e minuscole.

AllocatedStorage

Restituisce un numero intero che rappresenta la dimensione, in byte, del volume di dati.

DBInstanceClassMemory

Restituisce un numero intero per il numero di byte di memoria disponibili per il processo del database. Questo numero viene calcolato internamente a partire dalla quantità totale di memoria per la classe di istanza database. Da questo valore, il calcolo sottrae la memoria riservata al sistema operativo e ai processi RDS che gestiscono l'istanza. Pertanto, il numero è sempre

leggermente inferiore alle figure di memoria mostrate nelle tabelle della classe di istanza in Classi di istanze database Amazon Aurora. Il valore esatto dipende da una combinazione di fattori: come classe di istanza, motore di database e se si applica a un'istanza RDS o a un'istanza che fa parte di un cluster Aurora.

DBInstanceVCPU

Restituisce un numero intero che rappresenta il numero di unità di elaborazione centrali virtuali (vCPU) utilizzate da Amazon RDS per gestire l'istanza.

EndPointPort

Restituisce un numero intero che rappresenta la porta utilizzata durante la connessione all'istanza database.

TrueIfReplica

Restituisce 1 se l'istanza database è una replica di lettura e `0` in caso contrario. Questo è il valore predefinito per il parametro `read_only` in Aurora MySQL.

Operatori delle formule dei parametri database

Le formule dei parametri database supportano due operatori, di divisione e di moltiplicazione.

Operatore di divisione: /

Divide il dividendo per il divisore, restituendo un quoziente intero. I decimali nel quoziente vengono troncati, non arrotondati.

Sintassi

```
dividend / divisor
```

Gli argomenti del dividendo e del divisore devono essere espressioni intere.

Operatore di moltiplicazione: *

Moltiplica le espressioni, restituendone il prodotto. I decimali nelle espressioni vengono troncati, non arrotondati.

Sintassi

```
expression * expression
```

Entrambe le espressioni devono essere valori interi.

Funzioni dei parametri database

Puoi specificare gli argomenti delle funzioni dei parametri database come numeri interi o formule. Ogni funzione deve avere almeno un argomento. Specifica più argomenti come elenco separato da virgole. L'elenco non può includere membri vuoti, come argomento1,,argomento3. I nomi di funzione non fanno distinzione tra maiuscole e minuscole.

IF

Restituisce un argomento.

Sintassi

```
IF(argument1, argument2, argument3)
```

Restituisce il secondo argomento se il primo argomento restituisce true. In caso contrario, restituisce il terzo argomento.

GREATEST

Restituisce il valore più grande da un elenco di valori interi o formule di parametro.

Sintassi

```
GREATEST(argument1, argument2,...argumentn)
```

Restituisce un integer.

LEAST

Restituisce il valore più piccolo da un elenco di valori interi o formule di parametro.

Sintassi

```
LEAST(argument1, argument2,...argumentn)
```

Restituisce un integer.

SUM

Aggiunge i valori delle formule di parametro o dei numeri interi specificati.

Sintassi

```
SUM(argument1, argument2,...argumentn)
```

Restituisce un integer.

Espressioni di log di parametri database

Puoi impostare un valore del parametro database intero in una espressione di log. L'espressione va racchiusa tra parentesi graffe: {}. Per esempio:

```
{log(DBInstanceClassMemory/8187281418)*1000}
```

La funzione `log` rappresenta la base di log 2. In questo esempio viene utilizzato anche la variabile di formula `DBInstanceClassMemory`. Per informazioni, consulta Variabili di formula dei parametri database.

Esempi di valori dei parametri database

Questi esempi illustrano l'utilizzo di formule, funzioni ed espressioni per i valori dei parametri database.

> **Warning**
>
> L'impostazione errata dei parametri in un gruppo di parametri database può avere effetti negativi non intenzionali. Questi potrebbero includere prestazioni ridotte e instabilità del sistema. Presta sempre attenzione quando modifichi i parametri database ed esegui il backup dei dati prima di modificare il gruppo di parametri database. Prova le modifiche del gruppo di parametri in istanze database di test create tramite ripristini point-in-time prima di applicare queste modifiche alle istanze database di produzione.

Example utilizzando la funzione di parametro database LEAST

È possibile specificare la funzione `LEAST` in un valore di parametro Aurora MySQL `table_definition_cache`. Usalo per impostare il numero di definizioni di tabella che possono essere memorizzate nella cache delle definizioni sul valore minimo di `DBInstanceClassMemory`/393040 o 20.000.

```
LEAST({DBInstanceClassMemory/393040}, 20000)
```

Migrazione di dati a un cluster di database Amazon Aurora

Per la migrazione dei dati da un database esistente a un cluster database Amazon Aurora sono disponibili diverse opzioni, a seconda della compatibilità del motore di database. Le opzioni di migrazione dipendono anche dal database da cui esegui la migrazione e dalle dimensioni dei dati sottoposti a migrazione.

Migrazione di dati a un cluster di database Amazon Aurora MySQL

Puoi eseguire la migrazione dei dati da una delle origini seguenti a un cluster database Amazon Aurora MySQL.

- Un'istanza database RDS for MySQL
- Un database MySQL esterno a Amazon RDS
- Un database che non è compatibile con MySQL

Per ulteriori informazioni, consulta Migrazione di dati a un cluster di database Amazon Aurora MySQL.

Migrazione di dati a un cluster di database Amazon Aurora PostgreSQL

Puoi eseguire la migrazione dei dati da una delle origini seguenti a un cluster database Amazon Aurora PostgreSQL.

- Istanza database PostgreSQL Amazon RDS
- Database non compatibile con PostgreSQL

Per ulteriori informazioni, consulta Migrazione di dati su Amazon Aurora con compatibilità PostgreSQL.

Creazione di una cache Amazon ElastiCache utilizzando le impostazioni del cluster di database Aurora

ElastiCache è un servizio di cache in memoria completamente gestito che fornisce latenze di lettura e scrittura in microsecondi che supportano casi d'uso flessibili e in tempo reale. ElastiCache può aiutarti ad accelerare le prestazioni di applicazioni e database. Puoi utilizzare ElastiCache come archivio dati principale per i casi d'uso che non richiedono la durabilità dei dati, come classifiche di gioco, streaming e analisi dei dati. ElastiCache aiuta a rimuovere la complessità associata all'implementazione e alla gestione di un ambiente di calcolo distribuito. Per ulteriori informazioni, consulta Casi d’uso comuni relativi a ElastiCache e in che modo ElastiCache può essere d’aiuto per Memcached e Casi d’uso comuni relativi a ElastiCache e in che modo ElastiCache può essere d’aiuto per Redis OSS. Per creare i cluster ElastiCache è possibile utilizzare la console Amazon RDS.

È possibile utilizzare Amazon ElastiCache in due formati. Puoi iniziare con una cache serverless o scegliere di progettare il tuo cluster di cache. Se scegli di progettare un cluster di cache personalizzato, ElastiCache funziona con i motori Redis OSS e Memcached. Se non sai quale motore utilizzare, consulta Confronto tra Memcached e Redis. Per ulteriori informazioni su Amazon ElastiCache, consulta la Guida per l'utente di Amazon ElastiCache.

Argomenti

- Panoramica della creazione di una cache ElastiCache con le impostazioni del cluster di database Aurora
- Creazione di una cache ElastiCache con le impostazioni di un cluster di database Aurora esistente

Panoramica della creazione di una cache ElastiCache con le impostazioni del cluster di database Aurora

È possibile creare una cache ElastiCache da Amazon RDS utilizzando le stesse impostazioni di configurazione di un cluster di database Aurora appena creato o esistente.

Alcuni casi d’uso per associare una cache ElastiCache al cluster di database:

- È possibile risparmiare sui costi e migliorare le prestazioni utilizzando ElastiCache con RDS anziché eseguirlo solo su RDS.

- È possibile utilizzare la cache ElastiCache come datastore primario per applicazioni che non richiedono la durabilità dei dati. Le applicazioni che utilizzano Redis OSS o Memcached possono utilizzare ElastiCache pressoché senza modifiche.

Quando si crea una cache ElastiCache da RDS, la cache ElastiCache eredita le seguenti impostazioni dal cluster di database Aurora associato:

- Impostazioni di connettività di ElastiCache
- Impostazioni di sicurezza di ElastiCache

È anche possibile definire le impostazioni di configurazione della cache in base a specifiche esigenze.

Configurazione di ElastiCache nelle applicazioni

Per utilizzare la cache ElastiCache, le applicazioni devono essere configurate. È anche possibile ottimizzare e migliorare le prestazioni della cache configurando le applicazioni per l'uso di strategie di caching in base alle specifiche esigenze.

- Per accedere alla cache ElastiCache e iniziare a operare, consulta Getting started with ElastiCache (Redis OSS) e Getting started with ElastiCache (Memcached).
- Per ulteriori informazioni sulle strategie di caching, consulta Best practice e strategie di caching per Memcached e Best practice e strategie di caching per Redis OSS.
- Per ulteriori informazioni sulla disponibilità elevata nei cluster ElastiCache (Redis OSS), consulta Alta disponibilità utilizzando gruppi di replica.
- Potrebbe essere necessario sostenere i costi associati all'archiviazione di backup, al trasferimento dei dati all'interno o tra regioni o all'uso di AWS Outposts. Per ulteriori dettagli sui prezzi, consulta Prezzi di Amazon ElastiCache.

Creazione di una cache ElastiCache con le impostazioni di un cluster di database Aurora esistente

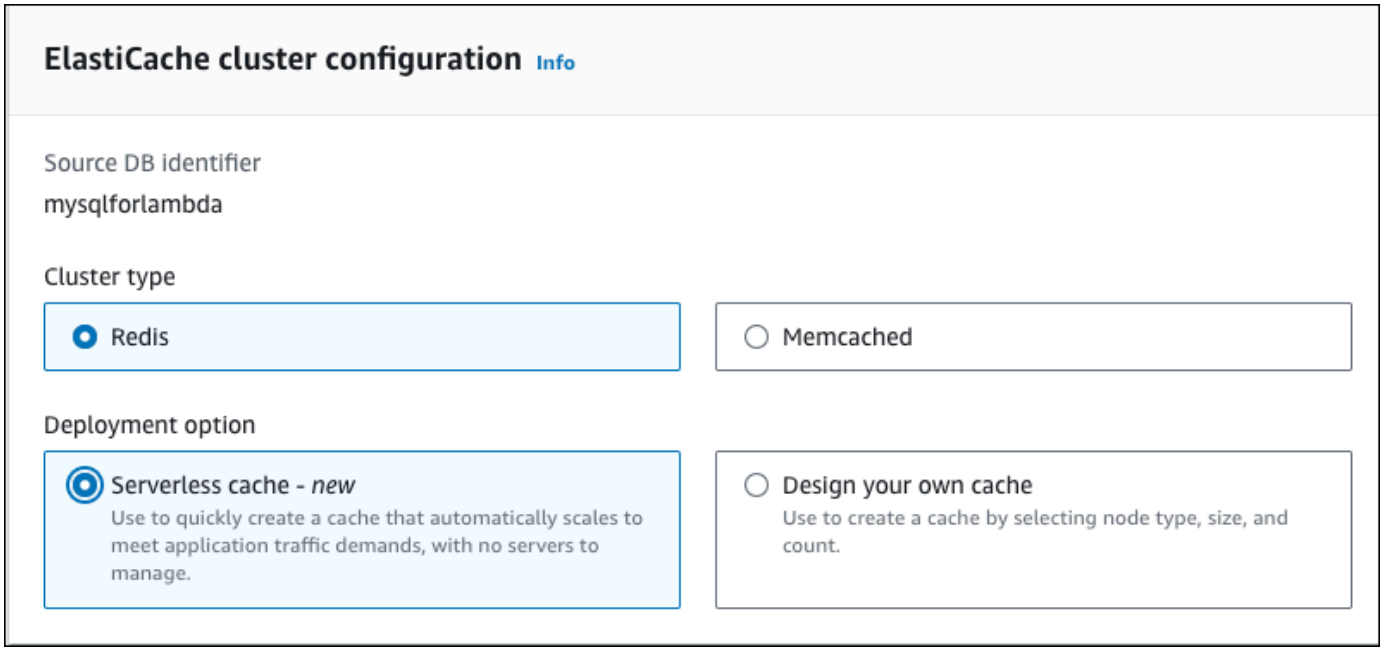
È possibile creare una cache ElastiCache per i cluster di database Aurora con le impostazioni ereditate dal cluster di database.

Creazione di una cache ElastiCache con le impostazioni di un cluster di database

1. Per creare un cluster di database, segui le istruzioni in Creazione di un cluster database Amazon Aurora.
2. Dopo aver creato un cluster di database Aurora, la finestra Componenti aggiuntivi suggeriti viene visualizzata nella console. Seleziona Crea un cluster ElastiCache da RDS utilizzando le impostazioni del database.

   Per un database esistente, nella pagina Database selezionare il cluster di database richiesta. Nel menu a discesa Operazioni scegliere Crea cluster ElastiCache per creare una cache ElastiCache in RDS con le stesse impostazioni del cluster di database Aurora esistente.

   Nella sezione di configurazione ElastiCache, il campo Identificatore database di origine visualizza da quale cluster di database eredita le impostazioni la cache ElastiCache.
3. Scegliere se creare un cluster Redis OSS o Memcached. Per ulteriori informazioni, consulta Confronto tra Memcached e Redis OSS.



4. Dopodiché, scegliere se creare una Cache serverless o selezionare Progetta la tua cache. Per ulteriori informazioni, consulta Scelta tra le opzioni di implementazione.

   Se si sceglie Cache serverless:

   a. In Impostazioni della cache, immettere i valori per Nome e Descrizione.

b. In Visualizza le impostazioni predefinite, lasciare le impostazioni predefinite per stabilire la connessione tra la cache e il cluster di database.

c. È anche possibile modificare le impostazioni predefinite scegliendo Personalizza le impostazioni predefinite. Selezionare le Impostazioni di connettività ElastiCache, le Impostazioni di sicurezza ElastiCache e i Limiti massimi di utilizzo.

5. Se si sceglie Progetta la tua cache:

a. Se si sceglie il cluster Redis OSS, indicare se la modalità cluster è Abilitata o Disabilitata. Per ulteriori informazioni, consulta Replication: Redis OSS (Cluster Mode Disabled) vs. Redis OSS (Cluster Mode Enabled).

b. Immetti i valori per Nome, Descrizione e Versione del motore.

Per Versione del motore, il valore predefinito consigliato è la versione più recente del motore. Per la cache ElastiCache è anche possibile scegliere un valore che meglio soddisfa le esigenze in Versione del motore.

c. Scegli il tipo di nodo per l'opzione Tipo di nodo. Per ulteriori informazioni, consulta Gestione di nodi.

Se si sceglie di creare un cluster Redis OSS con la Modalità cluster impostata su Abilitato, inserire il numero di shard (partizioni/gruppi di nodi) per l'opzione Numero di partizioni.

Immetti il numero di repliche di ogni partizione per l'opzione Numero di repliche.

> **Note**
>
> Il tipo di nodo selezionato, il numero di shard e il numero di repliche influiscono tutti sulle prestazioni della cache e sui costi delle risorse. Assicurati che queste impostazioni corrispondano alle esigenze del tuo database. Per informazioni sui prezzi, consulta Prezzi di Amazon ElastiCache.

d. Selezionare le Impostazioni di connettività ElastiCache e le Impostazioni di sicurezza ElastiCache. È possibile mantenere le impostazioni predefinite o personalizzarle in base alle proprie esigenze.

6. Verificare le impostazioni predefinite ed ereditate della cache ElastiCache. Alcune impostazioni non possono essere modificate dopo la creazione.

> **Note**
>
> RDS può modificare la finestra di backup della cache ElastiCache per soddisfare il requisito minimo di 60 minuti. La finestra di backup del database di origine rimane invariata.

7. Quando è tutto pronto, scegliere Crea una cache ElastiCache.

Nella console viene visualizzato un banner di conferma per la creazione della cache ElastiCache. Seguire il collegamento alla console ElastiCache presente nel banner per visualizzare i dettagli della cache. La console ElastiCache mostra la cache ElastiCache appena creata.

Migrazione automatica dei database EC2 ad Amazon Aurora utilizzando AWS Database Migration Service

È possibile utilizzare la console Aurora per eseguire la migrazione di un database EC2 ad Aurora. Aurora utilizza AWS Database Migration Service (AWS DMS) per eseguire la migrazione del database EC2 di origine. AWS DMS consente di migrare i database relazionali nel cloud AWS. Per ulteriori informazioni su AWS Database Migration Service, consulta Che cos'è AWS Database Migration Service? nella Guida per l'utente di AWS Database Migration Service.

Per iniziare la migrazione, è necessario creare un cluster di database Aurora equivalente in cui migrare i dati. Dopo aver creato il database di destinazione, è possibile importarvi il database EC2. Per i database di origine più piccoli di 1 TiB, questa azione di migrazione riduce il tempo e le risorse necessari per migrare i dati in Aurora .

Panoramica

La console Aurora consente di migrare i database EC2 in database Aurora equivalenti. È necessario creare un database Aurora per abilitare la migrazione dalla console.

È possibile migrare i database EC2 per i seguenti motori di database:

- MySQL
- PostgreSQL

Il processo di migrazione prevede i seguenti passaggi:

- Creare un database equivalente in Aurora. Affinché i database siano equivalenti, devono avere lo stesso motore di database e una versione del motore compatibile. Devono anche trovarsi nello stesso VPC. Per istruzioni sulla creazione del database, consulta Creazione di un cluster database Amazon Aurora .
- Scegliere il tipo di replica per il database:
  - Migrazione a pieno carico: Aurora copia l'intero database di origine nel database di destinazione, creando nuove tabelle nella destinazione quando necessario.

> **Note**
>
> Questa opzione causa un'interruzione del database Aurora.

- Migrazione a pieno carico e acquisizione dei dati di modifica (CDC): simile alla migrazione a pieno carico, con questa opzione, Aurora copia l'intero database di origine nel database di destinazione. Tuttavia, dopo la migrazione a pieno carico, Aurora applica le modifiche acquisite nell'origine al database di destinazione. L'acquisizione dei dati di modifica raccoglie le modifiche nei log del database utilizzando l'API nativa del motore di database.

  > **Note**
  >
  > Questa opzione causa un'interruzione del database Aurora.

- Acquisizione dei dati di modifica (CDC): questa opzione si utilizza per mantenere disponibile il database di destinazione durante la migrazione. Aurora migra le modifiche in corso dal database di origine nel database di destinazione.

- Aurora crea le risorse di rete necessarie per facilitare la migrazione. Una volta create le risorse necessarie, Aurora notifica all'utente le risorse create e consente di avviare il trasferimento dei dati.

  Il tempo necessario per completare la migrazione dipende dal tipo di replica e dalla dimensione del database di origine.

Prerequisiti

MySQL

Prima di iniziare a utilizzare un database MySQL come database di origine, è necessario che i seguenti prerequisiti siano soddisfatti. Questi prerequisiti si applicano alle origini gestite da AWS.

È necessario disporre di un account per AWS DMS che disponga del ruolo Replication Admin. Il ruolo richiede i seguenti privilegi:

- REPLICATION CLIENT: questo privilegio è richiesto per le attività di sola CDC. In altre parole, solo le attività di caricamento completo non richiedono questo privilegio.
- REPLICATION SLAVE: questo privilegio è richiesto per le attività di sola CDC. In altre parole, solo le attività di caricamento completo non richiedono questo privilegio.

L'utente AWS DMS deve disporre anche dei privilegi SELECT per le tabelle di origine designate per la replica.

Se si utilizzano valutazioni di premigrazione specifiche di MySQL, è necessario fornire i seguenti privilegi.

```
grant select on mysql.user to <dms_user>;
grant select on mysql.db to <dms_user>;
grant select on mysql.tables_priv to <dms_user>;
grant select on mysql.role_edges to <dms_user>  #only for MySQL version 8.0.11 and
 higher
```

PostgreSQL

Prima di migrare i dati da un database di origine PostgreSQL gestito da AWS, procedi come segue:

- Ti consigliamo di utilizzare un account utente AWS con le autorizzazioni minime richieste per l'istanza database PostgreSQL come account utente per l'endpoint di origine PostgreSQL per AWS DMS. L'utilizzo di un account master è sconsigliato. L'account deve avere il ruolo `rds_superuser` e il ruolo `rds_replication`. Il ruolo `rds_replication` fornisce le autorizzazioni per gestire gli slot logici e per eseguire lo streaming dei dati utilizzando gli slot logici.

> **Note**
>
> Alcune transazioni AWS DMS sono inattive per un po' di tempo prima che il motore DMS le utilizzi nuovamente. Il parametro `idle_in_transaction_session_timeout` in PostgreSQL versione 9.6 e successive consente di mandare in timeout e in errore le transazioni inattive.

Limitazioni

Le seguenti limitazioni si applicano al processo di migrazione automatica:

- Per iniziare la migrazione del database di origine, lo stato del database di destinazione deve essere Disponibile.
- Quando si esegue la migrazione da un database di origine MySQL, l’account Aurora deve avere il ruolo di amministratore della replica. È necessario anche avere i privilegi appropriati per il ruolo.
- L’istanza EC2 e il database di destinazione devono trovarsi nello stesso VPC.
- Non è possibile migrare il database EC2 ai seguenti database di destinazione quando si utilizza l’azione Migra dati dal database EC2:

- Aurora global database
- Aurora Limitless database
- Aurora Serverless v1
- Database con versione di MySQL precedente alla 5.7
- Database con versione di PostgreSQL precedente alla 10.4

Creazione di risorse IAM per migrazioni omogenee

Aurora utilizza AWS DMS per migrare i dati. Per accedere ai database e migrare i dati, AWS DMS crea un ambiente senza server per migrazioni di dati omogenee. In questo ambiente, AWS DMS richiede l'accesso al peering VPC, alle tabelle di routing, ai gruppi di sicurezza e ad altre risorse. AWS Inoltre, AWS DMS archivia log, metriche e progressi per ogni migrazione di dati in Amazon. CloudWatch Per creare un progetto di migrazione dei dati, è AWS DMS necessario accedere a questi servizi.

Inoltre, AWS DMS richiede l'accesso ai segreti che rappresentano un set di credenziali utente per autenticare la connessione al database per la connessione di origine e di destinazione.

> **Note**
>
> Utilizzando l'azione Migra dati dall' EC2 istanza, è possibile utilizzare la console RDS per generare queste risorse IAM. Salta questo passaggio se utilizzi le risorse IAM generate dalla console.

Per questo processo, sono necessarie le seguenti risorse IAM:

Argomenti

- Creazione di una policy IAM per migrazioni di dati omogenee
- Creazione di un ruolo IAM per migrazioni di dati omogenee
- Creazione di un ruolo e una policy di accesso ai segreti
- Creazione di un ruolo IAM per AWS DMS per la gestione di Amazon VPC

Creazione di una policy IAM per migrazioni di dati omogenee

In questa fase, crei una policy IAM che fornisce l'accesso AWS DMS ad Amazon EC2 e alle CloudWatch risorse. Quindi, crea un ruolo IAM e collegalo alla policy.

Per creare una policy IAM per migrazioni di dati omogenee

1. Accedi Console di gestione AWS e apri la console IAM all'indirizzo https://console.aws.amazon.com/iam/.
2. Nel riquadro di navigazione, scegli Policy.
3. Scegli Create Policy (Crea policy).
4. Nella pagina Crea policy seleziona la scheda JSON.
5. Incolla il seguente JSON nell'editor.

   JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "ec2:DescribeRouteTables",
                "ec2:DescribeSecurityGroups",
                "ec2:DescribeVpcPeeringConnections",
                "ec2:DescribeVpcs",
                "ec2:DescribePrefixLists",
                "logs:DescribeLogGroups"
            ],
            "Resource": "*"
        },
        {
            "Effect": "Allow",
            "Action": [
                "servicequotas:GetServiceQuota"
            ],
            "Resource": "arn:aws:servicequotas:*:*:vpc/L-0EA8095F"
        },
        {
            "Effect": "Allow",
```

```
            "Action": [
                "logs:CreateLogGroup",
                "logs:DescribeLogStreams"
            ],
            "Resource": "arn:aws:logs:*:*:log-group:dms-data-migration-*"
        },
        {
            "Effect": "Allow",
            "Action": [
                "logs:CreateLogStream",
                "logs:PutLogEvents"
            ],
            "Resource": "arn:aws:logs:*:*:log-group:dms-data-migration-*:log-
stream:dms-data-migration-*"
        },
        {
            "Effect": "Allow",
            "Action": "cloudwatch:PutMetricData",
            "Resource": "*"
        },
        {
            "Effect": "Allow",
            "Action": [
                "ec2:CreateRoute",
                "ec2:DeleteRoute"
            ],
            "Resource": "arn:aws:ec2:*:*:route-table/*"
        },
        {
            "Effect": "Allow",
            "Action": [
                "ec2:CreateTags"
            ],
            "Resource": [
                "arn:aws:ec2:*:*:security-group/*",
                "arn:aws:ec2:*:*:security-group-rule/*",
                "arn:aws:ec2:*:*:route-table/*",
                "arn:aws:ec2:*:*:vpc-peering-connection/*",
                "arn:aws:ec2:*:*:vpc/*"
            ]
        },
        {
            "Effect": "Allow",
            "Action": [
```

```
                "ec2:AuthorizeSecurityGroupEgress",
                "ec2:AuthorizeSecurityGroupIngress"
            ],
            "Resource": "arn:aws:ec2:*:*:security-group-rule/*"
        },
        {
            "Effect": "Allow",
            "Action": [
                "ec2:AuthorizeSecurityGroupEgress",
                "ec2:AuthorizeSecurityGroupIngress",
                "ec2:RevokeSecurityGroupEgress",
                "ec2:RevokeSecurityGroupIngress"
            ],
            "Resource": "arn:aws:ec2:*:*:security-group/*"
        },
        {
            "Effect": "Allow",
            "Action": [
                "ec2:AcceptVpcPeeringConnection",
                "ec2:ModifyVpcPeeringConnectionOptions"
            ],
            "Resource": "arn:aws:ec2:*:*:vpc-peering-connection/*"
        },
        {
            "Effect": "Allow",
            "Action": "ec2:AcceptVpcPeeringConnection",
            "Resource": "arn:aws:ec2:*:*:vpc/*"
        }
    ]
}
```

6. Scegli Next: Tags (Successivo: Tag) e Next: Review (Successivo: Rivedi).
7. Immetti **`HomogeneousDataMigrationsPolicy`** per Nome* e scegli Crea policy.

Creazione di un ruolo IAM per migrazioni di dati omogenee

In questa fase, crei un ruolo IAM che fornisce l'accesso a Gestione dei segreti AWS EC2, Amazon e CloudWatch.

Per creare un ruolo IAM per le migrazioni di dati

1. Accedi Console di gestione AWS e apri la console IAM all'indirizzo https://console.aws.amazon.com/iam/.
2. Nel pannello di navigazione, seleziona Roles (Ruoli).
3. Selezionare Create role (Crea ruolo).
4. Nella pagina Seleziona un'entità attendibile scegli Servizio AWS per Tipo di entità attendibile. Per Casi d'uso per altri servizi AWS scegli DMS.
5. Seleziona la casella di controllo DMS e scegli Successivo.
6. Nella pagina Aggiungi autorizzazioni, scegli HomogeneousDataMigrationsPolicyquello che hai creato in precedenza. Scegli Next (Successivo).
7. Nella pagina Nomina, verifica e crea immetti **`HomogeneousDataMigrationsRole`** per Nome ruolo e scegli Crea ruolo.
8. Nella pagina Ruoli immetti **`HomogeneousDataMigrationsRole`** per Nome ruolo. Scegli HomogeneousDataMigrationsRole.
9. Nella HomogeneousDataMigrationsRolepagina, scegli la scheda Relazioni di fiducia. Seleziona Modifica policy di attendibilità.
10. Nella pagina Modifica policy di attendibilità incolla il seguente codice JSON nell'editor, sostituendo il testo esistente.

    JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Sid": "",
            "Effect": "Allow",
            "Principal": {
                "Service": [
                    "dms-data-migrations.amazonaws.com",
                    "dms.your_region.amazonaws.com"
                ]
            },
            "Action": "sts:AssumeRole"
        }
    ]
```

```
}
```

Nell'esempio precedente, sostituisci *your_region* con il nome del tuoRegione AWS.

La precedente politica basata sulle risorse fornisce ai responsabili del AWS DMS servizio le autorizzazioni per eseguire attività in base alla politica gestita dal cliente. HomogeneousDataMigrationsPolicy

11. Scegli Aggiorna policy.

Creazione di un ruolo e una policy di accesso ai segreti

Le procedure seguenti consentono di creare un ruolo e una policy di accesso ai segreti che consentano a DMS di accedere alle credenziali utente per i database di origine e di destinazione.

Per creare la politica e il ruolo di accesso segreti, che consentano ad Amazon RDS di accedere Gestione dei segreti AWS per accedere al segreto appropriato

1. Accedi Console di gestione AWS e apri la console AWS Identity and Access Management (IAM) all'indirizzo https://console.aws.amazon.com/iam/.
2. Seleziona Policy, quindi scegli Crea policy.
3. Scegli JSON e inserisci la seguente policy per consentire l'accesso e la decrittografia del tuo segreto.

   JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": "secretsmanager:GetSecretValue",
            "Resource": "arn:aws:secretsmanager:us-
east-1:111122223333:secret:SecretName-ABCDEF"
        },
        {
            "Effect": "Allow",
            "Action": [
                "kms:Decrypt",
```

```
                "kms:DescribeKey"
            ],
            "Resource": "arn:aws:kms:us-
east-1:111122223333:key/1234abcd-12ab-34cd-56ef-1234567890ab"
        }
    ]
}
```

*secret_arn*Ecco l'ARN del tuo segreto, che puoi ottenere da entrambi `SecretsManagerSecretId` a seconda dei casi, ed *kms_key_arn* è l'ARN della AWS KMS chiave che stai usando per crittografare il tuo segreto, come nell'esempio seguente.

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": "secretsmanager:GetSecretValue",
            "Resource": "arn:aws:secretsmanager:us-
east-2:123456789012:secret:MySQLTestSecret-qeHamH"
        },
        {
             "Effect": "Allow",
             "Action": [
                        "kms:Decrypt",
                        "kms:DescribeKey"
                      ],
             "Resource": "arn:aws:kms:us-
east-2:123456789012:key/761138dc-0542-4e58-947f-4a3a8458d0fd"
        }
     ]
}
```

> **Note**
>
> Se si utilizza la chiave di crittografia predefinita creata daGestione dei segreti AWS, non è necessario specificare le AWS KMS autorizzazioni per. *kms_key_arn*

Se desideri che la tua policy fornisca l'accesso a entrambi i segreti, specifica semplicemente un oggetto risorsa JSON aggiuntivo per l'altro. *secret_arn*

4. Rivedi e crea la policy con un nome descrittivo e, facoltativamente, una descrizione.
5. Seleziona Ruoli, quindi scegli Crea ruolo.
6. Per il tipo di entità attendibile, scegli Servizio AWS.
7. Scegli DMS dall'elenco dei servizi come servizio attendibile, quindi seleziona Successivo: Autorizzazioni.
8. Cerca e collega la policy che hai creato nella fase 4, quindi procedi con l'aggiunta di eventuali tag ed esamina il ruolo. A questo punto, modifica le relazioni di trust per il ruolo in modo da utilizzare il principale del servizio regionale Amazon RDS come entità attendibile. Questo principale ha il seguente formato.

```
dms.region-name.amazonaws.com
```

Qui *region-name* è il nome della regione, ad esempio `us-east-1`. Segue quindi il principale del servizio regionale Amazon RDS per questa Regione.

```
dms.us-east-1.amazonaws.com
dms-data-migrations.amazonaws.com
```

Creazione di un ruolo IAM per AWS DMS per la gestione di Amazon VPC

Per gestire le impostazioni VPC per le risorse è necessario creare un ruolo IAM per AWS DMS. Questo ruolo deve essere disponibile per eseguire una migrazione ottimale.

Creazione di **`dms-vpc-role`** per la migrazione del database

<result>

Questa procedura crea il ruolo del DMS per gestire le impostazioni VPC per la migrazione.
</result>

1. Accedi a Console di gestione AWS e apri la console IAM all'indirizzo https://console.aws.amazon.com/iam/.
2. Nel riquadro di navigazione della console, selezionare Ruoli e Crea ruolo.
3. Scegli l'opzione Servizio AWS per l'opzione Seleziona un'entità attendibile.

Per Caso d’uso, seleziona DMS.

4. Per il passaggio Aggiungi autorizzazioni, seleziona `AmazonDMSVPCManagementRole` e scegli Successivo.
5. Nella pagina Nomina, verifica e crea imposta il Nome del ruolo su `dms-vpc-role` e scegli Crea ruolo.

Configurazione della migrazione dei dati per il database EC2

Per iniziare la migrazione dei dati del database di origine EC2, è necessario creare un database Aurora equivalente. Per istruzioni sulla creazione del database, consulta Creazione di un cluster database Amazon Aurora .

Dopo aver creato il database di destinazione, completa i seguenti passaggi per configurare la migrazione dei dati:

Configurazione del progetto di migrazione dei dati

1. Seleziona il database di destinazione nella pagina Database della console RDS.
2. Scegli il menu a discesa Operazioni e seleziona l’opzione Migra dati dal database EC2. Per i database di destinazione supportati, consulta Limitazioni.
3. Nella sezione Seleziona il database EC2 di origine:

    1. Controlla il Tipo di motore e assicurati che uguale a quello del database di origine.

        Inoltre, controlla se le versioni del motore sono compatibili.
    2. Per Istanza EC2, scegli l’istanza EC2 in cui risiede il database di origine.
    3. Per Porta, immetti la porta su cui il database di origine consente il traffico.
    4. Per Segreto, scegli Crea e usa un nuovo segreto, se non ne è disponibile uno. Immetti il Nome utente e Password per il database di origine. Scegli anche la chiave KMS con cui crittografare il segreto.

        Se disponi di un segreto esistente, seleziona Usa un segreto esistente, quindi scegli un segreto dal menu a discesa.
    5. Per il Ruolo IAM per il segreto, se disponi di un ruolo IAM esistente, seleziona Usa un ruolo IAM esistente e scegli dal menu a discesa un ruolo IAM che può accedere all’ID segreto del passaggio precedente.

Se non disponi di un ruolo IAM, scegli Crea e utilizza un nuovo ruolo IAM. Immetti il nome per il nuovo ruolo nel campo Nome ruolo IAM. Puoi vedere le autorizzazioni associate a questo ruolo nel link sottostante.

4. Nella sezione Visualizza database RDS di destinazione:

   1. Conferma le impostazioni del database di destinazione nella parte superiore della sezione.
   2. In Segreto, scegli Crea e usa un nuovo segreto se non disponi di un segreto esistente che contenga le credenziali del database di destinazione.

      Se disponi di un segreto esistente, seleziona il segreto dal menu a discesa.
   3. In Ruolo IAM per il segreto, seleziona un ruolo IAM in grado di accedere al segreto del passaggio precedente. È anche possibile creare un nuovo ruolo IAM se non si dispone di un ruolo IAM esistente.

      Se il menu a discesa non include ruoli IAM, specifica l'ARN del ruolo IAM nel formato `arn:aws:iam:`*`account_id`*`:role/`*`roleName`*.

5. Nella sezione Configura la migrazione dei dati:

   1. Seleziona il tipo di migrazione dei dati scegliendo tra A pieno carico, Pieno carico e acquisizione dei dati di modifica (CDC) o Acquisizione dei dati di modifica (CDC). Per ulteriori informazioni su queste opzioni, consulta Panoramica.

      Non è possibile modificare il tipo di migrazione dopo l'inizio della migrazione.
   2. In Ruolo IAM per la migrazione dei dati, se disponi di un ruolo IAM esistente, seleziona Usa un ruolo IAM esistente e scegli un ruolo IAM dal menu a discesa che fornisce a DMS le autorizzazioni per creare le risorse necessarie per la migrazione. Se non disponi di un ruolo IAM, scegli Crea e utilizza un nuovo ruolo IAM.

6. Verifica che la scheda Visualizza le impostazioni di migrazione mostri le impostazioni necessarie per configurare correttamente la migrazione dei dati.
7. Seleziona Migra per completare la configurazione della migrazione.

Dopo aver completato questi passaggi, è possibile vedere le risorse configurate per la migrazione dei dati scegliendo Visualizza dettagli nel banner di avanzamento della console. Una volta configurate le risorse necessarie, la migrazione viene avviata automaticamente. Se crei

Per migrare più database nel database di destinazione, inizia da capo questo processo con i dettagli sul nuovo database EC2.

Gestione delle migrazioni dei dati

Dopo aver utilizzato l'azione Migra dati dal database EC2 della console RDS, Aurora avvia automaticamente la migrazione.

Se hai utilizzato la console AWS DMS per creare le risorse di migrazione, puoi avviare il processo di migrazione.

Avvio della migrazione dei dati

Per avviare la migrazione dei dati, segui questi passaggi:

Avvio di una migrazione dei dati

1. Scegli il database di destinazione nella pagina Database della console RDS.
2. Nella pagina dei dettagli del database scegli la scheda Migrazioni di dati.
3. Nella scheda Migrazioni di dati, la sezione Migrazioni di dati associate elenca le migrazioni di dati disponibili.

   Le migrazioni configurate utilizzando la console Aurora si avviano automaticamente una volta configurate le risorse necessarie.

   Le migrazioni configurate utilizzando la console DMS sono impostate su Pronto.

   Per iniziare queste migrazioni, seleziona il menu a discesa Operazioni e seleziona Avvia.
4. La migrazione dei dati viene avviata per il database EC2.

Arresto della migrazione dei dati

Per le migrazioni di dati il cui tipo di replica è a pieno carico, l'arresto della migrazione causa l'arresto del processo che non potrà essere ripreso. Una volta arrestata, è necessario riavviare la migrazione.

Per le migrazioni con il tipo di replica impostato su Acquisizione dei dati di modifica (CDC) o A pieno carico e acquisizione dei dati di modifica (CDC), è possibile interrompere il processo di replica continua e riprenderlo in un secondo momento.

Arresto di una migrazione dei dati

1. Scegli il database di destinazione nella pagina Database della console RDS.
2. Nella pagina dei dettagli del database scegli la scheda Migrazioni di dati.
3. Nella scheda Migrazioni di dati, la sezione Migrazioni di dati associate elenca le migrazioni di dati in corso.

   Per arrestare una migrazione, seleziona una migrazione dei dati e seleziona Arresta nel menu a discesa Operazioni.
4. La migrazione dei dati viene arrestata per il database EC2.

Ripresa della migrazione dei dati

Per le migrazioni di dati il cui tipo di replica è A pieno carico e acquisizione dei dati di modifica (CDC) o Acquisizione dei dati di modifica (CDC), è possibile riprendere il processo CDC dall'ultimo punto di arresto.

Ripresa di una migrazione dei dati

1. Scegli il database di destinazione nella pagina Database della console RDS.
2. Nella pagina dei dettagli del database scegli la scheda Migrazioni di dati.
3. Nella scheda Migrazioni di dati, la sezione Migrazioni di dati associate elenca le migrazioni di dati arrestate.

   Per riprendere una migrazione, seleziona una migrazione dei dati e seleziona Riprendi l'elaborazione nel menu a discesa Operazioni.
4. La migrazione dei dati viene ripresa per il database EC2.

Eliminazione della migrazione dei dati

Utilizza le istruzioni seguenti per eliminare una migrazione dei dati associata.

Eliminazione di una migrazione dei dati

1. Scegli il database di destinazione nella pagina Database della console RDS.
2. Nella pagina dei dettagli del database scegli la scheda Migrazioni di dati.

3. Per eliminare una migrazione, seleziona una migrazione dei dati e seleziona Elimina nel menu a discesa Operazioni.
4. La migrazione dei dati viene eliminata.

L’eliminazione di una migrazione di dati in corso non ha alcun impatto sui dati già caricati nel database di destinazione.

Riavvio della migrazione dei dati

Utilizza le seguenti istruzioni per riavviare una migrazione di dati associata da un punto di inizio CDC

Riavvio di una migrazione dei dati

1. Scegli il database di destinazione nella pagina Database della console RDS.
2. Nella pagina dei dettagli del database scegli la scheda Migrazioni di dati.
3. Per riavviare una migrazione, seleziona una migrazione dei dati e seleziona Riavvia nel menu a discesa Operazioni.
4. La migrazione dei dati viene riavviata da un punto di inizio CDC.

Il riavvio di una migrazione di dati in corso non ha alcun impatto sui dati già caricati nel database di destinazione.

Monitoraggio delle migrazioni dei dati

Dopo aver avviato la migrazione dei dati, è possibile monitorarne lo stato e l’avanzamento. Le migrazioni di set di dati di grandi dimensioni richiedono ore per essere completate. Per mantenere l'affidabilità, la disponibilità e le prestazioni elevate della migrazione dei dati, è necessario monitorarne regolarmente l'avanzamento.

Per verificare lo stato e l'avanzamento della migrazione dei dati

1. Scegli il database di destinazione nella pagina Database della console RDS.
2. Nella pagina dei dettagli del database scegli la scheda Migrazioni di dati.
3. La sezione Migrazioni di dati associate elenca le migrazioni dei dati. Controlla la colonna Stato.
4. Per le migrazioni dei dati in corso, la colonna Stato di avanzamento della migrazione mostra la percentuale di dati migrati.

5. Per monitorare il processo in CloudWatch, utilizza il collegamento presente nella colonna CloudWatch.

Stati di migrazione

Per ogni migrazione di dati eseguita, la console Aurora visualizza lo Stato. L'elenco seguente include i possibili stati:

- `Ready`: la migrazione dei dati è pronta per iniziare.
- `Starting`: Aurora sta creando l'ambiente serverless per la migrazione dei dati.
- `Load running`: Aurora sta eseguendo la migrazione a pieno carico.
- `Load complete, replication ongoing`: Aurora ha completato la migrazione a pieno carico e ora replica le modifiche in corso. Questo stato si applica solo alle migrazioni di tipo A pieno carico e acquisizione dei dati di modifica (CDC).
- `Replication ongoing`: Aurora replica le modifiche in corso. Questo stato si applica solo a migrazioni di tipo Acquisizione dei dati di modifica (CDC).
- `Stopping`: Aurora arresta le migrazioni dei dati. Questo stato si applica quando si sceglie di arrestare la migrazione dei dati dal menu Operazioni.
- `Stopped`: Aurora ha arrestato la migrazione dei dati.
- `Failed`: la migrazione dei dati non è riuscita. Per ulteriori informazioni, consulta i file di log.
- `Restarting`: la migrazione dei dati ha riavviato una replica continua dei dati da un punto di inizio CDC.

Tutorial: creazione di un cluster di database MySQL con un gruppo di parametri personalizzati

In questo tutorial si crea un cluster di database MySQL con un gruppo di parametri personalizzati. Per ulteriori informazioni sui gruppi di parametri , consulta Gruppi di parametri del cluster di database per i cluster di database Amazon Aurora.

> **Important**
>
> Non vi sono costi aggiuntivi per la creazione di un account AWS. Tuttavia, completando l'esercitazione, potresti incorrere in costi per le risorse AWS utilizzate. È possibile eliminare queste risorse dopo aver completato l'esercitazione se non sono più necessarie.

Per creare un cluster di database con configurazioni e impostazioni personalizzate, è possibile utilizzare parametri personalizzati di opzioni. I parametri personalizzati sono particolarmente utili se si lavora con più database e si desidera configurare le relative impostazioni in modo uniforme.

Completando questi passaggi, imparerai:

- Come utilizzare Amazon Aurora per creare un cluster di database MySQL con un gruppo di parametri personalizzati.
- Come utilizzare parametri specifici per cluster di database MySQL.

Per completare questo tutorial, esegui queste attività:

1. Creare un gruppo di parametri di cluster di database con il parametro `default_password_lifetime` MySQL.
2. Crea un cluster di database MySQL con il gruppo di parametri del cluster di database che hai creato.

Argomenti

- Prerequisiti
- Creazione di un gruppo di parametri del cluster di database Amazon Aurora
- Modifica del valore del parametro nel gruppo di parametri personalizzato

- Creazione di un cluster di database MySQL con un gruppo di parametri di cluster di database

Prerequisiti

Questo tutorial richiede un Account AWS e un utente con accesso amministrativo. Se non sono già stati configurati, completare i passaggi delle seguenti sezioni:

- Registrati per un Account AWS
- Crea un utente con accesso amministrativo

Creazione di un gruppo di parametri del cluster di database Amazon Aurora

In questo tutorial si apprende come creare un gruppo di parametri personalizzato con default_password_lifetime per un cluster di database MySQL nella console. Il parametro `default_password_lifetime` controlla il numero di giorni prima della scadenza automatica della password del client. Per ulteriori informazioni su altri parametri disponibili per cluster di database MySQL, consulta Parametri di configurazione Aurora MySQL.

Per creare un gruppo di parametri

1. Apri la console Amazon RDS e scegli Gruppi di parametri.
2. In Gruppi di parametri personalizzati scegli Crea gruppo di parametri.
3. Imposta i dettagli del gruppo di parametri.
    1. Immetti un nome per il gruppo di parametri.
    2. Immetti una descrizione del gruppo di parametri.
    3. Per Tipo di motore, scegli Aurora MySQL.
    4. Per Famiglia del gruppo di parametri, scegli aurora-mysql8.0.
    5. Per Tipo, scegli Gruppo di parametri del cluster di database.
4. Scegli Crea.

Il nuovo gruppo di parametri cluster di database viene visualizzato nella pagina Gruppi di parametri della console Amazon RDS. I passaggi seguenti illustrano come modificare i valori dei parametri per personalizzare il gruppo di parametri.

Modifica del valore del parametro nel gruppo di parametri personalizzato

I seguenti passaggi consentono di modificare i valori dei parametri nel gruppo di parametri creato in Creazione di un gruppo di parametri del cluster di database Amazon Aurora.

Per modificare i valori dei parametri nel gruppo di parametri

1. Apri la console Amazon RDS e scegli Gruppi di parametri.
2. In Gruppi di parametri personalizzati, scegli il nome del gruppo di parametri cluster di database creato.
3. Scegli Modifica.
4. Nella casella di ricerca Filtra parametri, cerca il parametro personalizzato `default_password_lifetime`.
5. Seleziona la casella di controllo accanto al parametro e inserisci un valore che indica il numero di giorni da impostare per questo parametro della durata della password.
6. Seleziona Salva modifiche.

Il gruppo di parametri personalizzati è ora disponibile per l'associazione ad Amazon Aurora per il cluster di database MySQL 8.0.

Creazione di un cluster di database MySQL con un gruppo di parametri di cluster di database

Infine, si crea un cluster di database MySQL con il parametro personalizzato creato nei passaggi precedenti. I passaggi seguenti mostrano come creare un cluster di database MySQL con il parametro personalizzato .

Per creare un cluster di database con un parametro personalizzato e un nuovo gruppo di opzioni

1. Apri la console Amazon RDS e scegli Database.
2. Scegliere Crea database.
3. In Scegli un metodo di creazione del database seleziona Creazione standard.
4. In Opzioni motore, scegli Aurora (compatibile con MySQL).
5. Seleziona Configurazione aggiuntiva.
    - In Nome database iniziale, scegli un nome per cluster di database.

- Nel menu a discesa Gruppo di parametri cluster di database, seleziona il nome del gruppo di parametri personalizzato creato in precedenza.

6. Per questo tutorial, è possibile lasciare le impostazioni predefinite per qualsiasi altra impostazione del database o modificarle come necessario.
7. Scegliere Crea database.

RDS crea un nuovo cluster di database MySQL con un gruppo di parametri personalizzato di opzioni. Per ulteriori informazioni su questo database, consulta la pagina Database della console Amazon RDS.

In questo tutorial, è stata eseguita la configurazione di un cluster di database MySQL con impostazioni personalizzate utilizzando un parametro personalizzato . Il cluster di database appena creato gestisce la durata della password dell'utente utilizzando il parametro `default_password_lifetime`. Per ottimizzare il database, è possibile applicare impostazioni aggiuntive nel gruppo di parametri personalizzato e aggiungere opzioni.

Dopo aver completato la creazione di un cluster di database personalizzato, è necessario eliminare le risorse per evitare di incorrere in costi indesiderati. Per eliminare un cluster di database, seguire le istruzioni riportate in Eliminazione di cluster e istanze database di Aurora.

Gestione di un cluster DB Amazon Aurora

In questa sezione viene descritto come gestire e mantenere il cluster di database Aurora. Aurora opera su cluster di server di database che sono collegati in una topologia di replica. Di conseguenza, la gestione di Aurora spesso implica l'implementazione di modifiche a più server e la necessità di accertarsi che tutte le repliche di Aurora siano in grado di mantenere il passo con il server di origine. Poiché Aurora dimensiona in modo trasparente lo storage sottostante in base alla crescita dei dati, la gestione di Aurora richiede relativamente poca gestione dello storage su disco. Analogamente, dato che Aurora esegue backup continui in automatico, un cluster Aurora non richiede particolari pianificazioni o tempi di inattività per l'esecuzione dei backup.

Argomenti

- Avvio e arresto di un cluster di database Amazon Aurora
- Connessione automatica di un'istanza EC2 e di un cluster database Aurora
- Connessione automatica di una funzione Lambda e di un cluster database Aurora
- Modifica di un cluster database Amazon Aurora
- Aggiunta di repliche di Aurora a un cluster di database
- Gestione delle prestazioni e del dimensionamento dei cluster DB Aurora
- Clonazione di un volume per un cluster di database Amazon Aurora
- Integrazione di Aurora con altri servizi AWS
- Manutenzione di un cluster di database Amazon Aurora
- Riavvio di un cluster Amazon Aurora DB o di un'istanza Amazon Aurora DB
- Failover di un cluster di database Amazon Aurora
- Eliminazione di cluster e istanze database di Aurora
- Applicazione di tag alle risorse Amazon Aurora e Amazon RDS
- Nome della risorsa Amazon (ARN) in Amazon RDS
- Aggiornamenti di Amazon Aurora

Avvio e arresto di un cluster di database Amazon Aurora

Avviare e arrestare i cluster di database Aurora aiuta a gestire i costi degli ambienti di test e sviluppo. Puoi arrestare temporaneamente tutte le istanze database nel cluster invece di impostare e rimuovere tutte le istanze database ogni volta che utilizzi il cluster.

Argomenti

- Panoramica dell'avvio e dell'arresto di un cluster di database Aurora
- Limitazioni per l'arresto e l'avvio di cluster di database Aurora
- Arresto di un cluster di database Aurora
- Operazioni possibili durante l'arresto di un cluster di database Aurora
- Avvio di un cluster di database Aurora

Panoramica dell'avvio e dell'arresto di un cluster di database Aurora

Nei periodi in cui non hai bisogno di un cluster di database Aurora, puoi arrestare contemporaneamente tutte le istanze nel cluster. Puoi avviare nuovamente il cluster ogni volta che devi utilizzarlo. L'avvio e l'arresto semplificano i processi di impostazione e rimozione dei cluster utilizzati per lo sviluppo, i test o attività simili che non richiedono una disponibilità continua. Puoi eseguire tutte le procedure di Console di gestione AWS coinvolte con una sola operazione, indipendentemente dalla quantità di istanze presenti nel cluster.

Per tutta la durata della sospensione di un cluster di database, vengono addebitati solo i costi per l’archiviazione del cluster, gli snapshot manuali e l’archiviazione di backup automatici all’interno della finestra di conservazione specificata. Non è previsto alcun addebito per le ore dell'istanza database.

> **Important**
>
> È possibile arrestare un cluster di database per un massimo di sette giorni. Se non si avvia manualmente il cluster di database dopo sette giorni, esso viene avviato automaticamente in modo che non resti indietro rispetto agli aggiornamenti di manutenzione necessari.

Per ridurre gli addebiti per un cluster Aurora con poco carico, puoi arrestarlo anziché eliminarne tutte le repliche di Aurora. Per i cluster che hanno più di una o due istanze, eliminare frequentemente le istanze database e ricrearle è pratico solo se si utilizza AWS CLI o l'API di Amazon RDS.

Questa sequenza di operazioni può anche essere difficile da eseguire nell'ordine corretto, ad esempio eliminare tutte le repliche di Aurora prima dell'istanza primaria per evitare l'attivazione del meccanismo di failover.

Non utilizzare l'avvio e l'arresto se il cluster di database deve rimanere in esecuzione ma ha più capacità di quanta ne occorre. Se il cluster è troppo costoso o non molto impiegato, elimina una o più istanze database o modifica tutte le istanze database in una classe di istanze small. Non puoi arrestare una sola istanza database di Aurora.

Il tempo necessario per arrestare il cluster di database varia a seconda di fattori quali le classi di istanze database, lo stato della rete, il tipo di motore di database e lo stato del database. Il processo può richiedere alcuni minuti. Il servizio Amazon RDS esegue le operazioni seguenti:

- Arresta i processi del motore di database.
- Arresta i processi della piattaforma RDS.
- Termina le istanze Amazon EC2 sottostanti.

Il tempo necessario per riavviare il cluster di database varia a seconda di fattori quali la dimensione del database, le classi di istanze database, lo stato della rete, il tipo di motore di database e lo stato del database al momento dell'arresto del cluster. Il processo di avvio può richiedere fino a qualche ora, ma in genere dura alcuni minuti. Durante la creazione del piano di disponibilità, è opportuno tenere in considerazione la variabilità nel tempo di avvio.

Per avviare il cluster di database, il servizio esegue azioni come le seguenti:

- Effettua il provisioning delle istanze Amazon EC2 sottostanti.
- Avvia i processi della piattaforma RDS.
- Avvia i processi del motore di database.
- Ripristina le istanze database (il ripristino avviene anche dopo un normale arresto).

Limitazioni per l'arresto e l'avvio di cluster di database Aurora

Non è possibile arrestare e avviare alcuni cluster Aurora:

- È possibile arrestare e avviare un cluster parte di un Database globale Aurora solo se è l'unico cluster nel database globale.
- Non è possibile arrestare e avviare un cluster con una replica di lettura tra Regioni.

- Non è possibile arrestare e avviare un cluster che fa parte di un'implementazione blu/verde.
- Non è possibile arrestare e avviare un cluster Aurora Serverless v1. Con Aurora Serverless v2, è possibile arrestare e avviare un cluster.

Arresto di un cluster di database Aurora

Per utilizzare o amministrare un cluster di database di Aurora, si inizia sempre da un cluster di database Aurora in esecuzione, poi si arresta il cluster e lo si avvia di nuovo. Per tutta la durata dell'arresto di un cluster, vengono addebitati i costi per lo storage del cluster, gli snapshot manuali e lo storage di backup automatici all'interno della finestra di retention specificata, ma non per le ore di utilizzo dell'istanza database.

L'operazione di arresto interrompe prima le istanze di replica di Aurora e poi l'istanza primaria, per evitare l'attivazione del meccanismo di failover.

Console

Per arrestare un cluster Aurora

1. Accedi alla Console di gestione AWS e apri la console Amazon RDS all'indirizzo https://console.aws.amazon.com/rds/.
2. Nel pannello di navigazione, scegliere Databases (Database), quindi scegliere un cluster. Si può eseguire l'operazione di arresto da questa pagina o andare alla pagina dei dettagli del cluster di database da arrestare.
3. Per Actions (Operazioni), scegli Stop temporarily (Arresta temporaneamente).
4. Nella finestra Stop DB cluster temporarily, (Arresta temporaneamente il cluster di database) seleziona la conferma per il riavvio automatico del cluster database dopo 7 giorni.
5. Scegli Stop temporarily (Arresta temporaneamente) per arrestare il cluster database oppure Cancel (Annulla) per annullare l'operazione.

AWS CLI

Per arrestare un'istanza database tramite AWS CLI, chiamare il comando stop-db-cluster con i parametri seguenti:

- `--db-cluster-identifier` – il nome del cluster Aurora.

Example

```
aws rds stop-db-cluster --db-cluster-identifier mydbcluster
```

API RDS

Per arrestare un'istanza database tramite l'API Amazon RDS, chiamare l'operazione StopDBCluster con il parametro seguente:

- `DBClusterIdentifier` – il nome del cluster Aurora.

Operazioni possibili durante l'arresto di un cluster di database Aurora

Quando un cluster Aurora è in fase di arresto, è possibile eseguire ripristini point-in-time nella finestra di retention di backup automatica specificata. Per i dettagli su come eseguire un ripristino point-in-time, consulta Ripristino dei dati.

Non è possibile modificare la configurazione di un cluster di database Aurora o delle istanze database durante l'arresto del cluster. Inoltre, non è possibile aggiungere o rimuovere istanze database dal cluster o eliminarlo se ha ancora istanze database associate. Devi avviare il cluster prima di eseguire questo tipo di operazioni amministrative.

L'arresto di un cluster di database rimuove le azioni in sospeso, ad eccezione del gruppo di parametri del cluster di database o dei gruppi di parametri del database delle istanze del cluster di database.

Aurora applica la manutenzione pianificata al cluster in arresto dopo il suo riavvio. Ricorda che dopo sette giorni Aurora avvia automaticamente i cluster in arresto in modo che non rimangano troppo indietro rispetto allo stato di manutenzione.

Inoltre, Aurora non esegue backup automatici, perché i dati sottostanti non possono cambiare nel periodo di arresto del cluster. Aurora non estende il tempo di conservazione del backup per il cluster database mentre è arrestato.

Avvio di un cluster di database Aurora

Puoi sempre avviare un cluster di database Aurora partendo da un cluster Aurora che è già nello stato di arresto. Quando avvii il cluster, tutte le sue istanze database tornano disponibili. Il cluster mantiene le sue impostazioni di configurazione, come gli endpoint, i gruppi di parametri e i gruppi di sicurezza VPC.

L'avvio di un cluster di database richiede in genere alcuni minuti.

Console

Per avviare un cluster Aurora

1. Accedi alla Console di gestione AWS e apri la console Amazon RDS all'indirizzo https://console.aws.amazon.com/rds/.
2. Nel pannello di navigazione, scegliere Databases (Database), quindi scegliere un cluster. Si può eseguire l'operazione di avvio da questa pagina o andare alla pagina dei dettagli del cluster di database da avviare.
3. In Actions (Operazioni), scegliere Start (Avvia).

AWS CLI

Per avviare un cluster di database tramite AWS CLI, chiamare il comando start-db-cluster con i parametri seguenti:

- `--db-cluster-identifier` – il nome del cluster Aurora. Questo nome è sia lo specifico identificatore del cluster scelto durante la sua creazione sia l'identificatore istanze database scelto con `-cluster` aggiunto alla fine.

Example

```
aws rds start-db-cluster --db-cluster-identifier mydbcluster
```

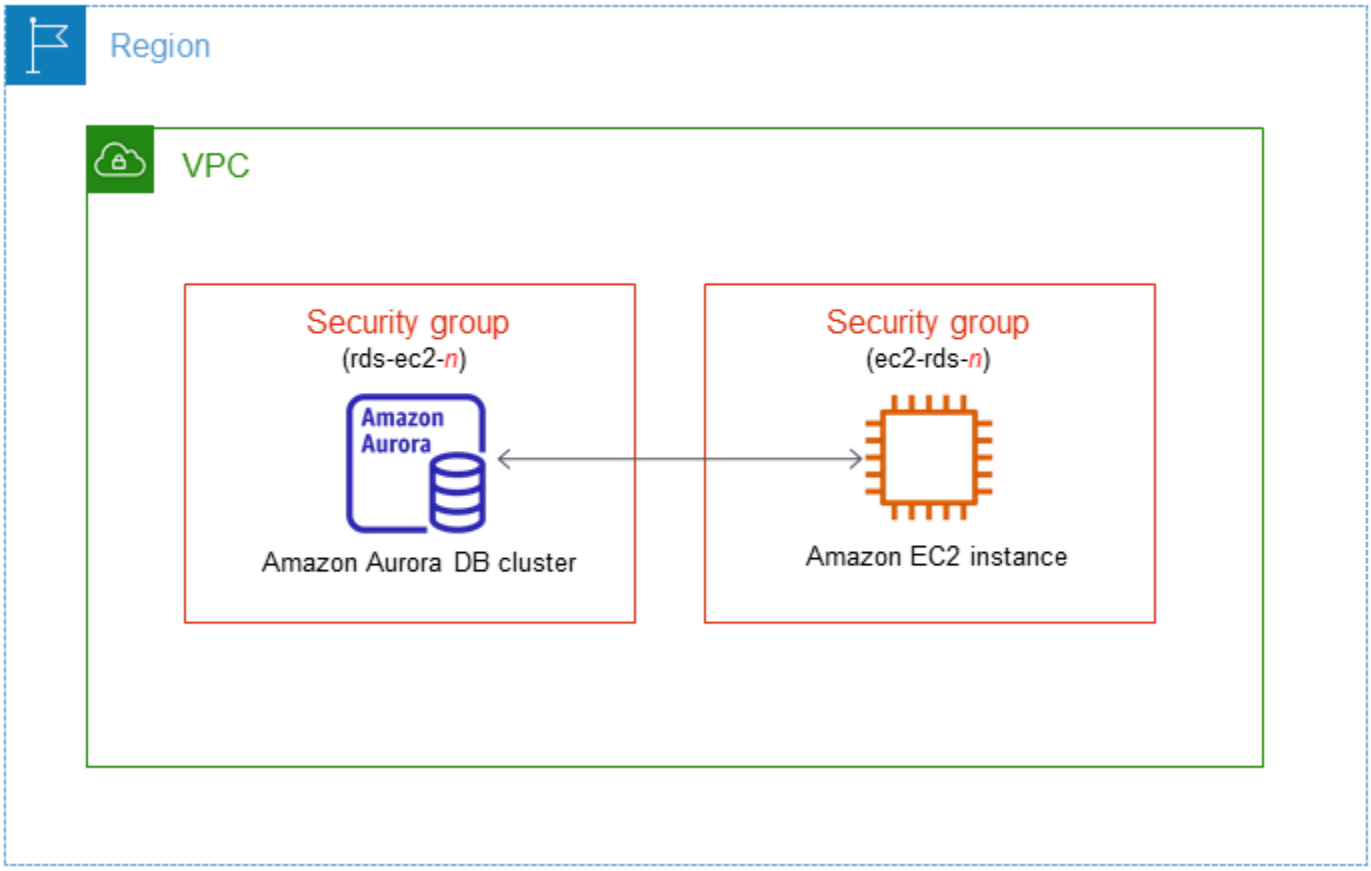
API RDS

Per avviare un cluster di database Aurora tramite l'API Amazon RDS, chiamare l'operazione StartDBCluster con il parametro seguente:

- `DBCluster` – il nome del cluster Aurora. Questo nome è sia lo specifico identificatore del cluster scelto durante la sua creazione sia l'identificatore istanze database scelto con `-cluster` aggiunto alla fine.

Connessione automatica di un'istanza EC2 e di un cluster database Aurora

È possibile usare la console Amazon RDS per semplificare l'impostazione di una connessione tra un'istanza Amazon Elastic Compute Cloud (Amazon EC2) e un cluster database Aurora. Spesso, il cluster database si trova in una sottorete privata e l'istanza EC2 si trova in una sottorete pubblica all'interno di un VPC. Puoi utilizzare un client SQL nell'istanza EC2 per effettuare la connessione al cluster di database. L'istanza EC2 può anche eseguire server web o applicazioni che accedono al cluster di database privato.



Se desideri connetterti a un'istanza EC2 che non si trova nello stesso VPC del cluster di database Aurora, consulta gli scenari in Scenari per accedere a un cluster database in un VPC.

Argomenti

- Panoramica della connettività automatica con un'istanza EC2
- Connessione automatica di un'istanza EC2 e di un cluster database Aurora
- Visualizzazione delle risorse di calcolo connesse

- Connessione a un'istanza database che esegue un motore DB specifico

Panoramica della connettività automatica con un'istanza EC2

Quando configuri una connessione tra un'istanza EC2 e un cluster database Aurora, Amazon RDS configura automaticamente il gruppo di sicurezza VPC per l'istanza EC2 e per il cluster database.

Di seguito sono riportati i requisiti per connettere un'istanza EC2 a un cluster database Aurora:

- L'istanza EC2 deve risiedere nello stesso VPC del cluster database.

  Se nello stesso VPC non esistono istanze EC2, allora la console fornisce un collegamento per crearne una.
- Attualmente, il cluster database non può essere un cluster database Aurora Serverless o parte di un database globale Aurora.
- L'utente che configura la connettività deve disporre delle autorizzazioni per eseguire le seguenti operazioni Amazon EC2:
  - `ec2:AuthorizeSecurityGroupEgress`
  - `ec2:AuthorizeSecurityGroupIngress`
  - `ec2:CreateSecurityGroup`
  - `ec2:DescribeInstances`
  - `ec2:DescribeNetworkInterfaces`
  - `ec2:DescribeSecurityGroups`
  - `ec2:ModifyNetworkInterfaceAttribute`
  - `ec2:RevokeSecurityGroupEgress`

Se l'istanza database e l'istanza EC2 si trovano in zone di disponibilità diverse, è possibile che all'account vengano addebitati costi tra zone di disponibilità.

Quando si configura una connessione a un'istanza EC2, Amazon RDS opera in base alla configurazione corrente dei gruppi di sicurezza associati al cluster database e all'istanza EC2, come descritto nella tabella seguente.

Configurazione del gruppo di sicurezza RDS corrente	Configurazione del gruppo di sicurezza EC2 corrente	Operazione RDS
Esistono uno o più gruppi di sicurezza associati al cluster database con un nome che corrisponde al modello `rds-ec2-`*n* (dove *n* è un numero). Un gruppo di sicurezza che corrisponde al modello non è stato modificato. Questo gruppo di sicurezza include una sola regola in uscita con il gruppo di sicurezza VPC dell'istanza EC2 come origine.	Esistono uno o più gruppi di sicurezza associati all'istanza EC2 con un nome che corrisponde al modello `ec2-rds-`*n* (dove *n* è un numero). Un gruppo di sicurezza che corrisponde al modello non è stato modificato. Questo gruppo di sicurezza ha una sola regola in uscita con il gruppo di sicurezza VPC del cluster database come origine.	RDS non esegue alcuna operazione. Una connessione è già configurata automaticamente tra l'istanza EC2 e il cluster database. Poiché esiste già una connessione tra l'istanza EC2 e il database RDS, i gruppi di sicurezza non vengono modificati.
Si applica una delle due condizioni seguenti: • Non esiste un gruppo di sicurezza associato al cluster database con un nome che corrisponde al modello `rds-ec2-`*n*. • Esistono uno o più gruppi di sicurezza associati al cluster database con un nome che corrisponde al modello `rds-ec2-`*n*. Tuttavia, Amazon RDS non può utilizzare nessuno di questi gruppi di sicurezza per la connessione all'istanza EC2. Amazon RDS non può utilizzare un gruppo di sicurezza che non dispone di una regola in entrata	Si applica una delle due condizioni seguenti: • Non esiste un gruppo di sicurezza associato all'istanza EC2 con un nome che corrisponde al modello `ec2-rds-`*n*. • Esistono uno o più gruppi di sicurezza associati all'istanza EC2 con un nome che corrisponde al modello `ec2-rds-`*n*. Tuttavia, Amazon RDS non può utilizzare nessuno di questi gruppi di sicurezza per la connessione al cluster database. Amazon RDS non può utilizzare un gruppo di sicurezza che non dispone di una regola in uscita con	RDS action: create new security groups

Configurazione del gruppo di sicurezza RDS corrente	Configurazione del gruppo di sicurezza EC2 corrente	Operazione RDS
con il gruppo di sicurezza VPC dell'istanza EC2 come origine. Inoltre, Amazon RDS non può utilizzare un gruppo di sicurezza che è stato modificato. Esempi di modifiche sono l'aggiunta di una regola o la modifica della porta di una regola esistente.	il gruppo di sicurezza VPC del cluster database come origine. Inoltre, Amazon RDS non può utilizzare un gruppo di sicurezza che è stato modificato.	
Esistono uno o più gruppi di sicurezza associati al cluster database con un nome che corrisponde al modello `rds-ec2-`*n*. Un gruppo di sicurezza che corrispon de al modello non è stato modificato. Questo gruppo di sicurezza include una sola regola in uscita con il gruppo di sicurezza VPC dell'istanza EC2 come origine.	Esistono uno o più gruppi di sicurezza associati all'istan za EC2 con un nome che corrisponde al modello `ec2-rds-`*n*. Tuttavia, Amazon RDS non può utilizzare nessuno di questi gruppi di sicurezza per la connessione al cluster database. Amazon RDS non può utilizzare un gruppo di sicurezza che non dispone di una regola in uscita con il gruppo di sicurezza VPC del cluster database come origine. Inoltre, Amazon RDS non può utilizzare un gruppo di sicurezza che è stato modificat o.	RDS action: create new security groups

Configurazione del gruppo di sicurezza RDS corrente	Configurazione del gruppo di sicurezza EC2 corrente	Operazione RDS
Esistono uno o più gruppi di sicurezza associati al cluster database con un nome che corrisponde al modello `rds-ec2-`*n*. Un gruppo di sicurezza che corrispon de al modello non è stato modificato. Questo gruppo di sicurezza include una sola regola in uscita con il gruppo di sicurezza VPC dell'istanza EC2 come origine.	Esiste un gruppo di sicurezza EC2 valido per la connessio ne, ma non è associato all'istanza EC2. Questo gruppo di sicurezza ha un nome che corrisponde al modello `ec2-rds-`*n*. Non è stato modificato. Dispone di una sola regola in uscita con il gruppo di sicurezza VPC del cluster database come origine.	RDS action: associate EC2 security group

Configurazione del gruppo di sicurezza RDS corrente	Configurazione del gruppo di sicurezza EC2 corrente	Operazione RDS
Si applica una delle due condizioni seguenti: • Non esiste un gruppo di sicurezza associato al cluster database con un nome che corrisponde al modello `rds-ec2-`*n*. • Esistono uno o più gruppi di sicurezza associati al cluster database con un nome che corrisponde al modello `rds-ec2-`*n*. Tuttavia, Amazon RDS non può utilizzare nessuno di questi gruppi di sicurezza per la connessione all'istanza EC2. Amazon RDS non può utilizzare un gruppo di sicurezza che non dispone di una regola in entrata con il gruppo di sicurezza VPC dell'istanza EC2 come origine. Inoltre, Amazon RDS non può utilizzare un gruppo di sicurezza che è stato modificato.	Esistono uno o più gruppi di sicurezza associati all'istanza EC2 con un nome che corrisponde al modello `ec2-rds-`*n*. Un gruppo di sicurezza che corrisponde al modello non è stato modificato. Questo gruppo di sicurezza ha una sola regola in uscita con il gruppo di sicurezza VPC del cluster database come origine.	RDS action: create new security groups

RDS: creazione di nuovi gruppi di sicurezza

Amazon RDS esegue le seguenti operazioni:

- Crea un nuovo gruppo di sicurezza che corrisponde al modello `rds-ec2-`*n*. Questo gruppo di sicurezza include una regola in uscita con il gruppo di sicurezza VPC dell'istanza EC2 come

origine. Questo gruppo di sicurezza è associato al cluster database e consente all'istanza EC2 di accedere al cluster database.

- Crea un nuovo gruppo di sicurezza che corrisponde al modello `ec2-rds-`*n*. Questo gruppo di sicurezza include una regola in uscita con il gruppo di sicurezza VPC del cluster di database come destinazione. Questo gruppo di sicurezza è associato all'istanza EC2 e consente all'istanza EC2 di inviare il traffico al cluster database.

Azione RDS: associazione del gruppo di sicurezza EC2

Amazon RDS associa il gruppo di sicurezza EC2 esistente, valido all'istanza EC2. Questo gruppo di sicurezza consente all'istanza EC2 di inviare traffico al cluster database.

Connessione automatica di un'istanza EC2 e di un cluster database Aurora

Prima di configurare una connessione tra un'istanza EC2 e un cluster database Aurora assicurati di aver soddisfatto i requisiti descritti in Panoramica della connettività automatica con un'istanza EC2.

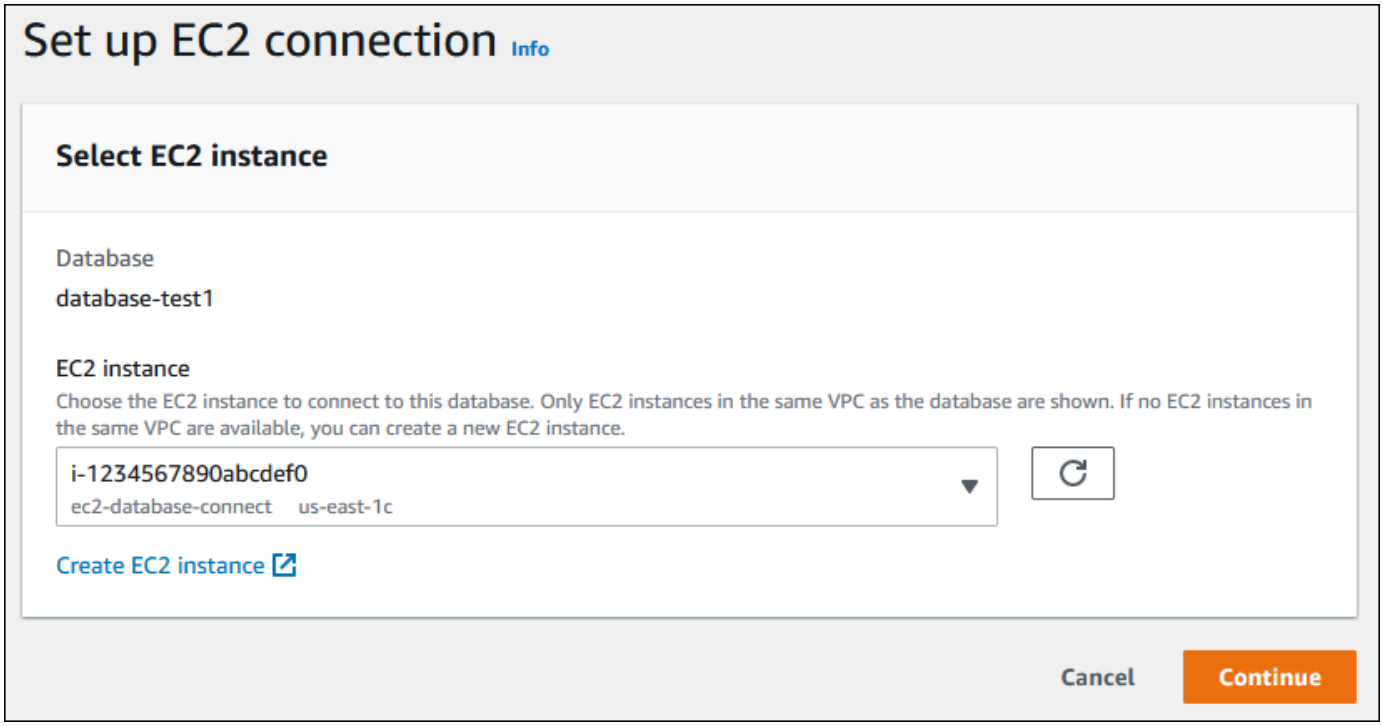
Se modifichi i gruppi di sicurezza dopo avere configurato la connettività, le modifiche potrebbero influenzare la connessione tra l'istanza EC2 e il cluster di database Aurora.

> **Note**
>
> È possibile configurare automaticamente una connessione tra un'istanza EC2 e un cluster database Aurora solo utilizzando la Console di gestione AWS. Non è possibile configurare automaticamente una connessione con AWS CLI o l'API RDS.

Per connettere automaticamente un'istanza EC2 e un cluster database Aurora

1. Accedi alla Console di gestione AWS e apri la console Amazon RDS all'indirizzo https://console.aws.amazon.com/rds/.
2. Nel riquadro di navigazione, scegli Databases (Database), quindi seleziona il cluster database.
3. In Operazioni, scegli Configura connessione EC2.

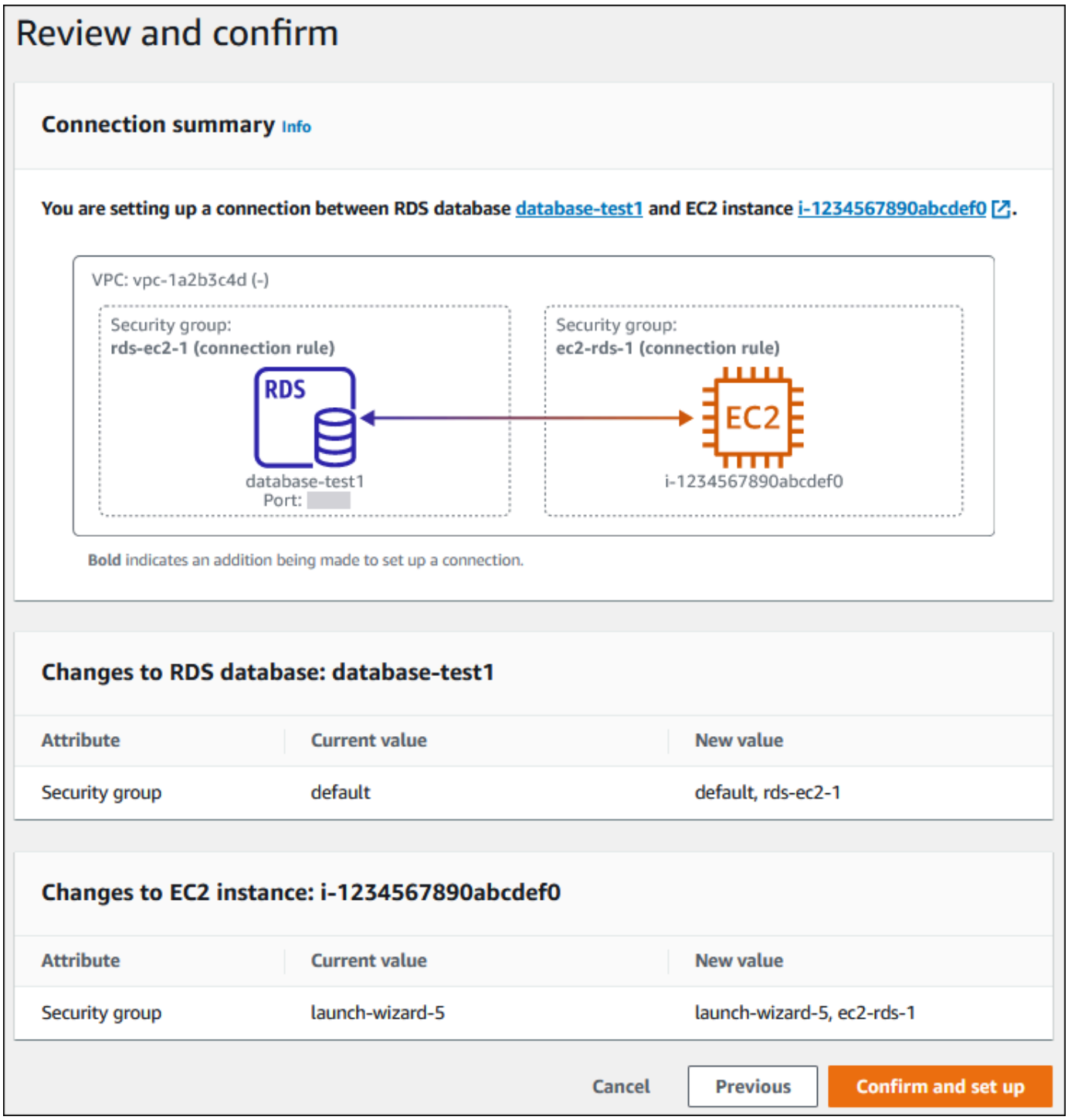
   Viene visualizzata la pagina Set up EC2 connection (Configura connessione EC2).
4. Nella pagina Set up EC2 connection (Configura connessione EC2), scegli l'istanza EC2.



Se nello stesso VPC non esistono istanze EC2, scegli Create EC2 instance (Crea istanza EC2) per crearne una. In questo caso, assicurati che la nuova istanza EC2 si trovi nello stesso VPC del cluster database.

5. Scegli Continue (Continua).

   Viene visualizzata la pagina Review and confirm (Rivedi e conferma).



6. Nella pagina Review and confirm (Rivedi e conferma), esamina le modifiche che RDS apporterà per configurare la connettività con l'istanza EC2.

   Se le modifiche sono corrette, scegli Conferma e configura.

   Se le modifiche non sono corrette, scegli Previous (Precedente) o Cancel (Annulla).

Visualizzazione delle risorse di calcolo connesse

È possibile utilizzare la Console di gestione AWS per visualizzare le risorse di calcolo connesse a un cluster di database Aurora. Le risorse mostrate includono le connessioni delle risorse di calcolo configurate automaticamente. È possibile configurare la connettività delle risorse di calcolo automaticamente nei modi seguenti:

- È possibile selezionare la risorsa di calcolo quando si crea il database.

  Per ulteriori informazioni, consulta Creazione di un cluster database Amazon Aurora.
- È possibile configurare la connettività tra un database esistente e una risorsa di calcolo.

  Per ulteriori informazioni, consulta Connessione automatica di un'istanza EC2 e di un cluster database Aurora.

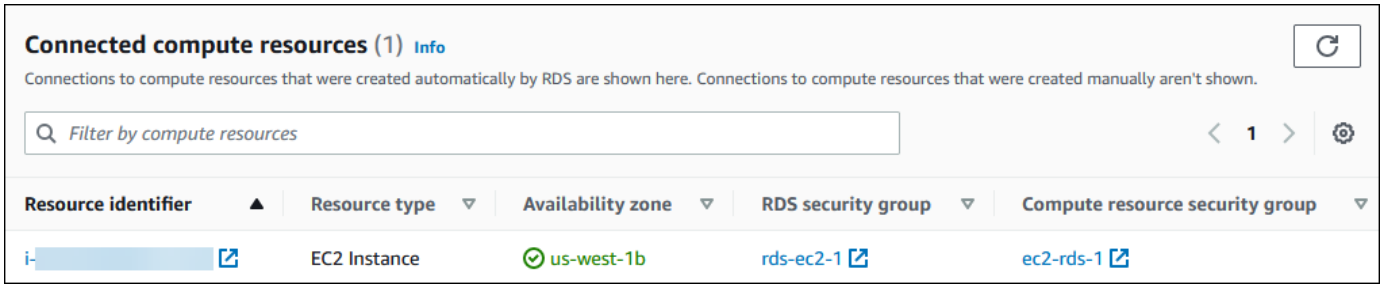
Le risorse di calcolo elencate non includono quelle connesse al database manualmente. Ad esempio, è possibile consentire manualmente a una risorsa di calcolo di accedere a un database aggiungendo una regola al gruppo di sicurezza VPC associato al database.

Per garantire la presenza della risorsa di calcolo nell'elenco, è necessario che siano soddisfatte le condizioni elencate di seguito.

- Il nome del gruppo di sicurezza associato alla risorsa di calcolo corrisponde al modello `ec2-rds-`*n* (dove *n* è un numero).
- Il gruppo di sicurezza associato alla risorsa di calcolo ha una regola in uscita con l'intervallo di porte impostato sulla porta utilizzata dal cluster database.
- Il gruppo di sicurezza associato alla risorsa di calcolo ha una regola in uscita con l'origine impostata su un gruppo di sicurezza associato al cluster database.
- Il nome del gruppo di sicurezza associato al cluster database corrisponde al modello `rds-ec2-`*n* (dove *n* è un numero).
- Il gruppo di sicurezza associato al cluster database ha una regola in entrata con l'intervallo di porte impostato sulla porta utilizzata dal cluster database.
- Il gruppo di sicurezza associato al cluster database ha una regola in entrata con l'origine impostata su un gruppo di sicurezza associato alla risorsa di calcolo.

Per visualizzare le risorse di calcolo connesse a un cluster database Aurora

1. Accedi alla Console di gestione AWS e apri la console Amazon RDS all'indirizzo https://console.aws.amazon.com/rds/.
2. Nel riquadro di navigazione, scegli Databases (Database), quindi seleziona il nome del cluster database.
3. Nella scheda Connectivity & security (Connettività e sicurezza), visualizza le risorse di calcolo in Connected compute resources (Risorse di calcolo connesse).



Connessione a un'istanza database che esegue un motore DB specifico

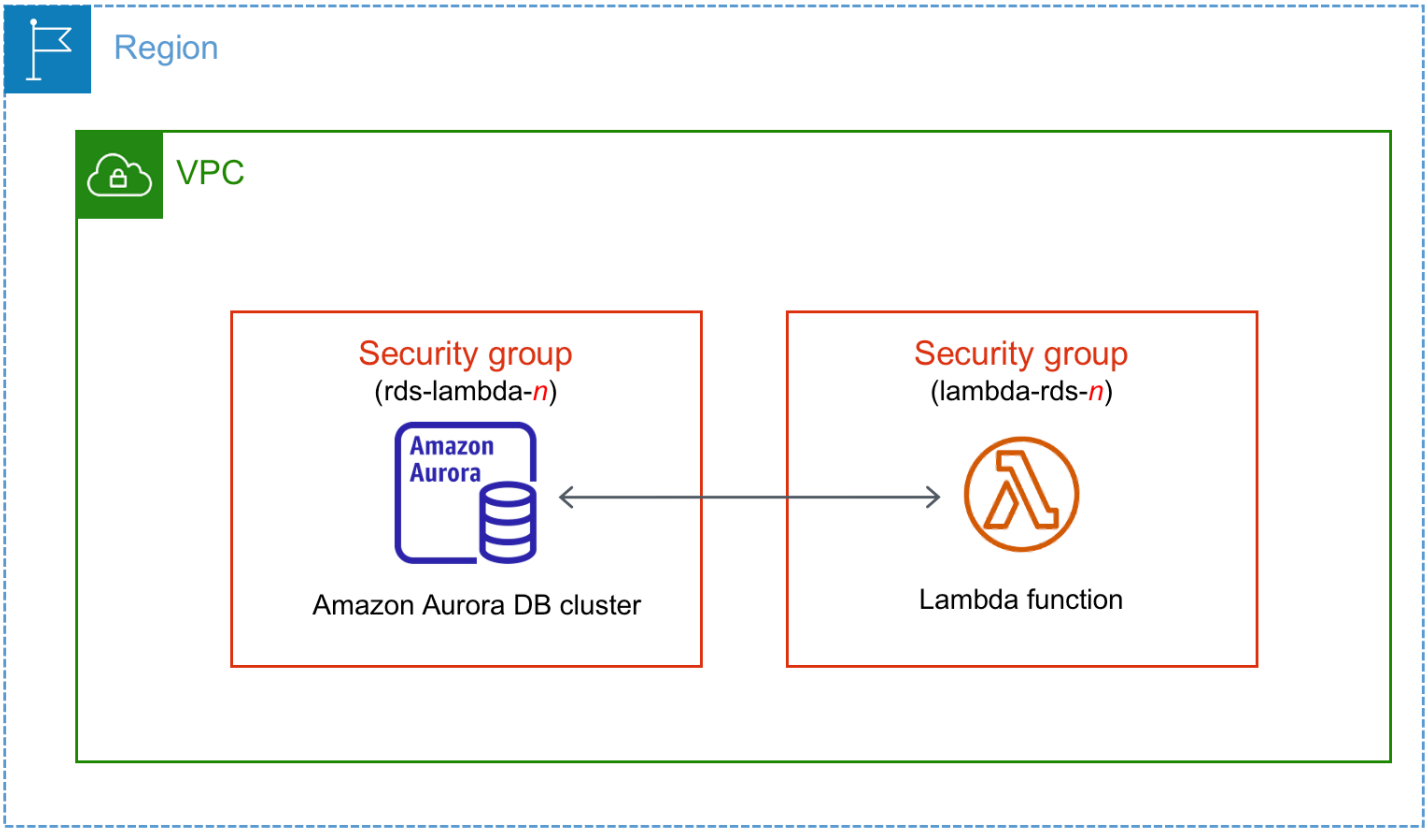
Per informazioni sulla connessione a un'istanza database che esegue un motore DB specifico, seguire le istruzioni per il motore DB:

- Connessione a un cluster di database Amazon Aurora MySQL
- Connessione a un cluster di database Amazon Aurora PostgreSQL

Connessione automatica di una funzione Lambda e di un cluster database Aurora

È possibile utilizzare la console Amazon RDS per semplificare la configurazione di una connessione tra una funzione Lambda e un cluster database Aurora. Spesso, il cluster database si trova in una sottorete privata all'interno di un VPC. La funzione Lambda può essere utilizzata dalle applicazioni per accedere al cluster database privato.

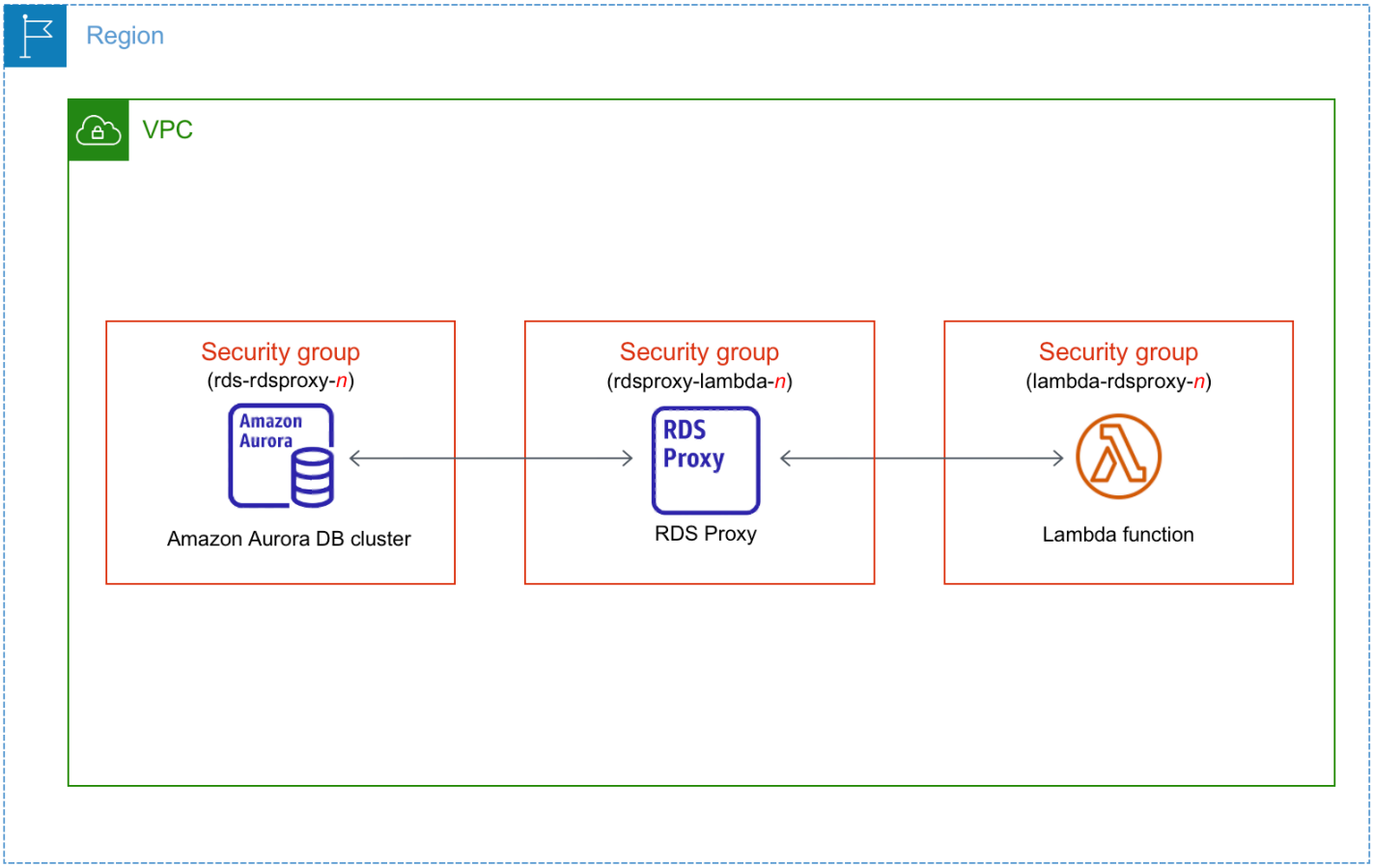
L'immagine seguente mostra una connessione diretta tra il cluster database e la funzione Lambda.



È possibile configurare la connessione tra la funzione Lambda e il cluster database tramite RDS Proxy per migliorare le prestazioni e la resilienza del database. Spesso, le funzioni Lambda effettuano connessioni database brevi e frequenti che traggono vantaggio dal pool di connessioni offerto da RDS Proxy. È possibile sfruttare qualsiasi autenticazione AWS Identity and Access Management (IAM) già disponibile per le funzioni Lambda, anziché gestire le credenziali del database nel codice dell'applicazione Lambda. Per ulteriori informazioni, consulta Server proxy per Amazon RDS per Aurora.

Quando si utilizza la console per connettersi a un proxy esistente, Amazon RDS aggiorna il gruppo di sicurezza proxy per consentire le connessioni dal cluster database e la funzione Lambda.

È anche possibile creare un nuovo proxy dalla stessa pagina della console. Quando si crea un proxy nella console, per accedere al cluster database, occorre inserire le credenziali del database o selezionare un segreto Gestione dei segreti AWS.



Tip

Per connettere rapidamente una funzione Lambda a un cluster di database Aurora, è anche possibile utilizzare la procedura guidata della console. Per aprire la procedura guidata, esegui queste operazioni:

1. Aprire la pagina Funzioni della console Lambda.
2. Seleziona la funzione a cui desideri connettere un database.
3. Nella scheda Configurazione, seleziona Database RDS.
4. Scegli Connessione al database RDS.

Dopo aver collegato la funzione a un database, scegli Aggiungi proxy se desideri creare un proxy.

Argomenti

- Panoramica della connettività automatica a una funzione Lambda
- Connessione automatica di un funzione Lambda e di un cluster database Aurora
- Visualizzazione delle risorse di calcolo connesse

Panoramica della connettività automatica a una funzione Lambda

Di seguito sono riportati i requisiti per connettere una funzione Lambda a un cluster database Aurora:

- La funzione Lambda deve esistere nello stesso VPC del cluster database.
- Attualmente, il cluster database non può essere un cluster database Aurora Serverless o parte di un database globale Aurora.
- L'utente che configura la connettività deve disporre delle autorizzazioni per eseguire le seguenti operazioni Amazon RDS, Amazon EC2, Lambda, Secrets Manager e IAM:
  - Amazon RDS
    - `rds:CreateDBProxies`
    - `rds:DescribeDBClusters`
    - `rds:DescribeDBProxies`
    - `rds:ModifyDBCluster`
    - `rds:ModifyDBProxy`
    - `rds:RegisterProxyTargets`
  - Amazon EC2
    - `ec2:AuthorizeSecurityGroupEgress`
    - `ec2:AuthorizeSecurityGroupIngress`
    - `ec2:CreateSecurityGroup`
    - `ec2:DeleteSecurityGroup`
    - `ec2:DescribeSecurityGroups`

    - `ec2:RevokeSecurityGroupEgress`
    - `ec2:RevokeSecurityGroupIngress`
- Lambda
    - `lambda:CreateFunctions`
    - `lambda:ListFunctions`
    - `lambda:UpdateFunctionConfiguration`
- Secrets Manager
    - `secretsmanager:CreateSecret`
    - `secretsmanager:DescribeSecret`
- IAM
    - `iam:AttachPolicy`
    - `iam:CreateRole`
    - `iam:CreatePolicy`
- AWS KMS
    - `kms:describeKey`

> **Note**
>
> Se il cluster database e la funzione Lambda si trovano in zone di disponibilità diverse, potrebbero essere addebitati costi tra zone di disponibilità all'account.

Quando si configura una connessione tra una funzione Lambda e un cluster database Aurora, Amazon RDS configura il gruppo di sicurezza VPC per la funzione e per il cluster database. Se si utilizza il proxy RDS, Amazon RDS configura anche il gruppo di sicurezza VPC per il proxy. Amazon RDS opera in base alla configurazione corrente dei gruppi di sicurezza associati al cluster database, alla funzione Lambda e al proxy, come descritto nella tabella seguente.

Configurazione del gruppo di sicurezza RDS corrente	Configurazione del gruppo di sicurezza Lambda corrente	Configurazione del gruppo di sicurezza proxy corrente	Operazione RDS
Esistono uno o più gruppi di sicurezza associati al cluster database con un nome che corrispon de al modello `rds-lambda-` *n* o se un proxy è già connesso al cluster database, RDS verifica se il parametro `TargetHea lth` di un proxy associato è `AVAILABLE` . Un gruppo di sicurezza che corrisponde al modello non è stato modificato. Questo gruppo di sicurezza include una sola regola in entrata con il gruppo di sicurezza VPC della funzione Lambda o del proxy come l'origine.	Esistono uno o più gruppi di sicurezza associati alla funzione Lambda con un nome che corrisponde al modello `lambda-rd s-` *n* o `lambda-rd sproxy-` *n* (dove *n* è un numero). Un gruppo di sicurezza che corrisponde al modello non è stato modificato. Questo gruppo di sicurezza dispone di una sola regola in uscita con il gruppo di sicurezza VPC del cluster database o il proxy come la destinazione.	Esistono uno o più gruppi di sicurezza associati al proxy con un nome che corrisponde al modello `rdsproxy-lambda-` *n* (dove *n* è un numero). Un gruppo di sicurezza che corrisponde al modello non è stato modificato. Questo gruppo di sicurezza dispone di regole in entrata e in uscita con i gruppi di sicurezza VPC della funzione Lambda e il cluster database.	Amazon RDS non esegue alcuna azione. Una connessione era già stata configurata automaticamente tra la funzione Lambda, il proxy (opzionale) e il cluster database. Poiché esiste già una connessione tra la funzione, il proxy e il database, i gruppi di sicurezza non vengono modificati.
Si applica una delle due condizioni seguenti:	Si applica una delle due condizioni seguenti:	Si applica una delle due condizioni seguenti:	RDS action: create new security groups

Configurazione del gruppo di sicurezza RDS corrente	Configurazione del gruppo di sicurezza Lambda corrente	Configurazione del gruppo di sicurezza proxy corrente	Operazione RDS
• Non esiste alcun gruppo di sicurezza associato al cluster database con un nome che corrispon de al modello `rds-lambda-` *n* o se `TargetHealth` di un proxy associato è `AVAILABLE` . • Esiste almeno un gruppo di sicurezza associato al cluster database con un nome che corrispon de al modello `rds-lambda-` *n* o se `TargetHealth` di un proxy associato è `AVAILABLE` . Tuttavia, nessuno di questi gruppi di sicurezza può essere utilizzato per la connessione alla funzione Lambda. Amazon RDS non può utilizzare un gruppo di sicurezza che	• Non esiste un gruppo di sicurezza associato alla funzione Lambda con un nome che corrisponde al modello `lambda-rds-` *n* o `lambda-rdsproxy-` *n*. • Esistono uno o più gruppi di sicurezza associati alla funzione Lambda con un nome che corrisponde al modello `lambda-rds-` *n* o `lambda-rdsproxy-` *n*. Tuttavia, Amazon RDS non può utilizzare nessuno di questi gruppi di sicurezza per la connessione al cluster database. Amazon RDS non può utilizzare un gruppo di sicurezza che non dispone di una regola in uscita con il gruppo di sicurezza VPC del	• Non esiste un gruppo di sicurezza associato al proxy con un nome che corrispon de al modello `rdsproxy-lambda-` *n*. • Esistono uno o più gruppi di sicurezza associati al proxy con un nome che corrispon de al modello `rdsproxy-lambda-` *n*. Tuttavia, Amazon RDS non può utilizzare nessuno di questi gruppi di sicurezza per la connessione al cluster database o alla funzione Lambda. Amazon RDS non può utilizzare un gruppo di sicurezza che non dispone di una regola in entrata e in uscita con il gruppo	

Configurazione del gruppo di sicurezza RDS corrente	Configurazione del gruppo di sicurezza Lambda corrente	Configurazione del gruppo di sicurezza proxy corrente	Operazione RDS
non dispone di una regola in entrata con il gruppo di sicurezza VPC della funzione Lambda o del proxy come l'origine. Inoltre, Amazon RDS non può utilizzare un gruppo di sicurezza che è stato modificato. Esempi di modifiche sono l'aggiunta di una regola o la modifica della porta di una regola esistente.	cluster database o del proxy come destinazione. Inoltre, Amazon RDS non può utilizzare un gruppo di sicurezza che è stato modificato.	di sicurezza VPC del cluster database e la funzione Lambda. Inoltre, Amazon RDS non può utilizzare un gruppo di sicurezza che è stato modificat o.	

Configurazione del gruppo di sicurezza RDS corrente	Configurazione del gruppo di sicurezza Lambda corrente	Configurazione del gruppo di sicurezza proxy corrente	Operazione RDS
Esiste almeno un gruppo di sicurezza associato al cluster database con un nome che corrispon de al modello `rds-lambda-` *n* o se `TargetHea lth` di un proxy associato è `AVAILABLE` . Un gruppo di sicurezza che corrisponde al modello non è stato modificato. Questo gruppo di sicurezza include una sola regola in entrata con il gruppo di sicurezza VPC della funzione Lambda o del proxy come l'origine.	Esistono uno o più gruppi di sicurezza associati alla funzione Lambda con un nome che corrisponde al modello `lambda-rd s-` *n* o `lambda-rd sproxy-` *n*. Tuttavia, Amazon RDS non può utilizzar e nessuno di questi gruppi di sicurezza per la connessione al cluster database. Amazon RDS non può utilizzare un gruppo di sicurezza che non dispone di una regola in uscita con il gruppo di sicurezza VPC del cluster database o del proxy come destinazione. Inoltre, Amazon RDS non può utilizzare un gruppo di sicurezza che è stato modificato.	Esistono uno o più gruppi di sicurezza associati al proxy con un nome che corrisponde al modello `rdsproxy-lambda-` *n*. Tuttavia, Amazon RDS non può utilizzar e nessuno di questi gruppi di sicurezza per la connessione al cluster database o alla funzione Lambda. Amazon RDS non può utilizzare un gruppo di sicurezza che non dispone di una regola in entrata e in uscita con il gruppo di sicurezza VPC del cluster database e la funzione Lambda. Inoltre, Amazon RDS non può utilizzare un gruppo di sicurezza che è stato modificat o.	RDS action: create new security groups

Configurazione del gruppo di sicurezza RDS corrente	Configurazione del gruppo di sicurezza Lambda corrente	Configurazione del gruppo di sicurezza proxy corrente	Operazione RDS
Esiste almeno un gruppo di sicurezza associato al cluster database con un nome che corrispon de al modello `rds-lambda-` *n* o se `TargetHea lth` di un proxy associato è `AVAILABLE` . Un gruppo di sicurezza che corrisponde al modello non è stato modificato. Questo gruppo di sicurezza include una sola regola in entrata con il gruppo di sicurezza VPC della funzione Lambda o del proxy come l'origine.	Esiste un gruppo di sicurezza Lambda valido per la connessione, ma non è associato alla funzione Lambda. Questo gruppo di sicurezza ha un nome che corrisponde al modello `lambda-rds-` *n* o `lambda-rdsproxy-` *n*. Non è stato modificato. Dispone di una sola regola in uscita con il gruppo di sicurezza VPC del cluster database o del proxy come la destinazione.	Esiste un gruppo di sicurezza proxy valido per la connessione, ma non è associato al proxy. Questo gruppo di sicurezza ha un nome che corrisponde al modello `rdsproxy-lambda-` *n*. Non è stato modificato. Dispone di regole in entrata e in uscita con i gruppi di sicurezza VPC del cluster database e la funzione Lambda.	RDS action: associate Lambda security group

Configurazione del gruppo di sicurezza RDS corrente	Configurazione del gruppo di sicurezza Lambda corrente	Configurazione del gruppo di sicurezza proxy corrente	Operazione RDS
Si applica una delle due condizioni seguenti: • Non esiste alcun gruppo di sicurezza associato al cluster database con un nome che corrisponde al modello `rds-lambda-` *n* o se `TargetHealth` di un proxy associato è `AVAILABLE`. • Esiste almeno un gruppo di sicurezza associato al cluster database con un nome che corrisponde al modello `rds-lambda-` *n* o se `TargetHealth` di un proxy associato è `AVAILABLE`. Tuttavia, Amazon RDS non può utilizzare nessuno di questi gruppi di sicurezza per la connessione alla	Esistono uno o più gruppi di sicurezza associati alla funzione Lambda con un nome che corrisponde al modello `lambda-rds-` *n* o `lambda-rdsproxy-` *n*. Un gruppo di sicurezza che corrisponde al modello non è stato modificato. Questo gruppo di sicurezza dispone di una sola regola in uscita con il gruppo di sicurezza VPC dell'istanza database o del proxy come la destinazione.	Esistono uno o più gruppi di sicurezza associati al proxy con un nome che corrisponde al modello `rdsproxy-lambda-` *n*. Un gruppo di sicurezza che corrisponde al modello non è stato modificato. Questo gruppo di sicurezza dispone di regole in entrata e in uscita con il gruppo di sicurezza VPC del cluster database e la funzione Lambda.	RDS action: create new security groups

Configurazione del gruppo di sicurezza RDS corrente	Configurazione del gruppo di sicurezza Lambda corrente	Configurazione del gruppo di sicurezza proxy corrente	Operazione RDS
funzione Lambda o al proxy. Amazon RDS non può utilizzare un gruppo di sicurezza che non dispone di una regola in entrata con il gruppo di sicurezza VPC della funzione Lambda o del proxy come l'origine. Inoltre, Amazon RDS non può utilizzare un gruppo di sicurezza che è stato modificato.			

Configurazione del gruppo di sicurezza RDS corrente	Configurazione del gruppo di sicurezza Lambda corrente	Configurazione del gruppo di sicurezza proxy corrente	Operazione RDS
Si applica una delle due condizioni seguenti: • Non esiste alcun gruppo di sicurezza associato al cluster database con un nome che corrispon de al modello `rds-lambda-` *n* o se `TargetHea lth` di un proxy associato è `AVAILABLE` . • Esiste almeno un gruppo di sicurezza associato al cluster database con un nome che corrispon de al modello `rds-lambda-` *n* o se `TargetHea lth` di un proxy associato è `AVAILABLE` . Tuttavia, Amazon RDS non può utilizzare nessuno di questi gruppi di sicurezza per la connessione alla	Si applica una delle due condizioni seguenti: • Non esiste un gruppo di sicurezza associato alla funzione Lambda con un nome che corrisponde al modello `lambda-rds-` *n* o `lambda-rdsproxy-` *n*. • Esistono uno o più gruppi di sicurezza associati alla funzione Lambda con un nome che corrisponde al modello `lambda-rds-` *n* o `lambda-rdsproxy-` *n*. Tuttavia, Amazon RDS non può utilizzare nessuno di questi gruppi di sicurezza per la connessione al cluster database. Amazon RDS non può utilizzare un gruppo	Si applica una delle due condizioni seguenti: • Non esiste un gruppo di sicurezza associato al proxy con un nome che corrispon de al modello `rdsproxy-lambda-` *n*. • Esistono uno o più gruppi di sicurezza associati al proxy con un nome che corrispon de al modello `rdsproxy-lambda-` *n*. Tuttavia, Amazon RDS non può utilizzare nessuno di questi gruppi di sicurezza per la connessione al cluster database o alla funzione Lambda. Amazon RDS non può utilizzare un gruppo	RDS action: create new security groups

Configurazione del gruppo di sicurezza RDS corrente	Configurazione del gruppo di sicurezza Lambda corrente	Configurazione del gruppo di sicurezza proxy corrente	Operazione RDS
funzione Lambda o al proxy. Amazon RDS non può utilizzare un gruppo di sicurezza che non dispone di una regola in entrata con il gruppo di sicurezza VPC della funzione Lambda o del proxy come l'origine. Inoltre, Amazon RDS non può utilizzare un gruppo di sicurezza che è stato modificato.	di sicurezza che non dispone di una regola in uscita con il gruppo di sicurezza VPC cluster database o del proxy come l'origine. Inoltre, Amazon RDS non può utilizzare un gruppo di sicurezza che è stato modificato.	di sicurezza che non dispone di una regola in entrata e in uscita con il gruppo di sicurezza VPC del cluster database e la funzione Lambda. Inoltre, Amazon RDS non può utilizzare un gruppo di sicurezza che è stato modificato.	

RDS: creazione di nuovi gruppi di sicurezza

Amazon RDS esegue le seguenti operazioni:

- Crea un nuovo gruppo di sicurezza che corrisponde al modello `rds-lambda-`*n* o `rds-rdsproxy-`*n* (se si sceglie di utilizzare RDS Proxy). Questo gruppo di sicurezza dispone di una regola in entrata con il gruppo di sicurezza VPC della funzione Lambda o proxy come l'origine. Questo gruppo di sicurezza è associato al cluster database e consente alla funzione o al proxy di accedere al cluster database.
- Crea un nuovo gruppo di sicurezza che corrisponde al modello `lambda-rds-`*n* o `lambda-rdsproxy-`*n*. Questo gruppo di sicurezza dispone di una regola in uscita con il gruppo di sicurezza VPC del cluster database o del proxy come la destinazione. Questo gruppo di sicurezza è associato alla funzione Lambda e consente alla funzione di inviare traffico al cluster database o inviare traffico tramite un proxy.

- Crea un nuovo gruppo di sicurezza che corrisponde al modello `rdsproxy-lambda-`*n*. Questo gruppo di sicurezza dispone di regole in entrata e in uscita con il gruppo di sicurezza VPC del cluster database e la funzione Lambda.

Azione RDS: associazione del gruppo di sicurezza Lambda

Amazon RDS associa il gruppo di sicurezza Lambda valido, esistente alla funzione Lambda. Questo gruppo di sicurezza consente alla funzione di inviare traffico al cluster database o inviare traffico tramite un proxy.

Connessione automatica di un funzione Lambda e di un cluster database Aurora

È possibile usare la console Amazon RDS per connettere automaticamente una funzione Lambda al cluster database. Ciò semplifica il processo di configurazione di una connessione tra queste risorse.

È anche possibile usare RDS Proxy per includere un proxy nella connessione. Funzioni Lambda effettuano connessioni database brevi e frequenti che traggono vantaggio dal pool di connessioni offerto da RDS Proxy. È anche possibile utilizzare qualsiasi autenticazione IAM che è già stata configurata per le funzioni Lambda, anziché gestire le credenziali del database nel codice dell'applicazione Lambda.

È possibile connettere un cluster database esistente a funzioni Lambda nuove ed esistenti utilizzando la pagina Configurazione della connessione Lambda. Il processo di configurazione consente di impostare automaticamente i gruppi di sicurezza richiesti.

Prima di configurare una connessione tra una funzione Lambda e un cluster database, assicurati che:

- La funzione Lambda e il cluster database si trovino nello stesso VPC.
- Disponi delle autorizzazioni corrette per l'account utente. Per ulteriori informazioni sui requisiti, consulta Panoramica della connettività automatica a una funzione Lambda.

Se si modificano i gruppi di sicurezza dopo la configurazione della connettività, le modifiche potrebbero influenzare la connessione tra la funzione Lambda e il cluster database.

> **Note**
>
> È possibile configurare automaticamente una connessione tra un cluster database e una funzione Lambda solo nella Console di gestione AWS. Per connettere una funzione Lambda, tutte le istanze nel cluster database devono essere nello stato Disponibile.

Per connettere automaticamente una funzione Lambda e un cluster database

<result>

Dopo aver confermato la configurazione, Amazon RDS avvia il processo di connessione della funzione Lambda, di RDS Proxy (se hai usato un proxy) e del cluster database. La console mostra la finestra di dialogo Dettagli di connessione, in cui sono elencate le modifiche al gruppo di sicurezza che consentono le connessioni tra le risorse.
</result>

1. Accedi alla Console di gestione AWS e apri la console Amazon RDS all'indirizzo https://console.aws.amazon.com/rds/.
2. Nel riquadro di navigazione, scegli Database, quindi seleziona il cluster database che desideri connettere a una funzione Lambda.
3. Per Operazioni, scegli Configura connessione Lambda.
4. Nella pagina Configurazione della connessione Lambda, in Seleziona la funzione Lambda, effettua una delle seguenti operazioni:
   - Se disponi di una funzione Lambda esistente nello stesso VPC del cluster database, seleziona Scegli una funzione esistente, quindi seleziona la funzione.
   - Se non disponi di una funzione Lambda nello stesso VPC, seleziona Crea una nuova funzione, quindi inserisci un Nome della funzione. Il runtime predefinito è impostato su Nodejs.18. Puoi modificare le impostazioni per la nuova funzione Lambda nella console Lambda dopo aver completato la configurazione della connessione.
5. (Facoltativo) In RDS Proxy, seleziona Connessione tramite RDS Proxy, quindi esegui una delle seguenti operazioni:
   - Se disponi di un proxy esistente che desideri utilizzare, seleziona Scegli un proxy esistente, quindi seleziona il proxy.
   - Se non disponi di un proxy e desideri che uno venga creato automaticamente da Amazon RDS, seleziona Crea nuovo proxy. Quindi, per Credenziali del database, esegui una delle seguenti operazioni:

a. Seleziona Nome utente e password del database, quindi inserisci Nome utente e Password per il cluster database.

b. Seleziona Segreti Secrets Manager. Quindi, per Seleziona segreto, scegli un segreto Gestione dei segreti AWS. Se non disponi di un segreto di Secrets Manager, seleziona Crea nuovo segreto di Secrets Manager per creare un nuovo segreto. Dopo aver creato il segreto, per Seleziona segreto, scegli il nuovo segreto.

Dopo aver creato il nuovo proxy, seleziona Scegli un proxy esistente, quindi seleziona il proxy. Tieni presente che prima che il proxy sia disponibile per la connessione, potrebbe trascorrere del tempo.

6. (Facoltativo) Espandi Riepilogo della connessione e verifica gli aggiornamenti evidenziati per le risorse.
7. Scegliere Set up (Configura).

Visualizzazione delle risorse di calcolo connesse

È possibile usare la Console di gestione AWS per visualizzare le funzioni Lambda collegate al cluster database. Le risorse mostrate includono le connessioni delle risorse di calcolo configurate automaticamente da Amazon RDS.

Le risorse di elaborazione elencate non includono quelle connesse manualmente al cluster database. Ad esempio, è possibile autorizzare manualmente una risorsa di calcolo ad accedere al cluster database aggiungendo una regola al gruppo di sicurezza VPC associato al database.

Affinché la console elenchi una funzione Lambda, devono essere soddisfatte le seguenti condizioni:

- Il nome del gruppo di sicurezza associato alla risorsa di calcolo corrisponde al modello `lambda-rds-`*n* o `lambda-rdsproxy-`*n* (dove *n* è un numero).
- Il gruppo di sicurezza associato alla risorsa di calcolo dispone di una regola in uscita con l'intervallo di porte impostato sulla porta del cluster database o un proxy associato. La destinazione della regola in uscita deve essere impostata su un gruppo di sicurezza associato al cluster database o a un proxy associato.
- Se la configurazione include un proxy, il nome del gruppo di sicurezza collegato al proxy associato al database corrisponde al modello `rdsproxy-lambda-`*n* (dove *n* è un numero).

- Il gruppo di sicurezza associato alla funzione dispone di una regola in uscita con l'intervallo di porte impostato sulla porta utilizzata dal cluster database o da un proxy associato. La destinazione deve essere impostata su un gruppo di sicurezza associato al cluster database o a un proxy associato.

Per visualizzare le risorse di elaborazione connesse automaticamente a un cluster database

1. Accedi alla Console di gestione AWS e apri la console Amazon RDS all'indirizzo https://console.aws.amazon.com/rds/.
2. Nel pannello di navigazione, scegli Database e seleziona il cluster database.
3. Nella scheda Connettività e sicurezza, visualizza le risorse di calcolo in Risorse di calcolo connesse.

Modifica di un cluster database Amazon Aurora

È possibile modificare le impostazioni di un cluster di database per completare attività come la modifica del periodo di retention dei backup o della porta del database. È anche possibile modificare le istanze database in un cluster di database per completare attività come la modifica della classe dell'istanza database o l'abilitazione di Performance Insights. In questo argomento viene illustrato come modificare un cluster di database Aurora, le istanze database e le relative impostazioni.

È consigliabile testare eventuali modifiche in un cluster di database o in un'istanza database di prova prima di modificare un cluster di database o un'istanza database di produzione, per comprendere pienamente l'impatto di ogni modifica. Ciò è importante soprattutto in caso di aggiornamento delle versioni di database.

Argomenti

- Modifica del cluster di database tramite la console, la CLI e l'API
- Modifica di un'istanza database in un cluster database
- Modifica della password per l’utente master del database
- Impostazioni per Amazon Aurora
- Impostazioni che non si applicano ai cluster di database Amazon Aurora
- Impostazioni che non si applicano alle istanze database Amazon Aurora

Modifica del cluster di database tramite la console, la CLI e l'API

Un cluster di database può essere modificato tramite Console di gestione AWS, AWS CLI o API RDS.

> **Note**
>
> La maggior parte delle modifiche può essere applicata immediatamente o durante la successiva finestra di manutenzione programmata. Alcune modifiche, come l'attivazione della protezione da eliminazione, vengono applicate immediatamente, indipendentemente da quando scegli di applicarle.
> La modifica della password master in Console di gestione AWS viene sempre applicata immediatamente.
> Se utilizzi endpoint SSL e modifichi l'identificatore del cluster database, interrompi e riavvia il cluster database per aggiornare gli endpoint SSL. Per ulteriori informazioni, consulta Avvio e arresto di un cluster di database Amazon Aurora.

Console

Per modificare un cluster di database

1. Accedi alla Console di gestione AWS e apri la console Amazon RDS all'indirizzo https://console.aws.amazon.com/rds/.
2. Nel riquadro di navigazione, scegliere Databases (Database), quindi selezionare il cluster di database che si desidera modificare.
3. Scegliere Modify (Modifica). Viene visualizzata la pagina Modify DB cluster (Modifica cluster di database).
4. Modificare le impostazioni desiderate. Per informazioni su ciascuna impostazione, consulta Impostazioni per Amazon Aurora.

   > **Note**
   >
   > Nella Console di gestione AWS alcune modifiche a livello di istanza si applicano solo all'istanza database corrente, mentre altre si applicano all'intero cluster di database. Per sapere se un'impostazione si applica all'istanza database o al cluster di database, vedi l'ambito relativo all'impostazione in Impostazioni per Amazon Aurora. Per cambiare un'impostazione che consente di modificare l'intero cluster di database a livello di istanza nella Console di gestione AWS, segui le istruzioni contenute in Modifica di un'istanza database in un cluster database.

5. Quando tutte le modifiche sono come le desideri, seleziona Continue (Continua) e controlla il riepilogo delle modifiche.
6. Per applicare le modifiche immediatamente, selezionare Apply Immediately (Applica immediatamente).
7. Nella pagina di conferma esaminare le modifiche. Se sono corrette, selezionare Modify cluster (Modifica cluster) per salvare le modifiche.

   In alternativa, scegliere Back (Indietro) per cambiare le modifiche o Cancel (Annulla) per annullare le modifiche.

AWS CLI

Per modificare un cluster di database tramite AWS CLI, chiamare il comando modify-db-cluster. Specifica l'identificatore del cluster di database e i valori per le impostazioni da modificare. Per informazioni su ciascuna impostazione, consulta Impostazioni per Amazon Aurora.

> **Note**
>
> Alcune impostazioni si applicano solo alle istanze database. Per modificare queste impostazioni, seguire le istruzioni in Modifica di un'istanza database in un cluster database.

Example

Il comando seguente modifica `mydbcluster` impostando il periodo di retention dei backup su 1 settimana (7 giorni).

Per Linux, macOS o Unix:

```
aws rds modify-db-cluster \
    --db-cluster-identifier mydbcluster \
    --backup-retention-period 7
```

Per Windows:

```
aws rds modify-db-cluster ^
    --db-cluster-identifier mydbcluster ^
    --backup-retention-period 7
```

API RDS

Per modificare un cluster di database tramite l'API Amazon RDS, chiamare l'operazione ModifyDBCluster. Specifica l'identificatore del cluster di database e i valori per le impostazioni da modificare. Per informazioni su ciascun parametro, consulta Impostazioni per Amazon Aurora.

> **Note**
>
> Alcune impostazioni si applicano solo alle istanze database. Per modificare queste impostazioni, seguire le istruzioni in Modifica di un'istanza database in un cluster database.

Modifica di un'istanza database in un cluster database

Un'istanza database può essere modificata in un cluster di database tramite Console di gestione AWS, AWS CLI o API RDS.

Quando modifichi un'istanza database, puoi applicare le modifiche immediatamente. Per applicare le modifiche immediatamente, devi selezionare l'opzione Apply Immediately (Applica immediatamente) nella Console di gestione AWS, usare il parametro `--apply-immediately` quando chiami AWS CLI oppure impostare il parametro `ApplyImmediately` su `true` quando usi l'API di Amazon RDS.

Se non si sceglie di applicare immediatamente le modifiche, le modifiche vengono rinviate alla finestra di manutenzione successiva. Durante la finestra di manutenzione successiva, vengono applicate tutte queste modifiche rinviate. Se si sceglie di applicare immediatamente le modifiche, le nuove modifiche verranno applicate insieme alle eventuali modifiche precedentemente rinviate.

Per vedere le modifiche in sospeso per la successiva finestra di manutenzione, utilizza il comando dell'interfaccia AWS CLI describe-db-clusters e controlla il campo `PendingModifiedValues`.

> **Important**
>
> Se nessuna delle modifiche rinviate richiede tempi di inattività, la selezione Apply immediately (Applica immediatamente) può provocare tempi di inattività imprevisti per l'istanza database. Non sono previsti tempi di inattività per le altre istanze database nel cluster di database.
> Le modifiche rinviate non sono elencate nell'output del comando CLI `describe-pending-maintenance-actions`. Le azioni di manutenzione includono solo gli aggiornamenti di sistema pianificati per la finestra di manutenzione successiva.

Console

Per modificare un'istanza database in un cluster di database

1. Accedi alla Console di gestione AWS e apri la console Amazon RDS all'indirizzo https://console.aws.amazon.com/rds/.
2. Nel riquadro di navigazione, scegliere Databases (Database), quindi selezionare l'istanza database che si desidera modificare.
3. Per Actions (Operazioni), scegliere Modify (Modifica). Viene visualizzata la pagina Modify DB Instance (Modifica istanza database).

4. Modificare le impostazioni desiderate. Per informazioni su ciascuna impostazione, consulta Impostazioni per Amazon Aurora.

   > **Note**
   >
   > Alcune impostazioni sono valide per l'intero cluster di database e devono essere modificate a livello di cluster. Per modificare queste impostazioni, seguire le istruzioni in Modifica del cluster di database tramite la console, la CLI e l'API.
   > Nella Console di gestione AWS alcune modifiche a livello di istanza si applicano solo all'istanza database corrente, mentre altre si applicano all'intero cluster di database. Per sapere se un'impostazione si applica all'istanza database o al cluster di database, vedi l'ambito relativo all'impostazione in Impostazioni per Amazon Aurora.

5. Quando tutte le modifiche sono come le desideri, seleziona Continue (Continua) e controlla il riepilogo delle modifiche.
6. Per applicare le modifiche immediatamente, selezionare Apply Immediately (Applica immediatamente).
7. Nella pagina di conferma esaminare le modifiche. Se sono corrette, seleziona Modifica istanza database per salvare le modifiche.

   In alternativa, scegliere Back (Indietro) per cambiare le modifiche o Cancel (Annulla) per annullare le modifiche.

AWS CLI

Per modificare un'istanza database in un cluster di database tramite AWS CLI, chiamare il comando modify-db-instance. Specifica l'identificatore istanze DB e i valori per le impostazioni da modificare. Per informazioni su ciascun parametro, consulta Impostazioni per Amazon Aurora.

> **Note**
>
> Alcune impostazioni si applicano all'intero cluster di database. Per modificare queste impostazioni, seguire le istruzioni in Modifica del cluster di database tramite la console, la CLI e l'API.

Example

Il codice seguente modifica `mydbinstance` impostando la classe di istanza database su `db.r4.xlarge`. Le modifiche vengono applicate durante la prossima finestra di manutenzione utilizzando `--no-apply-immediately`. Utilizza `--apply-immediately` per applicare immediatamente le modifiche.

Per Linux, macOS o Unix:

```
aws rds modify-db-instance \
    --db-instance-identifier mydbinstance \
    --db-instance-class db.r4.xlarge \
    --no-apply-immediately
```

Per Windows:

```
aws rds modify-db-instance ^
    --db-instance-identifier mydbinstance ^
    --db-instance-class db.r4.xlarge ^
    --no-apply-immediately
```

API RDS

Per modificare un'istanza database tramite l'API di Amazon RDS, chiama l'operazione ModifyDBInstance. Specifica l'identificatore istanze DB e i valori per le impostazioni da modificare. Per informazioni su ciascun parametro, consulta Impostazioni per Amazon Aurora.

> **Note**
>
> Alcune impostazioni si applicano all'intero cluster di database. Per modificare queste impostazioni, seguire le istruzioni in Modifica del cluster di database tramite la console, la CLI e l'API.

Modifica della password per l’utente master del database

Puoi utilizzare la Console di gestione AWS o l’interfaccia AWS CLI per cambiare la password dell’utente master.

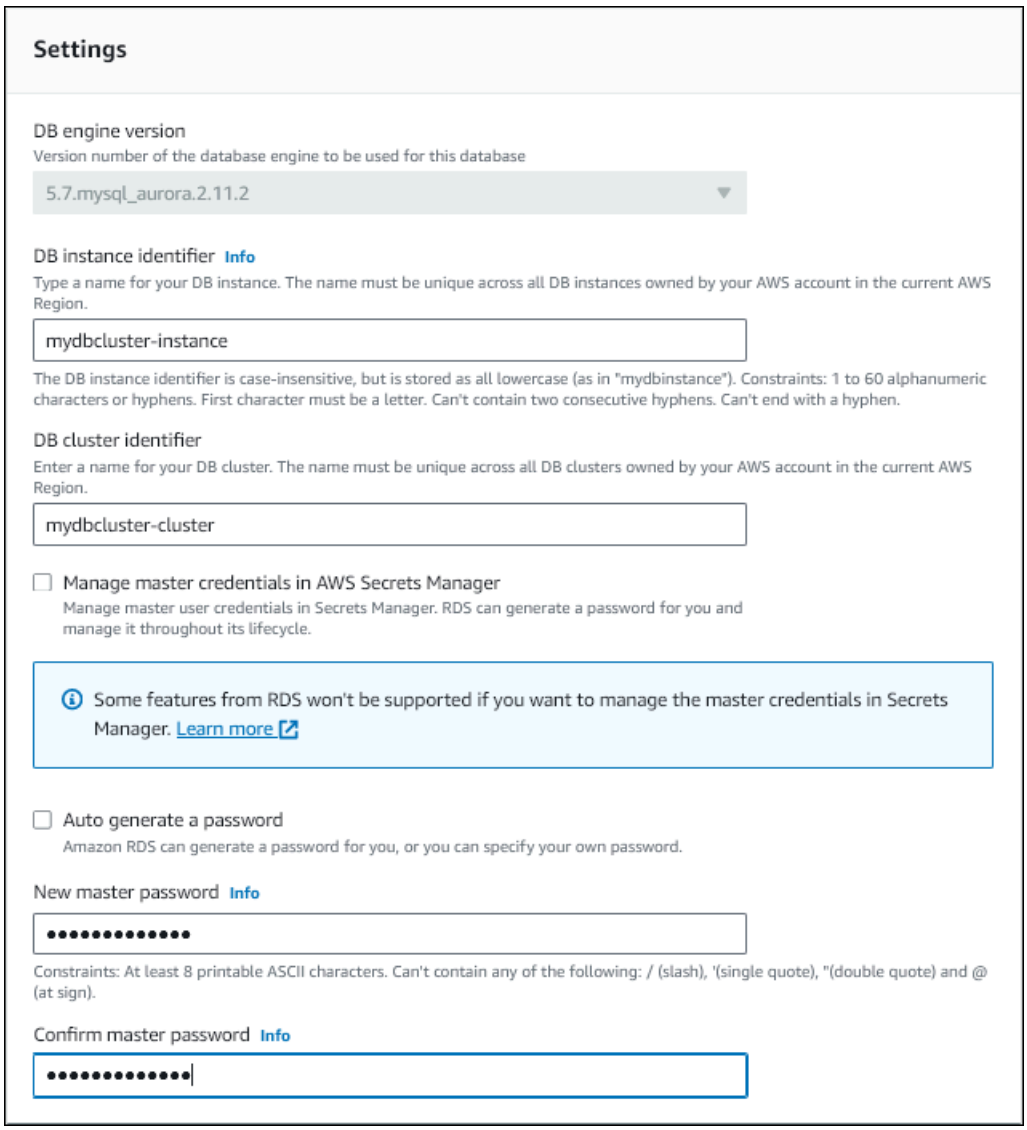
Console

È possibile modificare l’istanza database di scrittura per cambiare la password dell’utente master utilizzando Console di gestione AWS.

Come modificare la password dell’utente master

1. Accedi alla Console di gestione AWS e apri la console Amazon RDS all'indirizzo https://console.aws.amazon.com/rds/.
2. Nel riquadro di navigazione, scegliere Databases (Database), quindi selezionare l'istanza database che si desidera modificare.
3. Per Actions (Operazioni), scegliere Modify (Modifica).

   Viene visualizzata la pagina Modify DB Instance (Modifica istanza database).
4. Immetti una nuova password master.
5. Nel campo Conferma password master, inserisci la stessa nuova password.



6. Scegliere Continue (Continua) e controllare il riepilogo delle modifiche.

> **Note**
>
> Le modifiche alla password vengono sempre applicate immediatamente.

7. Nella pagina di conferma scegli Modify DB instance (Modifica istanza database).

CLI

Occorre chiamare il comando modify-db-cluster per modificare la password dell'utente master utilizzando l'interfaccia AWS CLI. Quindi è necessario specificare l'identificatore del cluster di database e la nuova password, come mostrato negli esempi seguenti.

Non è necessario specificare `--apply-immediately|--no-apply-immediately`, poiché le modifiche alla password vengono sempre applicate immediatamente.

Per Linux, macOS o Unix:

```
aws rds modify-db-cluster \
    --db-cluster-identifier mydbcluster \
    --master-user-password mynewpassword
```

Per Windows:

```
aws rds modify-db-cluster ^
    --db-cluster-identifier mydbcluster ^
    --master-user-password mynewpassword
```

Impostazioni per Amazon Aurora

La tabella seguente contiene informazioni dettagliate su quali impostazioni è possibile modificare, sui metodi per modificare l'impostazione e sull'ambito dell'impostazione. L'ambito determina se l'impostazione si applica all'intero cluster di database o se può essere impostata solo per specifiche istanze database.

> **Note**
>
> Per modificare un cluster database Aurora Serverless v1 o Aurora Serverless v2 sono disponibili impostazioni aggiuntive. Per informazioni su queste impostazioni, consulta Modifica di un cluster di database Aurora Serverless v1 e Gestione di cluster di database Aurora Serverless v2.
> Alcune impostazioni non sono disponibili per Aurora Serverless v1 e Aurora Serverless v2 a causa delle relative limitazioni. Per ulteriori informazioni, consultare Limitazioni di Aurora Serverless v1 e Requisiti e limitazioni per Aurora Serverless v2.

Impostazione e descrizione	Metodo	Ambito	Note sui tempi di inattività
Auto minor version upgrade (Aggiornamento automatico della versione secondaria) Specifica se l'istanza database deve ricevere automaticamente gli aggiornamenti della versione secondaria del motore preferiti quando diventano disponibili. Gli aggiornamenti vengono installati solo durante la finestra di manutenzione pianificata. Per ulteriori informazioni sugli aggiornamenti del motore, consulta Aggiornamenti del motore di database Amazon Aurora PostgreSQL per 02/06/2020 e Aggiornamenti del motore del database per Amazon Aurora MySQL. Per ulteriori informazioni sull'impostazione di Auto Minor	**Note** Questa impostazione è abilitata per impostazione predefinita. Per ogni nuovo cluster, scegli il valore appropriato per questa impostazione in base all'importanza, alla durata prevista e alla quantità di test di verifica eseguiti dopo ogni aggiornamento. Quando modifichi questa impostazione, esegui la modifica per ogni istanza database nel cluster Aurora. Se un'istanza database del cluster ha questa impostazione disattiva	L'intero cluster di database	Durante questa modifica non si verifica un'interruzione. Le interruzioni si verificano durante le finestre di manutenzione future quando Aurora applica gli aggiornamenti automatici.

Impostazione e descrizione	Metodo	Ambito	Note sui tempi di inattività
Version Upgrade (Aggiornamento automatico minore della versione) per Aurora MySQL, consulta Abilitazione degli aggiornamenti automatici tra versioni secondarie di Aurora MySQL.	ta, il cluster non viene aggiornato automatic amente. Tramite la Console di gestione AWS, Modifica di un'istanza database in un cluster database. Utilizzando l'interfa ccia AWS CLI, esegui il comando modify-db-instance e imposta l'opzione `--auto-minor-version-upgrade|--no-auto-minor-version-upgrade` . Utilizzando l'API RDS, chiama l'operazione ModifyDBInstance e imposta il parametro `AutoMinorVersionUpgrade` .		

Impostazione e descrizione	Metodo	Ambito	Note sui tempi di inattività
Backup retention period (Periodo di retention dei backup) Numero di giorni durante i quali vengono conservati i backup automatici. Il valore minimo è 1. Per ulteriori informazioni, consulta Backup.	Tramite la Console di gestione AWS, Modifica del cluster di database tramite la console, la CLI e l'API. Utilizzando l'interfaccia AWS CLI, esegui il comando modify-db-cluster e imposta l'opzione `--backup-retention-period`. Utilizzando l'API RDS, chiama l'operazione ModifyDBCluster e imposta il parametro `BackupRetentionPeriod` .	L'intero cluster di database	Durante questa modifica non si verifica un'interruzione.

Impostazione e descrizione	Metodo	Ambito	Note sui tempi di inattività
Finestra di backup (Ora di inizio) Intervallo di tempo in cui vengono eseguiti i backup automatici del database. L'ora di inizio della finestra di backup è indicata in base al formato Universal Coordinated Time (UTC) e la sua durata è in ore. I backup di Aurora sono continui e incrementali, tuttavia si utilizza una finestra di backup per creare un backup giornaliero del sistema che viene conservato per tutto il periodo di conservazione del backup. È anche possibile copiarlo per conservarlo al di fuori del periodo di conservazione. La finestra di manutenzione e la finestra di backup per il cluster di	Tramite la Console di gestione AWS, Modifica del cluster di database tramite la console, la CLI e l'API. Utilizzando l'interfaccia AWS CLI, esegui il comando modify-db-cluster e imposta l'opzione `--preferred-backup-window`. Utilizzando l'API RDS, chiama l'operazione ModifyDBCluster e imposta il parametro `PreferredBackupWindow` .	L'intero cluster di database	Durante questa modifica non si verifica un'interruzione.

Impostazione e descrizione	Metodo	Ambito	Note sui tempi di inattività
database non possono sovrapporsi. Per ulteriori informazioni, consulta Backup window (Finestra di backup).			
Impostazioni della capacità Proprietà di dimensionamento di un cluster database Aurora Serverless v1. Le proprietà di dimensionamento possono essere modificate solo per i cluster di database in modalità di motore DB `serverless` . Per informazioni su Aurora Serverless v1, consulta Utilizzo di Amazon Aurora Serverless v1.	Tramite la Console di gestione AWS, Modifica del cluster di database tramite la console, la CLI e l'API. Utilizzando l'interfaccia AWS CLI, esegui il comando modify-db-cluster e imposta l'opzione `--scaling-configuration` . Utilizzando l'API RDS, chiama l'operazione ModifyDBCluster e imposta il parametro `ScalingConfiguration` .	L'intero cluster di database	Durante questa modifica non si verifica un'interruzione. La modifica avviene immediatamente. Questa impostazione ignora l'impostazione Applica immediatamente.

Impostazione e descrizione	Metodo	Ambito	Note sui tempi di inattività
Autorità di certificazione L'autorità di certificazione (CA) per il certificato del server utilizzato dall'istanza database.	Tramite la Console di gestione AWS, Modifica di un'istanza database in un cluster database. Utilizzando l'interfaccia AWS CLI, esegui il comando modify-db-instance e imposta l'opzione `--ca-certificate-identifier` . Utilizzando l'API RDS, chiama l'operazione ModifyDBInstance e imposta il parametro `CACertificateIdentifier` .	Solo l'istanza database specificata	Un'interruzione si verifica solo se il motore database non supporta la rotazione senza riavvio. È possibile utilizzare il comando AWS CLI describe-db-engine-versions per determinare se il motore database supporta la rotazione senza riavvio.

Impostazione e descrizione	Metodo	Ambito	Note sui tempi di inattività
Configurazione dell'archiviazione del cluster Tipo di archiviazione per il cluster database: Aurora I/O-Optimized o Aurora Standard. Per ulteriori informazioni, consulta Configurazioni dell'archiviazione per i cluster database Amazon Aurora.	Tramite la Console di gestione AWS, Modifica del cluster di database tramite la console, la CLI e l'API. Utilizzando l'interfaccia AWS CLI, esegui il comando modify-db-cluster e imposta l'opzione `--storage-type` . Utilizzando l'API RDS, chiama l'operazione ModifyDBCluster e imposta il parametro `StorageType` .	L'intero cluster di database	La modifica del tipo di archiviazione di un cluster di database Aurora PostgreSQL con classi di istanze Letture ottimizzate causa un'interruzione. Ciò non si verifica quando si modificano i tipi di archiviazione per i cluster con altri tipi di classi di istanze. Per ulteriori informazioni sui tipi di classe di istanza database, consulta Tipi di classi di istanza database.

Impostazione e descrizione	Metodo	Ambito	Note sui tempi di inattività
Copy tags to snapshots (Copia tag in snapshot) Seleziona questa opzione per specificare di copiare i tag definiti per questo cluster di database in snapshot DB create da tale cluster di database. Per ulteriori informazioni, consulta Applicazione di tag alle risorse Amazon Aurora e Amazon RDS.	Tramite la Console di gestione AWS, Modifica del cluster di database tramite la console, la CLI e l'API. Tramite AWS CLI, esegui https://docs.aws.amazon.com/cli/latest/reference/rds/modify-db-cluster.html e imposta l'opzione `--copy-tags-to-snapshot` o `--no-copy-tags-to-snapshot`. Tramite l'API RDS, chiama https://docs.aws.amazon.com/AmazonRDS/latest/APIReference/API_ModifyDBCluster.html e imposta il parametro `CopyTagsToSnapshot`.	L'intero cluster di database	Durante questa modifica non si verifica un'interruzione.

Impostazione e descrizione	Metodo	Ambito	Note sui tempi di inattività
Data API (API dati Puoi accedere ad Aurora Serverless v1 con applicazioni basate su servizi Web, come AWS Lambda e AWS AppSync. Questa impostazione è valida solo per un cluster DB Aurora Serverless v1. Per ulteriori informazioni, consulta Utilizzo dell'API dati di Amazon RDS.	Tramite la Console di gestione AWS, Modifica del cluster di database tramite la console, la CLI e l'API. Tramite AWS CLI, esegui `modify-db-cluster` e imposta l'opzione `--enable-http-endpoint` . Tramite l'API RDS, chiama `ModifyDBCluster` e imposta il parametro `EnableHttpEndpoint` .	L'intero cluster di database	Durante questa modifica non si verifica un'interruzione.

Impostazione e descrizione	Metodo	Ambito	Note sui tempi di inattività
Database authentication (Autenticazione del database) L'autenticazione del database da utilizzare. Per MySQL: • Scegliere Password authentication (Autenticazione tramite password) per autenticare gli utenti di database solo con le password del database. • Seleziona Autenticazione database tramite password e IAM per autenticare gli utenti del database con password del database e credenziali utente tramite utenti e ruoli IAM. Per ulteriori informazioni, consulta Autenticazione del database IAM .	Tramite la Console di gestione AWS, Modifica del cluster di database tramite la console, la CLI e l'API. Utilizzando la AWS CLI, esegui il comando modify-db-cluster e imposta le seguenti opzioni: • Per l'autenticazione IAM, imposta l'opzione `--enable-iam-database-authentication|--no-enable-iam-database-authentication` . • Per l'autenticazione Kerberos imposta le opzioni `--domain` e `--domain-iam-role-name` . Utilizzando l'API RDS, richiama ModifyDBC	L'intero cluster di database	Durante questa modifica non si verifica un'interruzione.

Impostazione e descrizione	Metodo	Ambito	Note sui tempi di inattività
Per PostgreSQL: • Scegli IAM database authentication (Autenticazione database IAM) per autenticare gli utenti del database con password del database e credenziali utente tramite utenti e ruoli. Per ulteriori informazioni, consulta Autenticazione del database IAM . • Seleziona Autenticazione Kerberos per autenticare le password del database e le credenziali utente utilizzando l'autenticazione Kerberos. Per ulteriori informazioni, consulta Utilizzo di Autenticazione Kerberos con Aurora PostgreSQL.	luster e imposta i seguenti parametri: • Per l'autenticazione IAM, imposta il parametro `EnableIAMDatabaseAuthentication` . • Per l'autenticazione Kerberos, imposta i parametri `Domain` e `DomainIAMRoleName` .		

Impostazione e descrizione	Metodo	Ambito	Note sui tempi di inattività
Database port (Porta del database) Porta da utilizzare per accedere al cluster di database.	Tramite la Console di gestione AWS, Modifica del cluster di database tramite la console, la CLI e l'API. Tramite AWS CLI, esegui `modify-db-cluster` e imposta l'opzione `--port`. Tramite l'API RDS, chiama `ModifyDBCluster` e imposta il parametro `Port`.	L'intero cluster di database	Durante questa modifica si verifica un'interruzione. Tutte le istanze database nel cluster di database vengono riavviate immediatamente.

Impostazione e descrizione	Metodo	Ambito	Note sui tempi di inattività
DB Cluster Identifier (Identificatore cluster DB) L'identificatore del cluster di database. Questo valore è archiviato come stringa in caratteri minuscoli. Quando si modifica l'identificatore del cluster database, cambiano anche gli endpoint del cluster database. Gli endpoint delle istanze database nel cluster database non cambiano.	Tramite la Console di gestione AWS, Modifica del cluster di database tramite la console, la CLI e l'API. Tramite AWS CLI, esegui `modify-db-cluster` e imposta l'opzione `--new-db-cluster-identifier` . Tramite l'API RDS, chiama `ModifyDBCluster` e imposta il parametro `NewDBClusterIdentifier` .	L'intero cluster di database	Durante questa modifica non si verifica un'interruzione.

Impostazione e descrizione	Metodo	Ambito	Note sui tempi di inattività
DB cluster parameter group (Gruppo di parametri del cluster database) Gruppo di parametri del cluster di database da associare al cluster di database. Per ulteriori informazioni, consulta Gruppi di parametri per Amazon Aurora.	Tramite la Console di gestione AWS, Modifica del cluster di database tramite la console, la CLI e l'API. Tramite AWS CLI, esegui `modify-db-cluster` e imposta l'opzione `--db-cluster-parameter-group-name` . Tramite l'API RDS, chiama `ModifyDBCluster` e imposta il parametro `DBClusterParameterGroupName` .	L'intero cluster di database	Durante questa modifica non si verifica un'interruzione. Quando modifichi il gruppo di parametri, le modifiche apportate ad alcuni parametri vengono applicate alle istanze database nel cluster di database immediatamente senza un riavvio. Le modifiche apportate ad altri parametri vengono applicate solo dopo che le istanze database nel cluster di database vengono riavviate.

Impostazione e descrizione	Metodo	Ambito	Note sui tempi di inattività
DB instance class (Classe istanza database) Classe dell'istanza database da utilizzare. Per ulteriori informazioni, consulta Classi di istanze database Amazon Aurora.	Tramite la Console di gestione AWS, Modifica di un'istanza database in un cluster database. Tramite AWS CLI, esegui `modify-db-instance` e imposta l'opzione `--db-instance-class` . Tramite l'API RDS, chiama `ModifyDBInstance` e imposta il parametro `DBInstanceClass` .	Solo l'istanza database specificata	Durante questa modifica si verifica un'interruzione.

Impostazione e descrizione	Metodo	Ambito	Note sui tempi di inattività
DB instance identifier (Identificatore istanze DB) Identificatore istanze DB. Questo valore è archiviato come stringa in caratteri minuscoli.	Tramite la Console di gestione AWS, Modifica di un'istanza database in un cluster database. Tramite AWS CLI, esegui `modify-db-instance` e imposta l'opzione `--new-db-instance-identifier`. Tramite l'API RDS, chiama `ModifyDBInstance` e imposta il parametro `NewDBInstanceIdentifier`.	Solo l'istanza database specificata	Durante questa modifica si verifica un'interruzione. RDS riavvia l'istanza database per aggiornare i seguenti elementi: • Aurora MySQL: colonna `SERVER_ID` nella tabella `information_schema.replica_host_status` • Aurora PostgreSQL: colonna `server_id` nella funzione `aurora_replica_status()`

Impostazione e descrizione	Metodo	Ambito	Note sui tempi di inattività
DB parameter group (Gruppo di parametri database) Gruppo di parametri database da associare all'istanza database. Per ulteriori informazioni, consulta Gruppi di parametri per Amazon Aurora.	Tramite la Console di gestione AWS, Modifica di un'istanza database in un cluster database. Tramite AWS CLI, esegui `modify-db-instance` e imposta l'opzione `--db-parameter-group-name` . Tramite l'API RDS, chiama `ModifyDBInstance` e imposta il parametro `DBParameterGroupName` .	Solo l'istanza database specificata	Durante questa modifica non si verifica un'interruzione. Quando si associa un nuovo gruppo parametri del database a un'istanza database, i parametri statici e dinamici modificati vengono applicati solo dopo il riavvio dell'istanza database. Tuttavia, se modifichi i parametri dinamici nel gruppo di parametri database associato all'istanza database, tali modifiche vengono applicate immediatamente senza eseguire il riavvio. Per ulteriori informazioni, consultare Gruppi di parametri per Amazon Aurora e Riavvio di un cluster Amazon Aurora DB o di un'istanza Amazon Aurora DB.

Impostazione e descrizione	Metodo	Ambito	Note sui tempi di inattività
Deletion protection (Protezione da eliminazione) L'opzione Enable deletion protection (Abilita protezione da eliminazione) permette di impedire l'eliminazione del cluster di database. Per ulteriori informazioni, consulta Protezione dall'eliminazione per i cluster Aurora.	Tramite la Console di gestione AWS, Modifica del cluster di database tramite la console, la CLI e l'API. Tramite AWS CLI, esegui `modify-db-cluster` e imposta l'opzione `--deletion-protection|--no-deletion-protection` . Tramite l'API RDS, chiama `ModifyDBCluster` e imposta il parametro `DeletionProtection` .	L'intero cluster di database	Durante questa modifica non si verifica un'interruzione.

Impostazione e descrizione	Metodo	Ambito	Note sui tempi di inattività
Versione del motore La versione del motore di database da utilizzare. Prima di eseguire l'aggiornamento del cluster di database di produzione, è consigliabile testare il processo di aggiornamento in un cluster di database di test per verificarne la durata e convalidare le applicazioni.	Tramite la Console di gestione AWS, Modifica del cluster di database tramite la console, la CLI e l'API. Tramite AWS CLI, esegui `modify-db-cluster` e imposta l'opzione `--engine-version` . Tramite l'API RDS, chiama `ModifyDBCluster` e imposta il parametro `EngineVersion` .	L'intero cluster di database	Durante questa modifica si verifica un'interruzione.

Impostazione e descrizione	Metodo	Ambito	Note sui tempi di inattività
Enhanced Monitoring (Monitoraggio avanzato) L'opzione Enable enhanced monitoring (Abilita monitoraggio avanzato) permette di raccogliere i parametri in tempo reale per il sistema operativo in cui viene eseguita l'istanza database. Per ulteriori informazioni, consulta Monitoraggio dei parametri del sistema operativo con il monitoraggio avanzato.	Tramite la Console di gestione AWS, Modifica di un'istanza database in un cluster database. Tramite AWS CLI, esegui `modify-db-instance` e imposta le opzioni `--monitoring-role-arn` e `--monitoring-interval` . Tramite l'API RDS, chiama `ModifyDBInstance` e imposta i parametri `MonitoringRoleArn` e `MonitoringInterval` .	Solo l'istanza database specificata	Durante questa modifica non si verifica un'interruzione.

Impostazione e descrizione	Metodo	Ambito	Note sui tempi di inattività
Log exports (Esportazioni log) Seleziona i tipi di log da pubblicare su Amazon CloudWatch Logs. Per ulteriori informazioni, consulta File di log del database Aurora MySQL.	Tramite la Console di gestione AWS, Modifica del cluster di database tramite la console, la CLI e l'API. Tramite AWS CLI, esegui `modify-db-cluster` e imposta l'opzione `--cloudwatch-logs-export-configuration` . Tramite l'API RDS, chiama `ModifyDBCluster` e imposta il parametro `CloudwatchLogsExportConfiguration` .	L'intero cluster di database	Durante questa modifica non si verifica un'interruzione.

Impostazione e descrizione	Metodo	Ambito	Note sui tempi di inattività
Maintenance window (Finestra di manutenzione) Intervallo di tempo durante cui viene eseguita la manutenzione del sistema. La manutenzione del sistema include gli aggiornamenti, se applicabili. L'ora di inizio della finestra di manutenzione è indicata in base al formato Universal Coordinated Time (UTC) e la sua durata è in ore. Se imposti la finestra sull'ora corrente, vi devono essere almeno 30 minuti tra l'ora corrente e la fine della finestra per assicurare che tutte le modifiche in sospeso vengano applicate. Puoi impostare la finestra di manutenzione in modo indipendente per il	Per cambiare la finestra di manutenzione per il cluster di database tramite la Console di gestione AWS, Modifica del cluster di database tramite la console, la CLI e l'API. Per cambiare la finestra di manutenzione per un'istanza database tramite la Console di gestione AWS, Modifica di un'istanza database in un cluster database. Per cambiare la finestra di manutenzione per il cluster di database tramite AWS CLI, eseguire `modify-db-cluster` e impostare l'opzione `--preferred-maintenance-window` . Per cambiare la finestra di manutenzione per un'istanza database tramite	L'intero cluster di database o un'unica istanza database	Se sono presenti una o più operazioni in sospeso che causano un'interruzione e la finestra di manutenzione viene modificata per includere l'ora corrente, tali operazioni in sospeso vengono applicate immediatamente e si verifica un'interruzione.

Impostazione e descrizione	Metodo	Ambito	Note sui tempi di inattività
cluster di database e per ogni istanza database nel cluster di database. Quando l'ambito di una modifica è l'intero cluster di database, la modifica viene eseguita durante la finestra di manutenzione del cluster di database. Quando l'ambito di una modifica è un'istanza database, la modifica viene eseguita durante la finestra di manutenzione di tale istanza database. La finestra di manutenzione e la finestra di backup per il cluster di database non possono sovrapporsi. Per ulteriori informazioni, consulta Finestra di manutenzione Amazon RDS.	AWS CLI, eseguire `modify-db-instance` e impostare l'opzione `--preferred-maintenance-window` . Per cambiare la finestra di manutenzione per il cluster di database tramite l'API RDS, chiamare `ModifyDBCluster` e impostare il parametro `PreferredMaintenanceWindow` . Per cambiare la finestra di manutenzione per un'istanza database tramite l'API RDS, chiamare `ModifyDBInstance` e impostare il parametro `PreferredMaintenanceWindow` .		

Impostazione e descrizione	Metodo	Ambito	Note sui tempi di inattività
Manage master credentials in Gestione dei segreti AWS (Gestione credenziali master in Gestione dei segreti AWS) Seleziona Manage master credentials in Gestione dei segreti AWS (Gestione credenziali master in Gestione dei segreti AWS) per gestire la password dell'utente master in un segreto di Secrets Manager. Facoltativamente, scegli la chiave KMS da utilizzare per proteggere il segreto. Scegliere tra le chiavi KMS presenti nell'account o inserire la chiave da un altro account. Per ulteriori informazioni, consulta Gestione delle password con Amazon Aurora e	Tramite la Console di gestione AWS, Modifica di un'istanza database in un cluster database. Tramite AWS CLI, esegui `modify-db-cluster` e imposta le opzioni `--manage-master-user-password | --no-manage-master-user-password` e `--master-user-secret-kms-key-id` . Per ruotare immediatamente la password dell'utente master, imposta l'opzione `--rotate-master-user-password` . Tramite l'API RDS, chiama `ModifyDBCluster` e imposta i parametri `ManageMasterUserPassword` e `MasterUserSecretKmsKeyId` . Per	L'intero cluster di database	Durante questa modifica non si verifica un'interruzione.

Impostazione e descrizione	Metodo	Ambito	Note sui tempi di inattività
Gestione dei segreti AWS. Se Aurora sta già gestendo la password dell'utente master per il cluster database, puoi ruotare la password dell'utente master scegliendo Rotate secret immediately (Ruota il segreto immediatamente). Per ulteriori informazioni, consulta Gestione delle password con Amazon Aurora e Gestione dei segreti AWS.	ruotare immediatamente la password dell'utente master, imposta il parametro `RotateMasterUserPassword` su `true`.		

Impostazione e descrizione	Metodo	Ambito	Note sui tempi di inattività
Tipo di rete I protocolli di indirizzo IP supportati dal cluster database. IPv4 per specifica re che le risorse possono comunicare con il cluster database solo tramite il protocoll o di indirizzamento IPv4. Modalità dual-stack per specificare che le risorse possono comunicare con il cluster database su IPv4, IPv6 o entrambi. Utilizza la modalità dual-stack se le risorse devono comunicare con il cluster database sul protocollo di indirizzamento IPv6. Per utilizzare la modalità dual-stac k, assicurarsi che almeno due sottoreti si estendano su due zone di disponibi lità che supportano	Tramite la Console di gestione AWS, Modifica del cluster di database tramite la console, la CLI e l'API. Tramite AWS CLI, esegui `modify-db-cluster` e imposta l'opzione `--network-type` . Tramite l'API RDS, chiama `ModifyDBCluster` e imposta il parametro `NetworkType` .	L'intero cluster di database	Durante questa modifica non si verifica un'interr uzione.

Impostazione e descrizione	Metodo	Ambito	Note sui tempi di inattività
entrambi i protocolli di rete IPv4 e IPv6. Inoltre, assicurarsi di associare un blocco CIDR IPv6 alle sottoreti del gruppo di sottoreti DB specificato. Per ulteriori informazioni, consulta Assegnazione di indirizzi IP in Amazon Aurora.			

Impostazione e descrizione	Metodo	Ambito	Note sui tempi di inattività
New master password (Nuova password master) Password dell'utente master. • Per Aurora MySQL, la password deve contenere da 8 a 41 caratteri ASCII stampabili. • Per Aurora PostgreSQL, deve contenere da 8 a 99 caratteri ASCII stampabili. • Non può contenere /, ", @ o uno spazio.	Tramite la Console di gestione AWS, Modifica di un'istanza database in un cluster database. Tramite AWS CLI, esegui `modify-db-cluster` e imposta l'opzione `--master-user-password` . Tramite l'API RDS, chiama `ModifyDBCluster` e imposta il parametro `MasterUserPassword` .	L'intero cluster di database	Durante questa modifica non si verifica un'interruzione.

Impostazione e descrizione	Metodo	Ambito	Note sui tempi di inattività
Approfondimenti sulle prestazioni Specifica se abilitare Performance Insights, uno strumento che monitora il carico di istanze database per consentire di analizzare e risolvere i problemi di performance del database. Per ulteriori informazioni, consulta Monitoraggio del carico DB con Performance Insights su Amazon Aurora.	Tramite la Console di gestione AWS, Modifica di un'istanza database in un cluster database. Tramite AWS CLI, esegui `modify-db-instance` e imposta l'opzione `--enable-performance-insights|--no-enable-performance-insights` . Tramite l'API RDS, chiama `ModifyDBInstance` e imposta il parametro `EnablePerformanceInsights` .	Solo l'istanza database specificata	Durante questa modifica non si verifica un'interruzione.

Impostazione e descrizione	Metodo	Ambito	Note sui tempi di inattività
Performance Insights AWS KMS key L'identificatore di AWS KMS key per la crittografia dei dati di Performance Insights. L'identificatore della chiave KMS è il nome della risorsa Amazon Resource Name (ARN), l'identificatore della chiave o l'alias della chiave per la chiave KMS. Per ulteriori informazioni, consulta Attivazione e disattivazione di Approfondimenti sulle prestazioni per Aurora.	Tramite la Console di gestione AWS, Modifica di un'istanza database in un cluster database. Tramite AWS CLI, esegui `modify-db-instance` e imposta l'opzione `--performance-insights-kms-key-id` . Tramite l'API RDS, chiama `ModifyDBInstance` e imposta il parametro `PerformanceInsightsKMSKeyId` .	Solo l'istanza database specificata	Durante questa modifica non si verifica un'interruzione.

Impostazione e descrizione	Metodo	Ambito	Note sui tempi di inattività
Performance Insights retention period (Periodi di retention di Performance Insights Quantità di tempo, espressa in giorni, per cui conservare i dati di Performance Insights. L'impostazione del periodo di conservazione è Impostazione predefinita (7 giorni). Per mantenere i dati sulle prestazioni più a lungo, specifica da 1 a 24 mesi. Per altre informazioni sui periodi di conservazione, consulta Prezzi e conservazione dei dati per Performance Insights. Per ulteriori informazioni, consulta Attivazione e disattivazione di Approfondimenti sulle prestazioni per Aurora.	Tramite la Console di gestione AWS, Modifica di un'istanza database in un cluster database. Tramite AWS CLI, esegui `modify-db-instance` e imposta l'opzione `--performance-insights-retention-period` . Tramite l'API RDS, chiama `ModifyDBInstance` e imposta il parametro `PerformanceInsightsRetentionPeriod` .	Solo l'istanza database specificata	Durante questa modifica non si verifica un'interruzione.

Impostazione e descrizione	Metodo	Ambito	Note sui tempi di inattività
Promotion tier (Livello di promozione) Valore che specifica l'ordine di promozione di una replica di Aurora a istanza primaria in un cluster di database dopo un errore dell'istanza primaria esistente. Per ulteriori informazioni, consulta Tolleranza ai guasti di un cluster DB Aurora.	Tramite la Console di gestione AWS, Modifica di un'istanza database in un cluster database. Tramite AWS CLI, esegui `modify-db-instance` e imposta l'opzione `--promotion-tier` . Tramite l'API RDS, chiama `ModifyDBInstance` e imposta il parametro `PromotionTier` .	Solo l'istanza database specificata	Durante questa modifica non si verifica un'interruzione.

Impostazione e descrizione	Metodo	Ambito	Note sui tempi di inattività
Accesso pubblico Publicly accessibl e (Accessibile pubblicamente) per assegnare all'istanza database un indirizzo IP pubblico, ovvero renderla accessibile al di fuori del VPC. Per essere accessibi le pubblicamente, l'istanza database deve anche trovarsi in una sottorete pubblica nel VPC. Not publicly accessibl e (Non accessibi le pubblicamente) per rendere l'istanza database accessibile solo dall'interno del VPC. Per ulteriori informazi oni, consulta Nascondere cluster database in un VPC da Internet. Per connettersi a un'istanza DB dall'esterno Amazon VPC, l'istanza DB	Tramite la Console di gestione AWS, Modifica di un'istanza database in un cluster database. Tramite AWS CLI, esegui `modify-db-instance` e imposta l'opzione `--publicly-accessible|--no-publicly-accessible` . Tramite l'API RDS, chiama `ModifyDBInstance` e imposta il parametro `PubliclyAccessible` .	Solo l'istanza database specificata	Durante questa modifica non si verifica un'interr uzione.

Impostazione e descrizione	Metodo	Ambito	Note sui tempi di inattività
deve essere accessibile pubblicamente, l'accesso deve essere concesso utilizzando le regole in ingresso del gruppo di sicurezza dell'istanza DB e devono essere soddisfatti altri requisiti. Per ulteriori informazioni, consulta Impossibile connettersi all'istanza database di Amazon RDS. Se l'istanza database non è accessibile pubblicamente, puoi anche utilizzare una connessione AWS Site-to-Site VPN o una connessione Direct Connect per accedervi da una rete privata. Per ulteriori informazioni, consulta Riservatezza del traffico Internet.			

Impostazione e descrizione	Metodo	Ambito	Note sui tempi di inattività
Impostazioni della capacità Serverless v2 La capacità del database di un cluster database Aurora Serverless v2 misurata in unità di capacità di Aurora (ACU). Per ulteriori informazioni, consulta Impostazione dell'intervallo di capacità di Aurora Serverless v2 per un cluster.	Tramite la Console di gestione AWS, Modifica del cluster di database tramite la console, la CLI e l'API. Tramite AWS CLI, esegui `modify-db-cluster` e imposta l'opzione `--serverless-v2-scaling-configuration` . Tramite l'API RDS, chiama `ModifyDBCluster` e imposta il parametro `ServerlessV2ScalingConfiguration` .	L'intero cluster di database	Durante questa modifica non si verifica un'interruzione. La modifica avviene immediatamente. Questa impostazione ignora l'impostazione Applica immediatamente.

Impostazione e descrizione	Metodo	Ambito	Note sui tempi di inattività
Gruppo di sicurezza Gruppo di sicurezza da associare al cluster di database. Per ulteriori informazioni, consulta Controllo dell'accesso con i gruppi di sicurezza.	Tramite la Console di gestione AWS, Modifica del cluster di database tramite la console, la CLI e l'API. Tramite AWS CLI, esegui `modify-db-cluster` e imposta l'opzione `--vpc-security-group-ids`. Tramite l'API RDS, chiama `ModifyDBCluster` e imposta il parametro `VpcSecurityGroupIds` .	L'intero cluster di database	Durante questa modifica non si verifica un'interruzione.

Impostazione e descrizione	Metodo	Ambito	Note sui tempi di inattività
Target Backtrack window (Finestra di backtrack di destinazione) Quantità di tempo per cui è possibile eseguire il backtrack del cluster di database, espresso in secondi. Questa impostazione è disponibile solo per Aurora MySQL e solo se il cluster di database è stato creato con la funzionalità di backtrack abilitata.	Tramite la Console di gestione AWS, Modifica del cluster di database tramite la console, la CLI e l'API. Tramite AWS CLI, esegui `modify-db-cluster` e imposta l'opzione `--backtrack-window` . Tramite l'API RDS, chiama `ModifyDBCluster` e imposta il parametro `BacktrackWindow` .	L'intero cluster di database	Durante questa modifica non si verifica un'interruzione.

Impostazioni che non si applicano ai cluster di database Amazon Aurora

Le impostazioni seguenti nel comando della AWS CLI `modify-db-cluster` e l'operazione `ModifyDBCluster` dell'API RDS non si applicano ai cluster di database Amazon Aurora.

> **Note**
>
> Non è possibile utilizzare la Console di gestione AWS per modificare queste impostazioni per i cluster di database Aurora.

AWS CLIImpostazione	Impostazione API RDS
`--allocated-storage`	`AllocatedStorage`
`--auto-minor-version-upgrade | --no-auto-minor-version-upgrade`	`AutoMinorVersionUpgrade`
`--db-cluster-instance-class`	`DBClusterInstanceClass`
`--enable-performance-insights | --no-enable-performance-insights`	`EnablePerformanceInsights`
`--iops`	`Iops`
`--monitoring-interval`	`MonitoringInterval`
`--monitoring-role-arn`	`MonitoringRoleArn`
`--option-group-name`	`OptionGroupName`
`--performance-insights-kms-key-id`	`PerformanceInsightsKMSKeyId`
`--performance-insights-retention-period`	`PerformanceInsightsRetentionPeriod`

Impostazioni che non si applicano alle istanze database Amazon Aurora

Le impostazioni seguenti nel comando della AWS CLI `modify-db-instance` e l'operazione `ModifyDBInstance` dell'API RDS non si applicano alle istanze database Amazon Aurora.

> **Note**
>
> Non è possibile utilizzare la Console di gestione AWS per modificare queste impostazioni per le istanze database Aurora.

AWS CLIImpostazione	Impostazione API RDS
`--allocated-storage`	`AllocatedStorage`
`--allow-major-version-upgrade|--no-allow-major-version-upgrade`	`AllowMajorVersionUpgrade`
`--copy-tags-to-snapshot|--no-copy-tags-to-snapshot`	`CopyTagsToSnapshot`
`--domain`	`Domain`
`--db-security-groups`	`DBSecurityGroups`
`--db-subnet-group-name`	`DBSubnetGroupName`
`--domain-iam-role-name`	`DomainIAMRoleName`
`--multi-az|--no-multi-az`	`MultiAZ`
`--iops`	`Iops`
`--license-model`	`LicenseModel`
`--network-type`	`NetworkType`
`--option-group-name`	`OptionGroupName`
`--processor-features`	`ProcessorFeatures`
`--storage-type`	`StorageType`
`--tde-credential-arn`	`TdeCredentialArn`
`--tde-credential-password`	`TdeCredentialPassword`
`--use-default-processor-features|--no-use-default-processor-features`	`UseDefaultProcessorFeatures`

Aggiunta di repliche di Aurora a un cluster di database

Un cluster di database di Aurora con repliche ha un'istanza database primaria e fino a 15 repliche di Aurora. L'istanza database primaria supporta operazioni di lettura e scrittura ed esegue tutte le modifiche ai dati nel volume del cluster. Le repliche di Aurora si connettono allo stesso volume di storage dell'istanza DB primaria ma supportano solo le operazioni di lettura. È possibile utilizzare le repliche di Aurora per effettuare l'offload dei carichi di lavoro in lettura dall'istanza DB primaria. Per ulteriori informazioni, consulta Repliche di Aurora.

Le repliche di Amazon Aurora presentano le seguenti limitazioni:

- Non è possibile creare una replica di Aurora per un cluster database Aurora Serverless v1. Aurora Serverless v1 ha una singola istanza database che si ridimensiona automaticamente per supportare tutte le operazioni di lettura e scrittura del database.

  Puoi tuttavia aggiungere istanze di lettura ai cluster DB Aurora Serverless v2. Per ulteriori informazioni, consulta Aggiunta di un'istanza Aurora Serverless v2 di lettura.

Consigliamo di distribuire l'istanza primaria e le repliche di Aurora del cluster di database Aurora in più zone di disponibilità, in modo da migliorare la disponibilità del cluster di database. Per ulteriori informazioni, consulta Disponibilità nelle regioni.

Per rimuovere una replica di Aurora da un cluster di database Aurora, eliminare la replica di Aurora seguendo le istruzioni in Eliminazione di un'istanza database da un cluster database Aurora.

> **Note**
>
> Amazon Aurora supporta anche la replica tramite un database esterno o un'istanza database RDS. L'istanza DB RDS deve trovarsi nella stessa AWS regione di Amazon Aurora. Per ulteriori informazioni, consulta Replica con Amazon Aurora.

È possibile aggiungere repliche Aurora a un cluster DB utilizzando l'API Console di gestione AWSAWS CLI, the o RDS.

Console

Per aggiungere una replica di Aurora a un cluster di database

1. Accedi a Console di gestione AWS e apri la console Amazon RDS all'indirizzo https://console.aws.amazon.com/rds/.
2. Nel riquadro di navigazione, scegliere Databases (Database), quindi selezionare il cluster di database a cui aggiungere la nuova istanza DB.
3. Assicurarsi che sia il cluster che l'istanza primaria siano nello stato Available (Disponibile) . Se il cluster DB o l'istanza primaria si trovano in uno stato di transizione, ad esempio Creating (Creazione), non è possibile aggiungere una replica.

   Se il cluster non ha un'istanza primaria, creane una utilizzando il create-db-instanceAWS CLIcomando. Questa situazione può verificarsi se si utilizza l'interfaccia a riga di comando per ripristinare uno snapshot del cluster DB e quindi visualizzare il cluster nella Console di gestione AWS.
4. In Actions (Operazioni), scegliere Add reader (Aggiungi lettore).

   Viene visualizzata la pagina Add reader (Aggiungi lettore).
5. Nela pagina Add reader (Aggiungi lettore), specificare le opzioni per la replica Aurora. La tabella riportata di seguito mostra le impostazioni di una replica di Aurora.

Per questa opzione	Eseguire questa operazione
Availability zone (Zona di disponibilità)	Stabilire se si desidera specificare una zona di disponibilità particolare. L'elenco include solo le zone di disponibilità mappate al gruppo di sottoreti del database scelto al momento della creazione del cluster database. Per ulteriori informazioni sulle zone di disponibilità, consultare Regioni e zone di disponibilità.
Accessibile pubblicamente	Selezionare `Yes` per assegnare alla replica di Aurora un indirizzo IP pubblico. Altrimenti, selezionare `No`. Per ulteriori informazioni su come nascondere le repliche di Aurora per impedire l'accesso pubblico, consultare Nascondere cluster database in un VPC da Internet.

Per questa opzione	Eseguire questa operazione
Encryption (Crittografia)	Selezionare `Enable encryption` per abilitare la crittografia dei dati inattivi per questa replica di Aurora. Per ulteriori informazioni, consulta Crittografia delle risorse Amazon Aurora.
DB instance class (Classe istanza database)	Selezionare una classe di istanze DB che definisca i requisiti di elaborazione e di memoria per replica di Aurora. Per ulteriori informazioni sulle opzioni di classe di istanza DB, consulta Classi di istanze database Amazon Aurora.
Aurora replica source (Origine replica di)	Selezionare l'identificatore dell'istanza primaria per cui creare una replica di Aurora.
DB instance identifier (Identificatore istanze DB)	Inserisci un nome per l'istanza che sia unico per il tuo account nella AWS regione selezionata. Potresti scegliere di aggiungere alcune informazioni al nome, ad esempio includendo la AWS regione e il motore DB che hai selezionato**`aurora-read-instance1`** .
Priorità	Scegliere una priorità di failover per l'istanza. Se non si specifica alcun valore, l'impostazione predefinita è tier-1 (livello 1). Questa priorità determina l'ordine di promozione delle repliche di Aurora durante il recupero da un errore dell'istanza principale. Per ulteriori informazioni, consulta Tolleranza ai guasti di un cluster DB Aurora.
Database port (Porta del database)	La porta di una replica di Aurora corrisponde a quella per il cluster di database.
DB parameter group (Gruppo di parametri database)	Selezionare un gruppo di parametri. Si può utilizzare gruppo di parametri predefiniti forniti da Aurora oppure creare un gruppo di parametri personalizzato. Per ulteriori informazioni sui gruppi di parametri, consultare Gruppi di parametri per Amazon Aurora.

Per questa opzione	Eseguire questa operazione
Approfondimenti sulle prestazioni	La casella di controllo Turn on Performance Insights (Attiva Performance Insights) è selezionata per impostazione predefinita. Il valore non viene ereditato dall'istanza di scrittura. Per ulteriori informazioni, consulta Monitoraggio del carico DB con Performance Insights su Amazon Aurora.
Enhanced Monitoring (Monitoraggio avanzato)	Scegliere Enable enhanced monitoring (Abilita monitoraggio avanzato) per abilitare la raccolta di parametri in tempo reale per il sistema operativo su cui viene eseguito il cluster DB. Per ulteriori informazioni, consulta Monitoraggio dei parametri del sistema operativo con il monitoraggio avanzato.
Monitoring Role (Ruolo monitoraggio)	Disponibile solo se Enhanced Monitoring (Monitoraggio avanzato) è impostato su Enable enhanced monitoring (Abilita monitoraggio avanzato). Scegli il ruolo IAM che hai creato per consentire ad Amazon RDS di comunicare con Amazon CloudWatch Logs per te oppure scegli Default per fare in modo che RDS crei un ruolo per te denominato. `rds-monitoring-role` Per ulteriori informazioni, consulta Monitoraggio dei parametri del sistema operativo con il monitoraggio avanzato.
Granularity (Granularità)	Disponibile solo se Enhanced Monitoring (Monitoraggio avanzato) è impostato su Enable enhanced monitoring (Abilita monitoraggio avanzato). Impostare l'intervallo, in secondi, tra le operazioni di raccolta dei parametri per il cluster DB.

Per questa opzione	Eseguire questa operazione
Auto minor version upgrade (Aggiornamento automatico della versione secondaria)	Selezionare Enable auto minor version upgrade (Abilita aggiornamenti automatici versioni minori) per abilitare il cluster di database Aurora in modo che riceva automaticamente gli aggiornamenti minori della versione del motore di database non appena diventano disponibili. L'impostazione Auto Minor Version Upgrade (Aggiornamento automatico minore della versione) si applica ai cluster di database Aurora PostgreSQL e Aurora MySQL. Per i cluster Aurora MySQL 2.x, questa impostazione aggiorna i cluster alla versione massima 2.07.2. Per ulteriori informazioni sugli aggiornamenti del motore per Aurora PostgreSQL, consultare Aggiornamenti del motore di database Amazon Aurora PostgreSQL per 02/06/2020. Per ulteriori informazioni sugli aggiornamenti del motore per Aurora MySQL, consultare Aggiornamenti del motore del database per Amazon Aurora MySQL.

6. Scegliere Add reader (Aggiungi lettore) per creare la replica Aurora.

AWS CLI

Per creare una replica Aurora nel tuo cluster DB, esegui il comando. create-db-instanceAWS CLI Includere il nome del cluster di database come opzione `--db-cluster-identifier`. È anche possibile specificare una zona di disponibilità per la replica di Aurora utilizzando il parametro `--availability-zone`, come mostrato negli esempi seguenti.

Il comando seguente crea ad esempio una nuova replica di Aurora–compatibile con MySQL 5.7 denominata `sample-instance-us-west-2a`.

Per Linux, macOS o Unix:

```
aws rds create-db-instance --db-instance-identifier sample-instance-us-west-2a \
    --db-cluster-identifier sample-cluster --engine aurora-mysql --db-instance-class
 db.r5.large \
    --availability-zone us-west-2a
```

Per Windows:

```
aws rds create-db-instance --db-instance-identifier sample-instance-us-west-2a ^
    --db-cluster-identifier sample-cluster --engine aurora-mysql --db-instance-class
 db.r5.large ^
    --availability-zone us-west-2a
```

Il comando seguente crea una nuova replica di Aurora compatibile con MySQL 5.7 denominata `sample-instance-us-west-2a`.

Per Linux, macOS o Unix:

```
aws rds create-db-instance --db-instance-identifier sample-instance-us-west-2a \
    --db-cluster-identifier sample-cluster --engine aurora-mysql --db-instance-class
 db.r5.large \
    --availability-zone us-west-2a
```

Per Windows:

```
aws rds create-db-instance --db-instance-identifier sample-instance-us-west-2a ^
    --db-cluster-identifier sample-cluster --engine aurora --db-instance-class
 db.r5.large ^
    --availability-zone us-west-2a
```

Il comando seguente consente di creare una nuova replica di Aurora compatibile con PostgreSQL denominata `sample-instance-us-west-2a`.

Per Linux, macOS o Unix:

```
aws rds create-db-instance --db-instance-identifier sample-instance-us-west-2a \
    --db-cluster-identifier sample-cluster --engine aurora-postgresql --db-instance-
class db.r5.large \
    --availability-zone us-west-2a
```

Per Windows:

```
aws rds create-db-instance --db-instance-identifier sample-instance-us-west-2a ^
    --db-cluster-identifier sample-cluster --engine aurora-postgresql --db-instance-
class db.r5.large ^
    --availability-zone us-west-2a
```

API RDS

Per creare una replica Aurora nel tuo cluster DB, chiama l'operazione Create. DBInstance
Includere il nome del cluster di database come parametro `DBClusterIdentifier`. È anche possibile specificare una zona di disponibilità per la replica di Aurora utilizzando il parametro `AvailabilityZone`.

Per informazioni sul dimensionamento automatico di Amazon Aurora con le repliche Aurora, consultare le seguenti sezioni.

Argomenti

- Dimensionamento automatico di Amazon Aurora con le repliche Aurora
- Aggiunta di una policy di dimensionamento automatico a un cluster di database Amazon Aurora
- Modifica di una policy di dimensionamento automatico per un cluster di database Amazon Aurora
- Eliminazione di una policy di dimensionamento automatico dal cluster di database Amazon Aurora

Dimensionamento automatico di Amazon Aurora con le repliche Aurora

Per soddisfare i requisiti di connettività e carico di lavoro, il dimensionamento automatico Aurora modifica dinamicamente il numero di repliche Aurora (istanze database di lettura) assegnato a un cluster di database Aurora. Aurora Auto Scaling è disponibile per Aurora MySQL e Aurora PostgreSQL. Aurora Auto Scaling consente al cluster di database Aurora di gestire gli aumenti improvvisi di connettività o del carico di lavoro. Quando la connettività o il carico di lavoro diminuiscono, Aurora Auto Scaling rimuove le repliche Aurora inutili così da evitare di pagare per le istanze database non utilizzate.

Si definisce e si applica una politica di ridimensionamento a un cluster del database Aurora. La policy di dimensionamento definisce il numero minimo e massimo di repliche Aurora che Aurora Auto Scaling riesce a gestire. In base alla policy, Aurora Auto Scaling aumenta o diminuisce il numero di repliche Aurora in risposta ai carichi di lavoro effettivi, determinati utilizzando i parametri e i valori target di Amazon. CloudWatch

> **Note**
>
> Il servizio di dimensionamento automatico di Aurora non si applica al carico di lavoro dell'istanza database di scrittura. Il dimensionamento automatico di Aurora è utile solo per il carico di lavoro delle istanze di lettura.

Puoi utilizzare il Console di gestione AWS per applicare una politica di scalabilità basata su una metrica predefinita. In alternativa, puoi utilizzare l'API Aurora Auto Scaling per applicare una politica di scalabilità basata su una metrica predefinita AWS CLI o personalizzata.

Argomenti

- Prima di iniziare
- Policy di dimensionamento automatico Aurora
- Istanza IDs DB e tagging
- Dimensionamento automatico Aurora e Approfondimenti sulle prestazioni

Prima di iniziare

Prima di poter utilizzare il dimensionamento automatico Aurora con il cluster di database Aurora è necessario creare un cluster di database Aurora con un'istanza database (di scrittura) primaria.

Per ulteriori informazioni sulla creazione di un cluster di database Aurora, consulta Creazione di un cluster database Amazon Aurora.

Il dimensionamento automatico Aurora dimensiona un cluster di database solo se è nello stato disponibile.

Quando Aurora Auto Scaling aggiunge una nuova replica Aurora, la nuova replica Aurora è la stessa classe di istanze database di quella utilizzata dall'istanza primaria. Per altre informazioni sulle classi di istanza database, consulta Classi di istanze database Amazon Aurora. Inoltre, il livello di promozione per le nuove repliche Aurora è impostato sull'ultima priorità, che per impostazione predefinita è 15. Ciò significa che durante un failover una replica con una priorità più alta, ad esempio una creata manualmente, sarà promossa per prima. Per ulteriori informazioni, consulta Tolleranza ai guasti di un cluster DB Aurora.

Aurora Auto Scaling rimuove solo le repliche Aurora che ha creato.

Per sfruttare Aurora Auto Scaling, le applicazioni devono supportare le connessioni a nuove repliche Aurora. Per farlo, consigliamo di utilizzare l'endpoint di lettura Aurora. È possibile utilizzare un driver come il driver JDBC. AWS Per ulteriori informazioni, consulta Connessione a un cluster database Amazon Aurora.

> **Note**
>
> I database globali Aurora attualmente non supportano l'Auto Scaling Aurora per i cluster di database secondari.

Policy di dimensionamento automatico Aurora

Aurora Auto Scaling utilizza una policy di dimensionamento per regolare il numero di repliche Aurora in un cluster di database Aurora. Aurora Auto Scaling ha le seguenti componenti:

- Un ruolo collegato al servizio
- UN parametro target
- Capacità minima e massima
- Un periodo di attesa

Argomenti

- Ruolo legato al servizio
- Parametro target
- Capacità minima e massima
- Periodo di attesa
- Abilitazione o disabilitazione delle attività di riduzione
- Aggiunta, modifica o eliminazione di policy di dimensionamento automatico

Ruolo legato al servizio

Aurora Auto Scaling utilizza il ruolo collegato al servizio `AWSServiceRoleForApplicationAutoScaling_RDSCluster`. Per ulteriori informazioni, consulta Ruoli collegati ai servizi per Application Auto Scaling nella Guida per l'utente di Application Auto Scaling.

Parametro target

In questo tipo di policy, vengono specificati una metrica personalizzata o predefinita e un valore di destinazione per la metrica in una configurazione della policy di dimensionamento di monitoraggio delle destinazioni. Aurora Auto Scaling crea e CloudWatch gestisce allarmi che attivano la politica di scalabilità e calcola la regolazione della scalabilità in base alla metrica e al valore target. La policy di dimensionamento aggiunge o rimuove le repliche Aurora come richiesto per mantenere il parametro al valore di destinazione specificato o vicino a esso. Oltre a mantenere il parametro vicino al valore di destinazione, una policy di dimensionamento del monitoraggio di destinazione si adatta anche alle oscillazioni del parametro dovute a un carico di lavoro mutevole. Tale policy riduce anche le fluttuazioni rapide nel numero di repliche Aurora disponibili per il cluster di database.

Ad esempio, prendi una policy di dimensionamento che usa la metrica di utilizzo della CPU media predefinita. Tale policy può mantenere l'utilizzo della CPU a una percentuale specifica di utilizzo, come il 40 percento, o vicino ad essa.

> **Note**
>
> Per ogni cluster di database Aurora, è possibile creare solo una policy di Auto Scaling per ogni parametro di destinazione.

Quando configuri il servizio di dimensionamento automatico di Aurora, il valore della metrica di destinazione viene calcolato come media di tutte le istanze di lettura nel cluster. Il calcolo viene eseguito come descritto di seguito:

- Include tutte le istanze di lettura nel cluster Aurora, indipendentemente dal fatto che siano gestite tramite dimensionamento automatico o aggiunte manualmente.
- Include le istanze associate agli endpoint personalizzati. Gli endpoint personalizzati non influiscono sul calcolo delle metriche di destinazione.
- Non include l'istanza di scrittura del cluster.

Le metriche derivano dall'utilizzo delle seguenti dimensioni: CloudWatch

- `DBClusterIdentifier`
- `Role=READER`

Ad esempio, considera un cluster Aurora MySQL con la seguente configurazione:

- Istanze manuali (non controllate tramite dimensionamento automatico):
  - Istanze di scrittura con utilizzo della CPU al 50%
  - Istanza di lettura 1 (endpoint personalizzato: `custom-reader-1`) con utilizzo della CPU al 90%
  - Istanza di lettura 2 (endpoint personalizzato: `custom-reader-2`) con utilizzo della CPU al 90%
- Istanza con dimensionamento automatico:
  - Istanza di lettura 3 (aggiunta tramite dimensionamento automatico) con utilizzo della CPU al 10%

In questo scenario, la metrica di destinazione per la policy di dimensionamento automatico viene calcolata come segue:

```
Target metric = (CPU utilization of reader 1 + reader 2 + reader 3) / total number of
 readers

Target metric = (90 + 90 + 10) / 3 = 63.33%
```

La policy di dimensionamento automatico utilizza questo valore per valutare se ridurre o aumentare orizzontalmente le risorse in base alla soglia definita.

Considera i seguenti aspetti:

- Sebbene gli endpoint personalizzati determinino il modo in cui il traffico viene indirizzato a specifiche istanze di lettura, non escludono le istanze di lettura dal calcolo della metrica.
- Le istanze manuali sono sempre incluse nei calcoli delle metriche di destinazione.
- Per evitare comportamenti di dimensionamento imprevisti, assicurati che la configurazione del servizio di dimensionamento automatico tenga conto di tutte le istanze di lettura nel cluster.
- Se un cluster non ha istanze di lettura, la metrica non viene calcolata e gli allarmi della policy di dimensionamento automatico rimangono inattivi. Affinché la policy di dimensionamento automatico funzioni in modo efficace, deve essere sempre presente almeno un'istanza di lettura.

Capacità minima e massima

È possibile specificare il numero massimo di repliche Aurora che deve essere gestito da Application Auto Scaling. Questo valore deve essere impostato su 0–15 e deve essere uguale o maggiore rispetto al valore specificato per il numero minimo di repliche Aurora.

È anche possibile specificare il numero minimo di repliche Aurora che deve essere gestito da Application Auto Scaling. Questo valore deve essere impostato su 0–15 e deve essere uguale o minore rispetto al valore specificato per il numero massimo di repliche Aurora.

Affinché il dimensionamento automatico di Aurora funzioni, deve essere presente almeno un'istanza database di lettura. Se il cluster di database non ha istanze di lettura e imposti la capacità minima su 0, il servizio di dimensionamento automatico di Aurora non funziona.

> **Note**
>
> Sono impostate le capacità minima e massima per un cluster di database Aurora. I valori specificati vengono applicati a tutte le policy associate con quel cluster di database Aurora.

Periodo di attesa

È possibile sintonizzare i tempi di risposta di una policy di dimensionamento di monitoraggio della destinazione aggiungendo dei periodi di attesa che influiscano sul ridimensionamento del cluster di database Aurora in entrata e in uscita. Un periodo di attesa blocca le richieste di riduzione o aumento ulteriori finché il periodo non scade. Questi blocchi rallentano le eliminazioni delle repliche Aurora nel cluster di database Aurora per le richieste di riduzione e la creazione di repliche Aurora per le richieste di aumento.

Puoi specificare i seguenti periodi di attesa:

- Un'attività di riduzione riduce il numero di repliche Aurora nel cluster di database Aurora. Un periodo di attesa di riduzione specifica la quantità di tempo che deve passare, in secondi, tra il completamento di un'attività di riduzione e l'inizio di un'altra attività di questo tipo.
- Un'attività di aumento aumenta il numero di repliche Aurora nel cluster di database Aurora. Un periodo di attesa di aumento specifica la quantità di tempo che deve passare, in secondi, tra il completamento di un'attività di aumento e l'inizio di un'altra attività di questo tipo.

> **Note**
>
> Un tempo di raffreddamento di aumento orizzontale viene ignorato se una successiva richiesta di aumento orizzontale è relativa a un numero maggiore di repliche Aurora rispetto alla prima richiesta.

Se non si imposta un tempo di raffreddamento di riduzione orizzontale o aumento orizzontale, il valore predefinito per ciascuno è pari a 300 secondi.

Abilitazione o disabilitazione delle attività di riduzione

Puoi abilitare o disabilitare le attività di riduzione per una policy. Abilitare queste attività di riduzione consente alla policy di dimensionamento di eliminare le repliche Aurora. Quando le attività di riduzione sono abilitate, il periodo di attesa della riduzione nella policy di dimensionamento si applica alle attività di riduzione. Disabilitare le attività di riduzione evita alla policy di dimensionamento di eliminare le repliche Aurora.

> **Note**
>
> Le attività di aumento sono sempre abilitate, in modo che la policy di dimensionamento possa creare repliche Aurora in base alle esigenze.

Aggiunta, modifica o eliminazione di policy di dimensionamento automatico

È possibile aggiungere, modificare o eliminare le policy di auto scaling utilizzando o l'Console di gestione AWSAPI AWS CLI Application Auto Scaling. Per ulteriori informazioni su come aggiungere, modificare o eliminare le policy di dimensionamento automatico, consulta le sezioni successive.

- Aggiunta di una policy di dimensionamento automatico a un cluster di database Amazon Aurora
- Modifica di una policy di dimensionamento automatico per un cluster di database Amazon Aurora
- Eliminazione di una policy di dimensionamento automatico dal cluster di database Amazon Aurora

Istanza IDs DB e tagging

Quando una replica viene aggiunta da Aurora Auto Scaling, il relativo ID istanza database è preceduto da `application-autoscaling-`, ad esempio `application-autoscaling-61aabbcc-4e2f-4c65-b620-ab7421abc123`.

Il tag seguente viene aggiunto automaticamente all'istanza di database. È possibile visualizzarlo nella scheda Tag della pagina dei dettagli dell'istanza di database.

Tag	Valore
application-autoscaling:resourceId	cluster:mynewcluster-cluster

Per ulteriori informazioni sui tag delle risorse di Amazon RDS, consulta Applicazione di tag alle risorse Amazon Aurora e Amazon RDS.

Dimensionamento automatico Aurora e Approfondimenti sulle prestazioni

È possibile utilizzare Approfondimenti sulle prestazioni per monitorare le repliche aggiunte da Dimensionamento automatico Aurora, come con qualsiasi istanza database di lettura Aurora.

Per ulteriori informazioni sull'Approfondimenti sulle prestazioni per monitorare i cluster di database Aurora, consulta Monitoraggio del carico DB con Performance Insights su Amazon Aurora.

Aggiunta di una policy di dimensionamento automatico a un cluster di database Amazon Aurora

È possibile aggiungere una policy di dimensionamento con la Console di gestione AWS, AWS CLI o l'API Application Auto Scaling.

> **Note**
>
> Per un esempio che aggiunge una policy di dimensionamento utilizzando CloudFormation, consulta Dichiarazione di una policy di dimensionamento per un cluster di database Aurora nella Guida per l'utente di AWS CloudFormation.

Console

È possibile aggiungere una policy di dimensionamento a un cluster di database Aurora utilizzando la Console di gestione AWS.

Per aggiungere una policy di Auto Scaling a un cluster di database Aurora

1. Accedi alla Console di gestione AWS e apri la console Amazon RDS all'indirizzo https://console.aws.amazon.com/rds/.
2. Nel riquadro di navigazione, scegliere Databases (Database).
3. Scegliere il cluster di database Aurora a cui si desidera aggiungere una policy.
4. Scegliere la scheda Logs & events (Log ed eventi).
5. Nella sezione Auto scaling policies (Policy di Auto Scaling), scegliere Add (Aggiungi).

   Appare la finestra di dialogo Add Auto Scaling policy (Aggiungi policy di Auto Scaling).
6. Per Policy name (Nome policy), digitare il nome della policy.
7. Per il parametro di destinazione, scegliere in uno dei seguenti modi:

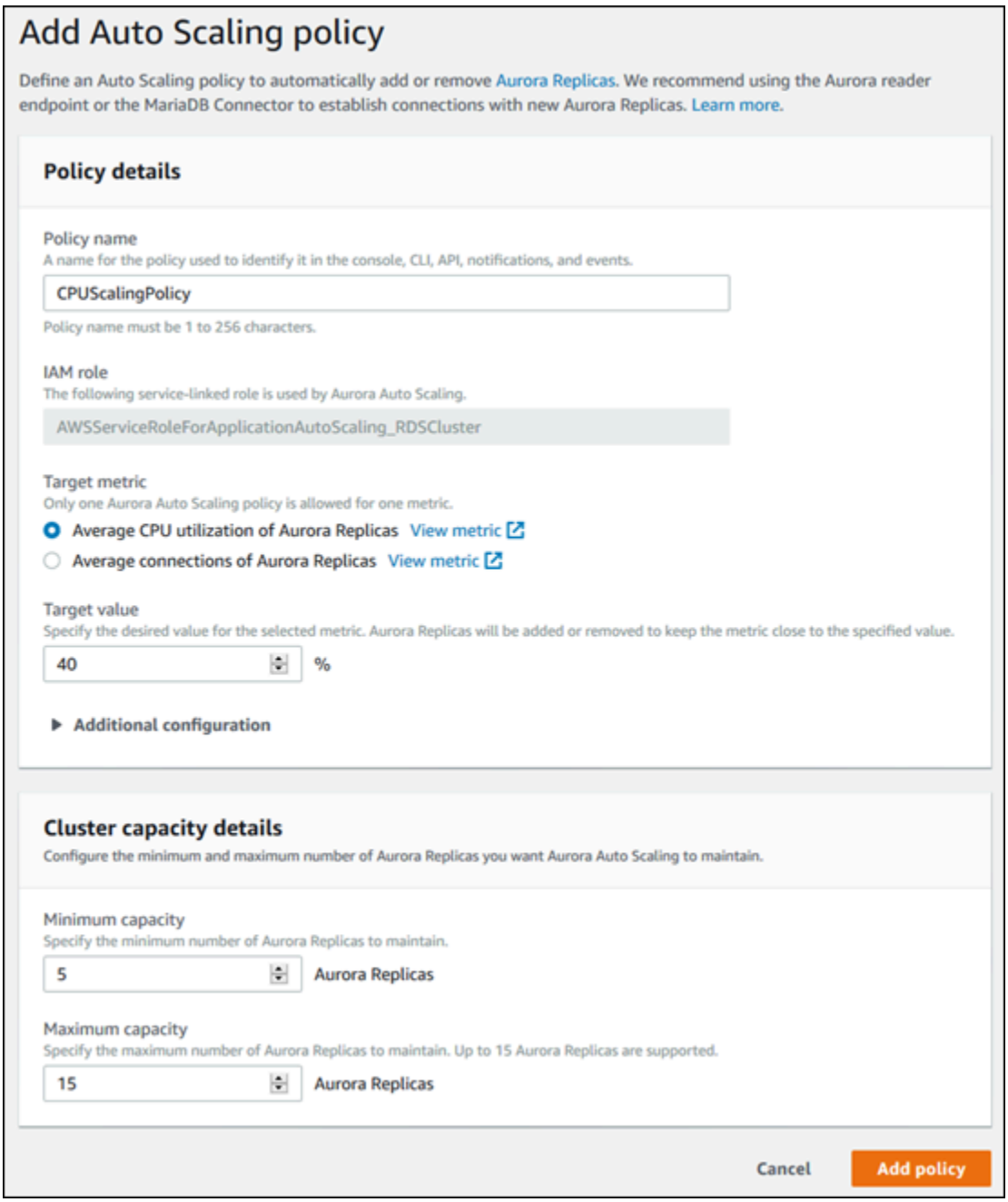
   - Average CPU utilization of Aurora Replicas (Utilizzo medio della CPU delle repliche Aurora) per creare una policy basata sull'utilizzo medio della CPU.
   - Average connections of Aurora Replicas (Connessioni medie delle repliche Aurora) per creare una policy basata sul numero medio di connessioni alle repliche Aurora.
8. Per il valore di destinazione, digitare uno dei seguenti modi:

   - Se si sceglie Average CPU utilization of Aurora Replicas (Utilizzo medio della CPU delle repliche Aurora), digitare la percentuale di utilizzo della CPU da mantenere sulle repliche Aurora.
   - Se si sceglie Average connections of Aurora Replicas (Connessioni medie delle repliche Aurora), digitare il numero di connessioni da mantenere.

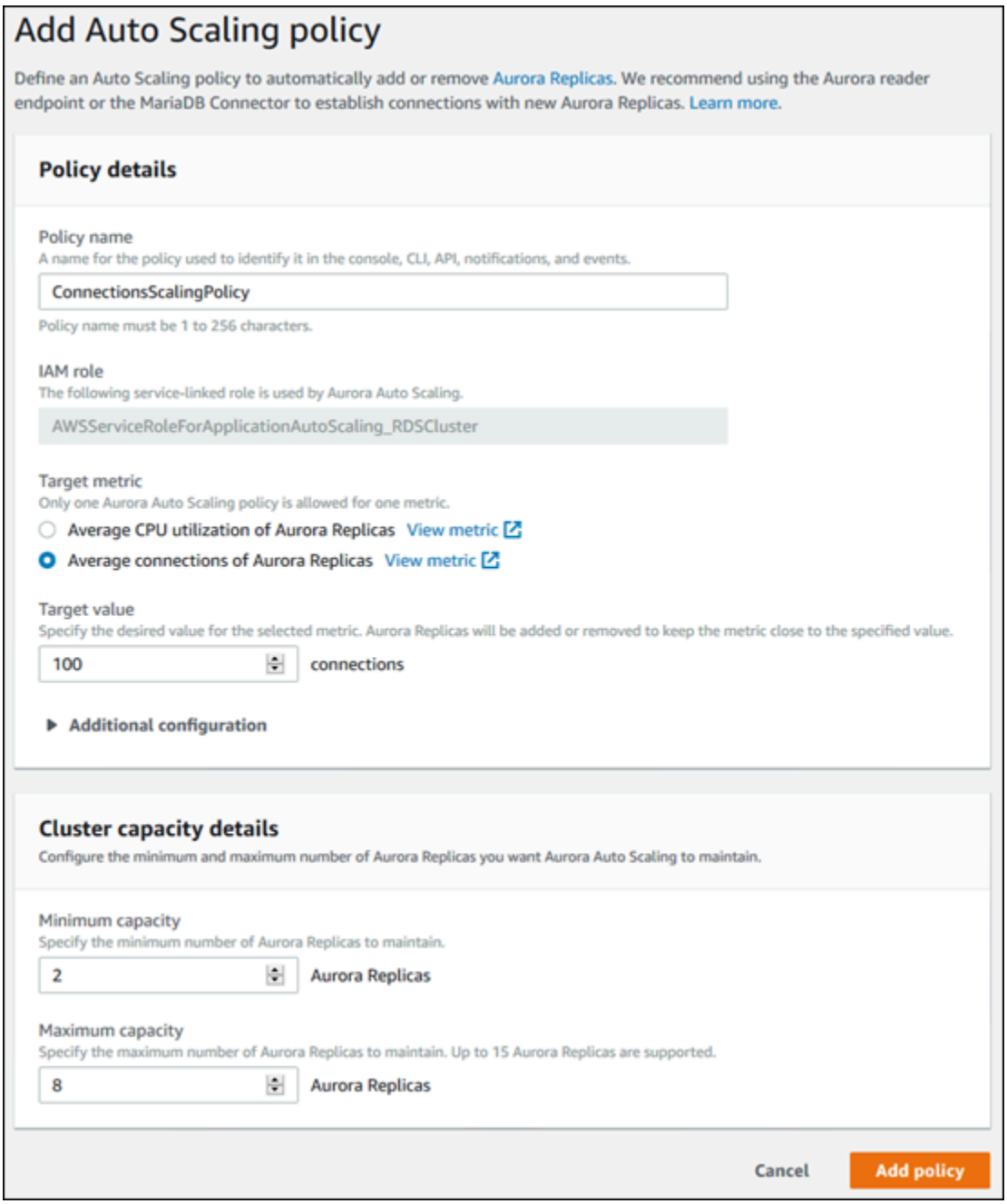
Le repliche Aurora vengono aggiunte o rimosse per tenere il parametro vicino al valore specificato.

9. (Facoltativo) Espandere Configurazione aggiuntiva per creare un tempo di raffreddamento di riduzione orizzontale o aumento orizzontale.
10. Per Minimum capacity (Capacità minima), digitare il numero minimo di repliche Aurora che la policy di Aurora Auto Scaling deve mantenere.
11. Per Maximum capacity (Capacità massima), digitare il numero massimo di repliche Aurora che la policy di Aurora Auto Scaling deve mantenere.
12. Scegliere Add policy (Aggiungi policy).

La seguente finestra di dialogo crea una politica di Auto Scaling basata su un utilizzo medio della CPU del 40 percento. La policy specifica un minimo di 5 repliche Aurora e un massimo di 15 repliche Aurora.



La seguente finestra di dialogo crea una policy di Auto Scaling basata su un numero di connessioni pari a 100. La policy specifica un minimo di due repliche Aurora e un massimo di otto repliche Aurora.



AWS CLI o API Application Auto Scaling

Puoi applicare una policy di dimensionamento basata un parametro di default o personalizzato. Per farlo, è possibile utilizzare AWS CLI o l'API Application Auto Scaling. La prima fase consiste nel registrare il cluster di database Aurora con Application Auto Scaling.

Registrazione di un cluster di database Aurora

Prima di poter utilizzare Aurora Auto Scaling con un cluster di database Aurora, registrare il cluster di database Aurora con Application Auto Scaling. Questa operazione consente di definire la dimensione e i limiti di dimensionamento da applicare al cluster. Application Auto Scaling ridimensiona dinamicamente il cluster di database Aurora lungo la dimensione scalabile `rds:cluster:ReadReplicaCount`, che rappresenta il numero di repliche Aurora.

Per registrare il cluster di database Aurora, si può utilizzare la AWS CLI o l'API Application Auto Scaling.

AWS CLI

Per registrare il cluster di database Aurora, utilizzare il comando della AWS CLI `register-scalable-target` con i parametri seguenti:

- `--service-namespace` – Impostare questo valore su `rds`.
- `--resource-id` – L'identificatore della risorsa per il cluster di database Aurora. Per questo parametro, il tipo di risorsa è `cluster` e l'identificatore univoco è il nome del cluster di database Aurora, ad esempio `cluster:myscalablecluster`.
- `--scalable-dimension` – Impostare questo valore su `rds:cluster:ReadReplicaCount`.
- `--min-capacity` – Il numero minimo di istanze database del lettore da gestire con Application Auto Scaling. Per informazioni sulla relazione tra `--min-capacity`, `--max-capacity` e il numero di istanze database nel cluster, consulta Capacità minima e massima.
- `--max-capacity` – Il numero massimo di istanze database del lettore da gestire con Application Auto Scaling. Per informazioni sulla relazione tra `--min-capacity`, `--max-capacity` e il numero di istanze database nel cluster, consulta Capacità minima e massima.

Example

Nell'esempio seguente viene registrato un cluster di database Aurora denominato `myscalablecluster`. La registrazione indica che il cluster di database deve essere dimensionato dinamicamente per avere da uno a otto repliche Aurora.

Per Linux, macOS o Unix:

```
aws application-autoscaling register-scalable-target \
    --service-namespace rds \
    --resource-id cluster:myscalablecluster \
```

```
    --scalable-dimension rds:cluster:ReadReplicaCount \
    --min-capacity 1 \
    --max-capacity 8 \
```

Per Windows:

```
aws application-autoscaling register-scalable-target ^
    --service-namespace rds ^
    --resource-id cluster:myscalablecluster ^
    --scalable-dimension rds:cluster:ReadReplicaCount ^
    --min-capacity 1 ^
    --max-capacity 8 ^
```

API Application Auto Scaling

Per registrare il cluster di database Aurora con Application Auto Scaling, utilizzare l'operazione API `RegisterScalableTarget` Application Auto Scaling con i parametri seguenti:

- `ServiceNamespace` – Impostare questo valore su `rds`.
- `ResourceID` – L'identificatore della risorsa per il cluster di database Aurora. Per questo parametro, il tipo di risorsa è `cluster` e l'identificatore univoco è il nome del cluster di database Aurora, ad esempio `cluster:myscalablecluster`.
- `ScalableDimension` – Impostare questo valore su `rds:cluster:ReadReplicaCount`.
- `MinCapacity` – Il numero minimo di istanze database del lettore da gestire con Application Auto Scaling. Per informazioni sulla relazione tra `MinCapacity`, `MaxCapacity` e il numero di istanze database nel cluster, consulta Capacità minima e massima.
- `MaxCapacity` – Il numero massimo di istanze database del lettore da gestire con Application Auto Scaling. Per informazioni sulla relazione tra `MinCapacity`, `MaxCapacity` e il numero di istanze database nel cluster, consulta Capacità minima e massima.

Example

Nell'esempio seguente viene registrato un cluster di database Aurora denominato `myscalablecluster` con l'API Application Auto Scaling. Questa registrazione indica che il cluster di database deve essere dimensionato dinamicamente per avere da uno a otto repliche Aurora.

```
POST / HTTP/1.1
```

```
Host: autoscaling.us-east-2.amazonaws.com
Accept-Encoding: identity
Content-Length: 219
X-Amz-Target: AnyScaleFrontendService.RegisterScalableTarget
X-Amz-Date: 20160506T182145Z
User-Agent: aws-cli/1.10.23 Python/2.7.11 Darwin/15.4.0 botocore/1.4.8
Content-Type: application/x-amz-json-1.1
Authorization: AUTHPARAMS

{
    "ServiceNamespace": "rds",
    "ResourceId": "cluster:myscalablecluster",
    "ScalableDimension": "rds:cluster:ReadReplicaCount",
    "MinCapacity": 1,
    "MaxCapacity": 8
}
```

Definizione di una politica di dimensionamento per un cluster di database Aurora

Una configurazione della policy di dimensionamento di monitoraggio degli obiettivi è rappresentata da un blocco JSON in cui sono definiti i valori dei parametri e della destinazione. Puoi salvare una configurazione della policy di dimensionamento come un blocco JSON in un file di testo. Questo file di testo viene utilizzato quando si chiama la AWS CLIo l'API Application Auto Scaling. Per ulteriori informazioni sulla sintassi della configurazione della policy, consulta `TargetTrackingScalingPolicyConfiguration` in Application Auto Scaling API Reference.

Le seguenti opzioni sono disponibili per definire una configurazione di una policy di dimensionamento di monitoraggio dei target.

Argomenti

- Utilizzo di un parametro predefinito
- Utilizzo di un parametro personalizzato
- Utilizzo di periodi di attesa
- Disabilitazione dell'attività di riduzione

Utilizzo di un parametro predefinito

Utilizzando dei parametri predefiniti, si può definire rapidamente una policy di dimensionamento per il monitoraggio della destinazione per un cluster di database Aurora che funziona bene sia con il monitoraggio della destinazione sia con il dimensionamento dinamico in Aurora Auto Scaling.

Attualmente, Aurora supporta i seguenti parametri predefiniti in Aurora Auto Scaling:

- RDSReaderAverageCPUUtilization – Il valore medio del parametro `CPUUtilization` in CloudWatch in tutte le repliche Aurora nel cluster di database Aurora.
- RDSReaderAverageDatabaseConnections – Il valore medio del parametro `DatabaseConnections` in CloudWatch in tutte le repliche Aurora nel cluster di database Aurora.

Per ulteriori informazioni sui parametri `CPUUtilization` e `DatabaseConnections`, consultare CloudWatch Parametri Amazon per Amazon Aurora.

Per utilizzare un parametro di default nella policy di dimensionamento, crea una configurazione di monitoraggio degli obiettivi per la policy di dimensionamento. La configurazione deve includere un `PredefinedMetricSpecification` per il parametro predefinito e un `TargetValue` per il valore di destinazione del parametro.

Example

L'esempio seguente descrive una tipica configurazione di policy per il dimensionamento di monitoraggio della destinazione per un cluster di database Aurora. In questa configurazione, il parametro predefinito `RDSReaderAverageCPUUtilization` viene utilizzato per regolare un cluster di database Aurora in base a un utilizzo medio della CPU del 40 per cento in tutte le repliche Aurora.

```
{
    "TargetValue": 40.0,
    "PredefinedMetricSpecification":
    {
        "PredefinedMetricType": "RDSReaderAverageCPUUtilization"
    }
}
```

Utilizzo di un parametro personalizzato

Utilizzando dei parametri personalizzati, è possibile definire una policy di dimensionamento di monitoraggio della destinazione che soddisfi i requisiti personalizzati. È possibile definire un parametro personalizzato in base a qualsiasi parametro Aurora che si modifichi in proporzione al dimensionamento.

Non tutti i parametri Aurora funzionano per il monitoraggio della destinazione. Il parametro deve essere un parametro di utilizzo valido e deve descrivere quanto è impegnata un'istanza. Il valore

del parametro deve aumentare o diminuire in proporzione al numero di repliche Aurora nel cluster di database Aurora. Questo aumento o riduzione proporzionale è necessario per utilizzare i dati del parametro per aumentare o diminuire in modo proporzionale il numero di repliche Aurora.

Example

Il seguente esempio descrive una configurazione di monitoraggio della destinazione per una policy di dimensionamento. In questa configurazione, un parametro personalizzato regola un cluster di database Aurora in base a un utilizzo medio della CPU del 50 per cento in tutte le repliche Aurora in un cluster di database Aurora denominato `my-db-cluster`.

```
{
    "TargetValue": 50,
    "CustomizedMetricSpecification":
    {
        "MetricName": "CPUUtilization",
        "Namespace": "AWS/RDS",
        "Dimensions": [
            {"Name": "DBClusterIdentifier","Value": "my-db-cluster"},
            {"Name": "Role","Value": "READER"}
        ],
        "Statistic": "Average",
        "Unit": "Percent"
    }
}
```

Utilizzo di periodi di attesa

È possibile specificare un valore, in secondi, per `ScaleOutCooldown` per aggiungere un periodo di attesa per l'aumento del cluster di database Aurora. Allo stesso modo, è possibile aggiungere un valore, in secondi, per `ScaleInCooldown` per aggiungere un periodo di attesa per la riduzione del cluster di database Aurora. Per ulteriori informazioni su `ScaleInCooldown` e `ScaleOutCooldown`, consulta `TargetTrackingScalingPolicyConfiguration` in Riferimento API Auto Scaling dell'applicazione.

Example

Il seguente esempio descrive una configurazione di monitoraggio della destinazione per una policy di dimensionamento. In questa configurazione, il parametro predefinito `RDSReaderAverageCPUUtilization` viene utilizzato per regolare un cluster di database Aurora; in base a un utilizzo medio della CPU del 40 per cento in tutte le repliche Aurora; in quel cluster

di database Aurora. La configurazione fornisce un periodo di attesa di riduzione di 10 minuti e un periodo di attesa di aumento di 5 minuti.

```
{
    "TargetValue": 40.0,
    "PredefinedMetricSpecification":
    {
        "PredefinedMetricType": "RDSReaderAverageCPUUtilization"
    },
    "ScaleInCooldown": 600,
    "ScaleOutCooldown": 300
}
```

Disabilitazione dell'attività di riduzione

Puoi prevenire la configurazione della policy di dimensionamento di monitoraggio della destinazione riducendo il cluster di database Aurora disabilitando l'attività di riduzione. La disabilitazione delle attività di riduzione impedisce alla policy di dimensionamento di eliminare le repliche Aurora, consentendo tuttavia alla policy di dimensionamento di crearle in base alle esigenze.

È possibile specificare un valore booleano per `DisableScaleIn` per abilitare o disabilitare l'attività di riduzione per il cluster di database Aurora. Per ulteriori informazioni su `DisableScaleIn`, consulta `TargetTrackingScalingPolicyConfiguration` in Riferimento API Auto Scaling dell'applicazione.

Example

Il seguente esempio descrive una configurazione di monitoraggio della destinazione per una policy di dimensionamento. In questa configurazione, il parametro predefinito `RDSReaderAverageCPUUtilization` regola un cluster di database Aurora in base a un utilizzo medio della CPU del 40 per cento in tutte le repliche Aurora in quel cluster di database Aurora. La configurazione disabilita l'attività di riduzione per la policy di dimensionamento.

```
{
    "TargetValue": 40.0,
    "PredefinedMetricSpecification":
    {
        "PredefinedMetricType": "RDSReaderAverageCPUUtilization"
    },
    "DisableScaleIn": true
}
```

Applicazione di una policy di dimensionamento a un cluster di database Aurora

Dopo la registrazione del cluster di database Aurora con Application Auto Scaling e la definizione di una policy di dimensionamento, applicare la policy di dimensionamento al cluster di database Aurora registrato. Per applicare una policy di dimensionamento a un cluster di database Aurora, è possibile utilizzare la AWS CLI oppure l'API Application Auto Scaling.

AWS CLI

Per applicare una policy di dimensionamento al cluster di database Aurora, utilizzare il comando della AWS CLI `put-scaling-policy` con i parametri seguenti:

- `--policy-name` – Il nome della policy di dimensionamento.
- `--policy-type` – Impostare questo valore su `TargetTrackingScaling`.
- `--resource-id` – L'identificatore della risorsa per il cluster di database Aurora. Per questo parametro, il tipo di risorsa è `cluster` e l'identificatore univoco è il nome del cluster di database Aurora, ad esempio `cluster:myscalablecluster`.
- `--service-namespace` – Impostare questo valore su `rds`.
- `--scalable-dimension` – Impostare questo valore su `rds:cluster:ReadReplicaCount`.
- `--target-tracking-scaling-policy-configuration` – La configurazione di una policy di dimensionamento per il monitoraggio della destinazione da utilizzare per il cluster di database Aurora.

Example

Nell'esempio seguente si applica una policy di dimensionamento per il monitoraggio della destinazione denominata `myscalablepolicy` a un cluster di database Aurora denominato `myscalablecluster` con Application Auto Scaling. Per fare ciò, usa la configurazione della policy salvata in un file denominato `config.json`.

Per Linux, macOS o Unix:

```
aws application-autoscaling put-scaling-policy \
    --policy-name myscalablepolicy \
    --policy-type TargetTrackingScaling \
    --resource-id cluster:myscalablecluster \
    --service-namespace rds \
    --scalable-dimension rds:cluster:ReadReplicaCount \
```

```
    --target-tracking-scaling-policy-configuration file://config.json
```

Per Windows:

```
aws application-autoscaling put-scaling-policy ^
    --policy-name myscalablepolicy ^
    --policy-type TargetTrackingScaling ^
    --resource-id cluster:myscalablecluster ^
    --service-namespace rds ^
    --scalable-dimension rds:cluster:ReadReplicaCount ^
    --target-tracking-scaling-policy-configuration file://config.json
```

API Application Auto Scaling

Per applicare una policy di dimensionamento al cluster di database Aurora con l'API Application Auto Scaling, utilizzare l'operazione API Application Auto Scaling `PutScalingPolicy` con i parametri seguenti:

- `PolicyName` – Il nome della policy di dimensionamento.
- `ServiceNamespace` – Impostare questo valore su `rds`.
- `ResourceID` – L'identificatore della risorsa per il cluster di database Aurora. Per questo parametro, il tipo di risorsa è `cluster` e l'identificatore univoco è il nome del cluster di database Aurora, ad esempio `cluster:myscalablecluster`.
- `ScalableDimension` – Impostare questo valore su `rds:cluster:ReadReplicaCount`.
- `PolicyType` – Impostare questo valore su `TargetTrackingScaling`.
- `TargetTrackingScalingPolicyConfiguration` – La configurazione di una policy di dimensionamento per il monitoraggio della destinazione da utilizzare per il cluster di database Aurora.

Example

Nell'esempio seguente si applica una policy di dimensionamento per il monitoraggio della destinazione denominata `myscalablepolicy` a un cluster di database Aurora denominato `myscalablecluster` con Application Auto Scaling. Si utilizza una policy di configurazione in base al parametro predefinito `RDSReaderAverageCPUUtilization`.

```
POST / HTTP/1.1
Host: autoscaling.us-east-2.amazonaws.com
Accept-Encoding: identity
Content-Length: 219
X-Amz-Target: AnyScaleFrontendService.PutScalingPolicy
X-Amz-Date: 20160506T182145Z
User-Agent: aws-cli/1.10.23 Python/2.7.11 Darwin/15.4.0 botocore/1.4.8
Content-Type: application/x-amz-json-1.1
Authorization: AUTHPARAMS

{
    "PolicyName": "myscalablepolicy",
    "ServiceNamespace": "rds",
    "ResourceId": "cluster:myscalablecluster",
    "ScalableDimension": "rds:cluster:ReadReplicaCount",
    "PolicyType": "TargetTrackingScaling",
    "TargetTrackingScalingPolicyConfiguration": {
        "TargetValue": 40.0,
        "PredefinedMetricSpecification":
        {
            "PredefinedMetricType": "RDSReaderAverageCPUUtilization"
        }
    }
}
```

Modifica di una policy di dimensionamento automatico per un cluster di database Amazon Aurora

È possibile modificare una policy di dimensionamento con la Console di gestione AWS, AWS CLI o l'API di Application Auto Scaling.

Console

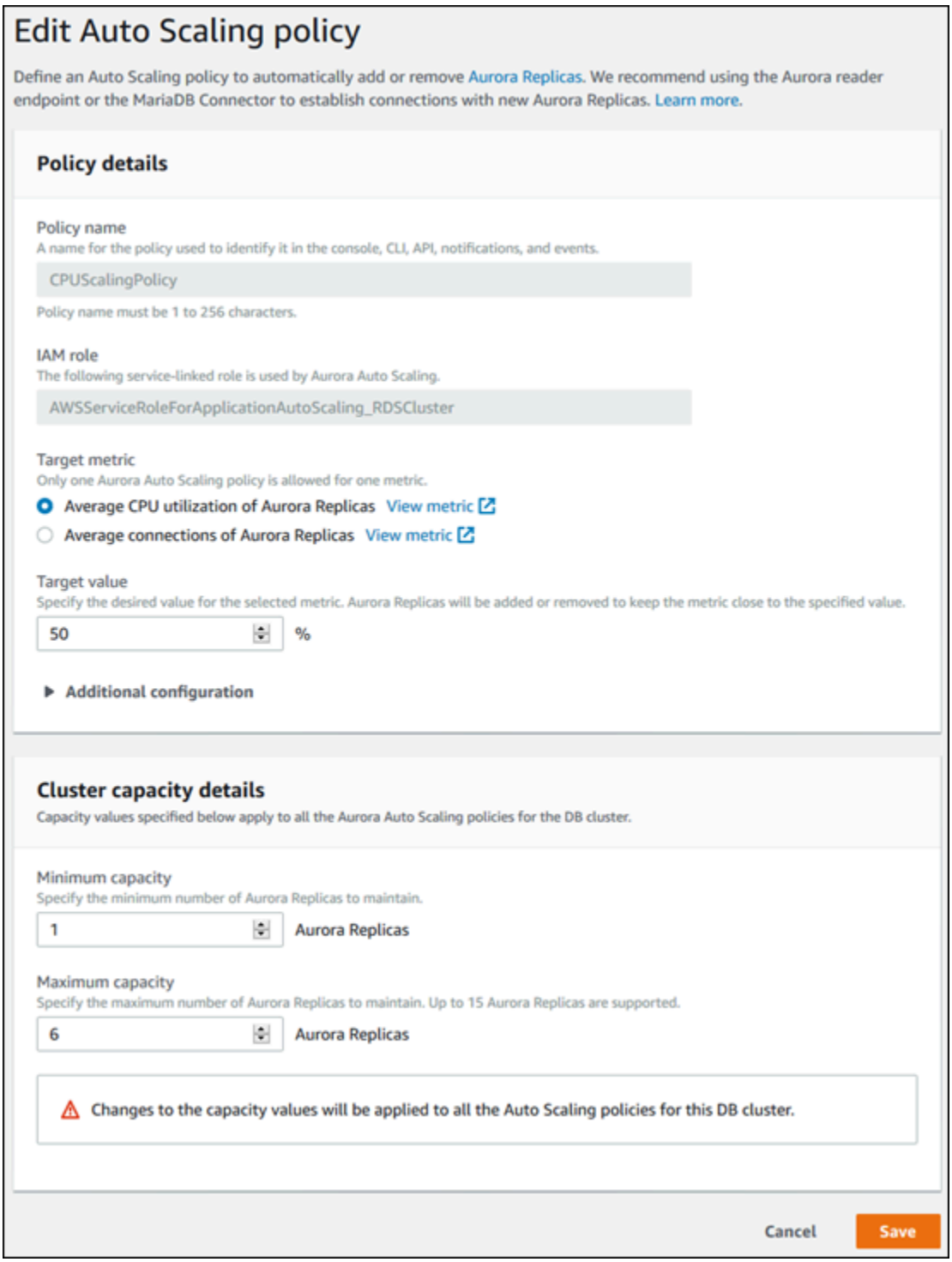
Puoi modificare una policy di dimensionamento usando la Console di gestione AWS.

Per modificare una policy di Auto Scaling per un cluster di database Aurora

1. Accedi alla Console di gestione AWS e apri la console Amazon RDS all'indirizzo https://console.aws.amazon.com/rds/.
2. Nel riquadro di navigazione, scegliere Databases (Database).
3. Scegliere il cluster di database Aurora di cui si desidera modificare la policy di Auto Scaling.

4. Scegliere la scheda Logs & events (Log ed eventi).
5. Nella sezione Auto Scaling Policies (Policy di Auto Scaling), scegliere la policy di Auto Scaling e successivamente Edit (Modifica).
6. Apportare modifiche alla policy.
7. Scegliere Save (Salva).

Quella che segue è una finestra di dialogo di esempio Edit Scaling policy (Modifica policy di dimensionamento).



AWS CLI o API Application Auto Scaling

È possibile utilizzare la AWS CLI o l'API Application Auto Scaling per modificare una policy di dimensionamento nello stesso modo con cui si applica una policy di dimensionamento:

- Quando si utilizza l'AWS CLI, specificare il nome della policy da modificare nel parametro `--policy-name`. Specifica i nuovi valori per i parametri che desideri modificare.
- Quando si utilizza l'API Application Auto Scaling, specificare il nome della policy da modificare nel parametro `PolicyName`. Specifica i nuovi valori per i parametri che desideri modificare.

Per ulteriori informazioni, consulta Applicazione di una policy di dimensionamento a un cluster di database Aurora.

Eliminazione di una policy di dimensionamento automatico dal cluster di database Amazon Aurora

È possibile eliminare una policy di dimensionamento con la Console di gestione AWS, AWS CLI o l'API di Application Auto Scaling.

Console

Puoi eliminare una policy di dimensionamento usando la Console di gestione AWS.

Per eliminare una policy di Auto Scaling per un cluster di database Aurora

1. Accedi alla Console di gestione AWS e apri la console Amazon RDS all'indirizzo https://console.aws.amazon.com/rds/.
2. Nel riquadro di navigazione, scegliere Databases (Database).
3. Scegliere il cluster di database Aurora con la policy di Auto Scaling che si desidera eliminare.
4. Scegliere la scheda Logs & events (Log ed eventi).
5. Nella sezione Auto Scaling Policies (Policy di Auto Scaling), scegliere la policy di Auto Scaling e successivamente Delete (Elimina).

AWS CLI

Per eliminare una policy di dimensionamento dal cluster di database Aurora, utilizzare il comando della AWS CLI `delete-scaling-policy` con i parametri seguenti:

- `--policy-name` – Il nome della policy di dimensionamento.
- `--resource-id` – L'identificatore della risorsa per il cluster di database Aurora. Per questo parametro, il tipo di risorsa è `cluster` e l'identificatore univoco è il nome del cluster di database Aurora, ad esempio `cluster:myscalablecluster`.

- `--service-namespace` – Impostare questo valore su `rds`.
- `--scalable-dimension` – Impostare questo valore su `rds:cluster:ReadReplicaCount`.

Example

Nell'esempio seguente, eliminare una policy di dimensionamento per il monitoraggio della destinazione denominata `myscalablepolicy` da un cluster di database Aurora denominato `myscalablecluster`.

Per Linux, macOS o Unix:

```
aws application-autoscaling delete-scaling-policy \
    --policy-name myscalablepolicy \
    --resource-id cluster:myscalablecluster \
    --service-namespace rds \
    --scalable-dimension rds:cluster:ReadReplicaCount \
```

Per Windows:

```
aws application-autoscaling delete-scaling-policy ^
    --policy-name myscalablepolicy ^
    --resource-id cluster:myscalablecluster ^
    --service-namespace rds ^
    --scalable-dimension rds:cluster:ReadReplicaCount ^
```

API Application Auto Scaling

Per eliminare una policy di dimensionamento dal cluster di database Aurora, utilizzare l'operazione API Application Auto Scaling `DeleteScalingPolicy` con i parametri seguenti:

- `PolicyName` – Il nome della policy di dimensionamento.
- `ServiceNamespace` – Impostare questo valore su `rds`.
- `ResourceID` – L'identificatore della risorsa per il cluster di database Aurora. Per questo parametro, il tipo di risorsa è `cluster` e l'identificatore univoco è il nome del cluster di database Aurora, ad esempio `cluster:myscalablecluster`.

- `ScalableDimension` – Impostare questo valore su `rds:cluster:ReadReplicaCount`.

Example

Nell'esempio seguente, eliminare una policy di dimensionamento per il monitoraggio della destinazione denominata `myscalablepolicy` da un cluster di database Aurora denominato `myscalablecluster` con l'API Application Auto Scaling.

```
POST / HTTP/1.1
Host: autoscaling.us-east-2.amazonaws.com
Accept-Encoding: identity
Content-Length: 219
X-Amz-Target: AnyScaleFrontendService.DeleteScalingPolicy
X-Amz-Date: 20160506T182145Z
User-Agent: aws-cli/1.10.23 Python/2.7.11 Darwin/15.4.0 botocore/1.4.8
Content-Type: application/x-amz-json-1.1
Authorization: AUTHPARAMS

{
    "PolicyName": "myscalablepolicy",
    "ServiceNamespace": "rds",
    "ResourceId": "cluster:myscalablecluster",
    "ScalableDimension": "rds:cluster:ReadReplicaCount"
}
```

Gestione delle prestazioni e del dimensionamento dei cluster DB Aurora

Puoi utilizzare le seguenti opzioni per gestire le prestazioni e il dimensionamento dei cluster DB e le istanze database Aurora:

Argomenti

- Dimensionamento dello storage
- Dimensionamento delle istanze
- Dimensionamento della lettura
- Gestione delle connessioni
- Gestione dei piani di esecuzione delle query

Dimensionamento dello storage

Lo storage Aurora viene automaticamente dimensionato con i dati del volume dei cluster. Man mano che i dati aumentano, lo spazio di archiviazione del volume dei cluster si espande a seconda della versione del motore di database. Per informazioni sulle dimensioni massime dei volumi dei cluster Aurora per ogni versione del motore, consulta Limiti di dimensione Amazon Aurora. Per informazioni sui tipi di dati inclusi nel volume del cluster, consulta Archiviazione Amazon Aurora. Per informazioni sulla dimensione massima per una versione specifica, consulta Limiti di dimensione Amazon Aurora.

Le dimensioni del volume del cluster vengono valutate ogni ora, per stabilire i costi di storage. Per informazioni sui prezzi, consulta la pagina dei prezzi di Aurora.

Anche se un volume del cluster Aurora può aumentare le dimensioni fino a molti tebibyte, solo lo spazio utilizzato nel volume viene addebitato. Il meccanismo per determinare lo spazio di storage fatturato dipende dalla versione del cluster Aurora.

- Quando i dati Aurora vengono rimossi dal volume del cluster, lo spazio fatturato complessivo diminuisce di una quantità comparabile. Questo comportamento di ridimensionamento dinamico si verifica quando i tablespace sottostanti vengono eliminati o riorganizzati per richiedere meno spazio. Pertanto, puoi ridurre le spese di archiviazione eliminando le tabelle e i database che non sono più necessari. Il ridimensionamento dinamico si applica a determinate versioni di Aurora. Di seguito sono riportate le versioni di Aurora in cui il volume del cluster viene ridimensionato dinamicamente durante la rimozione dei dati:

Motore del database	Versioni con ridimensionamento dinamico
Aurora MySQL	• Versione 3 (compatibile con MySQL 8.0): tutte le versioni supportate • Versione 2 (compatibile con MySQL 5.7): 2.11 e versioni successive
Aurora PostgreSQL	Tutte le versioni supportate
Aurora Serverless v2	Tutte le versioni supportate
Aurora Serverless v1	Non disponibile

- Nelle versioni di Aurora precedenti a quelle dell'elenco, il volume del cluster può riutilizzare lo spazio liberato dalla rimozione dei dati, ma le dimensioni del volume in sé non diminuiscono mai.

Il ridimensionamento dinamico si applica alle operazioni che rimuovono o ridimensionano fisicamente i tablespace all'interno del volume del cluster. Pertanto, si applica alle istruzioni SQL come `DROP TABLE`, `DROP DATABASE`, `TRUNCATE TABLE` e `ALTER TABLE ... DROP PARTITION`. Non si applica all'eliminazione di righe con l'istruzione `DELETE`. Se elimini un numero elevato di righe da una tabella, puoi eseguire l'istruzione Aurora MySQL `OPTIMIZE TABLE` o utilizzare l'estensione Aurora PostgreSQL `pg_repack` in seguito per riorganizzare la tabella e ridimensionare dinamicamente il volume del cluster.

Per Aurora MySQL, si applicano le seguenti considerazioni:

- Dopo aver aggiornato il cluster di database a una versione del motore di database che supporta il ridimensionamento dinamico e quando la funzionalità è abilitata nella specifica Regione AWS, lo spazio che viene successivamente liberato tramite determinate istruzioni SQL, ad esempio `DROP TABLE`, è recuperabile.

  Se la funzionalità è esplicitamente disabilitata in una determinata Regione AWS, lo spazio potrebbe essere solo riutilizzabile, e non recuperabile, anche nelle versioni che supportano il ridimensionamento dinamico.

  La funzionalità è stata abilitata per versioni specifiche del motore di database (da 1.23.0 a 1.23.4, da 2.09.0 a 2.09.3 e 2.10.0) tra novembre 2020 e marzo 2022, ed è abilitata per impostazione predefinita su tutte le versioni successive.

- Una tabella viene archiviata internamente in uno o più frammenti contigui di varie dimensioni. Durante l'esecuzione delle operazioni `TRUNCATE TABLE`, lo spazio corrispondente al primo frammento è riutilizzabile e non recuperabile. Gli altri frammenti sono recuperabili. Durante le operazioni `DROP TABLE`, lo spazio corrispondente all'intero tablespace è recuperabile.
- Il parametro `innodb_file_per_table` influisce sul modo in cui è organizzata l'archiviazione della tabella. Quando le tabelle fanno parte del tablespace di sistema, l'eliminazione della tabella non riduce le dimensioni del tablespace di sistema. Pertanto, è necessario accertarsi di impostare `innodb_file_per_table` su 1 per i cluster database Aurora MySQL per sfruttare al massimo il ridimensionamento dinamico.
- Nelle versioni 2.11 e successive, il tablespace temporaneo InnoDB viene eliminato e creato nuovamente al riavvio. In questo modo lo spazio occupato dal tablespace temporaneo viene rilasciato al sistema e il volume del cluster viene ridimensionato. Per utilizzare al meglio la funzionalità di ridimensionamento dinamico, è consigliabile aggiornare il cluster di database alle versioni 2.11 o successiva.

> **Note**
>
> La funzionalità di ridimensionamento dinamico non recupera immediatamente lo spazio quando le tabelle nei tablespace vengono eliminate, ma esegue tale operazione gradualmente, a un ritmo di circa 10 TB al giorno. Lo spazio del tablespace di sistema non viene recuperato perché il tablespace di sistema non viene mai eliminato. Lo spazio libero non reclamato in uno spazio tabella viene riutilizzato quando un'operazione richiede spazio in tale spazio tabella. La funzionalità di ridimensionamento dinamico può recuperare spazio di archiviazione solo quando il cluster è in uno stato disponibile.

Puoi verificare la quantità di spazio di storage utilizzata da un cluster monitorando il parametro `VolumeBytesUsed` in CloudWatch. Per ulteriori informazioni sulla fatturazione dello storage, consulta Come viene fatturato lo storage dei dati Aurora.

- Nella Console di gestione AWS puoi vedere questa cifra in un grafico visualizzando la scheda `Monitoring` nella pagina dei dettagli del cluster.
- Con l'AWS CLI, puoi eseguire un comando simile al seguente esempio di Linux. Sostituisci i tuoi valori per l'ora di inizio e di fine e il nome del cluster.

```
aws cloudwatch get-metric-statistics --metric-name "VolumeBytesUsed" \
```

```
  --start-time "$(date -d '6 hours ago')" --end-time "$(date -d 'now')" --period 60 \
  --namespace "AWS/RDS" \
  --statistics Average Maximum Minimum \
  --dimensions Name=DBClusterIdentifier,Value=my_cluster_identifier
```

Questo comando genera un output simile al seguente.

```
{
    "Label": "VolumeBytesUsed",
    "Datapoints": [
        {
            "Timestamp": "2020-08-04T21:25:00+00:00",
            "Average": 182871982080.0,
            "Minimum": 182871982080.0,
            "Maximum": 182871982080.0,
            "Unit": "Bytes"
        }
    ]
}
```

Gli esempi seguenti mostrano come è possibile tenere traccia dell'utilizzo dello storage per un cluster Aurora nel tempo utilizzando i comandi della AWS CLI su un sistema Linux. I parametri `--start-time` e `--end-time` definiscono l'intervallo di tempo complessivo come un giorno. Il parametro `--period` richiede le misurazioni a intervalli di un'ora. Non ha senso scegliere un valore `--period` piccolo perché i parametri vengono raccolti a intervalli, non in modo continuo. Inoltre, le operazioni di storage Aurora a volte continuano per qualche tempo in background al termine dell'istruzione SQL pertinente.

Il primo esempio restituisce l'output nel formato JSON predefinito. I punti dati vengono restituiti in ordine arbitrario, non in base al timestamp. Puoi importare questi dati JSON in uno strumento di grafici per eseguire l'ordinamento e la visualizzazione.

```
$ aws cloudwatch get-metric-statistics --metric-name "VolumeBytesUsed" \
  --start-time "$(date -d '1 day ago')" --end-time "$(date -d 'now')" --period 3600
  --namespace "AWS/RDS" --statistics Maximum --dimensions
 Name=DBClusterIdentifier,Value=my_cluster_id
{
    "Label": "VolumeBytesUsed",
    "Datapoints": [
        {
```

```
            "Timestamp": "2020-08-04T19:40:00+00:00",
            "Maximum": 182872522752.0,
            "Unit": "Bytes"
        },
        {
            "Timestamp": "2020-08-05T00:40:00+00:00",
            "Maximum": 198573719552.0,
            "Unit": "Bytes"
        },
        {
            "Timestamp": "2020-08-05T05:40:00+00:00",
            "Maximum": 206827454464.0,
            "Unit": "Bytes"
        },
        {
            "Timestamp": "2020-08-04T17:40:00+00:00",
            "Maximum": 182872522752.0,
            "Unit": "Bytes"
        },
... output omitted ...
```

In questo esempio vengono restituiti gli stessi dati del precedente. Il parametro `--output` rappresenta i dati in formato testo normale compatto. Il comando `aws cloudwatch` invia il suo output al comando `sort`. Il parametro `-k` del comando `sort` ordina l'output in base al terzo campo che è il timestamp in formato UTC (Universal Coordinated Time).

```
$ aws cloudwatch get-metric-statistics --metric-name "VolumeBytesUsed" \
  --start-time "$(date -d '1 day ago')" --end-time "$(date -d 'now')" --period 3600 \
  --namespace "AWS/RDS" --statistics Maximum --dimensions
 Name=DBClusterIdentifier,Value=my_cluster_id \
  --output text | sort -k 3
VolumeBytesUsed
DATAPOINTS  182872522752.0  2020-08-04T17:41:00+00:00 Bytes
DATAPOINTS  182872522752.0  2020-08-04T18:41:00+00:00 Bytes
DATAPOINTS  182872522752.0  2020-08-04T19:41:00+00:00 Bytes
DATAPOINTS  182872522752.0  2020-08-04T20:41:00+00:00 Bytes
DATAPOINTS  187667791872.0  2020-08-04T21:41:00+00:00 Bytes
DATAPOINTS  190981029888.0  2020-08-04T22:41:00+00:00 Bytes
DATAPOINTS  195587244032.0  2020-08-04T23:41:00+00:00 Bytes
DATAPOINTS  201048915968.0  2020-08-05T00:41:00+00:00 Bytes
DATAPOINTS  205368492032.0  2020-08-05T01:41:00+00:00 Bytes
DATAPOINTS  206827454464.0  2020-08-05T02:41:00+00:00 Bytes
DATAPOINTS  206827454464.0  2020-08-05T03:41:00+00:00 Bytes
```

```
DATAPOINTS  206827454464.0  2020-08-05T04:41:00+00:00 Bytes
DATAPOINTS  206827454464.0  2020-08-05T05:41:00+00:00 Bytes
DATAPOINTS  206827454464.0  2020-08-05T06:41:00+00:00 Bytes
DATAPOINTS  206827454464.0  2020-08-05T07:41:00+00:00 Bytes
DATAPOINTS  206827454464.0  2020-08-05T08:41:00+00:00 Bytes
DATAPOINTS  206827454464.0  2020-08-05T09:41:00+00:00 Bytes
DATAPOINTS  206827454464.0  2020-08-05T10:41:00+00:00 Bytes
DATAPOINTS  206827454464.0  2020-08-05T11:41:00+00:00 Bytes
DATAPOINTS  206827454464.0  2020-08-05T12:41:00+00:00 Bytes
DATAPOINTS  206827454464.0  2020-08-05T13:41:00+00:00 Bytes
DATAPOINTS  206827454464.0  2020-08-05T14:41:00+00:00 Bytes
DATAPOINTS  206833664000.0  2020-08-05T15:41:00+00:00 Bytes
DATAPOINTS  206833664000.0  2020-08-05T16:41:00+00:00 Bytes
```

L'output ordinato mostra quanto spazio di storage è stato utilizzato all'inizio e alla fine del periodo di monitoraggio. Puoi inoltre trovare i punti durante quel periodo quando Aurora ha allocato più spazio di storage per il cluster. Nell'esempio seguente vengono utilizzati i comandi Linux per riformattare i valori `VolumeBytesUsed` iniziali e finali come gigabyte (GB) e gibibyte (GiB). I gigabyte rappresentano unità misurate in potenze di 10 e sono comunemente utilizzati nelle discussioni di storage per dischi rigidi rotazionali. I gibibyte rappresentano le unità misurate in potenze di 2. Le misurazioni e i limiti di archiviazione di Aurora sono generalmente indicati nelle unità di potenza di 2, come gibibyte e tebibyte.

```
$ GiB=$((1024*1024*1024))
$ GB=$((1000*1000*1000))
$ echo "Start: $((182872522752/$GiB)) GiB, End: $((206833664000/$GiB)) GiB"
Start: 170 GiB, End: 192 GiB
$ echo "Start: $((182872522752/$GB)) GB, End: $((206833664000/$GB)) GB"
Start: 182 GB, End: 206 GB
```

Il parametro `VolumeBytesUsed` indica la quantità di spazio di storage nel cluster che sta causando costi. Quindi, è meglio ridurre al minimo questo numero quando possibile. Tuttavia, questo parametro non include alcune risorse di storage che Aurora utilizza internamente nel cluster senza addebitare alcun costo. Se il cluster si sta avvicinando al limite di storage e potrebbe esaurire lo spazio, è più utile monitorare il parametro `AuroraVolumeBytesLeftTotal` e cercare di massimizzare tale numero. Nell'esempio seguente viene eseguito un calcolo simile a quello precedente, ma per `AuroraVolumeBytesLeftTotal` invece di `VolumeBytesUsed`.

```
$ aws cloudwatch get-metric-statistics --metric-name "AuroraVolumeBytesLeftTotal" \
  --start-time "$(date -d '1 hour ago')" --end-time "$(date -d 'now')" --period 3600 \
```

```
  --namespace "AWS/RDS" --statistics Maximum --dimensions
 Name=DBClusterIdentifier,Value=my_old_cluster_id \
  --output text | sort -k 3
AuroraVolumeBytesLeftTotal
DATAPOINTS      140530528288768.0       2023-02-23T19:25:00+00:00       Count
$ TiB=$((1024*1024*1024*1024))
$ TB=$((1000*1000*1000*1000))
$ echo "$((69797067915264 / $TB)) TB remaining for this cluster"
69 TB remaining for this cluster
$ echo "$((69797067915264 / $TiB)) TiB remaining for this cluster"
63 TiB remaining for this cluster
```

Per un cluster che esegue Aurora MySQL versione 2.09 o successive oppure Aurora PostgreSQL, la dimensione libera riportata da `VolumeBytesUsed` aumenta quando i dati vengono aggiunti e diminuisce quando i dati vengono rimossi. L'esempio seguente mostra come. Questo report mostra le dimensioni massime e minime di storage per un cluster a intervalli di 15 minuti man mano che vengono create ed eliminate le tabelle con dati temporanei. Il report elenca il valore massimo prima del valore minimo. Pertanto, per capire come l'utilizzo dello storage è cambiato nell'intervallo di 15 minuti, interpreta i numeri da destra a sinistra.

```
$ aws cloudwatch get-metric-statistics --metric-name "VolumeBytesUsed" \
  --start-time "$(date -d '4 hours ago')" --end-time "$(date -d 'now')" --period 1800 \
  --namespace "AWS/RDS" --statistics Maximum Minimum --dimensions
 Name=DBClusterIdentifier,Value=my_new_cluster_id
  --output text | sort -k 4
VolumeBytesUsed
DATAPOINTS 14545305600.0 14545305600.0 2020-08-05T20:49:00+00:00 Bytes
DATAPOINTS 14545305600.0 14545305600.0 2020-08-05T21:19:00+00:00 Bytes
DATAPOINTS 22022176768.0 14545305600.0 2020-08-05T21:49:00+00:00 Bytes
DATAPOINTS 22022176768.0 22022176768.0 2020-08-05T22:19:00+00:00 Bytes
DATAPOINTS 22022176768.0 22022176768.0 2020-08-05T22:49:00+00:00 Bytes
DATAPOINTS 22022176768.0 15614263296.0 2020-08-05T23:19:00+00:00 Bytes
DATAPOINTS 15614263296.0 15614263296.0 2020-08-05T23:49:00+00:00 Bytes
DATAPOINTS 15614263296.0 15614263296.0 2020-08-06T00:19:00+00:00 Bytes
```

Nell'esempio seguente viene illustrato come, per un cluster che esegue versioni compatibili di Aurora MySQL o Aurora PostgreSQL, le dimensioni libere riportate da `AuroraVolumeBytesLeftTotal` riflettono il limite massimo di 256 TiB. Per ulteriori informazioni sulle versioni compatibili, consulta Limiti di dimensione Amazon Aurora.

```
$ aws cloudwatch get-metric-statistics --region us-east-1 --metric-name
 "AuroraVolumeBytesLeftTotal" \
  --start-time "$(date -d '4 hours ago')" --end-time "$(date -d 'now')" --period 1800 \
  --namespace "AWS/RDS" --statistics Minimum --dimensions
 Name=DBClusterIdentifier,Value=pq-57 \
  --output text | sort -k 3
AuroraVolumeBytesLeftTotal
DATAPOINTS 140515818864640.0 2020-08-05T20:56:00+00:00 Count
DATAPOINTS 140515818864640.0 2020-08-05T21:26:00+00:00 Count
DATAPOINTS 140515818864640.0 2020-08-05T21:56:00+00:00 Count
DATAPOINTS 140514866757632.0 2020-08-05T22:26:00+00:00 Count
DATAPOINTS 140511020580864.0 2020-08-05T22:56:00+00:00 Count
DATAPOINTS 140503168843776.0 2020-08-05T23:26:00+00:00 Count
DATAPOINTS 140503168843776.0 2020-08-05T23:56:00+00:00 Count
DATAPOINTS 140515818864640.0 2020-08-06T00:26:00+00:00 Count
$ TiB=$((1024*1024*1024*1024))
$ TB=$((1000*1000*1000*1000))
$ echo "$((140515818864640 / $TB)) TB remaining for this cluster"
140 TB remaining for this cluster
$ echo "$((140515818864640 / $TiB)) TiB remaining for this cluster"
256 TiB remaining for this cluster
```

Dimensionamento delle istanze

Puoi dimensionare il cluster database Aurora in base alle necessità, modificando la classe di istanza database per ogni istanza database nel cluster database. Aurora supporta diverse classi di istanze database ottimizzate per Aurora, a seconda della compatibilità del motore di database.

Motore di database	Dimensionamento delle istanze
Amazon Aurora MySQL	Per informazioni, consulta Dimensionamento delle istanze database Aurora MySQL
Amazon Aurora PostgreSQL	Per informazioni, consulta Dimensionamento delle istanze database Aurora PostgreSQL

Dimensionamento della lettura

Puoi ottenere il dimensionamento della lettura per il cluster di database Aurora creando fino a 15 repliche Aurora nel cluster di database. Ogni replica Aurora restituisce gli stessi dati dal volume

del cluster con ritardo di replica minimo—, in genere di molto inferiore a 100 millisecondi dopo che l'istanza primaria ha scritto un aggiornamento. Man mano che il traffico di lettura aumenta, puoi creare ulteriori repliche Aurora ed effettuare direttamente una connessione alle stesse per distribuire il carico di lettura del cluster DB. Non è necessario che le repliche Aurora appartengano alla stessa classe di istanze database dell'istanza primaria.

Per informazioni sull'aggiunta di repliche di Aurora a un cluster di database, consulta Aggiunta di repliche di Aurora a un cluster di database.

Gestione delle connessioni

Il numero massimo di connessioni consentite a un'istanza database di Aurora è determinato dal parametro `max_connections` nel gruppo di parametri a livello di istanza per l'istanza database. Il valore predefinito di tale parametro varia in base alla classe di istanze database utilizzata per l'istanza database e alla compatibilità del motore di database.

Motore di database	Valore predefinito di max_connections
Amazon Aurora MySQL	Per informazioni, consulta Numero massimo di connessioni a un'istanza database Aurora MySQL
Amazon Aurora PostgreSQL	Per informazioni, consultare Numero massimo di connessioni a un'istanza database Aurora PostgreSQL.

> **Tip**
>
> Se le applicazioni aprono e chiudono frequentemente connessioni o mantengono un numero elevato di connessioni di lunga durata aperte, ti consigliamo di utilizzare Amazon RDS Proxy. Proxy RDS è un proxy di database completamente gestito e ad alta disponibilità che utilizza il pooling di connessioni per condividere connessioni di database in modo sicuro ed efficiente. Per ulteriori informazioni sul Proxy RDS, consulta Server proxy per Amazon RDS per Aurora.

Gestione dei piani di esecuzione delle query

La gestione del piano di query per Aurora PostgreSQL ti consente di controllare quali piani verranno eseguiti dall'ottimizzatore. Per ulteriori informazioni, consulta Gestione dei piani di esecuzione delle query per Aurora PostgreSQL.

Clonazione di un volume per un cluster di database Amazon Aurora

La clonazione di Aurora ti consente di creare un nuovo cluster che abbia inizialmente le stesse pagine di dati dell'originale, ma sia un volume separato e indipendente. Il processo è progettato per essere veloce e conveniente. Il nuovo cluster con il relativo volume di dati associato è noto come clone. La creazione di un clone è più veloce ed efficiente in termini di spazio rispetto alla copia fisica dei dati utilizzando una tecnica diversa, ad esempio con il ripristino di uno snapshot.

Argomenti

- Panoramica sulla clonazione Aurora
- Limitazioni della clonazione Aurora
- Come funziona la clonazione Aurora
- Creazione di un clone Amazon Aurora
- Clonazione tra VPC con Amazon Aurora
- Clonazione tra account con AWS RAM e Amazon Aurora

Panoramica sulla clonazione Aurora

Per creare un clone, Aurora utilizza un protocollo copy-on-write. Questo meccanismo utilizza uno spazio aggiuntivo minimo per creare un clone iniziale. Quando il clone viene creato per la prima volta, Aurora conserva una singola copia dei dati utilizzati dal cluster database Aurora di origine e dal nuovo cluster database Aurora (clonato). L'archiviazione aggiuntiva viene allocata solo quando vengono apportate modifiche ai dati (sul volume di archiviazione Aurora) dal cluster database Aurora di origine o dal clone del cluster database Aurora. Per ulteriori informazioni sul protocollo copy-on-write, consulta Come funziona la clonazione Aurora.

La clonazione Aurora è particolarmente utile per configurare rapidamente ambienti di test utilizzando i dati di produzione, senza rischiare il danneggiamento dei dati. È possibile utilizzare i cloni per molti tipi di applicazioni di breve durata, ad esempio:

- Sperimenta potenziali cambiamenti (modifiche allo schema e modifiche ai gruppi di parametri, ad esempio) per valutare tutti gli impatti.
- Esegui operazioni che utilizzano in modo intensivo i carichi di lavoro, come l'esportazione di dati o l'esecuzione di query analitiche sul clone.
- Creare una copia del cluster database di produzione per lo sviluppo, il test o altri scopi.

È possibile creare più di un clone dallo stesso cluster database Aurora. È anche possibile creare più cloni da un altro clone.

Dopo aver creato un clone Aurora, è possibile configurare le istanze database Aurora in modo diverso dal cluster database Aurora di origine. Ad esempio, potrebbe non essere necessario un clone per scopi di sviluppo per soddisfare gli stessi requisiti di alta disponibilità del cluster database Aurora di produzione di origine. In questo caso, è possibile configurare il clone con una singola istanza database Aurora anziché con più istanze database utilizzate dal cluster database Aurora.

In caso di creazione di un clone con una configurazione di implementazione diversa da quella di origine, il clone viene creato utilizzando la versione secondaria più recente del motore di database Aurora di origine.

Quando crei cloni dai cluster database Aurora, i cloni vengono creati nell'account AWS, lo stesso account proprietario del cluster database Aurora di origine. Tuttavia, è anche possibile condividere Aurora Serverless v2, i cluster di database Aurora con provisioning e i cloni con altri account AWS. Per ulteriori informazioni, consulta Clonazione tra account con AWS RAM e Amazon Aurora.

Una volta terminato di utilizzare il clone per test, sviluppo o altri scopi, è possibile eliminarlo.

Limitazioni della clonazione Aurora

La clonazione Aurora presenta le seguenti limitazioni:

- Puoi creare tutti i cloni che desideri, fino al numero massimo di cluster database consentito nella Regione AWS.
- Puoi creare fino a 15 cloni con protocollo copy-on-write. Dopo che hai raggiunto 15 cloni, il clone successivo che crei è una copia completa. Il protocollo full-copy funziona come un ripristino point-in-time.
- Non puoi creare un clone in una regione AWS diversa rispetto a quella del cluster database Aurora di origine.
- Non è possibile creare un clone da un cluster database Aurora senza la funzionalità di query parallela a un cluster che utilizza query parallela. Per inserire dati in un cluster che usa la query parallela, crea uno snapshot del cluster originale ed eseguine il ripristino nel cluster in cui è abilitata l'opzione per la query parallela.
- Non è possibile creare un clone da un cluster database Aurora che non ha istanze database. È possibile clonare solo cluster database Aurora con almeno un'istanza database.

- È possibile creare un clone in un cloud privato virtuale (VPC) diverso da quello del cluster database Aurora. Tuttavia, le sottoreti in quei VPC devono essere associate alle stesse zone di disponibilità.
- È possibile creare un clone con provisioning Aurora da un cluster database Aurora con provisioning.
- I cluster con le istanze Aurora Serverless v2 seguono le stesse regole dei cluster con provisioning.
- Per Aurora Serverless v1:
  - È possibile creare un clone con provisioning da un cluster di database Aurora Serverless v1.
  - È possibile creare un clone Aurora Serverless v1 da un cluster di database con provisioning o Aurora Serverless v1.
  - Non è possibile creare un clone Aurora Serverless v1 da un cluster di database Aurora con provisioning non crittografato.
  - La clonazione tra più account al momento non supporta la clonazione di cluster database Aurora Serverless v1. Per ulteriori informazioni, consulta Restrizioni della clonazione tra più account.
  - Un cluster database Aurora Serverless v1 clonato ha lo stesso comportamento e le stesse limitazioni di qualsiasi altro cluster database Aurora Serverless v1. Per ulteriori informazioni, consulta Utilizzo di Amazon Aurora Serverless v1.
  - I cluster database Aurora Serverless v1 sono sempre crittografati. Quando si clona un cluster database Aurora Serverless v1 in un cluster database Aurora con provisioning, il cluster database Aurora con provisioning è crittografato. Puoi scegliere la chiave di crittografia, ma non è possibile disabilitare la crittografia. Per creare un clone da un cluster di database Aurora con provisioning in un cluster Aurora Serverless v1, occorre iniziare con un cluster di database Aurora con provisioning crittografato.

Come funziona la clonazione Aurora

La clonazione Aurora funziona a livello di archiviazione di un cluster database Aurora. Utilizza un protocollo copy-on-write, veloce ed efficiente in termini di spazio per il supporto durevole sottostante che supporta il volume di archiviazione Aurora. Per ulteriori informazioni, consulta la sezione relativa ai volumi cluster Aurora in Panoramica dell'archiviazione di Amazon Aurora.

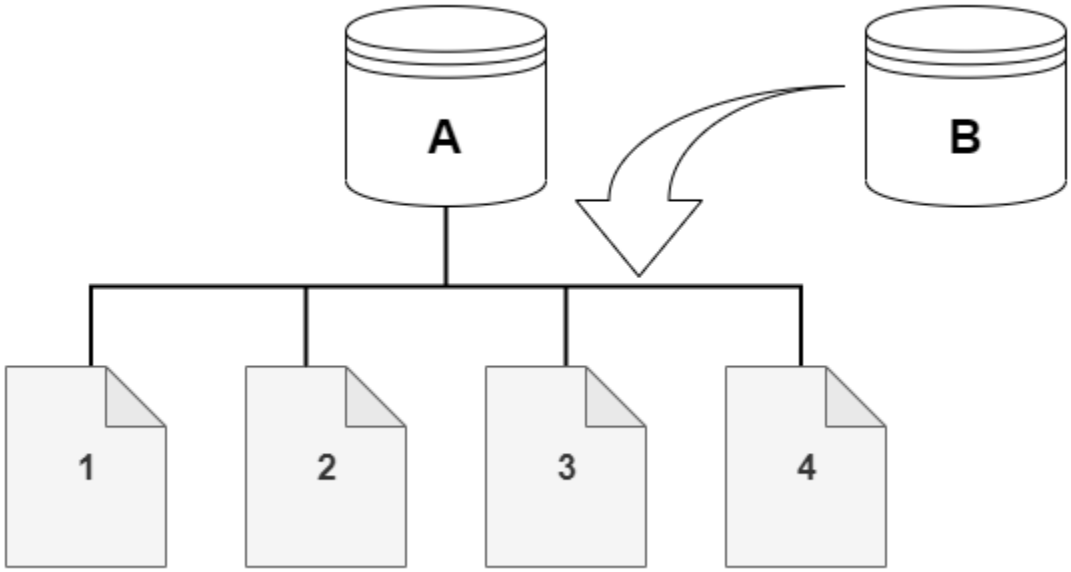
Argomenti

- Informazioni sul protocollo copy-on-write
- Eliminazione di un volume cluster di origine

Informazioni sul protocollo copy-on-write

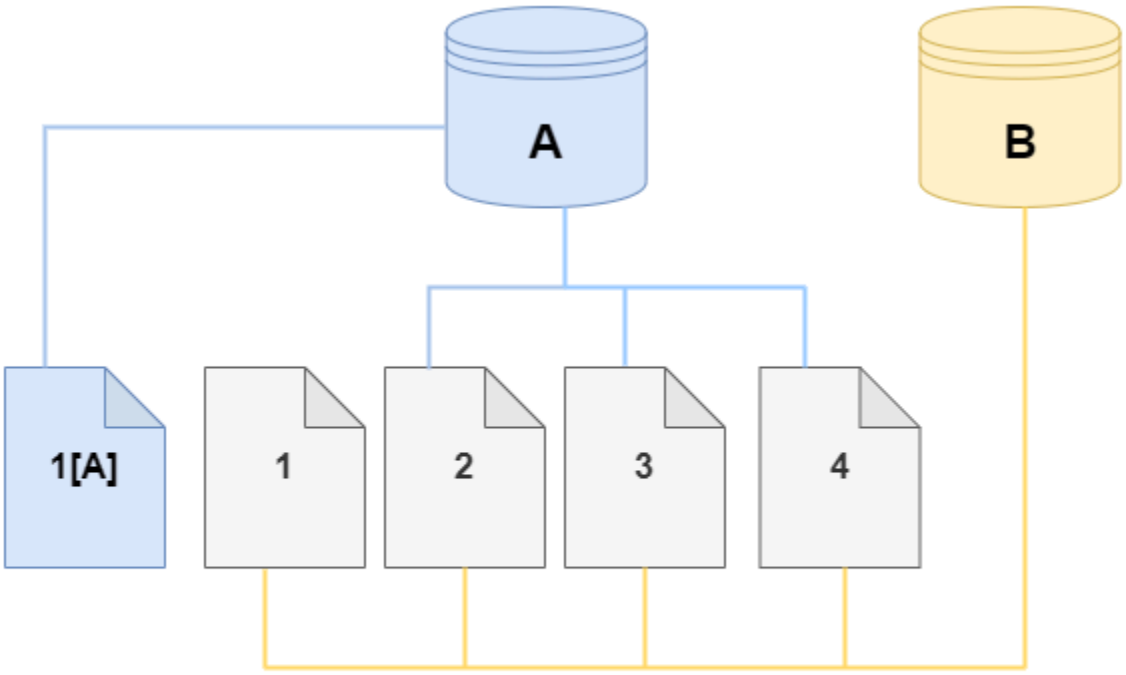
Un cluster database Aurora memorizza i dati nelle pagine del volume di archiviazione Aurora sottostante.

Ad esempio, nel diagramma seguente puoi vedere un cluster database Aurora (A) con quattro pagine dati, 1, 2, 3 e 4. Immagina che un clone, B, venga creato dal cluster database Aurora. Quando viene creato il clone, non viene copiato alcun dato. Piuttosto, il clone punta allo stesso insieme di pagine del cluster database Aurora di origine.

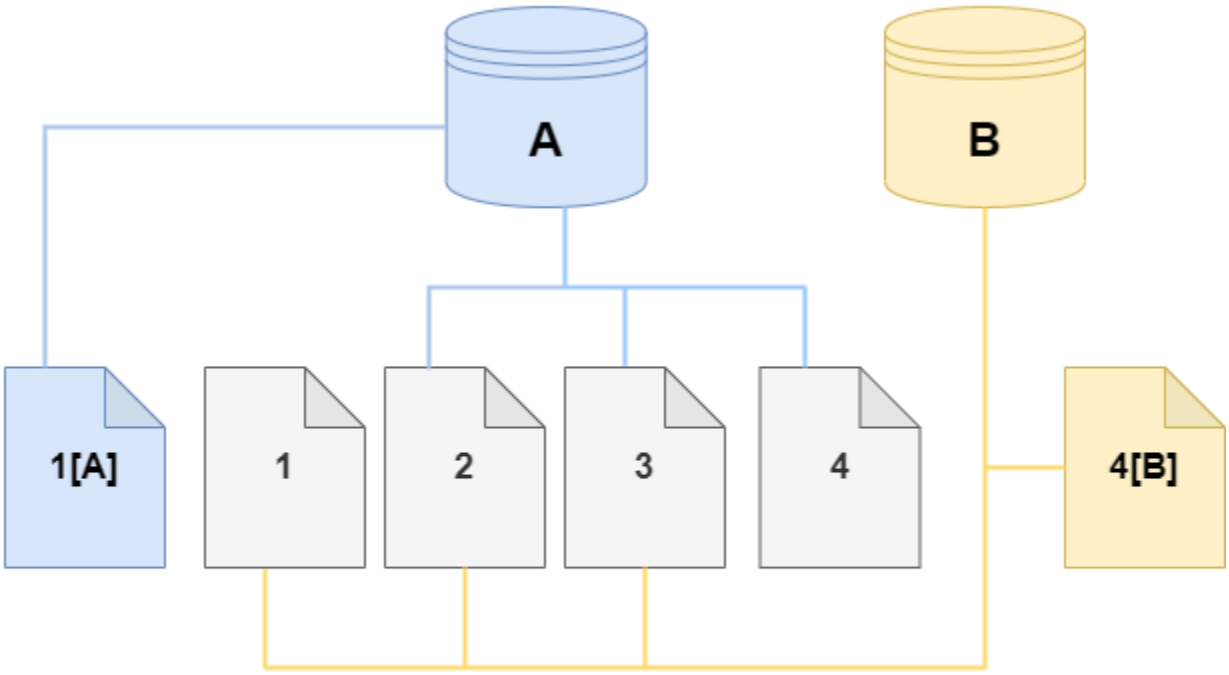


Quando viene creato il clone, in genere non è necessario alcuno spazio di archiviazione aggiuntivo. Il protocollo copy-on-write utilizza lo stesso segmento sul supporto di archiviazione fisico del segmento di origine. Lo spazio di archiviazione aggiuntivo è necessario solo se la capacità del segmento di origine non è sufficiente per l'intero segmento di clone. In questo caso, il segmento di origine viene copiato su un altro dispositivo fisico.

Nei diagrammi seguenti, puoi un esempio del protocollo copy-on-write in azione utilizzando lo stesso cluster A e il relativo clone, B, come illustrato in precedenza. Supponiamo che si apporti una modifica al cluster database Aurora (A) che si traduce in una modifica ai dati contenuti a pagina 1. Invece di scrivere sulla pagina originale 1, Aurora crea una nuova pagina 1[A]. Il volume del cluster database Aurora per cluster (A) punta ora alla pagina 1[A], 2, 3 e 4, mentre il clone (B) fa ancora riferimento alle pagine originali.



Sul clone, viene apportata una modifica a pagina 4 sul volume di archiviazione. Invece di scrivere sulla pagina originale 4, Aurora crea una nuova pagina 4[B]. Il clone punta ora alle pagine 1, 2, 3 e alla pagina 4[B], mentre il cluster (A) continua a puntare a 1[A], 2, 3 e 4.



Quando nel corso del tempo si verificano altre modifiche sia nel volume del cluster database Aurora di origine che nel clone, avrai bisogno di più spazio di archiviazione per acquisire e archiviare tali modifiche.

Eliminazione di un volume cluster di origine

Inizialmente, il volume clone condivide le stesse pagine di dati del volume originale a partire dal quale viene creato il clone. Finché esiste il volume originale, il volume clone viene considerato solo il proprietario delle pagine che il clone ha creato o modificato. Pertanto, la metrica `VolumeBytesUsed` per il volume clone all'inizio ha un valore ridotto, che aumenta solo quando i dati divergono tra il cluster originale e il clone. Per le pagine che sono identiche nel volume di origine e nel clone, i costi di archiviazione si applicano solo al cluster originale. Per ulteriori informazioni sul parametro `VolumeBytesUsed`, consulta Parametri a livello di cluster per Amazon Aurora.

Quando si elimina un volume cluster di origine a cui sono associati uno o più cloni, i dati nei volumi cluster dei cloni non sono interessati dall'operazione. Aurora conserva le pagine che in precedenza erano di proprietà del volume cluster di origine. Aurora ridistribuisce la fatturazione dello spazio di

archiviazione per le pagine che erano di proprietà del cluster eliminato. Supponiamo, ad esempio, che un cluster originale avesse due cloni e che a un certo punto il cluster originale sia stato eliminato. Di conseguenza, la metà delle pagine di dati di proprietà del cluster originale sarebbe di proprietà di un clone, mentre l'altra metà delle pagine sarebbe di proprietà dell'altro clone.

Una volta eliminato il cluster originale, man mano che crei o elimini altri cloni, Aurora continua a ridistribuire la proprietà delle pagine di dati tra tutti i cloni che condividono le stesse pagine. Potresti pertanto riscontrare che il valore della metrica `VolumeBytesUsed` cambia per il volume del cluster di un clone. Il valore della metrica può diminuire man mano che vengono creati altri cloni e la proprietà delle pagine viene distribuita tra più cluster. Inoltre, il valore della metrica può aumentare man mano che i cloni vengono eliminati e la proprietà delle pagine viene assegnata a un numero inferiore di cluster. Per informazioni sull'impatto delle operazioni di scrittura sulle pagine di dati nei volumi clonati, consulta Informazioni sul protocollo copy-on-write.

Quando il cluster originale e i cloni sono di proprietà dello stesso account AWS, tutti i costi di archiviazione per tali cluster si applicano allo stesso account AWS. Se alcuni cluster sono cloni multi-account, l'eliminazione del cluster originale può comportare l'addebito di costi di archiviazione aggiuntivi agli account AWS che possiedono i cloni multi-account.

Supponiamo, ad esempio, che un volume cluster abbia 1.000 pagine di dati utilizzate prima della creazione di uno o più cloni. Quando il cluster viene clonato, inizialmente il volume clone ha zero pagine utilizzate. Se il clone apporta modifiche a 100 pagine di dati, solo quelle 100 pagine vengono memorizzate nel volume clone e contrassegnate come utilizzate. Le altre 900 pagine del volume principale, che sono rimaste invariate, sono condivise da entrambi i cluster. In questo caso, il cluster principale prevede costi di archiviazione per 1.000 pagine e il volume clone per 100 pagine.

Se il volume di origine viene eliminato, i costi di archiviazione per il clone vanno a includere le 100 pagine modificate più le 900 pagine condivise dal volume originale, per un totale di 1.000 pagine.

Creazione di un clone Amazon Aurora

È possibile creare un clone nello stesso account AWS del cluster database Aurora di origine. A tale scopo, è possibile utilizzare la Console di gestione AWS o la AWS CLI e le procedure riportate di seguito.

Per consentire a un altro account AWS di creare un clone o condividere un clone con un altro account AWS, utilizzare le procedure descritte in Clonazione tra account con AWS RAM e Amazon Aurora.

Console

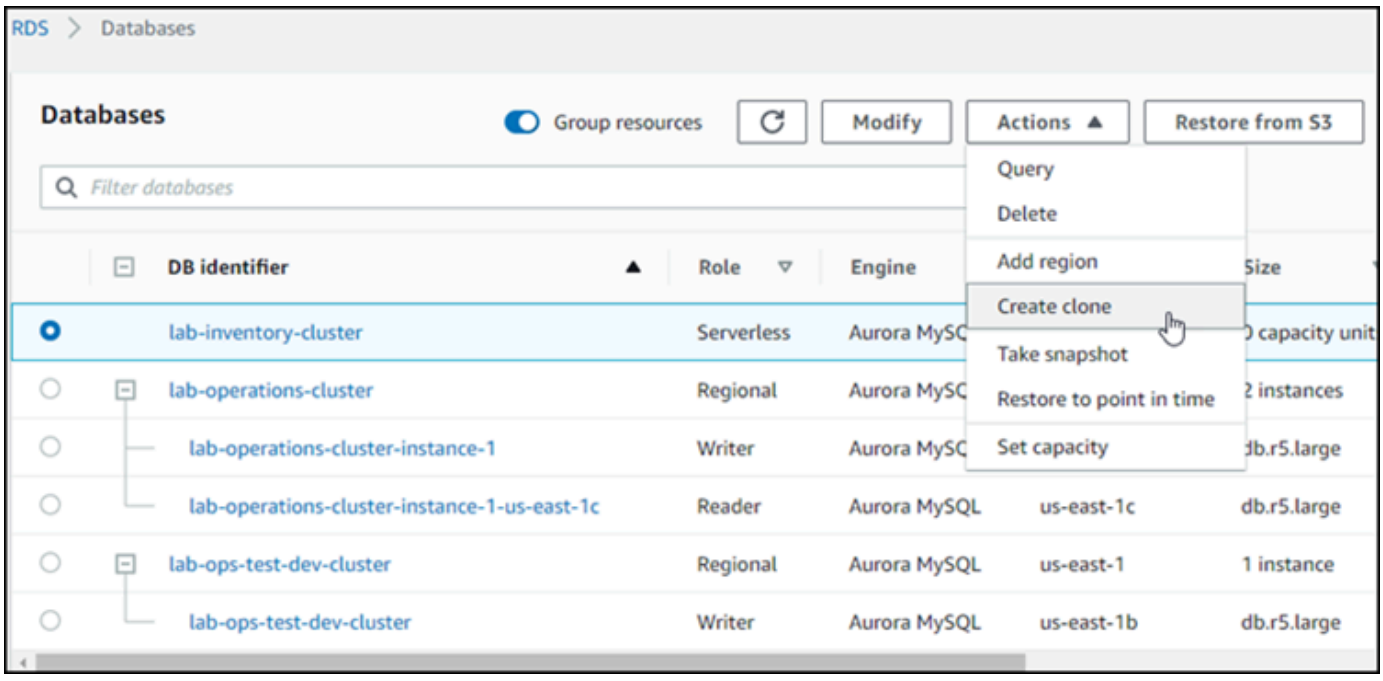
La procedura seguente descrive come clonare un cluster database Aurora utilizzando la Console di gestione AWS.

Creazione di un clone utilizzando i risultati Console di gestione AWS in un cluster database Aurora con una istanza database Aurora.

Queste istruzioni si applicano ai cluster di database di proprietà dello stesso account AWS che sta creando il clone. Se il cluster di database è di proprietà di un account AWS diverso, consulta Clonazione tra account con AWS RAM e Amazon Aurora.

Per creare un clone di un cluster di database di proprietà dell'account AWS tramite la Console di gestione AWS

1. Accedi alla Console di gestione AWS e apri la console Amazon RDS all'indirizzo https://console.aws.amazon.com/rds/.
2. Nel riquadro di navigazione, scegliere Databases (Database).
3. Seleziona il cluster database Aurora dall'elenco, quindi in Operazioni, seleziona Crea clone.



   Verrà visualizzata la pagina Crea clone, dove è possibile configurare le opzioni Impostazioni, Connettività e altre opzioni per il clone del cluster database Aurora.
4. Per Identificatore istanza database, immettere il nome che si desidera assegnare al cluster database Aurora clonato.

5. Per i cluster di database Aurora Serverless v1, scegli Con provisioning o Serverless per Tipo di capacità.

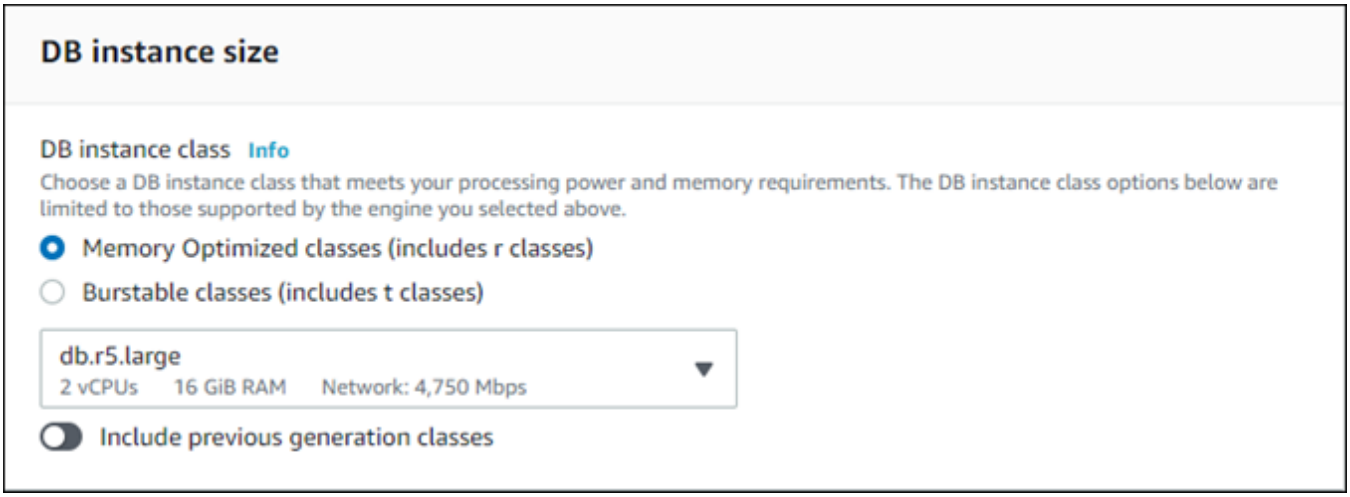
   È possibile scegliere Serverless solo se il cluster database Aurora di origine è un cluster database Aurora Serverless v1 o è un cluster database Aurora con provisioning crittografato.

6. Per i cluster di database con provisioning o Aurora Serverless v2, scegli Aurora I/O-Optimized o Aurora Standard per Configurazione dell'archiviazione del cluster.

   Per ulteriori informazioni, consulta Configurazioni dell'archiviazione per i cluster database Amazon Aurora.

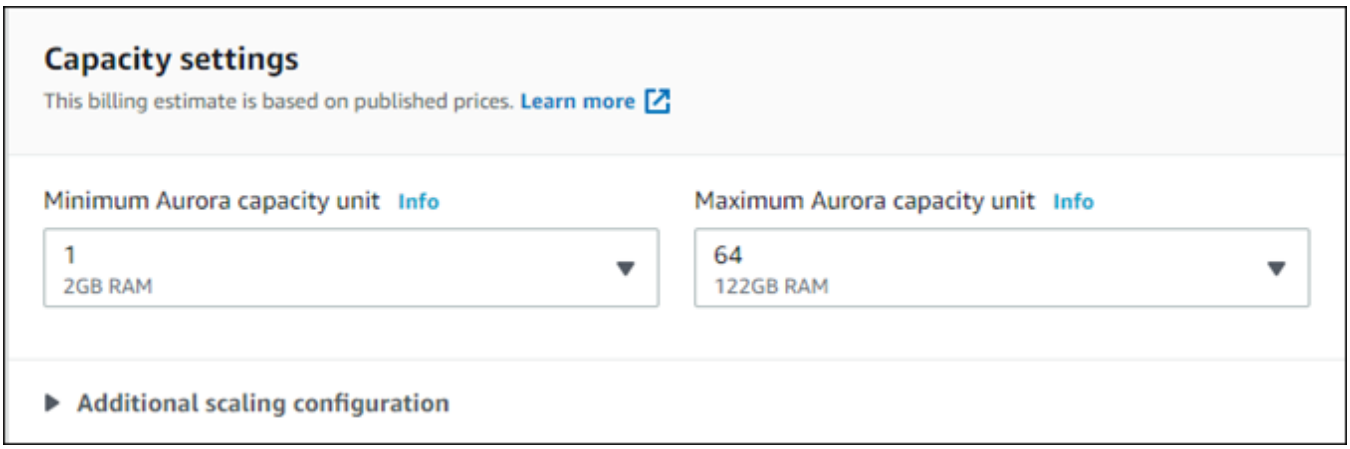
7. Scegliere la dimensione dell'istanza database o la capacità del cluster database:

   - Per un clone con provisioning, scegli una classe di istanza database.

     È possibile accettare l'impostazione predefinita oppure utilizzare una classe di istanza database diversa per il clone.

   - Per un clone Aurora Serverless v1 o Aurora Serverless v2, scegli un valore per Impostazioni della capacità.



È possibile accettare le impostazioni predefinite o modificarle per il tuo clone.

8. Scegli le altre impostazioni necessarie per il clone. Per ulteriori informazioni sulle impostazioni di cluster e istanza database Aurora, consulta Creazione di un cluster database Amazon Aurora.
9. Scegli Crea clone.

Quando viene creato il clone, viene elencato con gli altri cluster database Aurora nella sezione Database e se ne visualizza lo stato corrente. Il clone è pronto per l'utilizzo quando lo stato diventa Disponibile.

AWS CLI

L'utilizzo dell'interfaccia AWS CLI per la clonazione del cluster di database Aurora prevede fasi separate per la creazione del cluster clone e l'aggiunta di una o più istanze database al clone.

Il comando `restore-db-cluster-to-point-in-time` dell'interfaccia AWS CLI restituisce un cluster di database Aurora con gli stessi dati di archiviazione del cluster originale, ma senza istanze database Aurora. Le istanze database vengono create separatamente una volta che il clone diventa disponibile. È possibile scegliere il numero di istanze database e le relative classi di istanza per assegnare al clone una capacità di calcolo maggiore o minore rispetto a quella del cluster originale. Le fasi del processo sono descritte di seguito:

1. Creare il clone utilizzando il comando restore-db-cluster-to-point-in-time della CLI.
2. Creare l'istanza database di scrittura per il clone mediante il comando create-db-instance dell'interfaccia CLI.
3. (Facoltativo) Eseguire ulteriori comandi create-db-instance dell'interfaccia CLI per aggiungere una o più istanze di lettura al cluster clone. L'utilizzo di istanze di lettura aiuta a migliorare gli aspetti

relativi all'elevata disponibilità e alla scalabilità in lettura del clone. È possibile saltare questa fase qualora si intenda utilizzare il clone solo per scopi di sviluppo e test.

Argomenti

- Creazione del clone
- Controllo dello stato e ottenimento deii dettagli del clone
- Creazione dell'istanza database Aurora per il clone
- Parametri da utilizzare per la clonazione

Creazione del clone

Utilizza il comando `restore-db-cluster-to-point-in-time` dell'interfaccia CLI per creare il cluster clone iniziale.

Come creare un clone da un cluster di database Aurora di origine

- Utilizza il comando della CLI `restore-db-cluster-to-point-in-time`. Specifica i valori per i seguenti parametri. In questo caso tipico, il clone utilizza la stessa modalità motore del cluster originale (con provisioning o Aurora Serverless v1).
  - `--db-cluster-identifier`: scegliere un nome significativo per il clone. Il nome del clone viene assegnato utilizzando il comando restore-db-cluster-to-point-in-time della CLI. Il nome del clone viene quindi inviato nel comando create-db-instance della CLI.
  - `--restore-type`: utilizza `copy-on-write` per creare un clone del cluster database di origine. Senza questo parametro, il parametro `restore-db-cluster-to-point-in-time` ripristina il cluster database Aurora anziché creare un clone.
  - `--source-db-cluster-identifier`: utilizza il nome del cluster database Aurora di origine che si desidera clonare.
  - `--use-latest-restorable-time`: questo valore punta ai dati del volume ripristinabile più recenti per il cluster di database di origine. Utilizzalo per creare cloni.

Nell'esempio seguente viene creato un clone denominato `my-clone` da un cluster denominato `my-source-cluster`.

Per Linux, macOS o Unix:

```
aws rds restore-db-cluster-to-point-in-time \
    --source-db-cluster-identifier my-source-cluster \
    --db-cluster-identifier my-clone \
    --restore-type copy-on-write \
    --use-latest-restorable-time
```

Per Windows:

```
aws rds restore-db-cluster-to-point-in-time ^
    --source-db-cluster-identifier my-source-cluster ^
    --db-cluster-identifier my-clone ^
    --restore-type copy-on-write ^
    --use-latest-restorable-time
```

Il comando restituisce l'oggetto JSON contenente i dettagli del clone. Prima di provare a creare l'istanza database per il clone, verificare che il cluster database clonato sia disponibile. Per ulteriori informazioni, consulta Controllo dello stato e ottenimento deii dettagli del clone.

Si supponga, ad esempio, di disporre di un cluster denominato `tpch100g` che si desidera clonare. Nell'esempio seguente per Linux vengono creati un cluster clonato denominato `tpch100g-clone`, un'istanza di scrittura Aurora Serverless v2 denominata `tpch100g-clone-instance` e un'istanza di lettura con provisioning denominata `tpch100g-clone-instance-2` per il nuovo cluster.

Non è necessario fornire alcun parametro, come `--master-username` e `--master-user-password`. Aurora determina automaticamente quelli dal cluster originale. È necessario specificare il motore database da utilizzare. Pertanto, l'esempio verifica il nuovo cluster per determinare il valore corretto da utilizzare per il parametro `--engine`.

Questo esempio include anche l'opzione `--serverless-v2-scaling-configuration` per la creazione del cluster clone. In questo modo, è possibile aggiungere istanze Aurora Serverless v2 al clone anche se il cluster originale non utilizzava Aurora Serverless v2.

```
$ aws rds restore-db-cluster-to-point-in-time \
  --source-db-cluster-identifier tpch100g \
  --db-cluster-identifier tpch100g-clone \
  --serverless-v2-scaling-configuration MinCapacity=0.5,MaxCapacity=16 \
  --restore-type copy-on-write \
  --use-latest-restorable-time

$ aws rds describe-db-clusters \
  --db-cluster-identifier tpch100g-clone \
```

```
    --query '*[].[Engine]' \
    --output text
aurora-mysql

$ aws rds create-db-instance \
  --db-instance-identifier tpch100g-clone-instance \
  --db-cluster-identifier tpch100g-clone \
  --db-instance-class db.serverless \
  --engine aurora-mysql

$ aws rds create-db-instance \
  --db-instance-identifier tpch100g-clone-instance-2 \
  --db-cluster-identifier tpch100g-clone \
  --db-instance-class db.r6g.2xlarge \
  --engine aurora-mysql
```

Come creare un clone con una modalità motore differente rispetto a quella del cluster di database Aurora di origine

- Questa procedura si applica solo alle versioni di motore precedenti che supportano Aurora Serverless v1. Supponiamo di avere un cluster Aurora Serverless v1 e di voler creare un clone che sia un cluster con provisioning. In questo caso, occorre utilizzare il comando `restore-db-cluster-to-point-in-time` dell'interfaccia CLI e specificare valori di parametro simili a quelli dell'esempio precedente, più questi parametri aggiuntivi:

  - `--engine-mode`: utilizza questo parametro solo per creare cloni con modalità motore diversa rispetto a quella del cluster di database Aurora di origine. Questo parametro si applica solo alle versioni di motore precedenti che supportano Aurora Serverless v1. Scegli il valore da inviare con `--engine-mode` come riportato di seguito:

    - Utilizzare `--engine-mode provisioned` per creare un clone del cluster database Aurora da un cluster database Aurora Serverless.

      > **Note**
      >
      > Se intendi utilizzare Aurora Serverless v2 con un cluster che è stato clonato da Aurora Serverless v1, devi comunque specificare la modalità motore `provisioned` per il clone. Successivamente devi eseguire ulteriori fasi di aggiornamento e migrazione.

- Utilizza `--engine-mode serverless` per creare un clone Aurora Serverless v1 a partire da un cluster di database Aurora con provisioning. Quando specifichi la modalità motore `serverless`, puoi scegliere anche `--scaling-configuration`.

- `--scaling-configuration`: (facoltativo) da utilizzare con `--engine-mode serverless` per configurare la capacità minima e massima di un clone Aurora Serverless v1. Se non utilizzi questo parametro, Aurora crea un clone Aurora Serverless v1 utilizzando i valori di capacità Aurora Serverless v1 predefiniti per il motore di database.

Nel seguente esempio viene creato un clone con provisioning denominato `my-clone` a partire da un cluster di database Aurora Serverless v1 denominato `my-source-cluster`.

Per Linux, macOS o Unix:

```
aws rds restore-db-cluster-to-point-in-time \
    --source-db-cluster-identifier my-source-cluster \
    --db-cluster-identifier my-clone \
    --engine-mode provisioned \
    --restore-type copy-on-write \
    --use-latest-restorable-time
```

Per Windows:

```
aws rds restore-db-cluster-to-point-in-time ^
    --source-db-cluster-identifier my-source-cluster ^
    --db-cluster-identifier my-clone ^
    --engine-mode provisioned ^
    --restore-type copy-on-write ^
    --use-latest-restorable-time
```

Questi comandi restituiscono l'oggetto JSON contenente i dettagli del clone necessari per creare l'istanza database. Non puoi farlo finché lo stato del clone (il cluster database Aurora vuoto) non diventa Disponibile.

> **Note**
>
> Il comando restore-db-cluster-to-point-in-time della CLI AWS ripristina solo il cluster DB e non le istanze database per quel cluster. Per creare istanze database per il cluster di database ripristinato, esegui il comando create-db-instance. Questo comando consente di specificare

l'identificatore del cluster di database ripristinato come parametro `--db-cluster-identifier`. Puoi creare le istanze database solo dopo che il comando `restore-db-cluster-to-point-in-time` è terminato e il cluster database è disponibile.
Si supponga di iniziare con un cluster Aurora Serverless v1 e di volerne eseguire la migrazione a un cluster Aurora Serverless v2. La prima fase della migrazione prevede la creazione di un clone con provisioning del cluster Aurora Serverless v1. Per la procedura completa, inclusi eventuali aggiornamenti di versione richiesti, consulta Aggiornamento da un cluster Aurora Serverless v1 ad Aurora Serverless v2.

Controllo dello stato e ottenimento deii dettagli del clone

Puoi utilizzare il seguente comando per controllare lo stato del cluster clone appena creato.

```
$ aws rds describe-db-clusters --db-cluster-identifier my-clone --query '*[].[Status]'
 --output text
```

In alternativa, puoi ottenere lo stato e gli altri valori necessari per creare l'istanza database per il clone utilizzando la seguente query AWS CLI.

Per Linux, macOS o Unix:

```
aws rds describe-db-clusters --db-cluster-identifier my-clone \
  --query '*[].
{Status:Status,Engine:Engine,EngineVersion:EngineVersion,EngineMode:EngineMode}'
```

Per Windows:

```
aws rds describe-db-clusters --db-cluster-identifier my-clone ^
  --query "*[].
{Status:Status,Engine:Engine,EngineVersion:EngineVersion,EngineMode:EngineMode}"
```

Questa query restituisce un output simile al seguente:

```
[
  {
        "Status": "available",
        "Engine": "aurora-mysql",
```

```
        "EngineVersion": "8.0.mysql_aurora.3.04.1",
        "EngineMode": "provisioned"
    }
]
```

Creazione dell'istanza database Aurora per il clone

Utilizza il comando create-db-instance dell'interfaccia CLI per creare l'istanza database per il tuo clone con provisioning o Aurora Serverless v2. Per i cloni Aurora Serverless v1 non viene creata un'istanza database.

L'istanza database eredita le proprietà `--master-username` e `--master-user-password` dal cluster di database di origine.

Nell'esempio seguente viene creata un'istanza database per un clone con provisioning.

Per Linux, macOS o Unix:

```
aws rds create-db-instance \
    --db-instance-identifier my-new-db \
    --db-cluster-identifier my-clone \
    --db-instance-class db.r6g.2xlarge \
    --engine aurora-mysql
```

Per Windows:

```
aws rds create-db-instance ^
    --db-instance-identifier my-new-db ^
    --db-cluster-identifier my-clone ^
    --db-instance-class db.r6g.2xlarge ^
    --engine aurora-mysql
```

Nell'esempio seguente viene creata un'istanza database Aurora Serverless v2 per un clone che utilizza una versione di motore che supporta Aurora Serverless v2.

Per Linux, macOS o Unix:

```
aws rds create-db-instance \
    --db-instance-identifier my-new-db \
    --db-cluster-identifier my-clone \
```

```
    --db-instance-class db.serverless \
  --engine aurora-postgresql
```

Per Windows:

```
aws rds create-db-instance ^
    --db-instance-identifier my-new-db ^
    --db-cluster-identifier my-clone ^
    --db-instance-class db.serverless ^
    --engine aurora-mysql
```

Parametri da utilizzare per la clonazione

La tabella seguente riepiloga i vari parametri utilizzati con `restore-db-cluster-to-point-in-time` per clonare i cluster database Aurora.

Parametro	Descrizione
`--source-db-cluster-identifier`	Utilizza il nome del cluster database Aurora di origine che si desidera clonare.
`--db-cluster-identifier`	Quando crei un clone con il comando `restore-db-cluster-to-point-in-time` , scegli un nome significativo da assegnargli. Quindi questo nome viene inviato al comando `create-db-instance` .
`--restore-type`	Specifica `copy-on-write` come `--restore-type` per creare un clone del cluster database di origine anziché ripristinare il cluster database Aurora di origine.
`--use-latest-restorable-time`	Questo valore punta ai dati del volume ripristinabile più recenti per il cluster di database di origine. Utilizzalo per creare cloni.
`--serverless-v2-scaling-configuration`	Versioni più recenti che supportano Aurora Serverless v2. Utilizza questo parametro per configurare la capacità minima e massima di un clone Aurora Serverless v2. Se non specifichi questo parametro, non potrai creare alcuna istanza Aurora Serverless v2 nel cluster clone fintanto che non avrai modificato il cluster per aggiungere tale attributo.

Parametro	Descrizione
`--engine-mode`	Solo versioni meno recenti che supportano Aurora Serverless v1. Utilizza questo parametro per creare cloni di tipo diverso rispetto al cluster di database Aurora di origine, con uno dei seguenti valori: • Utilizza `provisioned` per creare un clone con provisioning a partire da un cluster di database Aurora Serverless v1. • Utilizza `serverless` per creare un clone Aurora Serverless v1 a partire da un cluster di database con provisioning o Aurora Serverless v2. Quando specifichi la modalità motore `serverless`, puoi scegliere anche `--scaling-configuration`.
`--scaling-configuration`	Solo versioni meno recenti che supportano Aurora Serverless v1. Utilizza questo parametro per configurare la capacità minima e massima di un clone Aurora Serverless v1. Se non specifichi questo parametro, Aurora creerà il clone utilizzando i valori di capacità predefiniti per il motore di database.

Per informazioni sulla clonazione multi-VPC e multi-account, consulta le seguenti sezioni.

Argomenti

- Clonazione tra VPC con Amazon Aurora
- Clonazione tra account con AWS RAM e Amazon Aurora

Clonazione tra VPC con Amazon Aurora

Supponiamo di voler imporre controlli degli accessi alla rete differenti sul cluster originale e sul clone. Ad esempio, è possibile utilizzare la clonazione per creare una copia di un cluster Aurora di produzione in un VPC diverso per lo sviluppo e il test. In alternativa, potresti creare un clone nell'ambito di una migrazione da sottoreti pubbliche a sottoreti private, per migliorare la sicurezza del database.

Le sezioni seguenti mostrano come impostare la configurazione di rete per il clone in modo che il cluster originale e il clone possano entrambi accedere agli stessi nodi di archiviazione Aurora, anche

da sottoreti diverse o VPC diversi. La verifica preventiva delle risorse di rete può evitare errori durante la clonazione che potrebbero essere difficili da diagnosticare.

Se non conosci il modo in cui Aurora interagisce con VPC, sottoreti e gruppi di sottoreti di database, consulta prima VPC Amazon e Amazon Aurora. Puoi seguire i tutorial in quella sezione per creare questo tipo di risorse nella console AWS e capire come si integrano tra loro.

Poiché la procedura prevede il passaggio da RDS a EC2 e viceversa, gli esempi utilizzano i comandi AWS dell'interfaccia CLI per aiutarti a capire come automatizzare tali operazioni e salvare l'output.

Argomenti

- Prima di iniziare
- Raccolta delle informazioni sull'ambiente di rete
- Creazione di risorse di rete per il clone
- Creazione di un clone Aurora con nuove impostazioni di rete
- Spostamento di un cluster da sottoreti pubbliche a sottoreti private
- Esempio end-to-end di creazione di un clone tra VPC

Prima di iniziare

Prima di iniziare a configurare un clone tra VPC, assicurati di disporre delle seguenti risorse:

- Un cluster di database Aurora da utilizzare come origine per la clonazione. Se è la prima volta che crei un cluster di database Aurora, consulta i tutorial in Nozioni di base su Amazon Aurora su come configurare un cluster utilizzando il motore di database MySQL o PostgreSQL.
- Un secondo VPC, se intendi creare un clone tra VPC. Se non disponi di un VPC da utilizzare per il clone, consulta Tutorial: Creazione di un Amazon VPC da utilizzare con un cluster database (solo IPv4) o Tutorial: Creazione di un VPC per l'utilizzo con un cluster database (modalità dual-stack).

Raccolta delle informazioni sull'ambiente di rete

Con la clonazione tra VPC, l'ambiente di rete può avere differenze sostanziali tra il cluster originale e il relativo clone. Prima di creare il clone, raccogli e registra le informazioni sul VPC, sulle sottoreti, sul gruppo di sottoreti di database e sulle AZ utilizzate nel cluster originale. In questo modo, puoi ridurre al minimo la possibilità di incorrere in problemi. Se si verifica un problema di rete, non sarà necessario interrompere alcuna attività di risoluzione dei problemi per cercare informazioni

diagnostiche. Le sezioni seguenti mostrano esempi dell'interfaccia CLI per raccogliere questo tipo di informazioni. È possibile salvare i dettagli nel formato più comodo da consultare durante la creazione del clone e la risoluzione dei problemi.

- Fase 1: verificare le zone di disponibilità del cluster originale
- Fase 2: verificare il gruppo di sottoreti di database del cluster originale
- Fase 3: verificare le sottoreti del cluster originale
- Fase 4: verificare le zone di disponibilità delle istanze database nel cluster originale
- Fase 5: verificare i VPC che puoi utilizzare per il clone

Fase 1: verificare le zone di disponibilità del cluster originale

Prima di creare il clone, verifica quali AZ utilizza il cluster originale per l'archiviazione. Come spiegato in Archiviazione Amazon Aurora, l'archiviazione per ogni cluster Aurora è associata esattamente a tre AZ. Poiché Cluster database Amazon Aurora sfrutta la separazione tra calcolo e archiviazione, questa regola è valida indipendentemente dal numero di istanze presenti nel cluster.

Ad esempio, puoi eseguire un comando dell'interfaccia CLI come il seguente, sostituendo il nome del tuo cluster con *my_cluster*. Nell'esempio seguente viene generato un elenco ordinato alfabeticamente in base al nome della AZ.

```
aws rds describe-db-clusters \
  --db-cluster-identifier my_cluster \
  --query 'sort_by(*[].AvailabilityZones[].{Zone:@},&Zone)' \
  --output text
```

Di seguito è riportato l'output di esempio del comando `describe-db-clusters` precedente. Dimostra che l'archiviazione per il cluster Aurora utilizza sempre tre AZ.

```
us-east-1c
us-east-1d
us-east-1e
```

Per creare un clone in un ambiente di rete che non dispone di tutte le risorse necessarie per connettersi a queste AZ, è necessario creare sottoreti associate ad almeno due di tali AZ e quindi creare un gruppo di sottoreti di database contenente queste due o tre sottoreti. Negli esempi seguenti viene mostrato come fare.

Fase 2: verificare il gruppo di sottoreti di database del cluster originale

Se desideri utilizzare per il clone lo stesso numero di sottoreti del cluster originale, puoi ottenere il numero di sottoreti dal gruppo di sottoreti di database del cluster originale. Un gruppo di sottoreti di database Aurora contiene almeno due sottoreti, ciascuna associata a una AZ diversa. Prendi nota delle AZ a cui sono associate le sottoreti.

L'esempio seguente mostra come trovare il gruppo di sottoreti di database del cluster originale e quindi lavorare a ritroso sulle AZ corrispondenti. Sostituisci il nome del cluster con *my_cluster* nel primo comando. Sostituire il nome del gruppo di sottoreti di database con *my_subnet* nel secondo comando.

```
aws rds describe-db-clusters --db-cluster-identifier my_cluster \
  --query '*[].DBSubnetGroup' --output text

aws rds describe-db-subnet-groups --db-subnet-group-name my_subnet_group \
  --query '*[].Subnets[].[SubnetAvailabilityZone.Name]' --output text
```

L'output di esempio potrebbe essere simile al seguente, per un cluster con un gruppo di sottoreti di database contenente due sottoreti. In questo caso, `two-subnets` è un nome specificato durante la creazione del gruppo di sottoreti di database.

```
two-subnets

us-east-1d
us-east-1c
```

L'output potrebbe essere simile al seguente, per un cluster con un gruppo di sottoreti di database contenente tre sottoreti.

```
three-subnets

us-east-1f
us-east-1d
us-east-1c
```

Fase 3: verificare le sottoreti del cluster originale

Se hai bisogno di maggiori dettagli sulle sottoreti nel cluster originale, esegui comandi AWS dell'interfaccia CLI simili ai seguenti. Puoi esaminare gli attributi delle sottoreti come gli intervalli di

indirizzi IP, il proprietario e così via. In questo modo, puoi determinare se utilizzare sottoreti diverse nello stesso VPC o creare sottoreti con caratteristiche simili in un VPC diverso.

Trova gli ID di tutte le sottoreti disponibili nel tuo VPC.

```
aws ec2 describe-subnets --filters Name=vpc-id,Values=my_vpc \
  --query '*[].[SubnetId]' --output text
```

Trova le sottoreti utilizzate nel gruppo di sottoreti di database.

```
aws rds describe-db-subnet-groups --db-subnet-group-name my_subnet_group \
  --query '*[].Subnets[].[SubnetIdentifier]' --output text
```

Quindi specifica le sottoreti che desideri esaminare in un elenco, come nel comando seguente. Sostituisci i nomi delle tue sottoreti con *my_subnet_1* e così via.

```
aws ec2 describe-subnets \
  --subnet-ids '["my_subnet_1","my_subnet2","my_subnet3"]'
```

L'esempio seguente mostra l'output parziale del comando `describe-subnets`. L'output mostra alcuni degli attributi importanti che puoi vedere per ogni sottorete, come la AZ associata e il VPC di cui fa parte.

```
{
    'Subnets': [
        {
            'AvailabilityZone': 'us-east-1d',
            'AvailableIpAddressCount': 54,
            'CidrBlock': '10.0.0.64/26',
            'State': 'available',
            'SubnetId': 'subnet-000a0bca00e0b0000',
            'VpcId': 'vpc-3f3c3fc3333b3ffb3',
            ...
        },
        {
            'AvailabilityZone': 'us-east-1c',
            'AvailableIpAddressCount': 55,
            'CidrBlock': '10.0.0.0/26',
            'State': 'available',
            'SubnetId': 'subnet-4b4dbfe4d4a4fd4c4',
            'VpcId': 'vpc-3f3c3fc3333b3ffb3',
```

```
            ...
```

Fase 4: verificare le zone di disponibilità delle istanze database nel cluster originale

Puoi utilizzare questa procedura per comprendere le AZ utilizzate per le istanze database nel cluster originale. In questo modo, puoi configurare esattamente le stesse AZ per le istanze database nel clone. Inoltre, puoi utilizzare più o meno istanze database nel clone a seconda che il clone venga utilizzato per la produzione, lo sviluppo e il test e così via.

Per ogni istanza del cluster originale, esegui un comando come quello riportato di seguito. Verifica che l'istanza abbia terminato la creazione e che si trovi prima nello stato `Available`. Sostituisci l'identificatore dell'istanza con *`my_instance`*.

```
aws rds describe-db-instances --db-instance-identifier my_instance \
  --query '*[].AvailabilityZone' --output text
```

L'esempio seguente mostra l'output dell'esecuzione del comando `describe-db-instances` precedente. Il cluster Aurora dispone di quattro istanze database. Pertanto, il comando deve essere eseguito quattro volte, sostituendo ogni volta un identificatore di istanza database diverso. L'output mostra come tali istanze database sono distribuite su un massimo di tre AZ.

```
us-east-1a
us-east-1c
us-east-1d
us-east-1a
```

Dopo aver creato il clone e avervi aggiunto delle istanze database, puoi specificare gli stessi nomi delle AZ nei comandi `create-db-instance`. Potresti eseguire tale operazione per configurare le istanze database nel nuovo cluster configurato esattamente per le stesse AZ del cluster originale.

Fase 5: verificare i VPC che puoi utilizzare per il clone

Se intendi creare il clone in un VPC diverso dall'originale, puoi ottenere un elenco degli ID VPC disponibili per il tuo account. Puoi eseguire questa fase anche se è necessario creare eventuali sottoreti aggiuntive nello stesso VPC del cluster originale. Quando esegui il comando per creare una sottorete, devi specificare l'ID VPC come parametro.

Per elencare tutti i VPC per il tuo account, esegui il comando dell'interfaccia CLI riportato di seguito.

```
aws ec2 describe-vpcs --query '*[].[VpcId]' --output text
```

Di seguito è riportato l'output di esempio del comando `describe-vpcs` precedente. L'output mostra che nell'account AWS corrente sono presenti quattro VPC che possono essere utilizzati come origine o destinazione per la clonazione tra VPC.

```
vpc-fd111111
vpc-2222e2cd2a222f22e
vpc-33333333a33333d33
vpc-4ae4d4de4a4444dad
```

Come destinazione del clone puoi utilizzare lo stesso VPC o un VPC diverso. Se il cluster originale e il clone si trovano nello stesso VPC, puoi riutilizzare lo stesso gruppo di sottoreti di database per il clone. Puoi anche creare un gruppo di sottoreti di database. Ad esempio, il nuovo gruppo di sottoreti di database potrebbe utilizzare sottoreti private, mentre il gruppo di sottoreti di database del cluster originale potrebbe utilizzare sottoreti pubbliche. Se crei il clone in un VPC diverso, assicurati che nel nuovo VPC sia presente un numero sufficiente di sottoreti e che le sottoreti siano associate alle AZ corrette del cluster originale.

Creazione di risorse di rete per il clone

Se durante la raccolta delle informazioni di rete hai scoperto che per il clone sono necessarie risorse di rete aggiuntive, puoi creare tali risorse prima di provare a configurare il clone. Ad esempio, potrebbe essere necessario creare più sottoreti, sottoreti associate ad AZ specifiche o un nuovo gruppo di sottoreti di database.

- Fase 1: creare le sottoreti per il clone
- Fase 2: creare il gruppo di sottoreti di database per il clone

Fase 1: creare le sottoreti per il clone

Se devi creare nuove sottoreti per il clone, esegui un comando simile al seguente. Potresti dover eseguire questa operazione quando crei il clone in un VPC diverso o quando apporti altre modifiche alla rete, ad esempio l'utilizzo di sottoreti private anziché di sottoreti pubbliche.

AWS genera automaticamente l'ID della sottorete. Sostituisci il nome del VPC del clone con *`my_vpc`*. Scegli l'intervallo di indirizzi per l'opzione `--cidr-block` al fine di consentire almeno 16 indirizzi IP nell'intervallo. Puoi includere qualsiasi altra proprietà che desideri specificare. Esegui il comando `aws ec2 create-subnet help` per visualizzare tutte le scelte.

```
aws ec2 create-subnet --vpc-id my_vpc \
```

```
  --availability-zone AZ_name --cidr-block IP_range
```

L'esempio seguente mostra alcuni attributi importanti di una sottorete appena creata.

```
{
    'Subnet': {
        'AvailabilityZone': 'us-east-1b',
        'AvailableIpAddressCount': 59,
        'CidrBlock': '10.0.0.64/26',
        'State': 'available',
        'SubnetId': 'subnet-44b4a44f4e44db444',
        'VpcId': 'vpc-555fc5df555e555dc',
        ...
        }
}
```

Fase 2: creare il gruppo di sottoreti di database per il clone

Se stai creando il clone in un VPC diverso o un diverso set di sottoreti all'interno dello stesso VPC, devi creare un nuovo gruppo di sottoreti di database e specificarlo durante la creazione del clone.

Assicurati di conoscere tutti i seguenti dettagli. Puoi trovarli tutti nell'output degli esempi precedenti.

1. VPC del cluster originale. Per istruzioni, consultare Fase 3: verificare le sottoreti del cluster originale.
2. VPC del clone, se stai eseguendo la creazione in un VPC diverso. Per istruzioni, consultare Fase 5: verificare i VPC che puoi utilizzare per il clone.
3. Tre AZ associate all'archiviazione Aurora per il cluster originale. Per istruzioni, consultare Fase 1: verificare le zone di disponibilità del cluster originale.
4. Due o tre AZ associate al gruppo di sottoreti di database per il cluster originale. Per istruzioni, consultare Fase 2: verificare il gruppo di sottoreti di database del cluster originale.
5. Gli ID di sottorete e le AZ associate di tutte le sottoreti nel VPC che intendi utilizzare per il clone. Utilizza lo stesso comando `describe-subnets` di Fase 3: verificare le sottoreti del cluster originale, sostituendo l'ID del VPC di destinazione.

Verifica quante AZ sono associate all'archiviazione del cluster originale e alle sottoreti nel VPC di destinazione. Per creare correttamente il clone, devono esserci due o tre AZ in comune. Se hai meno di due AZ in comune, torna a Fase 1: creare le sottoreti per il clone. Crea una, due o tre nuove sottoreti associate alle AZ che sono associate all'archiviazione del cluster originale.

Scegli le sottoreti nel VPC di destinazione associate alle stesse AZ dell'archiviazione Aurora nel cluster originale. Idealmente, scegli tre AZ. In questo modo avrai la massima flessibilità per distribuire le istanze database del clone su più AZ per un'elevata disponibilità delle risorse di calcolo.

Esegui un comando simile al seguente per creare il nuovo gruppo di sottoreti di database. Sostituisci gli ID delle tue sottoreti nell'elenco. Se specifichi gli ID di sottorete utilizzando variabili di ambiente, presta attenzione a citare l'elenco dei parametri `--subnet-ids` in modo da preservare le virgolette doppie intorno agli ID.

```
aws rds create-db-subnet-group --db-subnet-group-name my_subnet_group \
  --subnet-ids '["my_subnet_1","my_subnet_2","my_subnet3"]' \
  --db-subnet-group-description 'DB subnet group with 3 subnets for clone'
```

L'esempio seguente mostra l'output parziale del comando `create-db-subnet-group`.

```
{
    'DBSubnetGroup': {
        'DBSubnetGroupName': 'my_subnet_group',
        'DBSubnetGroupDescription': 'DB subnet group with 3 subnets for clone',
        'VpcId': 'vpc-555fc5df555e555dc',
        'SubnetGroupStatus': 'Complete',
        'Subnets': [
            {
                'SubnetIdentifier': 'my_subnet_1',
                'SubnetAvailabilityZone': {
                    'Name': 'us-east-1c'
                },
                'SubnetStatus': 'Active'
            },
            {
                'SubnetIdentifier': 'my_subnet_2',
                'SubnetAvailabilityZone': {
                    'Name': 'us-east-1d'
                },
                'SubnetStatus': 'Active'
            }
            ...
        ],
        'SupportedNetworkTypes': [
            'IPV4'
        ]
    }
```

```
}
```

A questo punto, non hai ancora creato il clone. Sono state create tutte le risorse di VPC e sottorete pertinenti in modo da poter specificare i parametri appropriati per i comandi `restore-db-cluster-to-point-in-time` e `create-db-instance` durante la creazione del clone.

Creazione di un clone Aurora con nuove impostazioni di rete

Dopo aver appurato che è disponibile la configurazione corretta di VPC, sottoreti, AZ e gruppi di sottoreti per l'utilizzo del nuovo cluster, puoi eseguire l'operazione di clonazione vera e propria. I seguenti esempi CLI evidenziano le opzioni come `--db-subnet-group-name`, `--availability-zone` e `--vpc-security-group-ids` che specifichi nei comandi per configurare il clone e le relative istanze database.

- Fase 1: specificare il gruppo di sottoreti di database per il clone
- Fase 2: specificare le impostazioni di rete per le istanze nel clone
- Fase 3: stabilire la connettività da un sistema client a un clone

Fase 1: specificare il gruppo di sottoreti di database per il clone

Quando crei il clone, puoi configurare tutte le impostazioni corrette per VPC, sottoreti e AZ specificando un gruppo di sottoreti di database. Utilizza i comandi degli esempi precedenti per verificare tutte le relazioni e le mappature che entrano nel gruppo di sottoreti di database.

Ad esempio, i comandi seguenti mostrano la clonazione di un cluster originale in un clone. Nel primo esempio, il cluster di origine è associato a due sottoreti e il clone è associato a tre sottoreti. Il secondo esempio mostra il caso opposto, ossia la clonazione da un cluster con tre sottoreti a un cluster con due sottoreti.

```
aws rds restore-db-cluster-to-point-in-time \
  --source-db-cluster-identifier cluster-with-3-subnets \
  --db-cluster-identifier cluster-cloned-to-2-subnets \
  --restore-type copy-on-write --use-latest-restorable-time \
  --db-subnet-group-name two-subnets
```

Se intendi utilizzare istanze Aurora Serverless v2 nel clone, includi un'opzione `--serverless-v2-scaling-configuration` quando crei il clone, come mostrato. In questo modo, puoi utilizzare la classe `db.serverless` durante la creazione di istanze database nel clone. Puoi anche

modificare il clone in un secondo momento per aggiungere questo attributo di configurazione del dimensionamento. I numeri di capacità mostrati in questo esempio consentono a ciascuna istanza Serverless v2 nel cluster di scalare tra 2 e 32 unità di capacità Aurora (ACU). Per informazioni sulla funzionalità di Aurora Serverless v2 e su come scegliere l'intervallo di capacità, consulta Uso di Aurora Serverless v2.

```
aws rds restore-db-cluster-to-point-in-time \
  --source-db-cluster-identifier cluster-with-2-subnets \
  --db-cluster-identifier cluster-cloned-to-3-subnets \
  --restore-type copy-on-write --use-latest-restorable-time \
  --db-subnet-group-name three-subnets \
  --serverless-v2-scaling-configuration 'MinCapacity=2,MaxCapacity=32'
```

Indipendentemente dal numero di sottoreti utilizzate dalle istanze database, l'archiviazione Aurora per il cluster di origine e il clone è associata a tre AZ. L'esempio seguente elenca le AZ associate sia al cluster originale che al clone, per entrambi i comandi `restore-db-cluster-to-point-in-time` degli esempi precedenti.

```
aws rds describe-db-clusters --db-cluster-identifier cluster-with-3-subnets \
  --query 'sort_by(*[].AvailabilityZones[].{Zone:@},&Zone)' --output text

us-east-1c
us-east-1d
us-east-1f

aws rds describe-db-clusters --db-cluster-identifier cluster-cloned-to-2-subnets \
  --query 'sort_by(*[].AvailabilityZones[].{Zone:@},&Zone)' --output text

us-east-1c
us-east-1d
us-east-1f

aws rds describe-db-clusters --db-cluster-identifier cluster-with-2-subnets \
  --query 'sort_by(*[].AvailabilityZones[].{Zone:@},&Zone)' --output text

us-east-1a
us-east-1c
us-east-1d

aws rds describe-db-clusters --db-cluster-identifier cluster-cloned-to-3-subnets \
  --query 'sort_by(*[].AvailabilityZones[].{Zone:@},&Zone)' --output text
```

```
us-east-1a
us-east-1c
us-east-1d
```

Poiché almeno due AZ si sovrappongono tra ogni coppia di cluster originali e cloni, entrambi i cluster possono accedere alla stessa archiviazione Aurora sottostante.

Fase 2: specificare le impostazioni di rete per le istanze nel clone

Quando crei istanze database nel clone, per impostazione predefinita queste ereditano il gruppo di sottoreti di database dal cluster stesso. In questo modo, Aurora assegna automaticamente ogni istanza a una specifica sottorete e la crea nella AZ associata alla sottorete. Questa scelta è comoda, soprattutto per i sistemi di sviluppo e test, perché non è necessario tenere traccia degli ID di sottorete o delle AZ mentre aggiungi nuove istanze al clone.

In alternativa, puoi specificare la AZ quando crei un'istanza database Aurora per il clone. La AZ che specifichi deve appartenere al set di AZ associate al clone. Se il gruppo di sottoreti di database che utilizzi per il clone contiene solo due sottoreti, puoi scegliere solo tra le AZ associate a queste due sottoreti. Questa scelta offre flessibilità e resilienza per sistemi ad alta disponibilità, perché puoi assicurarti che l'istanza di scrittura e l'istanza di lettura in standby si trovino in AZ diverse. Altrimenti, se aggiungi altre istanze di lettura al cluster, puoi assicurarti che siano distribuite su tre AZ. In questo modo, anche nell'improbabile eventualità di un problema con una AZ, avrai comunque un'istanza di scrittura e un'altra istanza di lettura in altre due AZ.

L'esempio seguente aggiunge un'istanza database con provisioning a un cluster Aurora PostgreSQL clonato che utilizza un gruppo di sottoreti di database personalizzato.

```
aws rds create-db-instance --db-cluster-identifier my_aurora_postgresql_clone \
  --db-instance-identifier my_postgres_instance \
  --db-subnet-group-name my_new_subnet \
  --engine aurora-postgresql \
  --db-instance-class db.t4g.medium
```

L'esempio seguente mostra l'output parziale del comando.

```
{
  'DBInstanceIdentifier': 'my_postgres_instance',
  'DBClusterIdentifier': 'my_aurora_postgresql_clone',
  'DBInstanceClass': 'db.t4g.medium',
  'DBInstanceStatus': 'creating'
```

```
  ...
}
```

L'esempio seguente aggiunge un'istanza database Aurora Serverless v2 a un clone Aurora MySQL che utilizza un gruppo di sottoreti di database personalizzato. Per poter utilizzare le istanze Serverless v2, assicurati di specificare l'opzione `--serverless-v2-scaling-configuration` per il comando `restore-db-cluster-to-point-in-time`, come mostrato negli esempi precedenti.

```
aws rds create-db-instance --db-cluster-identifier my_aurora_mysql_clone \
  --db-instance-identifier my_mysql_instance \
  --db-subnet-group-name my_other_new_subnet \
  --engine aurora-mysql \
  --db-instance-class db.serverless
```

L'esempio seguente mostra l'output parziale del comando.

```
{
  'DBInstanceIdentifier': 'my_mysql_instance',
  'DBClusterIdentifier': 'my_aurora_mysql_clone',
  'DBInstanceClass': 'db.serverless',
  'DBInstanceStatus': 'creating'
  ...
}
```

Fase 3: stabilire la connettività da un sistema client a un clone

Se ti stai già connettendo a un cluster Aurora da un sistema client, potresti voler consentire lo stesso tipo di connettività a un nuovo clone. Ad esempio, potresti connetterti al cluster originale da un'istanza Amazon Cloud9 o da un'istanza EC2. Per consentire connessioni dagli stessi sistemi client o dai sistemi nuovi che hai creato nel VPC di destinazione, devi configurare gruppi di sottoreti di database e gruppi di sicurezza VPC equivalenti a quelli del VPC. Specifica quindi il gruppo di sottoreti e i gruppi di sicurezza quando crei il clone.

I seguenti esempi mostrano la configurazione di un clone Aurora Serverless v2. Tale configurazione si basa sulla combinazione di `--engine-mode provisioned` e `--serverless-v2-scaling-configuration` durante la creazione del cluster di database e quindi di `--db-instance-class db.serverless` durante la creazione di ogni istanza database nel cluster. La modalità motore `provisioned` è l'impostazione predefinita, quindi puoi omettere questa opzione se preferisci.

```
aws rds restore-db-cluster-to-point-in-time \
  --source-db-cluster-identifier serverless-sql-postgres\
  --db-cluster-identifier serverless-sql-postgres-clone \
  --db-subnet-group-name 'default-vpc-1234' \
  --vpc-security-group-ids 'sg-4567' \
  --serverless-v2-scaling-configuration 'MinCapacity=0.5,MaxCapacity=16' \
  --restore-type copy-on-write \
  --use-latest-restorable-time
```

Quando crei le istanze database nel clone, specifica la stessa opzione `--db-subnet-group-name`. Facoltativamente, puoi includere l'opzione `--availability-zone` e specificare una delle AZ associate alle sottoreti nel gruppo di sottoreti in questione. Tale AZ deve inoltre essere una delle AZ associate al cluster originale.

```
aws rds create-db-instance \
  --db-cluster-identifier serverless-sql-postgres-clone \
  --db-instance-identifier serverless-sql-postgres-clone-instance \
  --db-instance-class db.serverless \
  --db-subnet-group-name 'default-vpc-987zyx654' \
  --availability-zone 'us-east-1c' \
  --engine aurora-postgresql
```

Spostamento di un cluster da sottoreti pubbliche a sottoreti private

Puoi utilizzare la clonazione per eseguire la migrazione di un cluster tra sottoreti pubbliche e private. Puoi eseguire questa operazione aggiungendo ulteriori livelli di sicurezza all'applicazione prima di distribuirla in produzione. Per questo esempio, dovresti aver già configurato le sottoreti private e il gateway NAT prima di iniziare il processo di clonazione con Aurora.

Per le fasi che coinvolgono Aurora, puoi seguire la stessa procedura generica degli esempi precedenti per Raccolta delle informazioni sull'ambiente di rete e Creazione di un clone Aurora con nuove impostazioni di rete. La differenza principale è che, anche se disponi di sottoreti pubbliche mappate a tutte le AZ del cluster originale, ora devi verificare di disporre di un numero sufficiente di sottoreti private per un cluster Aurora e che tali sottoreti siano associate alle stesse AZ utilizzate per l'archiviazione Aurora nel cluster originale. Analogamente ad altri casi d'uso di clonazione, puoi creare il gruppo di sottoreti di database per il clone con tre o due sottoreti private associate alle AZ richieste. Tuttavia, se utilizzi due sottoreti private nel gruppo di sottoreti di database, devi disporre di una terza sottorete privata associata alla terza AZ utilizzata per l'archiviazione Aurora nel cluster originale.

Puoi consultare questa lista di controllo per verificare che siano soddisfatti tutti i requisiti per eseguire questo tipo di operazione di clonazione.

- Registra le tre AZ che sono associate al cluster originale. Per istruzioni, consultare Fase 1: verificare le zone di disponibilità del cluster originale.
- Registra le tre o due AZ associate alle sottoreti pubbliche nel gruppo di sottoreti di database per il cluster originale. Per istruzioni, consultare Fase 3: verificare le sottoreti del cluster originale.
- Crea sottoreti private mappate a tutte e tre le AZ che sono associate al cluster originale. Esegui anche altre operazioni di configurazione di rete, ad esempio la creazione di un gateway NAT, per poter comunicare con le sottoreti private. Per istruzioni, consulta Creazione di una sottorete nella Guida per l'utente di Amazon Virtual Private Cloud.
- Crea un nuovo gruppo di sottoreti di database contenente tre o due delle sottoreti private che sono state associate alle AZ durante la prima fase. Per istruzioni, consultare Fase 2: creare il gruppo di sottoreti di database per il clone.

Quando tutti i prerequisiti sono soddisfatti, puoi sospendere l'attività del database sul cluster originale mentre crei il clone e cambi l'applicazione per utilizzarlo. Dopo aver creato il clone, verificato di poterti connettere a esso, eseguito il codice dell'applicazione e così via, puoi interrompere l'utilizzo del cluster originale.

Esempio end-to-end di creazione di un clone tra VPC

La creazione di un clone in un VPC diverso da quello originale avviene attraverso la stessa procedura generica mostrata negli esempi precedenti. Poiché l'ID VPC è una proprietà delle sottoreti, in realtà non specifichi l'ID VPC come parametro quando esegui uno dei comandi dell'interfaccia CLI di RDS. La differenza principale è che è più probabile che sia necessario creare nuove sottoreti, nuove sottoreti mappate ad AZ specifiche, un gruppo di sicurezza VPC e un nuovo gruppo di sottoreti di database. Questo è particolarmente vero se si tratta del primo cluster Aurora creato in quel VPC.

Puoi consultare questa lista di controllo per verificare che siano soddisfatti tutti i requisiti per eseguire questo tipo di operazione di clonazione.

- Registra le tre AZ che sono associate al cluster originale. Per istruzioni, consultare Fase 1: verificare le zone di disponibilità del cluster originale.
- Registra le tre o due AZ associate alle sottoreti nel gruppo di sottoreti di database per il cluster originale. Per istruzioni, consultare Fase 2: verificare il gruppo di sottoreti di database del cluster originale.

- Crea sottoreti mappate a tutte e tre le AZ che sono associate al cluster originale. Per istruzioni, consultare Fase 1: creare le sottoreti per il clone.
- Esegui altre operazioni di configurazione di rete, ad esempio la configurazione di un gruppo di sicurezza VPC, per sistemi client, server applicativi e così via per poter comunicare con le istanze database nel clone. Per istruzioni, consultare Controllo dell'accesso con i gruppi di sicurezza.
- Crea un nuovo gruppo di sottoreti di database contenente tre o due delle sottoreti che sono state associate alle AZ durante la prima fase. Per istruzioni, consultare Fase 2: creare il gruppo di sottoreti di database per il clone.

Quando tutti i prerequisiti sono soddisfatti, puoi sospendere l'attività del database sul cluster originale mentre crei il clone e cambi l'applicazione per utilizzarlo. Dopo aver creato il clone, verificato di poterti connettere a esso, eseguito il codice dell'applicazione e così via, puoi valutare se mantenere attivi sia l'originale che i cloni oppure interrompere l'utilizzo del cluster originale.

I seguenti esempi Linux mostrano la sequenza di operazioni AWS dell'interfaccia CLI per clonare un cluster di database Aurora da un VPC a un altro. Alcuni campi non rilevanti per gli esempi non vengono visualizzati nell'output del comando.

Innanzitutto, occorre controllare gli ID dei VPC di origine e di destinazione. Il nome descrittivo che assegni a un VPC quando lo crei è rappresentato come tag nei metadati del VPC.

```
$ aws ec2 describe-vpcs --query '*[].[VpcId,Tags]'
[
    [
        'vpc-0f0c0fc0000b0ffb0',
        [
            {
                'Key': 'Name',
                'Value': 'clone-vpc-source'
            }
        ]
    ],
    [
        'vpc-9e99d9f99a999bd99',
        [
            {
                'Key': 'Name',
                'Value': 'clone-vpc-dest'
            }
        ]
```

```
    ]
]
```

Il cluster originale esiste già nel VPC di origine. Per configurare il clone utilizzando lo stesso set di AZ per l'archiviazione Aurora, è necessario controllare le AZ utilizzate dal cluster originale.

```
$ aws rds describe-db-clusters --db-cluster-identifier original-cluster \
  --query 'sort_by(*[].AvailabilityZones[].{Zone:@},&Zone)' --output text

us-east-1c
us-east-1d
us-east-1f
```

Occorre assicurarsi che esistano sottoreti che corrispondono alle AZ utilizzate dal cluster originale: `us-east-1c`, `us-east-1d` e `us-east-1f`.

```
$ aws ec2 create-subnet --vpc-id vpc-9e99d9f99a999bd99 \
  --availability-zone us-east-1c --cidr-block 10.0.0.128/28
{
    'Subnet': {
        'AvailabilityZone': 'us-east-1c',
        'SubnetId': 'subnet-3333a33be3ef3e333',
        'VpcId': 'vpc-9e99d9f99a999bd99',
    }
}

$ aws ec2 create-subnet --vpc-id vpc-9e99d9f99a999bd99 \
--availability-zone us-east-1d --cidr-block 10.0.0.160/28
{
    'Subnet': {
        'AvailabilityZone': 'us-east-1d',
        'SubnetId': 'subnet-4eeb444cd44b4d444',
        'VpcId': 'vpc-9e99d9f99a999bd99',
    }
}

$ aws ec2 create-subnet --vpc-id vpc-9e99d9f99a999bd99 \
--availability-zone us-east-1f --cidr-block 10.0.0.224/28
{
    'Subnet': {
        'AvailabilityZone': 'us-east-1f',
        'SubnetId': 'subnet-66eea6666fb66d66c',
```

```
        'VpcId': 'vpc-9e99d9f99a999bd99',
    }
}
```

In questo esempio viene confermato che esistono sottoreti mappate alle AZ necessarie nel VPC di destinazione.

```
aws ec2 describe-subnets --query 'sort_by(*[] | [?VpcId == `vpc-9e99d9f99a999bd99`] |
[].{SubnetId:SubnetId,VpcId:VpcId,AvailabilityZone:AvailabilityZone},
 &AvailabilityZone)' --output table

-------------------------------------------------------------------------
|                            DescribeSubnets                            |
+------------------+---------------------------+------------------------+
| AvailabilityZone |         SubnetId          |          VpcId         |
+------------------+---------------------------+------------------------+
|  us-east-1a      |  subnet-000ff0e00000c0aea  |  vpc-9e99d9f99a999bd99  |
|  us-east-1b      |  subnet-1111d111111ca11b1  |  vpc-9e99d9f99a999bd99  |
|  us-east-1c      | subnet-3333a33be3ef3e333   |  vpc-9e99d9f99a999bd99  |
|  us-east-1d      | subnet-4eeb444cd44b4d444   |  vpc-9e99d9f99a999bd99  |
|  us-east-1f      | subnet-66eea6666fb66d66c   |  vpc-9e99d9f99a999bd99  |
+------------------+---------------------------+------------------------+
```

Prima di creare un cluster di database Aurora nel VPC, devi disporre di un gruppo di sottoreti di database con sottoreti mappate alle AZ utilizzate per l'archiviazione Aurora. Quando crei un cluster standard, puoi utilizzare qualsiasi set di tre AZ. Quando esegui la clonazione di un cluster esistente, il gruppo di sottoreti deve corrispondere ad almeno due delle tre AZ utilizzate per l'archiviazione Aurora.

```
$ aws rds create-db-subnet-group \
  --db-subnet-group-name subnet-group-in-other-vpc \
  --subnet-ids
 '["subnet-3333a33be3ef3e333","subnet-4eeb444cd44b4d444","subnet-66eea6666fb66d66c"]' \
  --db-subnet-group-description 'DB subnet group with 3 subnets:
 subnet-3333a33be3ef3e333,subnet-4eeb444cd44b4d444,subnet-66eea6666fb66d66c'

{
    'DBSubnetGroup': {
        'DBSubnetGroupName': 'subnet-group-in-other-vpc',
        'DBSubnetGroupDescription': 'DB subnet group with 3 subnets:
 subnet-3333a33be3ef3e333,subnet-4eeb444cd44b4d444,subnet-66eea6666fb66d66c',
        'VpcId': 'vpc-9e99d9f99a999bd99',
```

```
        'SubnetGroupStatus': 'Complete',
        'Subnets': [
            {
                'SubnetIdentifier': 'subnet-4eeb444cd44b4d444',
                'SubnetAvailabilityZone': { 'Name': 'us-east-1d' }
            },
            {
                'SubnetIdentifier': 'subnet-3333a33be3ef3e333',
                'SubnetAvailabilityZone': { 'Name': 'us-east-1c' }
            },
            {
                'SubnetIdentifier': 'subnet-66eea6666fb66d66c',
                'SubnetAvailabilityZone': { 'Name': 'us-east-1f' }
            }
        ]
    }
}
```

Ora le sottoreti e il gruppo di sottoreti di database sono predisposti. L'esempio seguente mostra il `restore-db-cluster-to-point-in-time` che clona il cluster. L'opzione `--db-subnet-group-name` associa il clone al set corretto di sottoreti mappate al set corretto di AZ del cluster originale.

```
$ aws rds restore-db-cluster-to-point-in-time \
  --source-db-cluster-identifier original-cluster \
  --db-cluster-identifier clone-in-other-vpc \
  --restore-type copy-on-write --use-latest-restorable-time \
  --db-subnet-group-name subnet-group-in-other-vpc

{
  'DBClusterIdentifier': 'clone-in-other-vpc',
  'DBSubnetGroup': 'subnet-group-in-other-vpc',
  'Engine': 'aurora-postgresql',
  'EngineVersion': '15.4',
  'Status': 'creating',
  'Endpoint': 'clone-in-other-vpc.cluster-c0abcdef.us-east-1.rds.amazonaws.com'
}
```

Nell'esempio seguente viene confermato che l'archiviazione Aurora nel clone utilizza lo stesso set di AZ del cluster originale.

```
$ aws rds describe-db-clusters --db-cluster-identifier clone-in-other-vpc \
```

```
  --query 'sort_by(*[].AvailabilityZones[].{Zone:@},&Zone)' --output text

us-east-1c
us-east-1d
us-east-1f
```

A questo punto, puoi creare le istanze database per il clone. Assicurati che il gruppo di sicurezza VPC associato a ciascuna istanza consenta le connessioni dagli intervalli di indirizzi IP che utilizzi per le istanze EC2, i server applicativi e così via che si trovano nel VPC di destinazione.

Clonazione tra account con AWS RAM e Amazon Aurora

Utilizzando AWS Resource Access Manager (AWS RAM) con Amazon Aurora, puoi condividere i cluster e i cloni di Aurora DB che appartengono al tuo AWS account con un altro account o organizzazione. AWS La clonazione tra più account è più veloce in queste situazioni piuttosto che con la creazione e il ripristino di uno snapshot di un database. Puoi creare un clone di uno dei tuoi cluster database Aurora e condividerlo. Oppure puoi condividere il tuo cluster Aurora DB con un altro AWS account e consentire al titolare dell'account di creare il clone. L'approccio scelto varia a seconda del caso d'uso.

Ad esempio, potrebbe essere necessario condividere regolarmente un clone del database finanziario con il team di verifica interno dell'organizzazione. In questo caso, il team di verifica utilizza il suo account AWS per le applicazioni. Puoi concedere all'AWSaccount del team di controllo l'autorizzazione ad accedere al tuo cluster Aurora DB e clonarlo secondo necessità.

D'altra parte, se un fornitore esterno verifica i tuoi dati finanziari, potresti preferire creare il clone da solo. È quindi possibile concedere al fornitore esterno solo l'accesso al clone.

Puoi anche utilizzare la clonazione tra account per supportare molti degli stessi casi d'uso per la clonazione all'interno dello stesso AWS account, ad esempio sviluppo e test. Ad esempio, l'organizzazione potrebbe utilizzare AWS account diversi per la produzione, lo sviluppo, i test e così via. Per ulteriori informazioni, consulta Panoramica sulla clonazione Aurora.

Pertanto, potresti voler condividere un clone con un altro AWS account o consentire a un altro AWS account di creare cloni dei tuoi cluster Aurora DB. In entrambi i casi, iniziate a usare per creare un oggetto AWS RAM di condivisione. Per informazioni complete sulla condivisione di AWS risorse tra AWS account, consulta la Guida AWS RAM per l'utente.

La creazione di un clone tra più account richiede azioni da parte dell'AWSaccount proprietario del cluster originale e dell'AWSaccount che crea il clone. Innanzitutto, il proprietario modifica il

cluster per permettere a uno o più account di poterlo clonare. Se uno degli account appartiene a un'AWSorganizzazione diversa, AWS genera un invito alla condivisione. L'altro account deve accettare l'invito prima di procedere. Dunque, ogni account autorizzato può clonare il cluster. Durante questo processo, il cluster è definito dal suo unico Amazon Resource Name (ARN).

Come per la clonazione all'interno dello stesso AWS account, lo spazio di archiviazione aggiuntivo viene utilizzato solo se vengono apportate modifiche ai dati dall'origine o dal clone. Gli addebiti per l'archiviazione vengono quindi applicati da quel momento. Se si elimina il cluster di origine, i costi di storage sono distribuiti equamente tra i cluster clonati rimanenti.

Argomenti

- Restrizioni della clonazione tra più account
- Consentire ad altri AWS account di clonare il cluster
- Clonazione di un cluster di proprietà di un altro account AWS

Restrizioni della clonazione tra più account

La clonazione tra più account di Aurora presenta le restrizioni seguenti:

- Non puoi clonare un Aurora Serverless v1 cluster su più account. AWS
- Non puoi visualizzare o accettare inviti a risorse condivise con. Console di gestione AWS Usa l'AWS CLIAPI Amazon RDS o la AWS RAM console per visualizzare e accettare inviti a risorse condivise.
- Puoi creare solo un nuovo clone da una risorsa condivisa con le tue. Account AWS Questo vale sia che la risorsa condivisa sia un cluster di database Aurora originale o un clone creato in precedenza.
- Puoi creare solo un nuovo clone da un clone che è stato condiviso con il tuo account. AWS
- Non puoi condividere risorse (cloni o cluster Aurora DB) che sono state condivise con il tuo account. AWS
- Non è possibile creare più di 15 cloni tra account da un singolo cluster di database Aurora.
- Ciascuno dei 15 cloni di più account deve appartenere a un account diverso. AWS In altre parole, è possibile creare solo un clone multiaccount di un cluster all'interno di qualsiasi account. AWS
- Dopo la clonazione, il cluster originale e il relativo clone sono considerati identici ai fini dell'applicazione dei limiti sui cloni tra account. Non è possibile creare cloni tra più account sia del cluster originale che del cluster clonato all'interno dello stesso account. AWS Il numero totale di cloni tra account per il cluster originale e uno dei suoi cloni non può superare 15.

- Non puoi condividere un cluster Aurora DB con altri AWS account a meno che il cluster non si trovi in uno `ACTIVE` stato.
- Non è possibile rinominare un cluster Aurora DB condiviso con AWS altri account.
- Non puoi creare un clone tra più account di un cluster crittografato con una chiave RDS predefinita.
- Non è possibile creare cloni non crittografati in un AWS account da cluster Aurora DB crittografati che sono stati condivisi da un altro account. AWS Il proprietario del cluster deve inoltre concedere l'autorizzazione per accedere alla AWS KMS key del cluster di origine. Puoi tuttavia utilizzare una chiave differente quando crei il clone.

Consentire ad altri AWS account di clonare il cluster

Per consentire ad altri AWS account di clonare un cluster di tua proprietà, utilizza AWS RAM per impostare l'autorizzazione di condivisione. In questo modo viene inoltre inviato un invito a ciascuno degli altri account che fanno parte di un'AWSorganizzazione diversa.

Per le procedure per condividere le risorse di tua proprietà nella AWS RAM console, consulta Condivisione delle risorse di tua proprietà nella Guida per l'AWS RAMutente.

Argomenti

- Concessione dell'autorizzazione ad altri AWS account per clonare il cluster
- Verifica se un cluster di tua proprietà è condiviso con altri account AWS

Concessione dell'autorizzazione ad altri AWS account per clonare il cluster

Se il cluster che condividi è crittografato, condividi anche la AWS KMS key per il cluster. Puoi consentire agli utenti o ai ruoli AWS Identity and Access Management (IAM) di un AWS account di utilizzare una chiave KMS in un account diverso.

Per fare ciò, devi prima aggiungere l'account esterno (utente root) alla politica chiave della chiave KMS tramite. AWS KMS Non aggiungi singoli utenti o ruoli alla policy delle chiavi, ma solo l'account proprietario esterno. Puoi condividere solamente una chiave KMS che hai creato tu stesso, non la chiave del servizio RDS di default. Per ulteriori informazioni sul controllo accessi per le chiavi KMS, consulta Autenticazione e controllo degli accessi per AWS KMS.

Console

Come concedere l'autorizzazione per clonare il tuo cluster

1. Accedi a Console di gestione AWS e apri la console Amazon RDS all'indirizzo https://console.aws.amazon.com/rds/.
2. Nel pannello di navigazione, seleziona Database.
3. Scegli il cluster database che desideri condividere per visualizzare la pagina Dettagli e scegli la scheda Connectivity & security (Connessione e sicurezza).
4. Nella sezione Condividi il cluster DB con altri AWS account, inserisci l'ID numerico dell'account a cui desideri consentire la AWS clonazione di questo cluster. Per un account IDs appartenente alla stessa organizzazione, puoi iniziare a digitare nella casella e quindi scegliere dal menu.

   > **Important**
   >
   > In alcuni casi, potresti volere la clonazione di un cluster da parte di un account che non appartiene alla stessa organizzazione AWS del tuo account. In questi casi, per ragioni di sicurezza la console non segnala chi sia il proprietario dell'ID account o se l'account esiste.
   > Fai attenzione a inserire numeri di conto che non appartengono alla stessa AWS organizzazione del tuo AWS account. Verifica immediatamente che la condivisione sia avvenuta con l'account desiderato.

5. Nella pagina di conferma, verifica che l'ID dell'account specificato sia corretto. Inserisci `share` nel riquadro di conferma per confermare.

   Nella pagina Dettagli, viene visualizzata una voce che mostra l'ID dell'AWSaccount specificato in Account con cui è condiviso questo cluster DB. La colonna Stato mostra inizialmente lo stato Pending (In attesa).
6. Contatta il proprietario dell'altro AWS account o accedi a quell'account se possiedi entrambi. Fornisci istruzioni al proprietario dell'altro account per accettare l'invito di condivisione e clonare il cluster database, nel modo descritto di seguito.

AWS CLI

Come concedere l'autorizzazione per clonare il tuo cluster

1. Raccogli le informazioni per i parametri richiesti. È necessario l'ARN per il cluster e l'ID numerico per l'altro account. AWS
2. Esegui il comando AWS RAM CLI. `create-resource-share`

   Per Linux, macOS o Unix:

   ```
   aws ram create-resource-share --name descriptive_name \
     --region region \
     --resource-arns cluster_arn \
     --principals other_account_ids
   ```

   Per Windows:

   ```
   aws ram create-resource-share --name descriptive_name ^
     --region region ^
     --resource-arns cluster_arn ^
     --principals other_account_ids
   ```

   Per includere più account IDs per il `--principals` parametro, separali l'uno IDs dall'altro con spazi. Per specificare se l'account consentito IDs può trovarsi all'esterno AWS dell'organizzazione, includi il `--no-allow-external-principals` parametro `--allow-external-principals` o per`create-resource-share`.

AWS RAMAPI

Come concedere l'autorizzazione per clonare il tuo cluster

1. Raccogli le informazioni per i parametri richiesti. È necessario l'ARN per il cluster e l'ID numerico per l'altro account. AWS
2. Chiama l'operazione AWS RAM API CreateResourceSharee specifica i seguenti valori:

   - Specificate l'ID dell'account per uno o più AWS account come `principals` parametro.
   - Specifica l'ARN di uno o più cluster database Aurora come parametro `resourceArns`.

- Specificate se l'account consentito IDs può essere esterno all'AWSorganizzazione includendo un valore booleano per il `allowExternalPrincipals` parametro.

Nuova creazione di un cluster che usa la chiave RDS predefinita

Se il cluster crittografato che desideri condividere utilizza la chiave RDS predefinita, assicurati di creare nuovamente il cluster. A tale scopo, crea uno snapshot manuale del cluster DB, utilizza una AWS KMS key e quindi ripristina il cluster in un nuovo cluster. Quindi condividi il nuovo cluster. Per eseguire questo processo, procedi come indicato di seguito.

Ricreare un cluster crittografato che usa una chiave RDS predefinita

1. Accedi a Console di gestione AWS e apri la console Amazon RDS all'indirizzo https://console.aws.amazon.com/rds/.
2. Scegli Snapshots (Snapshot) dal riquadro di navigazione.
3. Scegli il tuo snapshot.
4. Sotto Operazioni, scegli Copy Snapshot (Copia snapshot) e poi Enable encryption (Abilita crittografia).
5. Per AWS KMS key, scegli la nuova chiave di crittografia che desideri usare.
6. Ripristina lo snapshot copiato. A tale scopo, segui la procedura in Ripristino da uno snapshot cluster database. La nuova istanza database usa la nuova chiave di crittografia.
7. (Opzionale) Elimina il vecchio cluster database se non ne hai più bisogno. A tale scopo, segui la procedura in Eliminazione di una snapshot del cluster di database. Prima di farlo, conferma che il tuo nuovo cluster abbia tutti i dati necessari e che la tua applicazione può accedere correttamente.

Verifica se un cluster di tua proprietà è condiviso con altri account AWS

Puoi controllare se altri utenti hanno autorizzazione a condividere un cluster. Farlo ti aiuta a capire se il cluster si sta avvicinando al limite massimo di numeri di cloni tra più account.

Per le procedure per condividere le risorse utilizzando la AWS RAM console, consulta Condivisione delle risorse di tua proprietà nella Guida per l'AWS RAMutente.

AWS CLI

Per scoprire se un cluster di tua proprietà è condiviso con altri AWS account

- Chiama il comando AWS RAM CLI `list-principals`, utilizzando l'ID dell'account come proprietario della risorsa e l'ARN del cluster come ARN della risorsa. Puoi visualizzare tutte le condivisioni con il comando seguente. I risultati indicano a quali AWS account è consentito clonare il cluster.

```
aws ram list-principals \
    --resource-arns your_cluster_arn \
    --principals your_aws_id
```

AWS RAMAPI

Per scoprire se un cluster di tua proprietà è condiviso con altri AWS account

- Chiama l'operazione AWS RAM API ListPrincipals. Usa il tuo ID dell'account come proprietario di risorse e l'ARN del tuo cluster come l'ARN di risorse.

Clonazione di un cluster di proprietà di un altro account AWS

Per clonare un cluster di proprietà di un altro AWS account, usa AWS RAM per ottenere l'autorizzazione a creare il clone. Dopo aver ottenuto l'autorizzazione richiesta, puoi usare la procedura standard per clonare un cluster Aurora.

Puoi anche verificare se un cluster di tua proprietà è un clone di un cluster di proprietà di un account diverso. AWS

Per le procedure di utilizzo delle risorse di proprietà di altri utenti nella AWS RAM console, vedi Accesso alle risorse condivise con te nella Guida per l'AWS RAMutente.

Argomenti

- Visualizzazione degli inviti a clonare i cluster di proprietà di altri account AWS
- Accettare inviti a condividere cluster di proprietà di altri account AWS
- Clonazione di un cluster Aurora di proprietà di un altro account AWS
- Verifica se un cluster database è un clone tra più account

Visualizzazione degli inviti a clonare i cluster di proprietà di altri account AWS

Per gestire gli inviti a clonare i cluster di proprietà di AWS account di altre AWS organizzazioni, usa laAWS CLI, la console o l'AWS RAMAPI. AWS RAM Al momento, non puoi eseguire questa procedura utilizzando la console Amazon RDS.

Per le procedure di utilizzo degli inviti nella AWS RAM console, consulta Accesso alle risorse condivise con te nella Guida per l'utente. AWS RAM

AWS CLI

Per visualizzare gli inviti a clonare i cluster di proprietà di altri account AWS

1. Esegui il comando AWS RAM CLI. `get-resource-share-invitations`

   ```
   aws ram get-resource-share-invitations --region region_name
   ```

   I risultati del comando precedente mostrano tutti gli inviti per clonare cluster, inclusi gli inviti già accettati o rifiutati.
2. (Opzionale) Filtra l'elenco per vedere solamente gli inviti che necessitano una tua azione. A tale scopo, aggiungi il parametro `--query 'resourceShareInvitations[?status==`PENDING`]'`.

AWS RAMAPI

Per visualizzare gli inviti a clonare i cluster di proprietà di altri account AWS

1. Chiama l'operazione APIAWS RAM. `GetResourceShareInvitations` Questa operazione restituisce tutti questi inviti, inclusi gli inviti già accettati o rifiutati.
2. (Opzionale) Trova solamente gli inviti che necessitano una tua azione verificando il campo di restituzione `resourceShareAssociations` per un valore `status` di `PENDING`.

Accettare inviti a condividere cluster di proprietà di altri account AWS

Puoi accettare inviti a condividere cluster di proprietà di altri AWS account appartenenti a organizzazioni diverse. AWS Per lavorare con questi inviti, usa ilAWS CLI, the AWS RAM e RDS APIs o la console. AWS RAM Al momento, non puoi eseguire questa procedura utilizzando la console RDS.

Per le procedure di utilizzo degli inviti nella AWS RAM console, vedi Accesso alle risorse condivise con te nella Guida per l'AWS RAMutente.

AWS CLI

Per accettare un invito a condividere un cluster da un altro account AWS

1. Trova l'ARN dell'invito eseguendo il comando AWS RAM CLI `get-resource-share-invitations`, come mostrato in precedenza.
2. Accettate l'invito chiamando il comando AWS RAM CLI `accept-resource-share-invitation`, come illustrato di seguito.

   Per Linux, macOS o Unix:

   ```
   aws ram accept-resource-share-invitation \
     --resource-share-invitation-arn invitation_arn \
     --region region
   ```

   Per Windows:

   ```
   aws ram accept-resource-share-invitation ^
     --resource-share-invitation-arn invitation_arn ^
     --region region
   ```

AWS RAMe API RDS

Come accettare inviti per condividere il cluster di qualcuno

1. Trova l'ARN dell'invito chiamando l'operazione AWS RAM API `GetResourceShareInvitations`, come mostrato in precedenza.
2. Passa quell'ARN come `resourceShareInvitationArn` parametro all'operazione API RDS. AcceptResourceShareInvitation

Clonazione di un cluster Aurora di proprietà di un altro account AWS

Dopo aver accettato l'invito dall'AWSaccount proprietario del cluster DB, come mostrato in precedenza, è possibile clonare il cluster.

Console

Per clonare un cluster Aurora di proprietà di un altro account AWS

1. Accedi a Console di gestione AWS e apri la console Amazon RDS all'indirizzo https://console.aws.amazon.com/rds/.
2. Nel pannello di navigazione, seleziona Database.

   In cima all'elenco database, dovresti visualizzare uno o più elementi con un valore Ruolo di `Shared from account #account_id`. Per ragioni di sicurezza, puoi visualizzare solamente informazioni limitate riguardo i cluster originali. Puoi visualizzare le proprietà, come motore di database e versione, che devono essere uguali nel tuo cluster clonato.
3. Scegli il cluster che desideri clonare.
4. In Actions (Operazioni), selezionare Create clone (Crea clone).
5. Segui la procedura in Console per completare la configurazione del cluster clonato.
6. Abilita la crittografia per il cluster clonato come necessario. Se il cluster che stai clonando è crittografato, devi abilitare la crittografia per il cluster clonato. L'account AWS che ha condiviso con te il cluster deve condividere anche la chiave KMS utilizzata per crittografare il cluster. È possibile utilizzare la stessa chiave KMS per crittografare il clone o la propria chiave KMS. Non puoi creare un clone tra più account per un cluster crittografato con una chiave KMS di default.

   L'account che possiede la chiave di crittografia deve concedere l'autorizzazione per utilizzare la chiave all'account di destinazione utilizzando la policy delle chiavi. Questo processo è simile a quello per condividere gli snapshot crittografati, utilizzando una policy delle chiavi per concedere l'autorizzazione all'account di destinazione per utilizzare la chiave.

AWS CLI

Per clonare un cluster Aurora di proprietà di un altro account AWS

1. Accetta l'invito dell'AWSaccount proprietario del cluster DB, come mostrato in precedenza.
2. Clona il cluster specificando l'ARN completo del cluster di origine nel parametro `source-db-cluster-identifier` del comando RDS CLI `restore-db-cluster-to-point-in-time`, come mostrato di seguito.

   Se l'ARN passato come `source-db-cluster-identifier` non è stato condiviso, si presenterà lo stesso errore del caso in cui il cluster indicato non esista.

Per Linux, macOS o Unix:

```
aws rds restore-db-cluster-to-point-in-time \
  --source-db-cluster-identifier=arn:aws:rds:arn_details \
  --db-cluster-identifier=new_cluster_id \
  --restore-type=copy-on-write \
  --use-latest-restorable-time
```

Per Windows:

```
aws rds restore-db-cluster-to-point-in-time ^
  --source-db-cluster-identifier=arn:aws:rds:arn_details ^
  --db-cluster-identifier=new_cluster_id ^
  --restore-type=copy-on-write ^
  --use-latest-restorable-time
```

3. Se il cluster da clonare è crittografato, devi crittografare il tuo cluster clonato includendo un parametro `kms-key-id`. Questo valore `kms-key-id` può essere lo stesso di quello utilizzato per crittografare il cluster database originale o in alternativa la tua chiave KMS. Il tuo account deve avere l'autorizzazione a utilizzare la chiave di crittografia.

   Per Linux, macOS o Unix:

   ```
   aws rds restore-db-cluster-to-point-in-time \
     --source-db-cluster-identifier=arn:aws:rds:arn_details \
     --db-cluster-identifier=new_cluster_id \
     --restore-type=copy-on-write \
     --use-latest-restorable-time \
     --kms-key-id=arn:aws:kms:arn_details
   ```

   Per Windows:

   ```
   aws rds restore-db-cluster-to-point-in-time ^
     --source-db-cluster-identifier=arn:aws:rds:arn_details ^
     --db-cluster-identifier=new_cluster_id ^
     --restore-type=copy-on-write ^
     --use-latest-restorable-time ^
     --kms-key-id=arn:aws:kms:arn_details
   ```

L'account che possiede la chiave di crittografia deve concedere l'autorizzazione per utilizzare la chiave all'account di destinazione utilizzando la policy delle chiavi. Questo processo è simile a quello per condividere gli snapshot crittografati, utilizzando una policy delle chiavi per concedere l'autorizzazione all'account di destinazione per utilizzare la chiave. Di seguito, un esempio di una policy delle chiavi.

JSON

```
{
    "Id": "key-policy-1",
    "Version":"2012-10-17",
    "Statement": [
        {
            "Sid": "Allow use of the key",
            "Effect": "Allow",
            "Principal": {
                "AWS": [
                    "arn:aws:iam::111122223333:user/KeyUser",
                    "arn:aws:iam::111122223333:root"
                ]
            },
            "Action": [
                "kms:CreateGrant",
                "kms:Encrypt",
                "kms:Decrypt",
                "kms:ReEncrypt*",
                "kms:GenerateDataKey*",
                "kms:DescribeKey"
            ],
            "Resource": "*"
        },
        {
            "Sid": "Allow attachment of persistent resources",
            "Effect": "Allow",
            "Principal": {
                "AWS": [
                    "arn:aws:iam::111122223333:user/KeyUser",
                    "arn:aws:iam::111122223333:root"
                ]
            },
            "Action": [
```

```
                "kms:CreateGrant",
                "kms:ListGrants",
                "kms:RevokeGrant"
            ],
            "Resource": "*",
            "Condition": {
                "Bool": {
                    "kms:GrantIsForAWSResource": true
                }
            }
        }
    ]
}
```

> **Note**
>
> Il comando restore-db-cluster-to- point-in-time AWS CLI ripristina solo il cluster DB, non le istanze DB per quel cluster DB. Per creare istanze DB per il cluster DB ripristinato, richiama il comando. create-db-instance Specifica l'identificatore del cluster database ripristinato in `--db-cluster-identifier`.
> Puoi creare le istanze database solo dopo che il comando `restore-db-cluster-to-point-in-time` è terminato e il cluster database è disponibile.

API RDS

Per clonare un cluster Aurora di proprietà di un altro account AWS

1. Accetta l'invito dell'AWSaccount proprietario del cluster DB, come mostrato in precedenza.
2. Clona il cluster specificando l'ARN completo del cluster di origine nel parametro `SourceDBClusterIdentifier` dell'operazione API RDS `RestoreDBClusterToPointInTime`.

   Se l'ARN passato come `SourceDBClusterIdentifier` non è stato condiviso, allora si presenterà lo stesso errore del caso in cui il cluster indicato non esista.
3. Se il cluster da clonare è crittografato, includi un parametro `KmsKeyId` per crittografare il tuo cluster clonato. Questo valore `kms-key-id` può essere lo stesso di quello utilizzato per crittografare il cluster database originale o in alternativa la tua chiave KMS. Il tuo account deve avere l'autorizzazione a utilizzare la chiave di crittografia.

Durante la clonazione di un volume, l'account di destinazione deve essere autorizzato all'utilizzo della chiave di crittografia usata per crittografare il cluster di origine. Aurora crittografa il nuovo cluster clonato con la chiave di crittografia specificata in `KmsKeyId`.

L'account che possiede la chiave di crittografia deve concedere l'autorizzazione per utilizzare la chiave all'account di destinazione utilizzando la policy delle chiavi. Questo processo è simile a quello per condividere gli snapshot crittografati, utilizzando una policy delle chiavi per concedere l'autorizzazione all'account di destinazione per utilizzare la chiave. Di seguito, un esempio di una policy delle chiavi.

JSON

```
{
    "Id": "key-policy-1",
    "Version":"2012-10-17",
    "Statement": [
        {
            "Sid": "Allow use of the key",
            "Effect": "Allow",
            "Principal": {
                "AWS": [
                    "arn:aws:iam::111122223333:user/KeyUser",
                    "arn:aws:iam::111122223333:root"
                ]
            },
            "Action": [
                "kms:CreateGrant",
                "kms:Encrypt",
                "kms:Decrypt",
                "kms:ReEncrypt*",
                "kms:GenerateDataKey*",
                "kms:DescribeKey"
            ],
            "Resource": "*"
        },
        {
            "Sid": "Allow attachment of persistent resources",
            "Effect": "Allow",
            "Principal": {
                "AWS": [
                    "arn:aws:iam::111122223333:user/KeyUser",
```

```
                    "arn:aws:iam::111122223333:root"
                ]
            },
            "Action": [
                "kms:CreateGrant",
                "kms:ListGrants",
                "kms:RevokeGrant"
            ],
            "Resource": "*",
            "Condition": {
                "Bool": {
                    "kms:GrantIsForAWSResource": true
                }
            }
        }
    ]
}
```

> **Note**
>
> L'operazione DBClusterToPointInTimeRestore RDS API ripristina solo il cluster DB, non le istanze DB per quel cluster di database. Per creare istanze DB per il cluster DB ripristinato, richiama l'operazione Create RDS API. DBInstance Specifica l'identificatore del cluster database ripristinato in `DBClusterIdentifier`. È possibile creare le istanze database solo dopo che l'operazione `RestoreDBClusterToPointInTime` è completata e il cluster di database è disponibile.

Verifica se un cluster database è un clone tra più account

L'oggetto `DBClusters` identifica se ciascun cluster è un clone tra più account. Puoi visualizzare il cluster che puoi clonare utilizzando l'opzione`include-shared` quando esegui il comando RDS CLI `describe-db-clusters`. Tuttavia, non puoi visualizzare la maggior parte dei dettagli di configurazione per tali cluster.

AWS CLI

Come verificare se un cluster database è un clone tra più account

- Denomina il comando RDS CLI `describe-db-clusters`.

I seguenti esempi mostrano come sono visualizzati i cluster database effettivi o potenziali di cloni tra più account nell'output `describe-db-clusters`. Per i cluster esistenti di proprietà del tuo AWS account, il `CrossAccountClone` campo indica se il cluster è un clone di un cluster DB di proprietà di un altro account. AWS

In alcuni casi, una voce potrebbe avere un numero di AWS account diverso dal tuo nel `DBClusterArn` campo. In questo caso, quella voce rappresenta un cluster di proprietà di un AWS account diverso e che puoi clonare. Tali voci presentano pochi campi oltre `DBClusterArn`. Quando crei un cluster clonato, specifica che i valori per `StorageEncrypted`, `Engine` e `EngineVersion` siano gli stessi del cluster originale.

```
$aws rds describe-db-clusters --include-shared --region us-east-1
{
  "DBClusters": [
      {
          "EarliestRestorableTime": "2023-02-01T21:17:54.106Z",
          "Engine": "aurora-mysql",
          "EngineVersion": "8.0.mysql_aurora.3.02.0",
          "CrossAccountClone": false,
...
      },
      {
          "EarliestRestorableTime": "2023-02-09T16:01:07.398Z",
          "Engine": "aurora-mysql",
          "EngineVersion": "8.0.mysql_aurora.3.02.0",
          "CrossAccountClone": true,
...
      },
      {
          "StorageEncrypted": false,
          "DBClusterArn": "arn:aws:rds:us-east-1:12345678:cluster:cluster-
abcdefgh",
          "Engine": "aurora-mysql",
          "EngineVersion": "8.0.mysql_aurora.3.02.0
  ]
}
```

API RDS

Come verificare se un cluster database è un clone tra più account

- Richiama l'operazione Descrivi dell'API RDS. DBClusters

  Per i cluster esistenti di proprietà del tuo AWS account, il `CrossAccountClone` campo indica se il cluster è un clone di un cluster DB di proprietà di un altro account. AWS Le voci con un numero di AWS account diverso nel `DBClusterArn` campo rappresentano cluster che è possibile clonare e che sono di proprietà di altri account. AWS Tali voci presentano pochi campi oltre `DBClusterArn`. Quando crei un cluster clonato, specifica che i valori per `StorageEncrypted`, `Engine` e `EngineVersion` siano gli stessi del cluster originale.

  Il seguente esempio mostra un valore di ritorno che dimostra sia i cluster clonati effettivi che potenziali.

```
{
  "DBClusters": [
      {
          "EarliestRestorableTime": "2023-02-01T21:17:54.106Z",
          "Engine": "aurora-mysql",
          "EngineVersion": "8.0.mysql_aurora.3.02.0",
          "CrossAccountClone": false,
...
      },
      {
          "EarliestRestorableTime": "2023-02-09T16:01:07.398Z",
          "Engine": "aurora-mysql",
          "EngineVersion": "8.0.mysql_aurora.3.02.0",
          "CrossAccountClone": true,
...
      },
      {
          "StorageEncrypted": false,
          "DBClusterArn": "arn:aws:rds:us-east-1:12345678:cluster:cluster-
abcdefgh",
          "Engine": "aurora-mysql",
          "EngineVersion": "8.0.mysql_aurora.3.02.0"
      }
  ]
}
```

Integrazione di Aurora con altri servizi AWS

Integra Amazon Aurora con altri AWS servizi in modo da poter estendere il tuo cluster Aurora DB per utilizzare funzionalità aggiuntive nel cloud. AWS

Argomenti

- Integrazione di AWS servizi con Amazon Aurora MySQL
- Integrazione di AWS servizi con Amazon Aurora PostgreSQL

Integrazione di AWS servizi con Amazon Aurora MySQL

Amazon Aurora MySQL si integra con altri AWS servizi in modo da poter estendere il cluster Aurora MySQL DB per utilizzare funzionalità aggiuntive nel cloud. AWS Il cluster Aurora MySQL DB può utilizzare AWS i servizi per eseguire le seguenti operazioni:

- Richiama una funzione in modo sincrono o asincrono utilizzando le funzioni native o. AWS Lambda `lambda_sync` `lambda_async` In alternativa, chiamata asincrona di una funzione AWS Lambda utilizzando la procedura `mysql.lambda_async`.
- Caricamento dei dati da file di testo o XML archiviati in un bucket Amazon S3 nel cluster DB utilizzando il comando `LOAD DATA FROM S3` o `LOAD XML FROM S3`.
- Salvataggio dei dati nei file di testo archiviati in un bucket Amazon S3 dal cluster DB utilizzando il comando `SELECT INTO OUTFILE S3`.
- Aggiunta o rimozione automatica di repliche Aurora con Application Auto Scaling. Per ulteriori informazioni, consulta Dimensionamento automatico di Amazon Aurora con le repliche Aurora.

Per ulteriori informazioni sull'integrazione di Aurora MySQL AWS con altri servizi, consulta. Integrazione di Amazon Aurora MySQL con altri servizi AWS

Integrazione di AWS servizi con Amazon Aurora PostgreSQL

Amazon Aurora PostgreSQL si integra con altri servizi in modo da AWS poter estendere il cluster Aurora PostgreSQL DB per utilizzare funzionalità aggiuntive nel cloud. AWS Il cluster Aurora PostgreSQL DB può utilizzare i servizi per eseguire le seguenti operazioni: AWS

- Raccolta, visualizzazione e valutazione delle prestazioni in modo rapido per i carichi di lavoro del database relazionale con Performance Insights.

- Aggiunta o rimozione automatica di repliche Aurora con Aurora Auto Scaling. Per ulteriori informazioni, consulta Dimensionamento automatico di Amazon Aurora con le repliche Aurora.

Per ulteriori informazioni sull'integrazione di Aurora AWS PostgreSQL con altri servizi, consulta. Integrazione di Amazon Aurora PostgreSQL con altri servizi AWS

Manutenzione di un cluster di database Amazon Aurora

Periodicamente, Amazon RDS esegue la manutenzione delle risorse Amazon RDS. I seguenti argomenti descrivono queste azioni di manutenzione e come applicarle.

Panoramica degli aggiornamenti di manutenzione dei cluster di database

La manutenzione spesso comporta aggiornamenti alle seguenti risorse del cluster database:

- Hardware sottostante
- Sistema operativo (OS) sottostante
- Versione del motore del database

Gli aggiornamenti al sistema operativo si verificano generalmente per problemi di sicurezza. Consigliamo di eseguirli il prima possibile. Per ulteriori informazioni sugli aggiornamenti del sistema operativo, consulta the section called “Aggiornamenti del sistema operativo”.

Argomenti

- Risorse offline durante gli aggiornamenti di manutenzione
- Modifiche differite dell’istanza database e del cluster di database
- Eventuale coerenza per l'API DescribePendingMaintenanceActions

Risorse offline durante gli aggiornamenti di manutenzione

Per alcune operazioni di manutenzione, Amazon RDS deve portare offline il cluster di database per un breve intervallo di tempo. Tra le operazioni di manutenzione che richiedono l’impostazione offline di una risorsa si annovera l’applicazione delle patch necessarie al sistema operativo o al database. L’applicazione delle patch necessarie viene pianificata automaticamente solo per le patch correlate alla sicurezza e all’affidabilità dell’istanza. Tali patch si verificano raramente, in genere una volta ogni pochi mesi. Raramente richiedono più di una frazione del periodo di manutenzione.

Modifiche differite dell’istanza database e del cluster di database

Le modifiche dell’istanza e del cluster di database differite che si è scelto di non applicare immediatamente vengono applicate durante la finestra di manutenzione. Ad esempio, è possibile scegliere di modificare le classi di istanza database o i gruppi di parametri cluster o database durante

la finestra di manutenzione. Le modifiche specificate utilizzando l'impostazione di riavvio in sospeso non vengono visualizzate nell'elenco Manutenzione in sospeso . Per ulteriori informazioni sulla modifica di un cluster database, consulta Modifica di un cluster database Amazon Aurora.

Per vedere le modifiche in sospeso per la prossima finestra di manutenzione, usa il describe-db-clusters AWS CLI comando e controlla il campo. `PendingModifiedValues`

Eventuale coerenza per l'API DescribePendingMaintenanceActions

L'API `DescribePendingMaintenanceActions` di Amazon RDS segue un modello di consistenza finale. Ciò significa che il risultato dell'esecuzione del comando `DescribePendingMaintenanceActions` potrebbe non essere immediatamente visibile a tutti i comandi successivi di RDS. Tienilo a mente quando utilizzi `DescribePendingMaintenanceActions` immediatamente dopo aver utilizzato un comando API precedente.

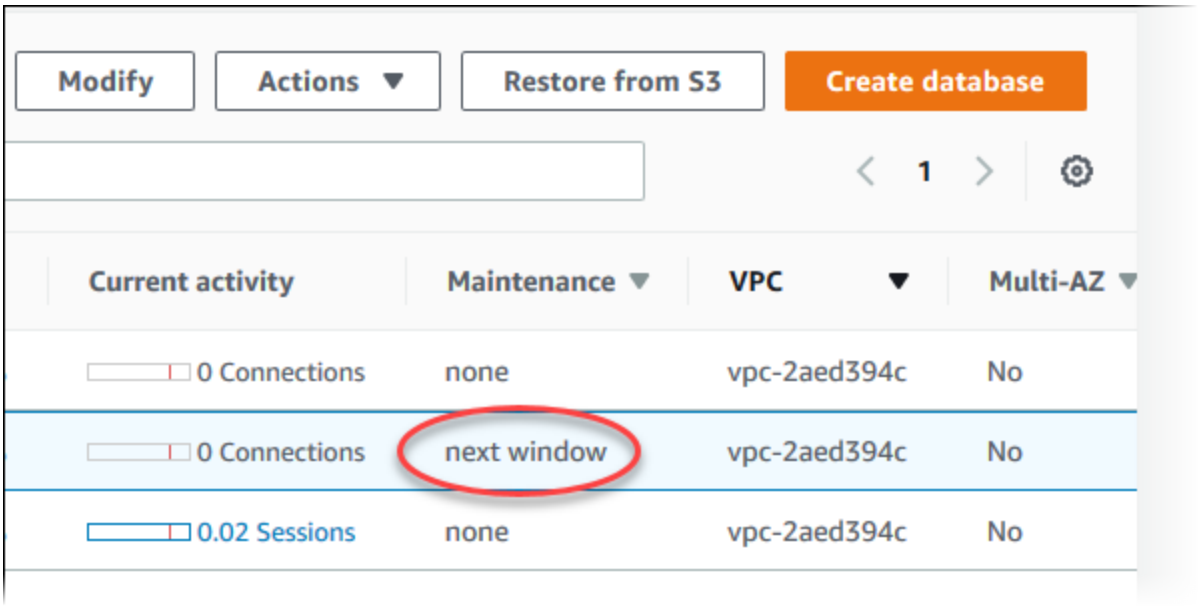
L'eventuale coerenza può influire sul modo in cui hai gestito gli aggiornamenti di manutenzione. Ad esempio, se si esegue il comando `ApplyPendingMaintenanceActions` per aggiornare la versione del motore di database per un cluster di database, alla fine sarà visibile a `DescribePendingMaintenanceActions`. In questo scenario, `DescribePendingMaintenanceActions` potrebbe mostrare che l'azione di manutenzione non è stata applicata anche nel caso lo sia stata.

Per gestire la coerenza finale, procedi nel modo seguente:

- Conferma lo stato di cluster di database prima di eseguire un comando per modificarlo. Esegui il comando `DescribePendingMaintenanceActions` appropriato utilizzando un algoritmo di backoff esponenziale per assicurarti di concedere tempo sufficiente per la propagazione del comando precedente nel sistema. A tale scopo, esegui il comando `DescribePendingMaintenanceActions` ripetutamente, iniziando con un paio di secondi di attesa e aumentando gradualmente fino a cinque minuti di attesa.
- Aggiungi il tempo di attesa tra i comandi successivi, anche se un comando `DescribePendingMaintenanceActions` restituisce una risposta accurata. Applica un algoritmo di backoff esponenziale a partire da un paio di secondi di attesa e aumenta gradualmente fino a circa cinque minuti di attesa.

Visualizzazione degli aggiornamento di manutenzione in sospeso

Verifica se è disponibile un aggiornamento di manutenzione per il tuo cluster di DB utilizzando la console RDS, l'API RDS o l' AWS CLI API RDS. Se è disponibile un aggiornamento, viene indicato nella colonna Manutenzione per il cluster di database nella console Amazon RDS, come illustrato in questa figura.



Se non è disponibile alcun aggiornamento di manutenzione per il cluster di un' database, il valore della colonna corrispondente è none (nessuno).

Se è disponibile un aggiornamento di manutenzione per il cluster di un' database, la colonna può avere i seguenti valori:

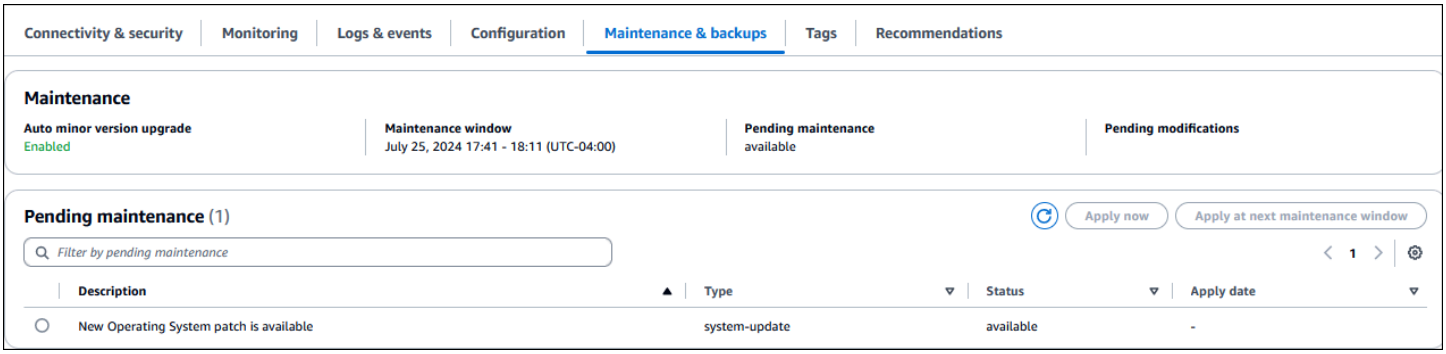
- richiesto – L'operazione di manutenzione sarà applicata alla risorsa e non può essere a tempo indeterminato.
- disponibile – L'operazione di manutenzione è disponibile ma non sarà automaticamente applicata alla risorsa. Puoi applicarla manualmente.
- finestra successiva – L'operazione di manutenzione sarà applicata alla risorsa durante la finestra di manutenzione successiva.
- In corso – L'operazione di manutenzione è in fase di applicazione alla risorsa.

Se è disponibile un aggiornamento, puoi scegliere una di queste operazioni:

- Se il valore di manutenzione è finestra successiva, posticipare le azioni di manutenzione scegliendo Posticipa aggiornamento da Operazioni. Non puoi rinviare un'azione di manutenzione che è già stata avviata.
- Applica immediatamente le operazioni di manutenzione.
- Applica le azioni di manutenzione durante la successiva finestra di manutenzione.
- Non eseguire alcuna operazione.

Per eseguire un'azione utilizzando il Console di gestione AWS

1. Scegli il nome dell'istanza o del cluster di database per visualizzarne i dettagli.
2. Scegliere Manutenzione e backup. Vengono visualizzate le azioni di manutenzione in attesa.
3. Scegli l'azione da intraprendere, quindi scegli quando applicarla.



La finestra di manutenzione determina l'avvio delle operazioni in sospeso, ma non limita il tempo di esecuzione totale di tali operazioni. Non è garantito che le operazioni di manutenzione terminino prima della fine della finestra di manutenzione e potrebbero continuare oltre l'ora di fine specificata. Per ulteriori informazioni, consulta Finestra di manutenzione Amazon RDS.

È inoltre possibile verificare se è disponibile un aggiornamento di manutenzione per il cluster di DB eseguendo il describe-pending-maintenance-actions AWS CLI comando.

Per informazioni sull'applicazione degli aggiornamenti di manutenzione, consulta Applicazione di aggiornamenti a un cluster di database.

Azioni di manutenzione per Amazon Aurora

Le seguenti azioni di manutenzione si applicano ai cluster di database Aurora:

- `os-upgrade` - Aggiorna i sistemi operativi di tutte le istanze database nel cluster di database, utilizzando aggiornamenti a rotazione. Per ulteriori informazioni, consulta the section called "Aggiornamenti del sistema operativo".
- `system-update`— Esegui patch del motore database per Aurora PostgreSQL.

Le seguenti azioni di manutenzione si applicano alle istanze database Aurora:

- `ca-certificate-rotation`: aggiorna il certificato Amazon RDS Certificate Authority per l'istanza database.
- `hardware-maintenance`: esegue la manutenzione sull'hardware sottostante per l'istanza database.
- `system-update`— Aggiorna il sistema operativo per l'istanza database.

Scelta della frequenza degli aggiornamenti di manutenzione di Aurora MySQL

Puoi scegliere con quale frequenza eseguire gli aggiornamenti di Aurora MySQL per ogni cluster di database. La scelta migliore dipende dall'utilizzo di Aurora MySQL e dalle priorità delle applicazioni eseguite in Aurora. Per informazioni sulle versioni con supporto a lungo termine (LTS) di Aurora MySQL che richiedono aggiornamenti meno frequenti, consultare Versioni con supporto a lungo termine (Long-Term Support, LTS) di Aurora MySQL.

Puoi scegliere di aggiornare un cluster Aurora MySQL più raramente se alcune o tutte le condizioni seguenti sono applicabili al tuo scenario:

- Il ciclo di test per l'applicazione richiede molto tempo per ogni aggiornamento al motore di database di Aurora MySQL.
- Hai molti cluster di database o molte applicazioni, tutti in esecuzione sulla stessa versione di Aurora MySQL. Preferisci aggiornare tutti i cluster di database e le applicazioni associate contemporaneamente.
- Utilizzi Aurora MySQL e RDS per MySQL e preferisci mantenere i cluster Aurora MySQL e le istanze database RDS per MySQL compatibili con lo stesso livello di MySQL.
- L'applicazione Aurora MySQL è in produzione o è comunque una risorsa business-critical. Non puoi avere tempi di inattività dovuti agli aggiornamenti, ad eccezione di rari casi per la distribuzione di patch critiche.

- L’applicazione Aurora MySQL non è limitata da problemi di prestazioni o da carenze nella funzionalità risolti nelle versioni di Aurora MySQL successive.

Se i fattori precedenti si applicano alla tua situazione, puoi limitare il numero di aggiornamenti forzati per un cluster di database Aurora MySQL. Per farlo, scegli una versione di Aurora MySQL specifica, nota come versione con supporto a lungo termine ("Long-Term Support", LTS) quando crei o aggiorni il cluster di database. In questo modo, riduci il numero di cicli di aggiornamenti, cicli di test e interruzioni correlate agli aggiornamenti per lo specifico cluster di database.

Puoi scegliere di aggiornare un cluster Aurora MySQL più spesso, se alcune o tutte le condizioni seguenti sono applicabili al tuo scenario:

- Il ciclo di test per l’applicazione è semplice e breve.
- L’applicazione è ancora in fase di sviluppo.
- L’ambiente del database utilizza diverse versioni di Aurora MySQL oppure versioni di Aurora MySQL e di RDS per MySQL. Ogni cluster Aurora MySQL ha il proprio ciclo di aggiornamento.
- Sei in attesa di determinati miglioramenti alle prestazioni o alle funzionalità per poter aumentare il tuo utilizzo di Aurora MySQL.

Se i fattori precedenti si applicano alla tua situazione, puoi abilitare Aurora per eseguire più spesso gli aggiornamenti importanti. Per farlo, esegui l’aggiornamento di un cluster database Aurora MySQL a una versione di Aurora MySQL più recente rispetto alla versione LTS. In questo modo, i miglioramenti alle prestazioni, le correzioni di bug e le funzionalità più recenti vengono resi disponibili più rapidamente.

Finestra di manutenzione Amazon RDS

La finestra di manutenzione è un intervallo di tempo settimanale durante il quale vengono applicate le modifiche al sistema. Ogni cluster di database ha una finestra di manutenzione settimanale. La finestra di manutenzione può essere considerata come un’opportunità per controllare quando vengono applicate le modifiche e le patch del software. Per ulteriori informazioni per regolare la finestra di manutenzione, consulta Impostazione della finestra di manutenzione preferita del cluster database.

Durante l’esecuzione delle attività di manutenzione, RDS utilizza alcune risorse del cluster database. Ciò potrebbe influire, in modo minimo, sulle prestazioni. Per un’istanza database, in rari casi,

potrebbe essere necessario un failover Multi-AZ per il completamento di un aggiornamento di manutenzione.

Se un evento di manutenzione è pianificato per una settimana specifica, viene avviato durante la finestra di manutenzione di 30 minuti indicata. La maggior parte degli eventi di manutenzione viene completata durante la finestra di manutenzione di 30 minuti, tuttavia l'esecuzione degli eventi di manutenzione di dimensioni maggiori può richiedere più di 30 minuti per il completamento. La finestra di manutenzione viene sospesa quando l'istanza di database viene arrestata/o.

La finestra di manutenzione di 30 minuti è selezionata a caso da un blocco di tempo di 8 ore per regione. Se non specifichi una finestra di manutenzione quando crei il cluster database, Amazon RDS assegna una finestra di manutenzione di 30 minuti in un giorno della settimana selezionato in modo casuale.

La tabella seguente mostra i blocchi temporali per ciascuno Regione AWS dei quali vengono assegnate le finestre di manutenzione predefinite.

Nome della regione	Regione	Periodo di tempo
US East (N. Virginia)	us-east-1	03:00 - 11:00 UTC
Stati Uniti orientali (Ohio)	us-east-2	03:00 - 11:00 UTC
US West (N. Californi a)	us-west-1	06:00 - 14:00 UTC
US West (Oregon)	us-west-2	06:00 - 14:00 UTC
Africa (Cape Town)	af-south-1	03:00 - 11:00 UTC
Asia Pacific (Hong Kong)	ap-east-1	06:00 - 14:00 UTC
Asia Pacific (Hyderabad)	ap-south-2	06:30 - 14:30 UTC
Asia Pacifico (Giacarta)	ap-southeast-3	08:00 - 16:00 UTC

Nome della regione	Regione	Periodo di tempo
Asia Pacifico (Malesia)	ap-southeast-5	09:00 - 17:00 UTC
Asia Pacifico (Melbourne)	ap-southeast-4	11:00 - 19:00 UTC
Asia Pacific (Mumbai)	ap-south-1	06:00 - 14:00 UTC
Asia Pacifico (Nuova Zelanda)	ap-southeast-6	13:00 - 21:00 UTC
Asia Pacifico (Osaka-Locale)	ap-northeast-3	22:00 - 23:59 UTC
Asia Pacific (Seoul)	ap-northeast-2	13:00 - 21:00 UTC
Asia Pacific (Singapore)	ap-southeast-1	14:00 - 22:00 UTC
Asia Pacific (Sydney)	ap-southeast-2	12:00 - 20:00 UTC
Asia Pacifico (Taipei)	ap-east-2	9:00 - 17:00 UTC
Asia Pacifico (Thailandia)	ap-southeast-7	8:00 - 16:00 UTC
Asia Pacifico (Tokyo)	ap-northeast-1	13:00 - 21:00 UTC
Canada (Central)	ca-central-1	03:00 - 11:00 UTC
Canada occidentale (Calgary)	ca-west-1	18:00 - 02:00 UTC
China (Beijing)	cn-north-1	06:00 - 14:00 UTC
China (Ningxia)	cn-northwest-1	06:00 - 14:00 UTC
Europe (Frankfurt)	eu-central-1	21:00 - 05:00 UTC

Nome della regione	Regione	Periodo di tempo
Europe (Ireland)	eu-west-1	22:00 - 06:00 UTC
Europe (London)	eu-west-2	22:00 - 06:00 UTC
Europe (Milan)	eu-south-1	02:00 - 10:00 UTC
Europe (Paris)	eu-west-3	23:59 - 07:29 UTC
Europa (Spagna)	eu-south-2	02:00 - 10:00 UTC
Europe (Stockholm)	eu-north-1	23:00 - 07:00 UTC
Europa (Zurigo)	eu-central-2	02:00 - 10:00 UTC
Israele (Tel Aviv)	il-central-1	03:00 - 11:00 UTC
Messico (Centrale)	mx-central-1	19:00 - 03:00 UTC
Medio Oriente (Bahrein)	me-south-1	06:00 - 14:00 UTC
Medio Oriente (Emirati Arabi Uniti)	me-central-1	05:00–13:00 UTC
Sud America (São Paulo)	sa-east-1	00:00 - 08:00 UTC
AWS GovCloud (Stati Uniti orientali)	us-gov-east-1	17:00 - 01:00 UTC
AWS GovCloud (Stati Uniti occidentali)	us-gov-west-1	06:00 - 14:00 UTC

Argomenti

- Impostazione della finestra di manutenzione preferita del cluster database

Impostazione della finestra di manutenzione preferita del cluster database

La finestra di manutenzione del cluster database Aurora deve essere impostata nel periodo di minore utilizzo e quindi potrebbe essere necessario apportare modifiche di tanto in tanto. Durante questo intervallo di tempo il cluster database non è disponibile solo se gli aggiornamenti applicati richiedono un'interruzione. L'interruzione dura il minimo necessario per apportare gli aggiornamenti richiesti.

> **Note**
>
> Per gli aggiornamenti al motore di database, Amazon Aurora gestisce la finestra di manutenzione preferita per un cluster database e non per le singole istanze.

Console

Per impostare la finestra di manutenzione preferita del cluster database

1. Accedi a Console di gestione AWS e apri la console Amazon RDS all'indirizzo https://console.aws.amazon.com/rds/.
2. Nel pannello di navigazione, seleziona Database.
3. Scegliere il cluster database per cui si desidera modificare la finestra di manutenzione.
4. Scegliere Modify (Modifica).
5. Nella sezione Manutenzione aggiornare a finestra di manutenzione.
6. Scegli Continue (Continua).

   Nella pagina di conferma esaminare le modifiche.
7. Per applicare immediatamente le modifiche alla finestra di manutenzione, scegli Immediately (Immediatamente) nella sezione Schedule of modifications (Pianificazione delle modifiche).
8. Scegliere Modify cluster (Modifica cluster) per salvare le modifiche.

   In alternativa, scegliere Back (Indietro) per cambiare le modifiche o Cancel (Annulla) per annullare le modifiche.

AWS CLI

Per regolare la finestra di manutenzione preferita del cluster DB, usa il AWS CLI `modify-db-cluster`comando con i seguenti parametri:

- `--db-cluster-identifier`
- `--preferred-maintenance-window`

Example

Nell'esempio di codice seguente la finestra di manutenzione viene impostata su martedì dalle 4:00 – alle 4:30.

Per Linux, macOS o Unix:

```
aws rds modify-db-cluster \
--db-cluster-identifier my-cluster \
--preferred-maintenance-window Tue:04:00-Tue:04:30
```

Per Windows:

```
aws rds modify-db-cluster ^
--db-cluster-identifier my-cluster ^
--preferred-maintenance-window Tue:04:00-Tue:04:30
```

API RDS

Per impostare la finestra di manutenzione del cluster database preferita, utilizzare l'operazione API Amazon RDS `ModifyDBCluster` con i parametri seguenti:

- `DBClusterIdentifier`
- `PreferredMaintenanceWindow`

Applicazione di aggiornamenti a un cluster di database

Con Amazon RDS, puoi scegliere quando eseguire le operazioni di manutenzione. Puoi decidere quando Amazon RDS applicare gli aggiornamenti utilizzando l'API Console di gestione AWS AWS CLI, o RDS.

Console

Per gestire un aggiornamento per un cluster di database

1. Accedi a Console di gestione AWS e apri la console Amazon RDS all'indirizzo https://console.aws.amazon.com/rds/.
2. Nel pannello di navigazione, seleziona Database.
3. Scegliere il cluster dell' database da aggiornare.
4. Per Actions (Operazioni), scegliere una delle seguenti opzioni:

   - Applica patch ora
   - Applica patch nella finestra successiva

   > **Note**
   >
   > Se si sceglie Applica patch nella finestra successiva e successivamente si desidera ritardare l'aggiornamento, è possibile scegliere Rinvia l'aggiornamento. Non puoi rinviare un'azione di manutenzione se è già stata avviata.
   > Per annullare un'azione di manutenzione, modifica l'istanza database e disabilita l'aggiornamento automatico della versione secondaria.

AWS CLI

Per applicare un aggiornamento in sospeso a un cluster di DB, usa il apply-pending-maintenance-action AWS CLI comando.

Example

Per Linux, macOS o Unix:

```
aws rds apply-pending-maintenance-action \
    --resource-identifier arn:aws:rds:us-west-2:001234567890:db:mysql-db \
```

```
    --apply-action system-update \
    --opt-in-type immediate
```

Per Windows:

```
aws rds apply-pending-maintenance-action ^
    --resource-identifier arn:aws:rds:us-west-2:001234567890:db:mysql-db ^
    --apply-action system-update ^
    --opt-in-type immediate
```

> **Note**
>
> Per rinviare un'azione di manutenzione, specificare `undo-opt-in` per `--opt-in-type`. Non è possibile specificare `undo-opt-in` per `--opt-in-type` se l'operazione di manutenzione è già stata avviata.
> Per annullare un'azione di manutenzione, esegui il modify-db-instance AWS CLI comando e specifica`--no-auto-minor-version-upgrade`.

Per restituire un elenco di risorse con almeno un aggiornamento in sospeso, usa il describe-pending-maintenance-actions AWS CLI comando.

Example

Per Linux, macOS o Unix:

```
aws rds describe-pending-maintenance-actions \
    --resource-identifier arn:aws:rds:us-west-2:001234567890:db:mysql-db
```

Per Windows:

```
aws rds describe-pending-maintenance-actions ^
    --resource-identifier arn:aws:rds:us-west-2:001234567890:db:mysql-db
```

È inoltre possibile restituire un elenco di risorse per un cluster di DB specificando il `--filters` parametro del `describe-pending-maintenance-actions` AWS CLI comando. Il formato del comando `--filters` è `Name=filter-name,Value=resource-id,...`.

Di seguito sono indicati i valori accettati per il parametro `Name` di un filtro:

- `db-instance-id`— Accetta un elenco di identificatori di istanze DB o Amazon Resource Names (ARNs). L'elenco restituito include solo le azioni di manutenzione in sospeso per le istanze DB identificate da questi identificatori o. ARNs
- `db-cluster-id`— Accetta un elenco di identificatori di cluster DB o ARNs per Amazon Aurora. L'elenco restituito include solo le azioni di manutenzione in sospeso per i cluster DB identificati da questi identificatori o. ARNs

L'esempio seguente restituisce le operazioni di manutenzione in sospeso per i cluster database `sample-cluster1` e `sample-cluster2`.

Example

Per Linux, macOS o Unix:

```
aws rds describe-pending-maintenance-actions \
 --filters Name=db-cluster-id,Values=sample-cluster1,sample-cluster2
```

Per Windows:

```
aws rds describe-pending-maintenance-actions ^
 --filters Name=db-cluster-id,Values=sample-cluster1,sample-cluster2
```

API RDS

Per applicare un aggiornamento a un cluster database, chiamare l'operazione `ApplyPendingMaintenanceAction` dell'API Amazon RDS.

Per ottenere un elenco delle risorse con almeno un aggiornamento in sospeso, chiamare l'operazione API Amazon RDS `DescribePendingMaintenanceActions`.

Aggiornamenti automatici delle versioni secondarie per i cluster di database Aurora

L'impostazione Aggiornamento automatico versione secondaria specifica se gli aggiornamenti Aurora vengono applicati automaticamente al cluster database. Questi aggiornamenti includono nuove versioni secondarie contenenti funzionalità aggiuntive e patch con correzioni di bug.

Gli aggiornamenti automatici delle versioni secondarie aggiornano periodicamente il database alle versioni recenti del motore di database. Tuttavia, l'aggiornamento potrebbe non includere

sempre la versione più recente del motore di database. Se è necessario mantenere i database su versioni specifiche in momenti particolari, consigliamo di eseguire manualmente l'aggiornamento alle versioni del database necessarie in base alla pianificazione richiesta. In caso di problemi di sicurezza critici o quando una versione raggiunge la end-of-support data di scadenza, Amazon Aurora potrebbe applicare un aggiornamento della versione secondaria anche se non hai abilitato l'opzione Aggiornamento automatico della versione secondaria. Per ulteriori informazioni, consulta la Guida all'aggiornamento del database nella documentazione di Oracle.

Consulta Aggiornamento della versione secondaria o del livello di patch di un cluster di database Aurora MySQL e Esecuzione di un aggiornamento a una versione secondaria.

> **Note**
>
> Database globale Aurora non supporta aggiornamenti automatici a versioni secondarie.

Questa impostazione è attivata per impostazione predefinita. Per ogni nuovo cluster database, scegliere il valore appropriato per questa impostazione. Il valore si basa sull'importanza, sulla durata prevista e sulla quantità di test di verifica eseguiti dopo ogni aggiornamento.

Per istruzioni su come attivare o disattivare l'impostazione Aggiornamento automatico versione secondaria, consulta quanto segue:

- Abilitazione degli aggiornamenti automatici delle versioni secondarie per un cluster database Aurora
- Abilitazione degli aggiornamenti automatici delle versioni secondarie per singole istanze database in un cluster database Aurora

> **Important**
>
> Si consiglia vivamente di applicare questa impostazione ai cluster database nuovi ed esistenti e non alle singole istanze database nel cluster. Se un'istanza database nel cluster ha questa impostazione disattivata, il cluster database non viene aggiornato automaticamente.

La tabella seguente mostra come funziona l'impostazione Aggiornamento automatico versione secondaria quando viene applicata a livello di cluster e istanza.

Azione	Impostazioni a livello di cluster	Impostazioni a livello di istanza	Il cluster è stato aggiornato automatic amente?
Impostazione su Vero a livello di cluster database.	True	Impostazione su Vero per tutte le istanze nuove ed esistenti	Sì
Impostazione su Falso a livello di cluster database.	False	Impostazione su Falso per tutte le istanze nuove ed esistenti	No
Precedente impostazione su Vero a livello di cluster database. Impostazione su Falso in almeno un'istanza database.	Impostazione modificata su Falso	Impostazione su Falso per una o più istanze	No
Precedente impostazione su Falso a livello di cluster database. Impostazione su Vero in almeno un'istanza database, ma non in tutte le istanze.	False	Impostazione su Vero in una o più istanze, ma non in tutte le istanze	No
Precedente impostazione su Falso a livello di cluster database. Impostazione su Vero in tutte le istanze database.	Impostazione modificata su Vero	Impostazione su Vero per tutte le istanze	Sì

Gli aggiornamenti automatici delle versioni secondarie vengono comunicati in anticipo tramite un evento cluster database Amazon RDS con una categoria di `maintenance` e un ID di `RDS-EVENT-0156`. Per ulteriori informazioni, consulta Categorie di eventi e messaggi di evento di Amazon RDS per Aurora.

Gli aggiornamenti automatici vengono eseguiti durante le finestre di manutenzione. Se le singole istanze database nel cluster database hanno finestre di manutenzione diverse dalla finestra di manutenzione del cluster, la finestra di manutenzione del cluster ha la precedenza.

Per ulteriori informazioni sugli aggiornamenti del motore per Aurora PostgreSQL, consultare Aggiornamenti del motore di database Amazon Aurora PostgreSQL per 02/06/2020.

Per ulteriori informazioni sull'impostazione di Auto Minor Version Upgrade (Aggiornamento automatico minore della versione) per Aurora MySQL, consulta Abilitazione degli aggiornamenti automatici tra versioni secondarie di Aurora MySQL. Per ulteriori informazioni sugli aggiornamenti del motore per Aurora MySQL, consulta Aggiornamenti del motore del database per Amazon Aurora MySQL.

Abilitazione degli aggiornamenti automatici delle versioni secondarie per un cluster database Aurora

Eseguire la procedura generale descritta in Modifica del cluster di database tramite la console, la CLI e l'API.

Console

Nella pagina Modifica il cluster DB, nella sezione Manutenzione selezionare la casella di controllo Abilita aggiornamento automatico versione secondaria.

AWS CLI

Chiamare il comando modify-db-cluster AWS CLI . Specificare il nome del cluster database per l'opzione `--db-cluster-identifier` e `true` per l'opzione `--auto-minor-version-upgrade`. Facoltativamente, specificare l'opzione `--apply-immediately` per attivare immediatamente questa impostazione per il cluster database.

API RDS

Richiama l'operazione Modify DBCluster API e specifica il nome del cluster DB per il `DBClusterIdentifier` parametro e `true` per il `AutoMinorVersionUpgrade` parametro. Facoltativamente, impostare il parametro `ApplyImmediately` su `true` per attivare immediatamente questa impostazione per il cluster database.

Abilitazione degli aggiornamenti automatici delle versioni secondarie per singole istanze database in un cluster database Aurora

Eseguire la procedura generale descritta in Modifica di un'istanza database in un cluster database.

Console

Nella pagina Modifica istanza database, nella sezione Manutenzione selezionare la casella di controllo Abilita aggiornamento automatico versione secondaria.

AWS CLI

Chiamare il comando modify-db-instance AWS CLI . Specifica il nome dell'istanza database per l'opzione `--db-instance-identifier` e `true` per l'opzione `--auto-minor-version-upgrade`. Facoltativamente, specifica l'opzione `--apply-immediately` per attivare immediatamente questa impostazione per l'istanza database. Esegui un comando `modify-db-instance` separato per ogni istanza database presente nel cluster.

API RDS

Chiama l'operazione Modify DBInstance API e specifica il nome del cluster DB per il `DBInstanceIdentifier` parametro e `true` per il `AutoMinorVersionUpgrade` parametro. Facoltativamente, imposta il parametro `ApplyImmediately` su `true` per attivare immediatamente questa impostazione per l'istanza database. Chiama un'operazione `ModifyDBInstance` separata per ogni istanza database presente nel cluster.

È possibile utilizzare un comando CLI come il seguente per verificare lo stato dell'impostazione `AutoMinorVersionUpgrade` per tutte le istanze database nei cluster Aurora MySQL.

```
aws rds describe-db-instances \
  --query '*[].
{DBClusterIdentifier:DBClusterIdentifier,DBInstanceIdentifier:DBInstanceIdentifier,AutoMinorVer
```

Questo comando genera un output simile al seguente:

```
[
  {
      "DBInstanceIdentifier": "db-writer-instance",
      "DBClusterIdentifier": "my-db-cluster-57",
```

```
        "AutoMinorVersionUpgrade": true
    },
    {
        "DBInstanceIdentifier": "db-reader-instance1",
        "DBClusterIdentifier": "my-db-cluster-57",
        "AutoMinorVersionUpgrade": false
    },
    {
        "DBInstanceIdentifier": "db-writer-instance2",
        "DBClusterIdentifier": "my-db-cluster-80",
        "AutoMinorVersionUpgrade": true
    },
... output omitted ...
```

In questo esempio, l'impostazione Abilita aggiornamento automatico versione secondaria è disattivata per il cluster di database `my-db-cluster-57`, perché è disattivata per una delle istanze database del cluster.

Aggiornamenti del sistema operativo per i cluster Aurora DB

Le istanze database nei cluster database Aurora MySQL e Aurora PostgreSQL richiedono occasionalmente aggiornamenti del sistema operativo. Amazon RDS aggiorna il sistema operativo a una versione più recente per migliorare le prestazioni del database e la posizione di sicurezza generale dei clienti. In genere, gli aggiornamenti richiedono circa 10 minuti. Gli aggiornamenti del sistema operativo non modificano la versione del motore database o la classe istanza database di un'istanza database.

Esistono due tipi di aggiornamenti del sistema operativo, differenziati dalla descrizione dell'azione di manutenzione in sospeso:

- Aggiornamento della distribuzione del sistema operativo: utilizzato per eseguire la migrazione alla versione principale supportata più recente di Amazon Linux. La sua descrizione è `New Operating System upgrade is available`.
- Patch del sistema operativo: utilizzata per applicare varie correzioni di sicurezza e talvolta per migliorare le prestazioni del database. La sua descrizione è `New Operating System patch is available`.

Gli aggiornamenti del sistema operativo possono essere facoltativi o obbligatori:

- Un aggiornamento facoltativo può essere applicato in qualsiasi momento. Sebbene questi aggiornamenti siano facoltativi, ti consigliamo di applicarli periodicamente per mantenere aggiornato il parco istanze RDS. RDS non applica automaticamente questi aggiornamenti.

  Per ricevere una notifica quando diventa disponibile una nuova patch facoltativa del sistema operativo, è possibile iscriversi a RDS-EVENT-0230 nella categoria degli eventi di applicazione delle patch di sicurezza. Per informazioni sulla sottoscrizione agli eventi RDS, consulta Sottoscrizione alle notifiche eventi di Amazon RDS.

  > **Note**
  >
  > `RDS-EVENT-0230` non si applica agli aggiornamenti di distribuzione del sistema operativo.

- È richiesto un aggiornamento obbligatorio. Viene inviata una notifica prima dell'aggiornamento obbligatorio. La notifica potrebbe contenere una data di scadenza. Pianificare la programmazione dell'aggiornamento prima di questa data di scadenza. Dopo la data di scadenza specificata, Amazon RDS aggiorna automaticamente il sistema operativo per l'istanza database alla versione più recente durante una delle finestre di manutenzione assegnate.

  Gli aggiornamenti della distribuzione del sistema operativo sono obbligatori.

> **Note**
>
> Potrebbe essere necessario rimanere aggiornati su tutti gli aggiornamenti facoltativi e obbligatori per soddisfare vari obblighi di conformità. Si consiglia di applicare regolarmente tutti gli aggiornamenti resi disponibili da RDS durante le finestre di manutenzione.

Per i cluster Aurora database, è possibile utilizzare l'opzione di manutenzione a livello di cluster per eseguire aggiornamenti del sistema operativo (OS). Trova l'opzione per eseguire aggiornamenti a livello di cluster nella scheda Manutenzione e backup quando selezioni il nome del tuo cluster di database nella console `os-upgrade` o usi il comando in AWS CLI. Questo metodo preserva la disponibilità di lettura con aggiornamenti continui che applicano automaticamente gli aggiornamenti a poche istanze database di lettura alla volta. Per evitare failover multipli e ridurre tempo di inattività non necessario, Aurora aggiorna l'istanza database di scrittura per ultima.

Gli aggiornamenti del sistema operativo a livello di cluster vengono eseguiti durante la finestra di manutenzione specificata per il cluster. Questo garantisce aggiornamenti coordinati su tutto il cluster.

Per la compatibilità con le versioni precedenti, Aurora mantiene anche l'opzione di manutenzione a livello di istanza. Consigliamo, tuttavia, di utilizzare aggiornamenti a livello di cluster. Se è necessario utilizzare aggiornamenti a livello di istanza, aggiornare prima le istanze database di lettura in un cluster di database, quindi aggiornare l'istanza database di scrittura. Se aggiorni le istanze di lettura e scrittura contemporaneamente, aumenti la possibilità di tempi di inattività dovuti al failover. Trova l'opzione per eseguire aggiornamenti a livello di istanza nella scheda Manutenzione e backup quando selezioni il nome dell'istanza database nella console o utilizzi il comando `system-update` in AWS CLI.

Gli aggiornamenti del sistema operativo a livello di istanza avvengono durante la finestra di manutenzione specificata per ogni rispettiva istanza. Ad esempio, se un cluster e due istanze di lettura hanno orari di manutenzione diversi, un aggiornamento del sistema operativo a livello di cluster si allinea alla finestra di manutenzione del cluster.

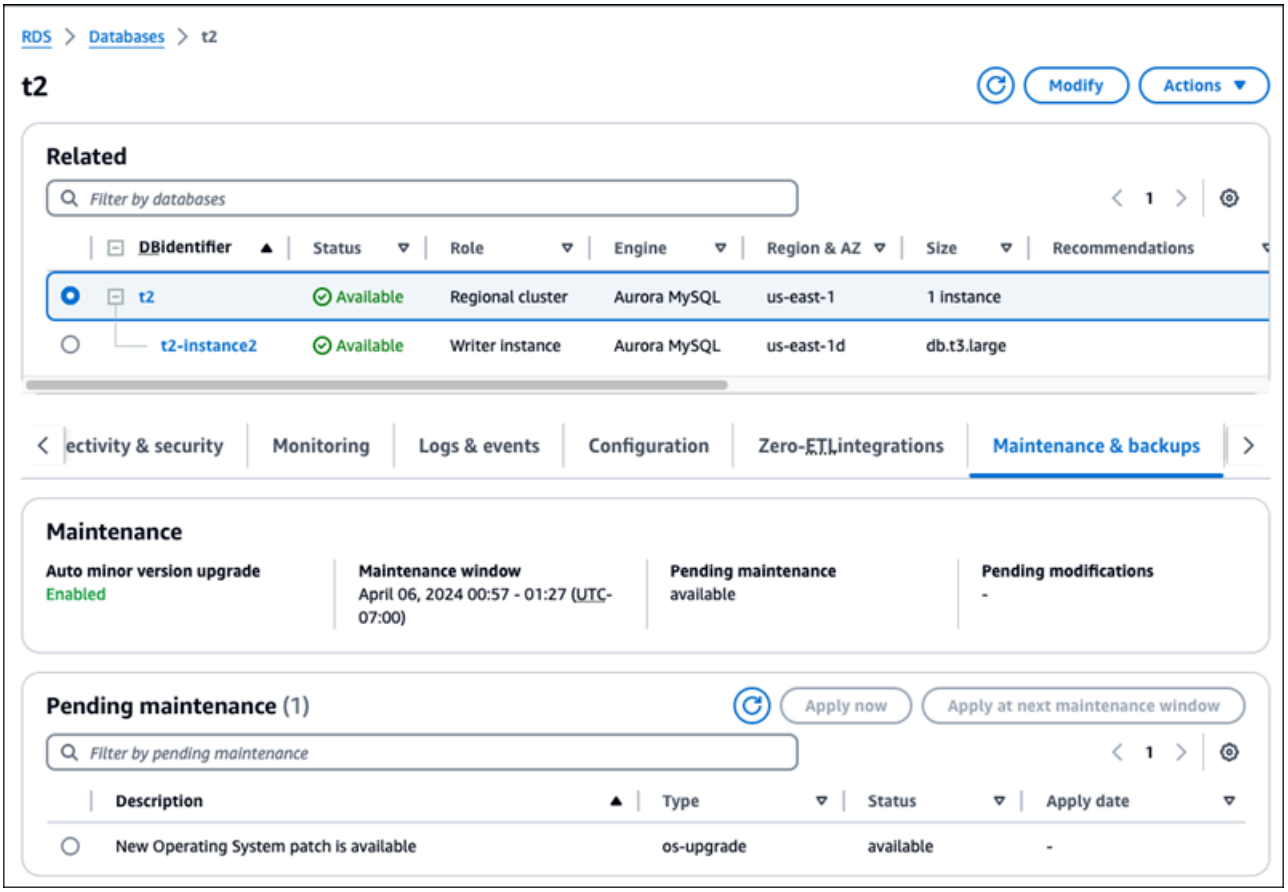
È possibile utilizzare Console di gestione AWS o the AWS CLI per ottenere informazioni sul tipo di aggiornamento del sistema operativo.

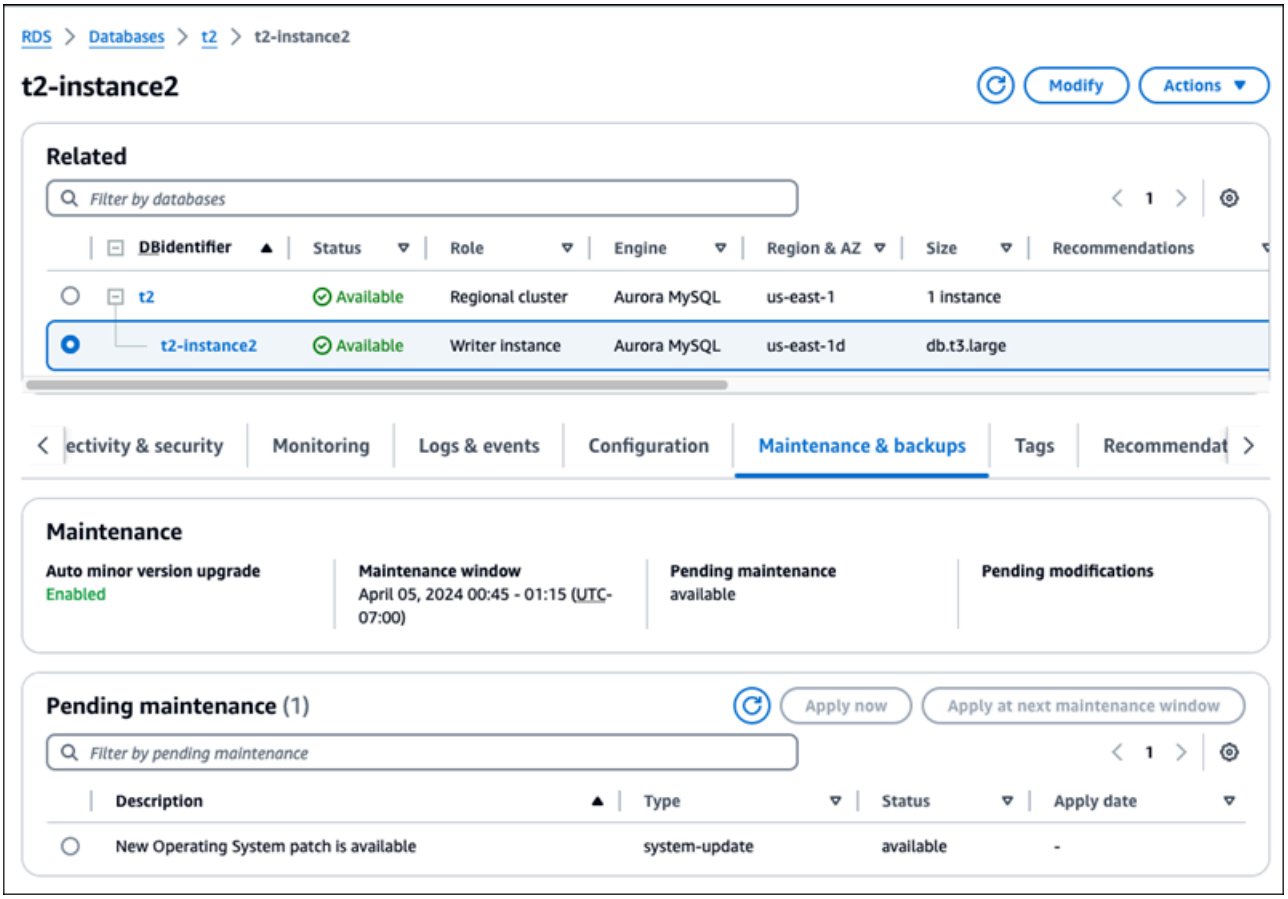
Console

Per ottenere informazioni di aggiornamento, utilizzare il Console di gestione AWS

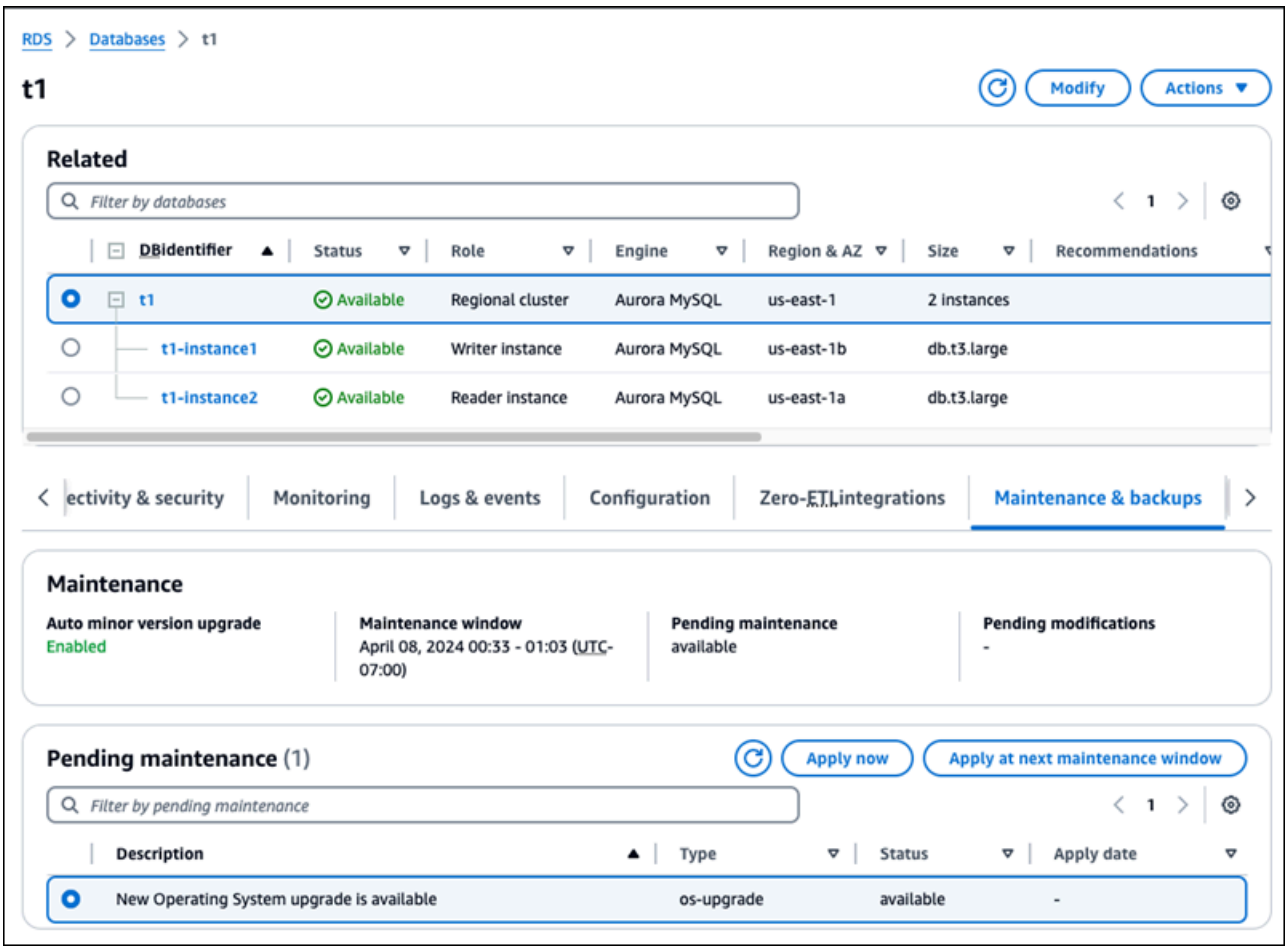
1. Accedi a Console di gestione AWS e apri la console Amazon RDS all'indirizzo https://console.aws.amazon.com/rds/.
2. Nel riquadro di navigazione, seleziona Databases (Database), quindi scegli l'istanza database.
3. Scegliere Manutenzione e backup.
4. Nella sezione Manutenzione in sospeso, trova l'aggiornamento del sistema operativo e controlla il valore Descrizione.

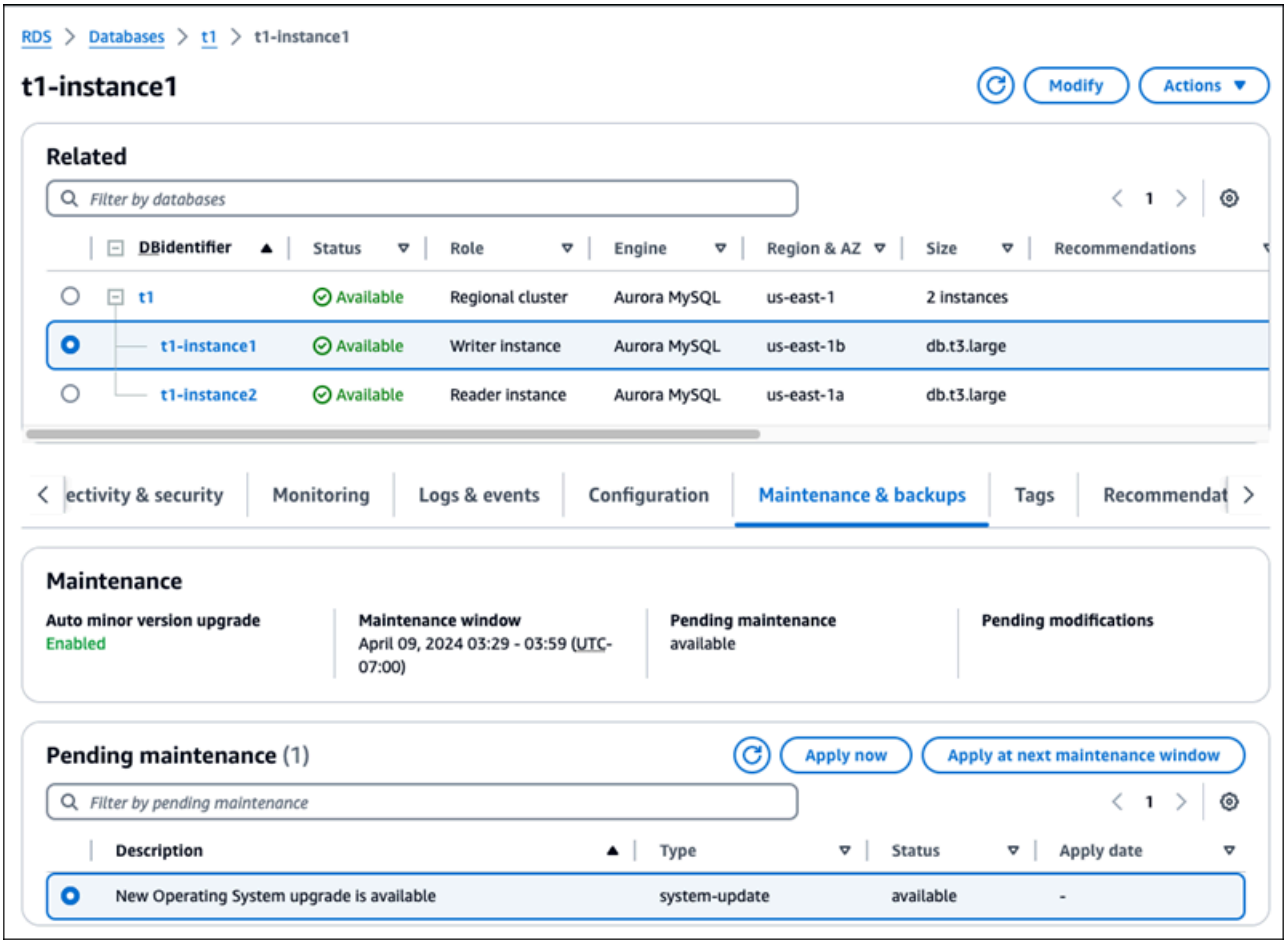
Le immagini seguenti mostrano un cluster di database con un'istanza database di scrittura con una patch del sistema operativo disponibile.

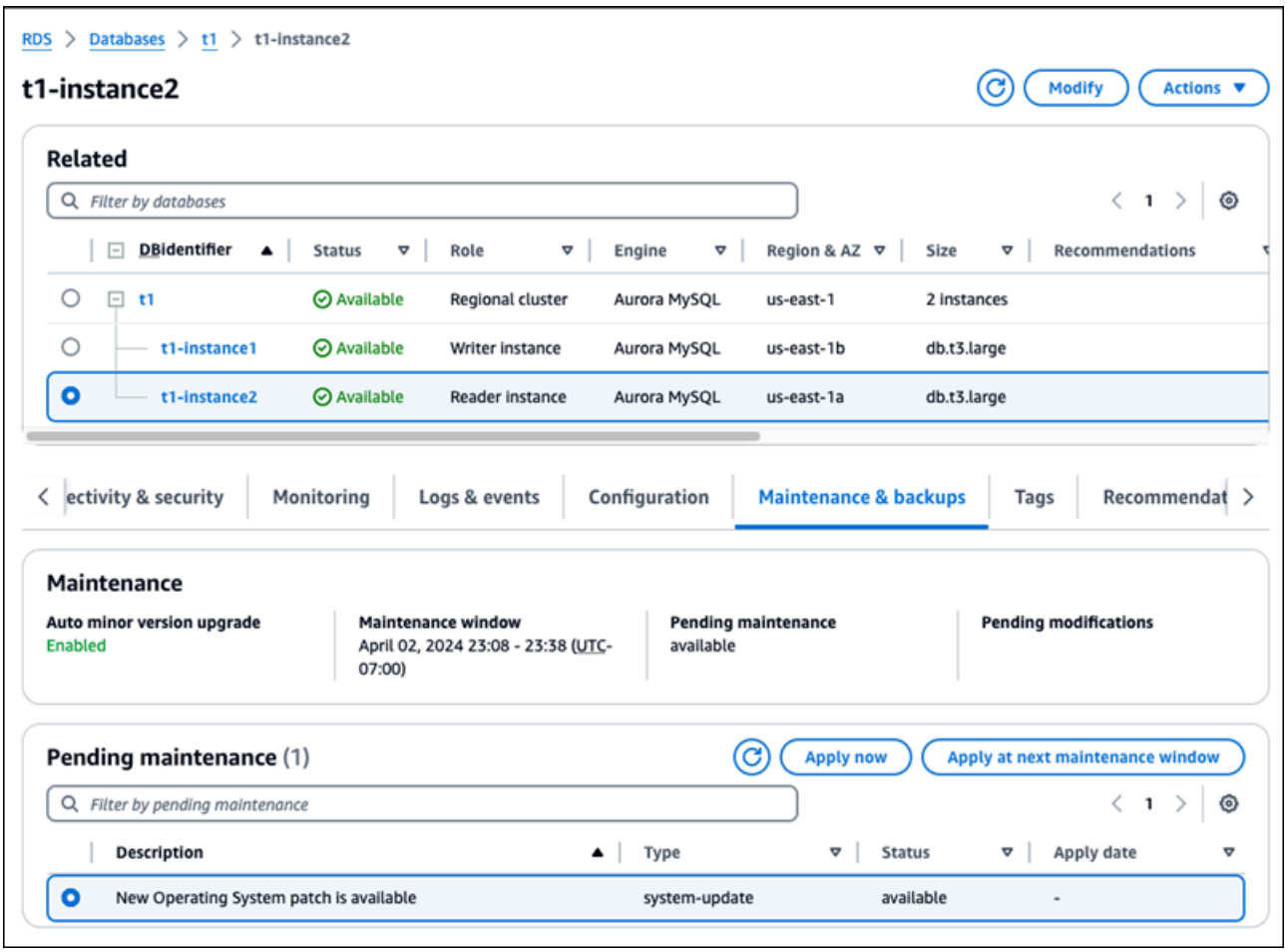

RDS > Databases > t2
t2
Modify
Actions
Related
Filter by databases
1
DBidentifier
Status
Role
Engine
Region & AZ
Size
Recommendations
t2
Available
Regional cluster
Aurora MySQL
us-east-1
1 instance
t2-instance2
Available
Writer instance
Aurora MySQL
us-east-1d
db.t3.large
ectivity & security
Monitoring
Logs & events
Configuration
Zero-ETL integrations
Maintenance & backups
Maintenance
Auto minor version upgrade
Enabled
Maintenance window
April 06, 2024 00:57 - 01:27 (UTC-07:00)
Pending maintenance
available
Pending modifications
-
Pending maintenance (1)
Apply now
Apply at next maintenance window
Filter by pending maintenance
1
Description
Type
Status
Apply date
New Operating System patch is available
os-upgrade
available
-



Le seguenti immagini mostrano un cluster di database con un'istanza database di scrittura e un'istanza database di lettura. L'istanza di scrittura dispone di un aggiornamento obbligatorio del sistema operativo. L'istanza di lettura dispone di una patch del sistema operativo.


RDS > Databases > t1
t1
Modify
Actions
Related
Filter by databases
1
DBidentifier
Status
Role
Engine
Region & AZ
Size
Recommendations
t1
Available
Regional cluster
Aurora MySQL
us-east-1
2 instances
t1-instance1
Available
Writer instance
Aurora MySQL
us-east-1b
db.t3.large
t1-instance2
Available
Reader instance
Aurora MySQL
us-east-1a
db.t3.large
ectivity & security
Monitoring
Logs & events
Configuration
Zero-ETL integrations
Maintenance & backups
Maintenance
Auto minor version upgrade
Enabled
Maintenance window
April 08, 2024 00:33 - 01:03 (UTC-07:00)
Pending maintenance
available
Pending modifications
-
Pending maintenance (1)
Apply now
Apply at next maintenance window
Filter by pending maintenance
1
Description
Type
Status
Apply date
New Operating System upgrade is available
os-upgrade
available
-


RDS > Databases > t1 > t1-instance1
t1-instance1
Modify
Actions
Related
Filter by databases
1
DBidentifier
Status
Role
Engine
Region & AZ
Size
Recommendations
t1
Available
Regional cluster
Aurora MySQL
us-east-1
2 instances
t1-instance1
Available
Writer instance
Aurora MySQL
us-east-1b
db.t3.large
t1-instance2
Available
Reader instance
Aurora MySQL
us-east-1a
db.t3.large
ectivity & security
Monitoring
Logs & events
Configuration
Maintenance & backups
Tags
Recommendat
Maintenance
Auto minor version upgrade
Enabled
Maintenance window
April 09, 2024 03:29 - 03:59 (UTC-07:00)
Pending maintenance
available
Pending modifications
Pending maintenance (1)
Apply now
Apply at next maintenance window
Filter by pending maintenance
1
Description
Type
Status
Apply date
New Operating System upgrade is available
system-update
available
-



AWS CLI

Per ottenere informazioni di aggiornamento da AWS CLI, usa il describe-pending-maintenance-actionscomando.

```
aws rds describe-pending-maintenance-actions
```

Il seguente output mostra un aggiornamento della distribuzione del sistema operativo per un cluster di database e un'istanza database.

```
{
  "PendingMaintenanceActions": [
    {
      "ResourceIdentifier": "arn:aws:rds:us-east-1:123456789012:cluster:t3",
      "PendingMaintenanceActionDetails": [
        {
          "Action": "os-upgrade",
          "Description": "New Operating System upgrade is available"
```

```
        }
      ]
    },
    {
      "ResourceIdentifier": "arn:aws:rds:us-east-1:123456789012:db:t3-instance1",
      "PendingMaintenanceActionDetails": [
        {
          "Action": "system-update",
          "Description": "New Operating System upgrade is available"
        }
      ]
    },
  ]
}
```

Il seguente output mostra un patch del sistema operativo per un'istanza database.

```
{
  "ResourceIdentifier": "arn:aws:rds:us-east-1:123456789012:db:mydb2",
  "PendingMaintenanceActionDetails": [
    {
      "Action": "system-update",
      "Description": "New Operating System patch is available"
    }
  ]
}
```

Disponibilità di aggiornamenti del sistema operativo

Gli aggiornamenti del sistema operativo sono specifici per la versione del motore database e la classe istanza database. Pertanto, le istanze database ricevono o richiedono aggiornamenti in momenti diversi. Se è disponibile un aggiornamento del sistema operativo per l'istanza database basato sulla versione del motore e sulla classe di istanza, l'aggiornamento viene visualizzato nella console. Può essere visualizzato anche eseguendo il describe-pending-maintenance-actions AWS CLI comando o chiamando l'operazione API DescribePendingMaintenanceActionsRDS. Se è disponibile un aggiornamento per l'istanza, puoi aggiornare il sistema operativo seguendo le istruzioni in Applicazione di aggiornamenti a un cluster di database.

Utilizzo della politica di implementazione degli AWS Organizations aggiornamenti per gli aggiornamenti automatici delle versioni secondarie

Aurora supporta la politica di implementazione degli AWS Organizations aggiornamenti per gestire gli aggiornamenti automatici delle versioni secondarie su più risorse di database e. Account AWS Questa policy consente di implementare una strategia di aggiornamento controllata per i cluster mediante:

Come funziona la politica di implementazione degli aggiornamenti

Quando una nuova versione secondaria del motore diventa idonea per l'aggiornamento automatico, la policy controlla la sequenza di aggiornamento in base a ordini definiti:

- Le risorse contrassegnate come [prime] (in genere ambienti di sviluppo) diventano idonee per gli aggiornamenti durante le rispettive finestre di manutenzione.
- Dopo un determinato periodo di cottura, le risorse contrassegnate come [seconde] diventano idonee.
- Dopo un altro periodo di cottura designato, le risorse contrassegnate come [ultime] (in genere ambienti di produzione) diventano idonee.
- Monitoraggio dell'avanzamento dell'aggiornamento tramite notifiche AWS Health.

Puoi definire i tuoi ordini di upgrade in base a:

- Politiche a livello di account: si applicano a tutte le risorse idonee in account specifici.
- Tag delle risorse: si applicano a risorse specifiche in base ai tag.

> **Note**
>
> Le risorse non configurate con una politica di aggiornamento o escluse dalla politica ricevono automaticamente un ordine di aggiornamento di [secondo].

Prerequisiti

- Account AWS È necessario far parte di un'organizzazione in Organizzazioni con la politica di implementazione degli aggiornamenti abilitata

- Abilita gli aggiornamenti automatici delle versioni secondarie per i tuoi cluster
- I tag non sono strettamente necessari per la politica di implementazione degli aggiornamenti. Se desideri definire ordini di aggiornamento specifici per ambienti diversi (ad esempio sviluppo, test, controllo qualità, produzione), puoi utilizzare i tag. Se non includi le impostazioni dei tag nella tua politica, tutte le risorse soggette a tale politica seguono l'ordine di aggiornamento predefinito. Per le risorse Aurora, vengono utilizzati solo tag a livello di cluster per la politica di implementazione dell'aggiornamento, anche se i tag sono definiti a livello di istanza.

Per etichettare le tue risorse

1. Accedi a Console di gestione AWS e apri la console Amazon RDS all'indirizzo https://console.aws.amazon.com/rds/.
2. Nel pannello di navigazione, seleziona Database.
3. Scegli il cluster a cui vuoi taggare.
4. Scegli Azioni, quindi scegli Gestisci tag.
5. Seleziona Aggiungi tag.
6. Inserisci la chiave del tag (ad esempio «Ambiente») e il valore (ad esempio, «Sviluppo»)
7. Scegli Aggiungi tag, quindi Salva.

Puoi anche aggiungere tag usando AWS CLI:

```
aws rds add-tags-to-resource \
    --resource-name arn:aws:rds:region:account-number:cluster:cluster-name \
    --tags Key=Environment,Value=Development
```

Ordine e fasi di aggiornamento

La politica di implementazione dell'aggiornamento supporta tre ordini di aggiornamento:

- [primo] - In genere utilizzato per ambienti di sviluppo o test
- [secondo] - Utilizzato in genere per ambienti di controllo qualità. Ordine predefinito per le risorse se la politica non è configurata in modo specifico
- [ultimo] - Solitamente riservato agli ambienti di produzione

Quando una nuova versione secondaria del motore diventa idonea per l'aggiornamento automatico:

- Le risorse con ordine di aggiornamento [first] diventano idonee per gli aggiornamenti durante le finestre di manutenzione configurate.
- Dopo un determinato periodo di attesa, le risorse con ordine di aggiornamento [secondo] diventano idonee per gli aggiornamenti durante i periodi di manutenzione.
- Dopo un altro periodo di attesa prestabilito, le risorse con ordine di aggiornamento [ultimo] diventano idonee per gli aggiornamenti durante le rispettive finestre di manutenzione.
- La campagna di aggiornamento automatico della versione secondaria si chiude dopo che tutte le risorse idonee con ordini di upgrade [primo], [secondo] e [ultimo] sono stati aggiornati o quando la campagna raggiunge la data di fine pianificata, a seconda di quale evento si verifica per primo.

> **Note**
>
> Tutti gli aggiornamenti automatici delle versioni secondarie vengono eseguiti durante la finestra di manutenzione configurata di ciascun cluster per ridurre al minimo il potenziale impatto sulle applicazioni.

Osservabilità

AWS Health e monitoraggio

Ricevi notifiche AWS sanitarie:

- Prima dell'inizio di una campagna automatica di aggiornamento di una versione secondaria
- Transizione tra una fase e l'altra per tracciare e monitorare l'avanzamento dell'aggiornamento
- Aggiornamenti sui progressi che mostrano il numero di risorse potenziate per tutta la flotta nella console AWS Health

Notifiche di eventi Amazon RDS:

- Notifiche per le risorse abilitate per gli aggiornamenti automatici delle versioni secondarie, tra cui:
  - Quando la risorsa diventa idonea per l'aggiornamento in base al relativo ordine di aggiornamento ([primo], [secondo] o [ultimo])
  - Cronologia pianificata dell'aggiornamento durante la finestra di manutenzione
  - Stato di inizio e completamento dell'aggiornamento individuale del database

- Iscriviti a questi eventi tramite Amazon EventBridge 0 per il monitoraggio automatizzato

Considerazioni

Alcune considerazioni da tenere a mente:

- La politica si applica a tutte le future campagne di aggiornamento automatico delle versioni secondarie, comprese le modifiche alle norme apportate durante le campagne attive.
- Se partecipi a una campagna di upgrade in corso, le tue risorse seguono l'ordine di aggiornamento in corso e non attendono una politica configurata.
- Le risorse non configurate con una politica di aggiornamento o escluse dalla politica ricevono automaticamente un ordine di aggiornamento di [secondo].
- La policy prevede periodi di convalida tra le fasi di aggiornamento prima di passare alla fase successiva.
- Le modifiche ai criteri o ai tag delle risorse richiedono tempo per propagarsi prima che venga applicato il nuovo ordine di aggiornamento.
- La policy si applica solo alle risorse Aurora con aggiornamenti automatici delle versioni secondarie abilitati.
- Se si rileva un problema all'interno di un ambiente, è possibile disattivare gli aggiornamenti automatici delle versioni secondarie per gli ambienti successivi o utilizzare il periodo di convalida per risolvere i problemi prima che gli aggiornamenti passino all'ordine di aggiornamento successivo.

Per ulteriori informazioni sull'etichettatura delle risorse RDS, vedere. Applicazione di tag alle risorse Amazon Aurora e Amazon RDS Per istruzioni dettagliate sulla configurazione e l'utilizzo della politica di implementazione degli aggiornamenti, consulta Guida introduttiva AWS Organizations nella Guida per l'AWS Organizations utente.

Riavvio di un cluster Amazon Aurora DB o di un'istanza Amazon Aurora DB

Potrebbe essere necessario riavviare il cluster DB o alcune istanze all'interno del cluster, di solito per motivi di manutenzione. Ad esempio, si supponga di modificare i parametri all'interno di un gruppo di parametri o di associare un gruppo di parametri diverso al cluster. In questi casi, è necessario riavviare il cluster affinché le modifiche abbiano effetto. Analogamente, è possibile riavviare una o più istanze DB di lettura all'interno del cluster. È possibile organizzare le operazioni di riavvio per singole istanze per ridurre al minimo i tempi di inattività per l'intero cluster.

Il tempo necessario per riavviare ogni istanza DB nel cluster dipende dall'attività del database al momento del riavvio. Dipende anche dal processo di ripristino dello specifico motore di DB. Se è pratico, ridurre l'attività del database su quella particolare istanza prima di avviare il processo di riavvio. In questo modo è possibile ridurre il tempo necessario per riavviare il database.

È possibile riavviare ogni istanza DB nel cluster solo quando è nello stato disponibile. Un'istanza database può non essere disponibile per vari motivi. Questi includono lo stato di arresto del cluster, una modifica applicata all'istanza e un'azione della finestra di manutenzione, ad esempio un aggiornamento della versione.

Il riavvio di un'istanza database comporta il riavvio del processo del motore di database. Il riavvio di un'istanza database comporta un'interruzione temporanea, durante la quale lo stato dell'istanza database viene impostato su rebooting (riavvio in corso).

> **Note**
>
> Se l'istanza database non usa le modifiche più recenti apportate al gruppo di parametri database associato, la Console di gestione AWS mostra il gruppo di parametri database con stato pending-reboot (riavvio in attesa). Lo stato dei gruppi di parametro pending-reboot non prevede un riavvio automatico nel corso della prossima finestra di manutenzione. Per applicare le ultime modifiche del parametro su quella istanza database, riavviare manualmente l'istanza database. Per ulteriori informazioni sui gruppi di parametri, consultare Gruppi di parametri per Amazon Aurora.

Argomenti

- Riavvio di un'istanza database in un cluster Aurora
- Riavvio di un cluster Aurora con disponibilità di lettura

- Riavvio di un cluster Aurora senza disponibilità di lettura
- Verifica del tempo di attività per i cluster e le istanze Aurora
- Esempi di operazioni di riavvio Aurora

Riavvio di un'istanza database in un cluster Aurora

Questa procedura è l'operazione più importante che si esegue durante il riavvio con Aurora. Molte delle procedure di manutenzione prevedono il riavvio di una o più istanze database Aurora in un determinato ordine.

Console

Per riavviare un'istanza database

1. Accedi alla Console di gestione AWS e apri la console Amazon RDS all'indirizzo https://console.aws.amazon.com/rds/.
2. Nel riquadro di navigazione, scegliere Databases (Database) e l'istanza DB da riavviare.
3. In Actions (Operazioni), scegliere Reboot (Riavvia).

   Viene visualizzata la pagina Reboot DB Instance (Riavvia istanza DB).
4. Scegliere Reboot (Riavvia) per riavviare l'istanza DB.

   Oppure scegliere Cancel (Annulla).

AWS CLI

Per riavviare un'istanza database tramite AWS CLI, chiamare il comando `reboot-db-instance`.

Example

Per Linux, macOS o Unix:

```
aws rds reboot-db-instance \
    --db-instance-identifier mydbinstance
```

Per Windows:

```
aws rds reboot-db-instance ^
    --db-instance-identifier mydbinstance
```

API RDS

Per riavviare un'istanza DB tramite l'API Amazon RDS, chiamare l'operazione `RebootDBInstance`.

Riavvio di un cluster Aurora con disponibilità di lettura

Se si utilizza la funzionalità della disponibilità di lettura, è possibile riavviare l'istanza di scrittura del cluster Aurora senza riavviare le istanze di lettura nel cluster di database primario o secondario. In questo modo si contribuisce a mantenere alta la disponibilità del cluster per le operazioni di lettura durante il riavvio dell'istanza di scrittura. È possibile riavviare le istanze di lettura in un secondo momento, secondo una pianificazione adatta a te. Ad esempio, per un cluster di produzione, è possibile riavviare le istanze di lettura una alla volta, iniziando solo dopo il completamento del riavvio dell'istanza primaria. Per ogni istanza DB che si riavvia, attenersi alla procedura in Riavvio di un'istanza database in un cluster Aurora.

La funzionalità della disponibilità di lettura per i cluster di database primari è disponibile in Aurora MySQL 2.10 e versioni successive. La disponibilità di lettura per i cluster di database secondari è disponibile in Aurora MySQL 3.06 e versioni successive.

Per Aurora PostgreSQL, questa funzione è disponibile per impostazione predefinita per le seguenti versioni:

- 15.2 o versioni successive alla 15
- 14.7 o versioni successive alla 14
- 13.10 o versioni successive alla 13
- 12.14 e versioni successive alla 12

Per ulteriori informazioni sulla funzionalità della disponibilità di lettura in Aurora PostgreSQL, consulta Miglioramento della disponibilità di lettura delle repliche Aurora.

Prima di questa funzionalità, il riavvio dell'istanza primaria causava contemporaneamente un riavvio per ogni istanza di lettura. Se il cluster Aurora sta eseguendo una versione precedente, utilizzare invece la procedura di riavvio in Riavvio di un cluster Aurora senza disponibilità di lettura.

> **Note**
>
> La modifica del comportamento di riavvio nei cluster di database Aurora con la funzionalità della disponibilità di lettura è diversa per i database Aurora MySQL precedenti alla versione

3.06 Se si riavvia l'istanza di scrittura per il cluster principale in un database globale Aurora, le istanze di lettura nel cluster primario rimangono disponibili. Tuttavia, le istanze database in qualsiasi cluster secondario si riavviano contemporaneamente.
Una versione limitata della funzionalità di disponibilità in lettura migliorata è supportata dai database globali Aurora PostgreSQL versioni 12.16, 13.12, 14.9, 15.4 e versioni successive.

Si riavvia spesso il cluster dopo aver apportato modifiche ai gruppi di parametri del cluster. È possibile apportare modifiche ai parametri seguendo le procedure in Gruppi di parametri per Amazon Aurora. Si supponga di riavviare l'istanza database di scrittura in un cluster Aurora per applicare le modifiche ai parametri del cluster. Alcune o tutte le istanze database di lettura potrebbero continuare a utilizzare le vecchie impostazioni dei parametri. Tuttavia, le diverse impostazioni dei parametri non influiscono sull'integrità dei dati del cluster. Tutti i parametri del cluster che influiscono sull'organizzazione dei file di dati vengono utilizzati solo dall'istanza database di scrittura.

Ad esempio, in un cluster Aurora MySQL è possibile aggiornare i parametri del cluster come `binlog_format` e `innodb_purge_threads` sull'istanza di scrittura prima delle istanze di lettura. Solo l'istanza di scrittura sta scrivendo registri binari e eliminando i record di annullamento. Per i parametri che modificano il modo in cui le query interpretano le istruzioni SQL o l'output della query, potrebbe essere necessario fare attenzione nel riavviare immediatamente le istanze di lettura. Lo si fa per evitare comportamenti imprevisti dell'applicazione durante le query. Ad esempio, supponiamo di modificare il parametro `lower_case_table_names` e di riavviare l'istanza di scrittura. In questo caso, le istanze di lettura potrebbero non essere in grado di accedere a una tabella appena creata finché non vengono tutte riavviate.

Per un elenco di tutti i parametri del cluster Aurora MySQL, consulta Parametri a livello di cluster.

Per un elenco di tutti i parametri per cluster Aurora MySQL, consulta Parametri a livello di cluster Aurora PostgreSQL.

Tip

Aurora MySQL potrebbe comunque riavviare alcune istanze di lettura insieme all'istanza di scrittura se il cluster sta elaborando un carico di lavoro con un throughput elevato.
La riduzione del numero di riavvii si applica anche durante le operazioni di failover. Aurora MySQL riavvia solo l'istanza database di scrittura e la destinazione del failover durante un failover. Altre istanze DB di lettura nel cluster rimangono disponibili per continuare a elaborare le query tramite connessioni all'endpoint di lettura. Pertanto, è possibile migliorare

la disponibilità durante un failover se è presente più di un'istanza database di lettura in un cluster.

Riavvio di un cluster Aurora senza disponibilità di lettura

Se non si utilizza la funzionalità della disponibilità di lettura, è possibile riavviare l'intero cluster database Aurora riavviando l'istanza database di scrittura di tale cluster. A tale scopo, segui la procedura in Riavvio di un'istanza database in un cluster Aurora.

Il riavvio dell'istanza DB di scrittura avvia anche un riavvio per ogni istanza DB di lettura nel cluster. In questo modo, tutte le modifiche dei parametri a livello di cluster vengono applicate contemporaneamente a tutte le istanze DB. Tuttavia, il riavvio di tutte le istanze DB causa una breve interruzione del cluster. Le istanze DB di lettura rimangono non disponibili fino a quando l'istanza DB di scrittura termina il riavvio e diventa disponibile.

Questo comportamento di riavvio si applica a tutti i cluster database creati in Aurora MySQL versione 2.09 e precedenti.

Per Aurora PostgreSQL questo comportamento si applica alle seguenti versioni:

- 14.6 e versioni precedenti alla 14
- 13.9 e versioni precedenti alla 13
- 12.13 e versioni precedenti alla 12
- Tutte le versioni di PostgreSQL 11

Nella console RDS, l'istanza DB scrittore ha il valore Writer (Di scrittura) nella colonna Role (Ruolo) nella pagina Database (Database). Nella CLI di RDS, l'output del comando `describe-db-clusters` include una sezione `DBClusterMembers`. L'elemento `DBClusterMembers` che rappresenta l'istanza DB di scrittura ha un valore di `true` per il campo `IsClusterWriter`.

Important

Con la funzionalità della disponibilità di lettura, il comportamento di riavvio è diverso in Aurora MySQL e Aurora PostgreSQL: le istanze database di lettura in genere rimangono disponibili durante il riavvio dell'istanza di scrittura. Quindi è possibile riavviare le istanze di lettura al momento opportuno. È possibile riavviare le istanze di lettura con una pianificazione

scaglionata se si desidera che alcune istanze di lettura siano sempre disponibili. Per ulteriori informazioni, consulta Riavvio di un cluster Aurora con disponibilità di lettura.

Verifica del tempo di attività per i cluster e le istanze Aurora

È possibile controllare e monitorare il periodo di tempo dall'ultimo riavvio per ogni istanza database nel cluster Aurora. Il CloudWatch parametro Amazon `EngineUptime` riporta il numero di secondi trascorsi dall'ultima volta in cui è stata avviata un'istanza DB. È possibile esaminare questo parametro in un point-in-time per scoprire il tempo di attività per l'istanza database. È inoltre possibile monitorare il parametro nel tempo per rilevare quando l'istanza viene riavviata.

È altresì possibile esaminare il parametro `EngineUptime` a livello del cluster. Le dimensioni `Minimum` e `Maximum` segnalano i valori di attività minori e maggiori per tutte le istanze database nel cluster. Per verificare l'ultima volta in cui una qualsiasi istanza di lettura in un cluster è stata riavviata, o riavviata per un altro motivo, monitorare il parametro a livello di cluster utilizzando la dimensione `Minimum`. Per verificare quale istanza nel cluster è durata più a lungo senza un riavvio, monitorare il parametro a livello di cluster utilizzando la dimensione `Maximum`. Ad esempio, è possibile confermare che tutte le istanze database nel cluster siano state riavviate dopo una modifica della configurazione.

Tip

Per il monitoraggio a lungo termine, si consiglia di monitorare il parametro `EngineUptime` per singole istanze anziché a livello di cluster. Il parametro `EngineUptime` a livello di cluster viene impostato su zero quando viene aggiunta una nuova istanza database al cluster. Tali modifiche del cluster possono avvenire nell'ambito di operazioni di manutenzione e dimensionamento come quelle eseguite da Auto Scaling.

Gli esempi di CLI riportati di seguito mostrano come esaminare i parametro `EngineUptime` per le istanze di scrittura e lettura in un cluster. Gli esempi utilizzano un cluster denominato `tpch100g`. Questo cluster ha un'istanza database di scrittura `instance-1234`. Dispone inoltre di due istanze database di lettura `instance-7448` e `instance-6305`.

Innanzitutto, il comando `reboot-db-instance` riavvia una delle istanze di lettura. Il comando `wait` attende fino al termine del riavvio dell'istanza.

```
$ aws rds reboot-db-instance --db-instance-identifier instance-6305
{
```

```
    "DBInstance": {
        "DBInstanceIdentifier": "instance-6305",
        "DBInstanceStatus": "rebooting",
...
$ aws rds wait db-instance-available --db-instance-id instance-6305
```

Il CloudWatch `get-metric-statistics` comando esamina la `EngineUptime` metrica negli ultimi cinque minuti a intervalli di un minuto. Il tempo di attività dell'istanza `instance-6305` viene ripristinato a zero e ricomincia a contare verso l'alto. Questo AWS CLI esempio per Linux utilizza la sostituzione `$( )` delle variabili per inserire i timestamp appropriati nei comandi CLI. Utilizza anche il comando Linux `sort` per ordinare l'output nel momento in cui è stato raccolto il parametro. Il valore del timestamp è il terzo campo di ogni riga di output.

```
$ aws cloudwatch get-metric-statistics --metric-name "EngineUptime" \
  --start-time "$(date -d '5 minutes ago')" --end-time "$(date -d 'now')" \
  --period 60 --namespace "AWS/RDS" --statistics Maximum \
  --dimensions Name=DBInstanceIdentifier,Value=instance-6305 --output text \
  | sort -k 3
EngineUptime
DATAPOINTS 231.0 2021-03-16T18:19:00+00:00 Seconds
DATAPOINTS 291.0 2021-03-16T18:20:00+00:00 Seconds
DATAPOINTS 351.0 2021-03-16T18:21:00+00:00 Seconds
DATAPOINTS 411.0 2021-03-16T18:22:00+00:00 Seconds
DATAPOINTS 471.0 2021-03-16T18:23:00+00:00 Seconds
```

Il tempo di attività minimo per il cluster viene ripristinato a zero perché una delle istanze del cluster è stata riavviata. Il tempo di attività massimo per il cluster non viene ripristinato perché almeno una delle istanze database nel cluster è rimasta disponibile.

```
$ aws cloudwatch get-metric-statistics --metric-name "EngineUptime" \
  --start-time "$(date -d '5 minutes ago')" --end-time "$(date -d 'now')" \
  --period 60 --namespace "AWS/RDS" --statistics Minimum \
  --dimensions Name=DBClusterIdentifier,Value=tpch100g --output text \
  | sort -k 3
EngineUptime
DATAPOINTS 63099.0 2021-03-16T18:12:00+00:00 Seconds
DATAPOINTS 63159.0 2021-03-16T18:13:00+00:00 Seconds
DATAPOINTS 63219.0 2021-03-16T18:14:00+00:00 Seconds
DATAPOINTS 63279.0 2021-03-16T18:15:00+00:00 Seconds
DATAPOINTS 51.0 2021-03-16T18:16:00+00:00 Seconds
```

```
$ aws cloudwatch get-metric-statistics --metric-name "EngineUptime" \
  --start-time "$(date -d '5 minutes ago')" --end-time "$(date -d 'now')" \
  --period 60 --namespace "AWS/RDS" --statistics Maximum \
  --dimensions Name=DBClusterIdentifier,Value=tpch100g --output text \
  | sort -k 3
EngineUptime
DATAPOINTS 63389.0 2021-03-16T18:16:00+00:00 Seconds
DATAPOINTS 63449.0 2021-03-16T18:17:00+00:00 Seconds
DATAPOINTS 63509.0 2021-03-16T18:18:00+00:00 Seconds
DATAPOINTS 63569.0 2021-03-16T18:19:00+00:00 Seconds
DATAPOINTS 63629.0 2021-03-16T18:20:00+00:00 Seconds
```

Quindi un altro comando `reboot-db-instance` riavvia l'istanza di scrittura del cluster. Un altro comando `wait` si interrompe fino al termine del riavvio dell'istanza di scrittura.

```
$ aws rds reboot-db-instance --db-instance-identifier instance-1234
{
  "DBInstanceIdentifier": "instance-1234",
  "DBInstanceStatus": "rebooting",
...
$ aws rds wait db-instance-available --db-instance-id instance-1234
```

Ora il parametro `EngineUptime` per l'istanza di scrittura mostra che l'istanza `instance-1234` è stata riavviata di recente. Anche l'istanza di lettura `instance-6305` è stata riavviata automaticamente insieme all'istanza di scrittura. Questo cluster esegue Aurora MySQL 2.09, il che non mantiene le istanze di lettura in esecuzione al riavvio dell'istanza di scrittura.

```
$ aws cloudwatch get-metric-statistics --metric-name "EngineUptime" \
  --start-time "$(date -d '5 minutes ago')" --end-time "$(date -d 'now')" \
  --period 60 --namespace "AWS/RDS" --statistics Maximum \
  --dimensions Name=DBInstanceIdentifier,Value=instance-1234 --output text \
  | sort -k 3
EngineUptime
DATAPOINTS 63749.0 2021-03-16T18:22:00+00:00 Seconds
DATAPOINTS 63809.0 2021-03-16T18:23:00+00:00 Seconds
DATAPOINTS 63869.0 2021-03-16T18:24:00+00:00 Seconds
DATAPOINTS 41.0 2021-03-16T18:25:00+00:00 Seconds
DATAPOINTS 101.0 2021-03-16T18:26:00+00:00 Seconds

$ aws cloudwatch get-metric-statistics --metric-name "EngineUptime" \
  --start-time "$(date -d '5 minutes ago')" --end-time "$(date -d 'now')" \
  --period 60 --namespace "AWS/RDS" --statistics Maximum \
```

```
  --dimensions Name=DBInstanceIdentifier,Value=instance-6305 --output text \
  | sort -k 3
EngineUptime
DATAPOINTS 411.0 2021-03-16T18:22:00+00:00 Seconds
DATAPOINTS 471.0 2021-03-16T18:23:00+00:00 Seconds
DATAPOINTS 531.0 2021-03-16T18:24:00+00:00 Seconds
DATAPOINTS 49.0 2021-03-16T18:26:00+00:00 Seconds
```

Esempi di operazioni di riavvio Aurora

Gli esempi Aurora MySQL seguenti mostrano diverse combinazioni di operazioni di riavvio per le istanze database di lettura e scrittura in un cluster DB Aurora. Dopo ogni riavvio, le query SQL dimostrano il tempo di attività per le istanze nel cluster.

Argomenti

- Individuazione delle istanze di scrittore e lettore per un cluster Aurora
- Riavvio di una singola istanza di lettura
- Riavvio dell'istanza di scrittura
- Riavvio indipendente di scrittore e lettori
- Applicazione di una modifica dei parametri del cluster a un cluster Aurora MySQL versione 2.10

Individuazione delle istanze di scrittore e lettore per un cluster Aurora

In un cluster Aurora MySQL con più istanze database, è importante sapere quale è lo scrittore e quali sono i lettori. Le istanze di scrittura e lettura possono anche cambiare ruolo quando si verifica un'operazione di failover. Pertanto, è meglio eseguire un controllo come il seguente prima di eseguire qualsiasi operazione che richieda un'istanza di scrittura o lettura. In questo caso, i valori `False` per `IsClusterWriter` identificano le istanze di lettura `instance-6305` e `instance-7448`. Il valore `True` identifica l'istanza di scrittura, `instance-1234`.

```
$ aws rds describe-db-clusters --db-cluster-id tpch100g \
  --query "*[].['Cluster:',DBClusterIdentifier,DBClusterMembers[*].
['Instance:',DBInstanceIdentifier,IsClusterWriter]]" \
  --output text
Cluster:     tpch100g
Instance:    instance-6305    False
Instance:    instance-7448    False
Instance:    instance-1234    True
```

Prima di iniziare gli esempi di riavvio, l'istanza di scrittura ha un tempo di attività di circa una settimana. La query SQL in questo esempio mostra un modo specifico di MySQL per controllare il tempo di attività. È possibile utilizzare questa tecnica in un'applicazione di database. Per un'altra tecnica che utilizza AWS CLI e funziona per entrambi i motori Aurora, consulta Verifica del tempo di attività per i cluster e le istanze Aurora.

```
$ mysql -h instance-7448.a12345.us-east-1.rds.amazonaws.com -P 3306 -u my-user -p
...
mysql> select date_sub(now(), interval variable_value second) "Last Startup",
    -> time_format(sec_to_time(variable_value),'%Hh %im') as "Uptime"
    -> from performance_schema.global_status
    -> where variable_name='Uptime';
+----------------------------+---------+
| Last Startup               | Uptime  |
+----------------------------+---------+
| 2021-03-08 17:49:06.000000 | 174h 42m|
+----------------------------+---------+
```

Riavvio di una singola istanza di lettura

Questo esempio riavvia una delle istanze DB di lettura. Probabilmente questa istanza è stata sovraccaricata da una query sovradimensionata o da molte connessioni simultanee. Oppure è rimasta indietro rispetto all'istanza di scrittura a causa di un problema di rete. Dopo aver avviato l'operazione di riavvio, l'esempio utilizza un comando `wait` per sospendere fino a quando l'istanza non diventa disponibile. A quel punto, l'istanza ha un tempo di attività di alcuni minuti.

```
$ aws rds reboot-db-instance --db-instance-identifier instance-6305
{
    "DBInstance": {
        "DBInstanceIdentifier": "instance-6305",
        "DBInstanceStatus": "rebooting",
...
    }
}
$ aws rds wait db-instance-available --db-instance-id instance-6305
$ mysql -h instance-6305.a12345.us-east-1.rds.amazonaws.com -P 3306 -u my-user -p
...
mysql> select date_sub(now(), interval variable_value second) "Last Startup",
    -> time_format(sec_to_time(variable_value),'%Hh %im') as "Uptime"
    -> from performance_schema.global_status
    -> where variable_name='Uptime';
```

```
+----------------------------+---------+
| Last Startup               | Uptime  |
+----------------------------+---------+
| 2021-03-16 00:35:02.000000 | 00h 03m |
+----------------------------+---------+
```

Il riavvio dell'istanza di lettura non ha influito sul tempo di attività dell'istanza di scrittura. Ha ancora un tempo di attività di circa una settimana.

```
$ mysql -h instance-7448.a12345.us-east-1.rds.amazonaws.com -P 3306 -u my-user -p
...
mysql> select date_sub(now(), interval variable_value second) "Last Startup",
    -> time_format(sec_to_time(variable_value),'%Hh %im') as "Uptime"
    -> from performance_schema.global_status where variable_name='Uptime';
+----------------------------+----------+
| Last Startup               | Uptime   |
+----------------------------+----------+
| 2021-03-08 17:49:06.000000 | 174h 49m |
+----------------------------+----------+
```

Riavvio dell'istanza di scrittura

Questo esempio riavvia l'istanza di scrittura. Questo cluster esegue Aurora MySQL versione 2.09. Poiché la versione Aurora MySQL è inferiore alla 2.10, il riavvio dell'istanza di scrittura riavvia anche le istanze di lettura nel cluster.

Un comando `wait` si interrompe fino al termine del riavvio. Ora il tempo di attività per quell'istanza viene ripristinato a zero. È possibile che un'operazione di riavvio possa richiedere tempi sostanzialmente diversi per le istanze DB di scrittura e lettura. Le istanze database di scrittura e lettura eseguono diversi tipi di operazioni di pulizia a seconda dei ruoli.

```
$ aws rds reboot-db-instance --db-instance-identifier instance-1234
{
    "DBInstance": {
        "DBInstanceIdentifier": "instance-1234",
        "DBInstanceStatus": "rebooting",
...
    }
}
$ aws rds wait db-instance-available --db-instance-id instance-1234
$ mysql -h instance-1234.a12345.us-east-1.rds.amazonaws.com -P 3306 -u my-user -p
...
```

```
mysql> select date_sub(now(), interval variable_value second) "Last Startup",
    -> time_format(sec_to_time(variable_value),'%Hh %im') as "Uptime"
    -> from performance_schema.global_status where variable_name='Uptime';
+----------------------------+---------+
| Last Startup               | Uptime  |
+----------------------------+---------+
| 2021-03-16 00:40:27.000000 | 00h 00m |
+----------------------------+---------+
```

Dopo il riavvio per l'istanza database di scrittura, entrambe le istanze database di lettura hanno anche il ripristino del tempo di attività. Il riavvio dell'istanza di scrittura ha causato il riavvio delle istanze di lettura. Questo comportamento si applica ai cluster Aurora PostgreSQL e ai cluster Aurora MySQL prima della versione 2.10.

```
$ mysql -h instance-7448.a12345.us-east-1.rds.amazonaws.com -P 3306 -u my-user -p
...
mysql> select date_sub(now(), interval variable_value second) "Last Startup",
    -> time_format(sec_to_time(variable_value),'%Hh %im') as "Uptime"
    -> from performance_schema.global_status where variable_name='Uptime';
+----------------------------+---------+
| Last Startup               | Uptime  |
+----------------------------+---------+
| 2021-03-16 00:40:35.000000 | 00h 00m |
+----------------------------+---------+

$ mysql -h instance-6305.a12345.us-east-1.rds.amazonaws.com -P 3306 -u my-user -p
...
mysql> select date_sub(now(), interval variable_value second) "Last Startup",
    -> time_format(sec_to_time(variable_value),'%Hh %im') as "Uptime"
    -> from performance_schema.global_status where variable_name='Uptime';
+----------------------------+---------+
| Last Startup               | Uptime  |
+----------------------------+---------+
| 2021-03-16 00:40:33.000000 | 00h 01m |
+----------------------------+---------+
```

Riavvio indipendente di scrittore e lettori

Gli esempi seguenti mostrano un cluster che esegue Aurora MySQL versione 2.10. In questa versione Aurora MySQL e versioni successive, è possibile riavviare l'istanza di scrittura senza causare il riavvio per tutte le istanze di lettura. In questo modo, le applicazioni a uso intensivo di query non subiscono interruzioni quando si riavvia l'istanza di scrittura. È possibile riavviare le istanze

di lettura in un secondo momento. È possibile eseguire questi riavvii in un momento di scarso traffico di query. È inoltre possibile riavviare le istanze di lettura una alla volta. In questo modo, almeno un'istanza di lettura è sempre disponibile per il traffico di query dell'applicazione.

Nell'esempio seguente viene utilizzato un cluster denominato `cluster-2393`, che esegue Aurora MySQL versione `5.7.mysql_aurora.2.10.0`. Questo cluster ha un'istanza di scrittura denominata `instance-9404` e tre istanze di lettura denominate `instance-6772`, `instance-2470` e `instance-5138`.

```
$ aws rds describe-db-clusters --db-cluster-id cluster-2393 \
  --query "*[].['Cluster:',DBClusterIdentifier,DBClusterMembers[*].
['Instance:',DBInstanceIdentifier,IsClusterWriter]]" \
  --output text
Cluster:        cluster-2393
Instance:       instance-5138        False
Instance:       instance-2470        False
Instance:       instance-6772        False
Instance:       instance-9404        True
```

Il controllo del valore `uptime` di ogni istanza di database tramite il comando `mysql` mostra che ognuna ha approssimativamente lo stesso tempo di attività. Ad esempio, ecco il tempo di attività per `instance-5138`.

```
mysql> SHOW GLOBAL STATUS LIKE 'uptime';
+---------------+-------+
| Variable_name | Value |
+---------------+-------+
| Uptime        | 3866  |
+---------------+-------+
```

Utilizzando CloudWatch, possiamo ottenere le informazioni relative al tempo di attività senza accedere effettivamente alle istanze. In questo modo, un amministratore può monitorare il database ma non può visualizzare o modificare alcun dato della tabella. In tal caso, specifichiamo un periodo di tempo che dura cinque minuti e controlliamo il valore del tempo di attività ogni minuto. I valori di attività crescenti dimostrano che le istanze non sono state riavviate durante quel periodo.

```
$ aws cloudwatch get-metric-statistics --metric-name "EngineUptime" \
  --start-time "$(date -d '5 minutes ago')" --end-time "$(date -d 'now')" --period 60 \
  --namespace "AWS/RDS" --statistics Minimum --dimensions
 Name=DBInstanceIdentifier,Value=instance-9404 \
  --output text | sort -k 3
```

```
EngineUptime
DATAPOINTS 4648.0 2021-03-17T23:42:00+00:00 Seconds
DATAPOINTS 4708.0 2021-03-17T23:43:00+00:00 Seconds
DATAPOINTS 4768.0 2021-03-17T23:44:00+00:00 Seconds
DATAPOINTS 4828.0 2021-03-17T23:45:00+00:00 Seconds
DATAPOINTS 4888.0 2021-03-17T23:46:00+00:00 Seconds

$ aws cloudwatch get-metric-statistics --metric-name "EngineUptime" \
  --start-time "$(date -d '5 minutes ago')" --end-time "$(date -d 'now')" --period 60 \
  --namespace "AWS/RDS" --statistics Minimum --dimensions
 Name=DBInstanceIdentifier,Value=instance-6772 \
  --output text | sort -k 3
EngineUptime
DATAPOINTS 4315.0 2021-03-17T23:42:00+00:00 Seconds
DATAPOINTS 4375.0 2021-03-17T23:43:00+00:00 Seconds
DATAPOINTS 4435.0 2021-03-17T23:44:00+00:00 Seconds
DATAPOINTS 4495.0 2021-03-17T23:45:00+00:00 Seconds
DATAPOINTS 4555.0 2021-03-17T23:46:00+00:00 Seconds
```

Ora riavviamo una delle istanze di lettura, `instance-5138`. Aspettiamo che l'istanza diventi nuovamente disponibile dopo il riavvio. Ora il monitoraggio del tempo di attività per un periodo di cinque minuti mostra che il tempo di attività è stato ripristinato a zero durante quel periodo. Il valore di attività più recente è stato misurato cinque secondi dopo il completamento del riavvio.

```
$ aws rds reboot-db-instance --db-instance-identifier instance-5138
{
  "DBInstanceIdentifier": "instance-5138",
  "DBInstanceStatus": "rebooting"
}
$ aws rds wait db-instance-available --db-instance-id instance-5138

$ aws cloudwatch get-metric-statistics --metric-name "EngineUptime" \
  --start-time "$(date -d '5 minutes ago')" --end-time "$(date -d 'now')" --period 60 \
  --namespace "AWS/RDS" --statistics Minimum --dimensions
 Name=DBInstanceIdentifier,Value=instance-5138 \
  --output text | sort -k 3
EngineUptime
DATAPOINTS 4500.0 2021-03-17T23:46:00+00:00 Seconds
DATAPOINTS 4560.0 2021-03-17T23:47:00+00:00 Seconds
DATAPOINTS 4620.0 2021-03-17T23:48:00+00:00 Seconds
DATAPOINTS 4680.0 2021-03-17T23:49:00+00:00 Seconds
DATAPOINTS  5.0 2021-03-17T23:50:00+00:00 Seconds
```

Successivamente, eseguiamo un riavvio per l'istanza di scrittur, `instance-9404`. Confrontiamo i valori di attività per l'istanza di scrittura e una delle istanze di lettura. Così facendo, possiamo vedere che il riavvio dello scrittore non ha causato un riavvio per i lettori. Nelle versioni precedenti a Aurora MySQL 2.10, i valori di attività per tutti i lettori sarebbero stati reimpostati contemporaneamente allo scrittore.

```
$ aws rds reboot-db-instance --db-instance-identifier instance-9404
{
  "DBInstanceIdentifier": "instance-9404",
  "DBInstanceStatus": "rebooting"
}
$ aws rds wait db-instance-available --db-instance-id instance-9404

$ aws cloudwatch get-metric-statistics --metric-name "EngineUptime" \
  --start-time "$(date -d '5 minutes ago')" --end-time "$(date -d 'now')" --period 60 \
  --namespace "AWS/RDS" --statistics Minimum --dimensions
 Name=DBInstanceIdentifier,Value=instance-9404 \
  --output text | sort -k 3
EngineUptime
DATAPOINTS 371.0 2021-03-17T23:57:00+00:00 Seconds
DATAPOINTS 431.0 2021-03-17T23:58:00+00:00 Seconds
DATAPOINTS 491.0 2021-03-17T23:59:00+00:00 Seconds
DATAPOINTS 551.0 2021-03-18T00:00:00+00:00 Seconds
DATAPOINTS  37.0  2021-03-18T00:01:00+00:00 Seconds

$ aws cloudwatch get-metric-statistics --metric-name "EngineUptime" \
  --start-time "$(date -d '5 minutes ago')" --end-time "$(date -d 'now')" --period 60 \
  --namespace "AWS/RDS" --statistics Minimum --dimensions
 Name=DBInstanceIdentifier,Value=instance-6772 \
  --output text | sort -k 3
EngineUptime
DATAPOINTS 5215.0 2021-03-17T23:57:00+00:00 Seconds
DATAPOINTS 5275.0 2021-03-17T23:58:00+00:00 Seconds
DATAPOINTS 5335.0 2021-03-17T23:59:00+00:00 Seconds
DATAPOINTS 5395.0 2021-03-18T00:00:00+00:00 Seconds
DATAPOINTS 5455.0 2021-03-18T00:01:00+00:00 Seconds
```

Per assicurarsi che tutte le istanze di lettura abbiano le stesse modifiche ai parametri di configurazione dell'istanza di scrittura, riavviare tutte le istanze di lettura dopo quelle di scrittura. Questo esempio riavvia tutti i lettori e quindi attende che tutti siano disponibili prima di procedere.

```
$ aws rds reboot-db-instance --db-instance-identifier instance-6772
```

```
{
  "DBInstanceIdentifier": "instance-6772",
  "DBInstanceStatus": "rebooting"
}

$ aws rds reboot-db-instance --db-instance-identifier instance-2470
{
  "DBInstanceIdentifier": "instance-2470",
  "DBInstanceStatus": "rebooting"
}

$ aws rds reboot-db-instance --db-instance-identifier instance-5138
{
  "DBInstanceIdentifier": "instance-5138",
  "DBInstanceStatus": "rebooting"
}

$ aws rds wait db-instance-available --db-instance-id instance-6772
$ aws rds wait db-instance-available --db-instance-id instance-2470
$ aws rds wait db-instance-available --db-instance-id instance-5138
```

Ora possiamo vedere che l'istanza database di scrittura ha il tempo di attività più alto. Il valore di attività di questa istanza è aumentato costantemente durante tutto il periodo di monitoraggio. Le istanze database di lettura sono state tutte riavviate dopo il lettore. Possiamo vedere il punto nel periodo di monitoraggio in cui ogni lettore è stato riavviato e il suo tempo di attività è stato ripristinato a zero.

```
$ aws cloudwatch get-metric-statistics --metric-name "EngineUptime" \
  --start-time "$(date -d '5 minutes ago')" --end-time "$(date -d 'now')" --period 60 \
  --namespace "AWS/RDS" --statistics Minimum --dimensions
 Name=DBInstanceIdentifier,Value=instance-9404 \
  --output text | sort -k 3
EngineUptime
DATAPOINTS 457.0 2021-03-18T00:08:00+00:00 Seconds
DATAPOINTS 517.0 2021-03-18T00:09:00+00:00 Seconds
DATAPOINTS 577.0 2021-03-18T00:10:00+00:00 Seconds
DATAPOINTS 637.0 2021-03-18T00:11:00+00:00 Seconds
DATAPOINTS  697.0 2021-03-18T00:12:00+00:00 Seconds

$ aws cloudwatch get-metric-statistics --metric-name "EngineUptime" \
  --start-time "$(date -d '5 minutes ago')" --end-time "$(date -d 'now')" --period 60 \
  --namespace "AWS/RDS" --statistics Minimum --dimensions
 Name=DBInstanceIdentifier,Value=instance-2470 \
```

```
  --output text | sort -k 3
EngineUptime
DATAPOINTS 5819.0 2021-03-18T00:08:00+00:00 Seconds
DATAPOINTS  35.0  2021-03-18T00:09:00+00:00 Seconds
DATAPOINTS 95.0 2021-03-18T00:10:00+00:00 Seconds
DATAPOINTS 155.0 2021-03-18T00:11:00+00:00 Seconds
DATAPOINTS 215.0 2021-03-18T00:12:00+00:00 Seconds

$ aws cloudwatch get-metric-statistics --metric-name "EngineUptime" \
  --start-time "$(date -d '5 minutes ago')" --end-time "$(date -d 'now')" --period 60 \
  --namespace "AWS/RDS" --statistics Minimum --dimensions
 Name=DBInstanceIdentifier,Value=instance-5138 \
  --output text | sort -k 3
EngineUptime
DATAPOINTS 1085.0 2021-03-18T00:08:00+00:00 Seconds
DATAPOINTS 1145.0 2021-03-18T00:09:00+00:00 Seconds
DATAPOINTS 1205.0 2021-03-18T00:10:00+00:00 Seconds
DATAPOINTS  49.0  2021-03-18T00:11:00+00:00 Seconds
DATAPOINTS 109.0 2021-03-18T00:12:00+00:00 Seconds
```

Applicazione di una modifica dei parametri del cluster a un cluster Aurora MySQL versione 2.10

Nell'esempio seguente viene illustrato come applicare una modifica dei parametri a tutte le istanze database nel cluster Aurora MySQL 2.10. Con questa versione Aurora MySQL, si riavviano l'istanza di scrittura e tutte le istanze di lettura in modo indipendente.

L'esempio utilizza il parametro di configurazione MySQL `lower_case_table_names` per la dimostrazione. Quando questa impostazione dei parametri è diversa tra le istanze database di scrittura e lettura, una query potrebbe non essere in grado di accedere a una tabella dichiarata con un nome maiuscolo o minuscolo misto. Oppure, se due nomi di tabelle differiscono solo in termini di lettere maiuscole e minuscole, una query potrebbe accedere alla tabella errata.

Questo esempio mostra come determinare le istanze di scrittura e lettura nel cluster esaminando l'attributo `IsClusterWriter` di ciascuna istanza. Il cluster è denominato `cluster-2393`. Il cluster ha un'istanza di scrittura denominata `instance-9404`. Le istanze di lettura nel cluster sono denominate `instance-5138` e `instance-2470`.

```
$ aws rds describe-db-clusters --db-cluster-id cluster-2393 \
  --query '*[].[DBClusterIdentifier,DBClusterMembers[*].
[DBInstanceIdentifier,IsClusterWriter]]' \
```

```
  --output text
cluster-2393
instance-5138        False
instance-2470        False
instance-9404        True
```

Per dimostrare gli effetti della modifica del parametro `lower_case_table_names`, abbiamo impostato due gruppi di parametri del cluster DB. Il gruppo di parametri `lower-case-table-names-0` ha questo parametro impostato su 0. Il gruppo di parametri `lower-case-table-names-1` ha questo gruppo di parametri impostato su 1.

```
$ aws rds create-db-cluster-parameter-group --description 'lower-case-table-names-0' \
  --db-parameter-group-family aurora-mysql5.7 \
  --db-cluster-parameter-group-name lower-case-table-names-0
{
    "DBClusterParameterGroup": {
        "DBClusterParameterGroupName": "lower-case-table-names-0",
        "DBParameterGroupFamily": "aurora-mysql5.7",
        "Description": "lower-case-table-names-0"
    }
}

$ aws rds create-db-cluster-parameter-group --description 'lower-case-table-names-1' \
  --db-parameter-group-family aurora-mysql5.7 \
  --db-cluster-parameter-group-name lower-case-table-names-1
{
    "DBClusterParameterGroup": {
        "DBClusterParameterGroupName": "lower-case-table-names-1",
        "DBParameterGroupFamily": "aurora-mysql5.7",
        "Description": "lower-case-table-names-1"
    }
}

$ aws rds modify-db-cluster-parameter-group \
  --db-cluster-parameter-group-name lower-case-table-names-0 \
  --parameters
 ParameterName=lower_case_table_names,ParameterValue=0,ApplyMethod=pending-reboot
{
    "DBClusterParameterGroupName": "lower-case-table-names-0"
}

$ aws rds modify-db-cluster-parameter-group \
  --db-cluster-parameter-group-name lower-case-table-names-1 \
```

```
    --parameters
 ParameterName=lower_case_table_names,ParameterValue=1,ApplyMethod=pending-reboot
{
    "DBClusterParameterGroupName": "lower-case-table-names-1"
}
```

Il valore predefinito di `lower_case_table_names` è 0. Con questa impostazione dei parametri, la tabella `foo` è distinta dalla tabella F00. Questo esempio verifica che il parametro sia ancora all'impostazione predefinita. Quindi l'esempio crea tre tabelle che differiscono solo per lettere maiuscole e minuscole nei nomi.

```
mysql> create database lctn;
Query OK, 1 row affected (0.07 sec)

mysql> use lctn;
Database changed
mysql> select @@lower_case_table_names;
+--------------------------+
| @@lower_case_table_names |
+--------------------------+
|                        0 |
+--------------------------+

mysql> create table foo (s varchar(128));
mysql> insert into foo values ('Lowercase table name foo');

mysql> create table Foo (s varchar(128));
mysql> insert into Foo values ('Mixed-case table name Foo');

mysql> create table FOO (s varchar(128));
mysql> insert into FOO values ('Uppercase table name FOO');

mysql> select * from foo;
+--------------------------+
| s                        |
+--------------------------+
| Lowercase table name foo |
+--------------------------+

mysql> select * from Foo;
+--------------------------+
| s                        |
+--------------------------+
```

```
| Mixed-case table name Foo |
+---------------------------+

mysql> select * from FOO;
+---------------------------+
| s                         |
+---------------------------+
| Uppercase table name FOO |
+---------------------------+
```

Successivamente, associamo il gruppo di parametri DB al cluster per impostare il parametro `lower_case_table_names` su 1. Questa modifica ha effetto solo dopo il riavvio di ogni istanza database.

```
$ aws rds modify-db-cluster --db-cluster-identifier cluster-2393 \
  --db-cluster-parameter-group-name lower-case-table-names-1
{
  "DBClusterIdentifier": "cluster-2393",
  "DBClusterParameterGroup": "lower-case-table-names-1",
  "Engine": "aurora-mysql",
  "EngineVersion": "5.7.mysql_aurora.2.10.0"
}
```

Il primo riavvio che facciamo è per l'istanza database di scrittura. Quindi aspettiamo che l'istanza diventi nuovamente disponibile. A quel punto, ci connettiamo all'endpoint di scrittura e verifichiamo che l'istanza di scrittura abbia il valore del parametro modificato. Il comando `SHOW TABLES` conferma che il database contiene le tre tabelle diverse. Tuttavia, le query che fanno riferimento a tabelle denominate `foo`, `Foo` o `F00` accedono tutte alla tabella il cui nome è interamente minuscolo, `foo`.

```
# Rebooting the writer instance
$ aws rds reboot-db-instance --db-instance-identifier instance-9404
$ aws rds wait db-instance-available --db-instance-id instance-9404
```

Ora, le query che utilizzano l'endpoint cluster mostrano gli effetti della modifica dei parametri. Indipendentemente dal fatto che il nome della tabella nella query sia maiuscolo, minuscolo o misto, l'istruzione SQL accede alla tabella il cui nome è tutto minuscolo.

```
mysql> select @@lower_case_table_names;
+---------------------------+
| @@lower_case_table_names |
```

```
+--------------------------+
|                        1 |
+--------------------------+

mysql> use lctn;
mysql> show tables;
+----------------+
| Tables_in_lctn |
+----------------+
| FOO            |
| Foo            |
| foo            |
+----------------+

mysql> select * from foo;
+--------------------------+
| s                        |
+--------------------------+
| Lowercase table name foo |
+--------------------------+

mysql> select * from Foo;
+--------------------------+
| s                        |
+--------------------------+
| Lowercase table name foo |
+--------------------------+

mysql> select * from FOO;
+--------------------------+
| s                        |
+--------------------------+
| Lowercase table name foo |
+--------------------------+
```

L'esempio successivo mostra le stesse query della precedente. In questo caso, le query utilizzano l'endpoint di lettura e vengono eseguite su una delle istanze database di lettura. Queste istanze non sono ancora state riavviate. Pertanto, hanno ancora l'impostazione originale per il parametro `lower_case_table_names`. Ciò significa che le query possono accedere a ciascuna delle tabelle `foo`, `Foo` e `FOO`.

```
mysql> select @@lower_case_table_names;
+--------------------------+
```

```
| @@lower_case_table_names |
+--------------------------+
|                        0 |
+--------------------------+

mysql> use lctn;

mysql> select * from foo;
+--------------------------+
| s                        |
+--------------------------+
| Lowercase table name foo |
+--------------------------+

mysql> select * from Foo;
+---------------------------+
| s                         |
+---------------------------+
| Mixed-case table name Foo |
+---------------------------+

mysql> select * from FOO;
+--------------------------+
| s                        |
+--------------------------+
| Uppercase table name FOO |
+--------------------------+
```

Successivamente, riavviamo una delle istanze di lettura e aspettiamo che diventi nuovamente disponibile.

```
$ aws rds reboot-db-instance --db-instance-identifier instance-2470
{
  "DBInstanceIdentifier": "instance-2470",
  "DBInstanceStatus": "rebooting"
}
$ aws rds wait db-instance-available --db-instance-id instance-2470
```

Mentre è connesso all'endpoint dell'istanza per `instance-2470`, una query mostra che il nuovo parametro è attivo.

```
mysql> select @@lower_case_table_names;
```

```
+--------------------------+
| @@lower_case_table_names |
+--------------------------+
|                        1 |
+--------------------------+
```

A questo punto, le due istanze di lettura nel cluster sono in esecuzione con impostazioni `lower_case_table_names` diverse. Pertanto, qualsiasi connessione all'endpoint di lettura del cluster utilizza un valore per questa impostazione che è imprevedibile. È importante riavviare immediatamente l'altra istanza di lettura in modo che entrambe abbiano impostazioni coerenti.

```
$ aws rds reboot-db-instance --db-instance-identifier instance-5138
{
  "DBInstanceIdentifier": "instance-5138",
  "DBInstanceStatus": "rebooting"
}
$ aws rds wait db-instance-available --db-instance-id instance-5138
```

L'esempio seguente conferma che tutte le istanze del lettore hanno la stessa impostazione per il parametro `lower_case_table_names`. I comandi controllano il valore di impostazione `lower_case_table_names` su ogni istanza di lettura. Quindi lo stesso comando che utilizza l'endpoint di lettura dimostra che ogni connessione all'endpoint di lettura utilizza una delle istanze di lettura, ma quale sia non è prevedibile.

```
# Check lower_case_table_names setting on each reader instance.

$ mysql -h instance-5138.a12345.us-east-1.rds.amazonaws.com \
  -u my-user -p -e 'select @@aurora_server_id, @@lower_case_table_names'
+--------------------------+--------------------------+
| @@aurora_server_id       | @@lower_case_table_names |
+--------------------------+--------------------------+
| instance-5138            |                        1 |
+--------------------------+--------------------------+

$ mysql -h instance-2470.a12345.us-east-1.rds.amazonaws.com \
  -u my-user -p -e 'select @@aurora_server_id, @@lower_case_table_names'
+--------------------------+--------------------------+
| @@aurora_server_id       | @@lower_case_table_names |
+--------------------------+--------------------------+
| instance-2470            |                        1 |
+--------------------------+--------------------------+
```

```
# Check lower_case_table_names setting on the reader endpoint of the cluster.

$ mysql -h cluster-2393.cluster-ro-a12345.us-east-1.rds.amazonaws.com \
  -u my-user -p -e 'select @@aurora_server_id, @@lower_case_table_names'
+--------------------------+--------------------------+
| @@aurora_server_id       | @@lower_case_table_names |
+--------------------------+--------------------------+
| instance-5138            |                        1 |
+--------------------------+--------------------------+

# Run query on writer instance

$ mysql -h cluster-2393.cluster-a12345.us-east-1.rds.amazonaws.com \
  -u my-user -p -e 'select @@aurora_server_id, @@lower_case_table_names'
+--------------------------+--------------------------+
| @@aurora_server_id       | @@lower_case_table_names |
+--------------------------+--------------------------+
| instance-9404            |                        1 |
+--------------------------+--------------------------+
```

Con la modifica dei parametri applicata ovunque, possiamo vedere l'effetto dell'impostazione `lower_case_table_names=1`. Se la tabella viene definita `foo`, `Foo` o `F00` la query converte il nome in `foo` e accede alla stessa tabella in ogni caso.

```
mysql> use lctn;

mysql> select * from foo;
+--------------------------+
| s                        |
+--------------------------+
| Lowercase table name foo |
+--------------------------+

mysql> select * from Foo;
+--------------------------+
| s                        |
+--------------------------+
| Lowercase table name foo |
+--------------------------+

mysql> select * from FOO;
+--------------------------+
```

```
| s                         |
+---------------------------+
| Lowercase table name foo |
+---------------------------+
```

Failover di un cluster di database Amazon Aurora

È possibile eseguire un failover manuale di un cluster di database Aurora, ad esempio, quando si desidera sostituire un'istanza database di scrittura con provisioning con un'istanza di scrittura Aurora Serverless v2.

Aurora esegue il failover su una nuova istanza database primaria in uno dei due modi seguenti:

- Promuovendo un'istanza database di lettura esistente come nuova istanza primaria.
- Creando una nuova istanza primaria

Se il cluster di database include una o più istanze di lettura, una di queste verrà promossa a istanza primaria durante un evento di errore. Per aumentare la disponibilità del cluster di database, è consigliabile creare almeno una o più istanze di lettura in due o più due zone di disponibilità. Per ulteriori informazioni sul meccanismo di failover, consulta Tolleranza ai guasti di un cluster DB Aurora.

È possibile utilizzare la Console di gestione AWS, la AWS CLI o l'API RDS per eseguire un failover manuale.

Console

Per eseguire il failover di un cluster di database

1. Accedi alla Console di gestione AWS e apri la console Amazon RDS all'indirizzo https://console.aws.amazon.com/rds/.
2. Nel riquadro di navigazione, scegli Database, quindi seleziona l'istanza database nel cluster di database di cui vuoi eseguire il failover.
3. Per Actions (Operazioni), scegliere Failover.

   Viene visualizzata una pagina di conferma.
4. Scegli Failover.

   La pagina Database mostra che lo stato del cluster di database è Failing-over. Al termine del failover, lo stato torna Disponibile e vengono visualizzati i ruoli per l'istanza database primaria nuova e precedente.

AWS CLI

Per eseguire il failover di un cluster di database mediante la AWS CLI, chiama il comando failover-db-cluster. Specifica i seguenti parametri:

- `--db-cluster-identifier`: il cluster di database di cui vuoi eseguire il failover.
- `--target-db-instance-identifier`: il nome dell'istanza database da promuovere a istanza database primaria.

Example

Per Linux, macOS o Unix:

```
aws rds failover-db-cluster \
    --db-cluster-identifier mydbcluster \
    --target-db-instance-identifier mydbcluster-instance-2
```

Per Windows:

```
aws rds failover-db-cluster ^
    --db-cluster-identifier mydbcluster ^
    --target-db-instance-identifier mydbcluster-instance-2
```

API RDS

Per modificare un cluster di database tramite l'API Amazon RDS, chiama l'operazione FailoverDBCluster. Specifica i seguenti parametri:

- DBClusterIdentifier
- TargetDBInstanceIdentifier

Eliminazione di cluster e istanze database di Aurora

È possibile eliminare un cluster database Aurora quando non è più necessario. L'eliminazione del cluster rimuove il volume del cluster contenente tutti i dati. Prima di eliminare il cluster, puoi decidere di salvare una snapshot dei dati. È possibile ripristinare lo snapshot in un secondo momento per creare un nuovo cluster contenente gli stessi dati.

Inoltre, è possibile eliminare le istanze database da un cluster, conservando però il cluster stesso e i dati che contiene. L'eliminazione delle istanze database consente di ridurre i costi se il cluster non è occupato e non ha bisogno della capacità di calcolo di più istanze database.

Argomenti

- Eliminazione di un cluster database Aurora
- Protezione dall'eliminazione per i cluster Aurora
- Eliminazione di un cluster Aurora arrestato
- Eliminazione di cluster Aurora MySQL che sono repliche di lettura
- Lo snapshot finale durante l'eliminazione di un cluster
- Eliminazione di un'istanza database da un cluster database Aurora

Eliminazione di un cluster database Aurora

Aurora non fornisce un metodo con una singola operazione per eliminare un cluster database. Questa scelta progettuale ha lo scopo di impedire la perdita accidentale di dati o di portare l'applicazione offline. Le applicazioni Aurora sono in genere mission-critical e richiedono elevata disponibilità. Pertanto, Aurora facilita il dimensionamento del cluster verso l'alto e verso il basso aggiungendo e rimuovendo istanze database. La rimozione del cluster database stesso richiede di effettuare un'eliminazione separata.

Utilizza la procedura generale seguente per rimuovere tutte le istanze database da un cluster e quindi eliminare il cluster stesso.

1. Elimina tutte le istanze reader nel cluster. Utilizza la procedura descritta in Eliminazione di un'istanza database da un cluster database Aurora.

   Se il cluster dispone di più istanze di lettura, l'eliminazione di una delle istanze riduce la capacità di calcolo del cluster. L'eliminazione delle istanze reader garantisce innanzitutto che il cluster rimanga disponibile per tutta la procedura e non esegua operazioni di failover non necessarie.

2. Elimina l'istanza writer dal cluster. Ancora una volta, utilizza la procedura in Eliminazione di un'istanza database da un cluster database Aurora.

   Quando elimini le istanze database, il cluster e il relativo volume cluster associato vengono mantenuti anche dopo l'eliminazione di tutte le istanze database.

3. Elimina il cluster database.

   - Console di gestione AWS: scegli il cluster, quindi scegli Elimina nel menu lOperazioni. Puoi scegliere le opzioni seguenti per conservare i dati del cluster in caso di necessità in un secondo momento:
     - Crea uno snapshot finale del volume del cluster. L'impostazione predefinita prevede la creazione di uno snapshot finale.
     - Mantieni backup automatici. L'impostazione predefinita non prevede il mantenimento dei backup automatici.

     > **Note**
     >
     > I backup automatici per i cluster di database Aurora Serverless v1 non vengono conservati.

     Aurora richiede inoltre di confermare che intendi veramente eliminare il cluster.
   - CLI e API: chiama il comando CLI `delete-db-cluster` o l'operazione API `DeleteDBCluster`. Puoi scegliere le opzioni seguenti per conservare i dati del cluster in caso di necessità in un secondo momento:
     - Crea uno snapshot finale del volume del cluster.
     - Mantieni backup automatici.

     > **Note**
     >
     > I backup automatici per i cluster di database Aurora Serverless v1 non vengono conservati.

Argomenti

- Eliminazione di un cluster Aurora vuoto
- Eliminazione di un cluster Aurora con una singola istanza database

- Eliminazione di un cluster Aurora con più istanze database

Eliminazione di un cluster Aurora vuoto

Puoi eliminare un cluster DB vuoto utilizzando Console di gestione AWS AWS CLI, o l'API Amazon RDS.

> **Tip**
>
> Puoi conservare un cluster senza istanze database in modo da mantenere i dati senza incorrere in costi di CPU per il cluster. Puoi iniziare a utilizzare di nuovo il cluster molto rapidamente creando una o più nuove istanze database per il cluster. Tuttavia, puoi aggiungere nuove istanze DB solo utilizzando l'API AWS CLI o l'API RDS. Non puoi aggiungere nuove istanze database utilizzando la console Amazon RDS. Puoi eseguire operazioni amministrative specifiche di Aurora sul cluster se questo non dispone di istanze database associate. Non puoi accedere ai dati o eseguire operazioni che richiedono la connessione a un'istanza database.

Console

Per eliminare un cluster database

1. Accedi a Console di gestione AWS e apri la console Amazon RDS all'indirizzo https://console.aws.amazon.com/rds/.
2. Nel pannello di navigazione, scegli Database, quindi scegli il cluster database multi-AZ che vuoi eliminare.
3. In Actions (Azioni), scegliere Delete (Elimina).
4. Per creare uno snapshot database multi-AZ finale per il cluster di database, scegli Crea snapshot finale. Questa è l'impostazione predefinita.
5. Se si è scelto di creare uno snapshot finale, immettere il Final snapshot name (Nome dello snapshot finale).
6. Per mantenere i backup automatici, scegliere Retain automated backups (Mantieni backup automatici). Questa non è l'impostazione predefinita.
7. Immettere **`delete me`** nella casella.
8. Scegliere Delete (Elimina).

CLI

Per eliminare un cluster Aurora DB vuoto utilizzando AWS CLI, chiamare il delete-db-clustercomando.

Supponiamo che il cluster vuoto `deleteme-zero-instances` sia stato utilizzato solo per lo sviluppo e il test e non contenga dati importanti. In tal caso, non è necessario conservare una snapshot del volume del cluster quando si elimina il cluster. Nell'esempio seguente viene illustrato un cluster che non contiene istanze database e quindi elimina il cluster vuoto senza creare uno snapshot finale o mantenendo backup automatici.

```
$ aws rds describe-db-clusters --db-cluster-identifier deleteme-zero-instances --output
 text \
  --query '*[].["Cluster:",DBClusterIdentifier,DBClusterMembers[*].
["Instance:",DBInstanceIdentifier,IsClusterWriter]]
Cluster:        deleteme-zero-instances

$ aws rds delete-db-cluster --db-cluster-identifier deleteme-zero-instances \
  --skip-final-snapshot \
  --delete-automated-backups
{
  "DBClusterIdentifier": "deleteme-zero-instances",
  "Status": "available",
  "Engine": "aurora-mysql"
}
```

API RDS

Per eliminare un cluster Aurora DB vuoto utilizzando l'API Amazon RDS, chiama l'operazione Delete. DBCluster

Eliminazione di un cluster Aurora con una singola istanza database

Puoi anche eliminare l'ultima istanza database anche se il cluster database è protetto dall'eliminazione. In questo caso, il cluster di database continua a esistere e i dati vengono conservati. Puoi accedere di nuovo ai dati collegando una nuova istanza database al cluster.

Nell'esempio seguente viene illustrato come il comando `delete-db-cluster` non funziona quando al cluster sono ancora associate istanze database. Questo cluster ha una singola istanza database writer. Esaminando le istanze database nel cluster, si controlla l'attributo `IsClusterWriter` di ciascuna di esse. Il cluster potrebbe avere zero o un'istanza database writer. Un valore `true` indica

un'istanza database writer. Un valore `false` indica un'istanza database reader. Il cluster potrebbe avere zero, una o più istanze database reader. In questo caso, eliminiamo l'istanza database writer utilizzando il comando `delete-db-instance`. Non appena l'istanza database entra nello stato `deleting`, possiamo eliminare anche il cluster. Anche per questo esempio, supponiamo che il cluster non contenga dati che valga la pena conservare. Pertanto, non creiamo uno snapshot del volume del cluster né manteniamo backup automatici.

```
$ aws rds delete-db-cluster --db-cluster-identifier deleteme-writer-only --skip-final-
snapshot
An error occurred (InvalidDBClusterStateFault) when calling the DeleteDBCluster
 operation:
  Cluster cannot be deleted, it still contains DB instances in non-deleting state.

$ aws rds describe-db-clusters --db-cluster-identifier deleteme-writer-only \
  --query '*[].[DBClusterIdentifier,Status,DBClusterMembers[*].
[DBInstanceIdentifier,IsClusterWriter]]'
[
    [
        "deleteme-writer-only",
        "available",
        [
            [
                "instance-2130",
                true
            ]
        ]
    ]
]

$ aws rds delete-db-instance --db-instance-identifier instance-2130
{
  "DBInstanceIdentifier": "instance-2130",
  "DBInstanceStatus": "deleting",
  "Engine": "aurora-mysql"
}

$ aws rds delete-db-cluster --db-cluster-identifier deleteme-writer-only \
  --skip-final-snapshot \
  --delete-automated-backups
{
  "DBClusterIdentifier": "deleteme-writer-only",
  "Status": "available",
  "Engine": "aurora-mysql"
```

```
}
```

Eliminazione di un cluster Aurora con più istanze database

Se il cluster contiene più istanze database, in genere esiste una singola istanza writer e una o più istanze reader. Le istanze reader aiutano con la disponibilità elevata, rimanendo in standby per assumere il controllo se l'istanza writer riporta un problema. Le istanze reader possono inoltre essere utilizzate per il dimensionamento del cluster in modo da gestire un carico di lavoro a uso intensivo di lettura senza aggiungere sovraccarico all'istanza writer.

Per eliminare un cluster con più istanze database reader, elimina prima le istanze reader, quindi l'istanza writer. L'eliminazione dell'istanza di scrittura lascia il cluster e i relativi dati al loro posto. Il cluster viene eliminato mediante un'azione distinta.

- Per la procedura di eliminazione di un'istanza database Aurora, consulta Eliminazione di un'istanza database da un cluster database Aurora.
- Per la procedura di eliminazione di un'istanza database writer in un cluster Aurora, consulta Eliminazione di un cluster Aurora con una singola istanza database.
- Per la procedura di eliminazione di un cluter Aurora vuoto, consulta Eliminazione di un cluster Aurora vuoto.

In questo esempio della CLI viene illustrato come eliminare un cluster contenente sia un'istanza database di scrittura che una singola istanza database di lettura. L'output di `describe-db-clusters` mostra che `instance-7384` è l'istanza writer e `instance-1039` è l'istanza reader. Nell'esempio viene eliminata prima l'istanza reader, poiché l'eliminazione dell'istanza writer mentre esiste ancora un'istanza reader provocherebbe un'operazione di failover. Non ha senso promuovere l'istanza reader a writer se prevedi di eliminare anche quell'istanza. Di nuovo, supponiamo che queste istanze `db.t2.small` vengano utilizzate solo per lo sviluppo e il test e quindi l'operazione di eliminazione ignora lo snapshot finale e non mantiene i backup automatici.

```
$ aws rds delete-db-cluster --db-cluster-identifier deleteme-writer-and-reader --skip-
final-snapshot

An error occurred (InvalidDBClusterStateFault) when calling the DeleteDBCluster
 operation:
  Cluster cannot be deleted, it still contains DB instances in non-deleting state.

$ aws rds describe-db-clusters --db-cluster-identifier deleteme-writer-and-reader --
output text \
```

```
  --query '*[].["Cluster:",DBClusterIdentifier,DBClusterMembers[*].
["Instance:",DBInstanceIdentifier,IsClusterWriter]]
Cluster:        deleteme-writer-and-reader
Instance:       instance-1039  False
Instance:       instance-7384   True

$ aws rds delete-db-instance --db-instance-identifier instance-1039
{
  "DBInstanceIdentifier": "instance-1039",
  "DBInstanceStatus": "deleting",
  "Engine": "aurora-mysql"
}

$ aws rds delete-db-instance --db-instance-identifier instance-7384
{
  "DBInstanceIdentifier": "instance-7384",
  "DBInstanceStatus": "deleting",
  "Engine": "aurora-mysql"
}

$ aws rds delete-db-cluster --db-cluster-identifier deleteme-writer-and-reader \
  --skip-final-snapshot \
  --delete-automated-backups
{
  "DBClusterIdentifier": "deleteme-writer-and-reader",
  "Status": "available",
  "Engine": "aurora-mysql"
}
```

Nell'esempio seguente viene illustrato come eliminare un cluster database contenente un'istanza database writer e più istanze database reader. Utilizza un output conciso dal comando `describe-db-clusters` per ottenere un report delle istanze writer e reader. Ancora una volta, eliminiamo tutte le istanze database reader prima di eliminare l'istanza database writer. Non importa con quale ordine eliminiamo le istanze database reader.

Supponiamo che questo cluster con diverse istanze database contenga dati che vale la pena conservare. Pertanto, il comando `delete-db-cluster` di questo esempio include i parametri `--no-skip-final-snapshot` e `--final-db-snapshot-identifier` per specificare i dettagli della snapshot da creare. Include anche il parametro `--no-delete-automated-backups` per mantenere i backup automatici.

```
$ aws rds describe-db-clusters --db-cluster-identifier deleteme-multiple-readers --
output text \
  --query '*[].["Cluster:",DBClusterIdentifier,DBClusterMembers[*].
["Instance:",DBInstanceIdentifier,IsClusterWriter]]
Cluster:        deleteme-multiple-readers
Instance:       instance-1010   False
Instance:       instance-5410   False
Instance:       instance-9948   False
Instance:       instance-8451   True

$ aws rds delete-db-instance --db-instance-identifier instance-1010
{
  "DBInstanceIdentifier": "instance-1010",
  "DBInstanceStatus": "deleting",
  "Engine": "aurora-mysql"
}

$ aws rds delete-db-instance --db-instance-identifier instance-5410
{
  "DBInstanceIdentifier": "instance-5410",
  "DBInstanceStatus": "deleting",
  "Engine": "aurora-mysql"
}

$ aws rds delete-db-instance --db-instance-identifier instance-9948
{
  "DBInstanceIdentifier": "instance-9948",
  "DBInstanceStatus": "deleting",
  "Engine": "aurora-mysql"
}

$ aws rds delete-db-instance --db-instance-identifier instance-8451
{
  "DBInstanceIdentifier": "instance-8451",
  "DBInstanceStatus": "deleting",
  "Engine": "aurora-mysql"
}

$ aws rds delete-db-cluster --db-cluster-identifier deleteme-multiple-readers \
  --no-delete-automated-backups \
  --no-skip-final-snapshot \
  --final-db-snapshot-identifier deleteme-multiple-readers-final-snapshot
{
```

```
  "DBClusterIdentifier": "deleteme-multiple-readers",
  "Status": "available",
  "Engine": "aurora-mysql"
}
```

Nell'esempio seguente viene illustrato come confermare che Aurora abbia creato la snapshot richiesta. Puoi richiedere i dettagli per la snapshot specifica specificandone l'identificativo `deleteme-multiple-readers-final-snapshot`. Puoi inoltre ottenere un report di tutte le snapshot per il cluster che è stato eliminato specificando l'identificativo del cluster `deleteme-multiple-readers`. Entrambi i comandi restituiscono informazioni sulla stessa snapshot.

```
$ aws rds describe-db-cluster-snapshots \
  --db-cluster-snapshot-identifier deleteme-multiple-readers-final-snapshot
{
    "DBClusterSnapshots": [
        {
            "AvailabilityZones": [],
            "DBClusterSnapshotIdentifier": "deleteme-multiple-readers-final-snapshot",
            "DBClusterIdentifier": "deleteme-multiple-readers",
            "SnapshotCreateTime": "11T01:40:07.354000+00:00",
            "Engine": "aurora-mysql",
...

$ aws rds describe-db-cluster-snapshots --db-cluster-identifier deleteme-multiple-
readers
{
    "DBClusterSnapshots": [
        {
            "AvailabilityZones": [],
            "DBClusterSnapshotIdentifier": "deleteme-multiple-readers-final-snapshot",
            "DBClusterIdentifier": "deleteme-multiple-readers",
            "SnapshotCreateTime": "11T01:40:07.354000+00:00",
            "Engine": "aurora-mysql",
...
```

Protezione dall'eliminazione per i cluster Aurora

Non puoi eliminare cluster per i quali è abilitata la protezione dall'eliminazione. Puoi eliminare istanze database all'interno del cluster, ma non il cluster stesso. In questo modo, il volume del cluster contenente tutti i dati è al sicuro dall'eliminazione accidentale. Aurora applica la protezione

dall'eliminazione per un cluster DB indipendentemente dal fatto che si tenti di eliminare il cluster utilizzando la console, l'API RDS o l' AWS CLI API RDS.

La protezione da eliminazione viene abilitata per impostazione predefinita quando crei un cluster di database di produzione con la Console di gestione AWS. Tuttavia, la protezione dall'eliminazione è disabilitata per impostazione predefinita se si crea un cluster utilizzando l' AWS CLI API o. L'attivazione o la disattivazione della protezione contro l'eliminazione non causa un'interruzione. Per poter eliminare il cluster, modifica il cluster e disabilita la protezione dall'eliminazione. Per ulteriori informazioni sull'attivazione e sulla disattivazione della protezione da eliminazione, consulta Modifica del cluster di database tramite la console, la CLI e l'API.

> **Tip**
>
> Anche se tutte le istanze database sono state eliminate, potrai comunque accedere ai dati creando una nuova istanza database nel cluster.

Eliminazione di un cluster Aurora arrestato

Non puoi eliminare un cluster se si trova nello stato `stopped`. In questo caso, avvia il cluster prima di eliminarlo. Per ulteriori informazioni, consulta Avvio di un cluster di database Aurora.

Eliminazione di cluster Aurora MySQL che sono repliche di lettura

Con Aurora MySQL, non puoi eliminare un'istanza database in un cluster di database se si verificano entrambe le condizioni seguenti:

- Il cluster database è una replica di lettura di un altro cluster di database Aurora.
- L'istanza database è l'unica istanza nel cluster database.

Per eliminare un'istanza database in questo caso, prima di tutto promuovi il cluster di database in modo che non sia più una replica di lettura. Al termine della promozione, puoi eliminare l'istanza database finale nel cluster database. Per ulteriori informazioni, consulta Repliche di cluster di database Amazon Aurora MySQL tra Regioni AWS.

Lo snapshot finale durante l'eliminazione di un cluster

In questa sezione, gli esempi mostrano come è possibile scegliere se eseguire un'istantanea finale quando si elimina un cluster Aurora. Se si sceglie di creare uno snapshot finale ma il nome

specificato corrisponde a uno snapshot esistente, l'operazione si interrompe con un errore. In questo caso, esamina i dettagli dello snapshot per confermare se rappresenta il dettaglio corrente o se si tratta di uno snapshot precedente. Se lo snapshot esistente non contiene i dati più recenti che desideri conservare, rinomina lo snapshot e riprova oppure specifica un nome diverso per il parametro snapshot finale.

Eliminazione di un'istanza database da un cluster database Aurora

Puoi eliminare un'istanza database da un cluster database Aurora come parte del processo di eliminazione dell'intero cluster. Se il cluster contiene un certo numero di istanze database, l'eliminazione del cluster richiede l'eliminazione di ogni singola istanza database. Puoi eliminare una o più istanze reader da un cluster anche se il cluster in esecuzione. Questa operazione può essere effettuata per ridurre la capacità di calcolo e i costi associati se il cluster non è occupato.

Per eliminare un'istanza database, devi specificare il nome dell'istanza.

È possibile eliminare un'istanza DB utilizzando Console di gestione AWS AWS CLI, the o l'API RDS.

> **Note**
>
> Quando una replica Aurora viene eliminata, l'endpoint dell'istanza viene rimosso immediatamente e la replica Aurora viene rimossa dall'endpoint di lettura. Se sono presenti istruzioni in esecuzione nella replica Aurora da eliminare, è disponibile un periodo di tolleranza di tre minuti. L'esecuzione delle istruzioni esistenti può essere completata correttamente durante questo periodo di tolleranza. Quando finisce il periodo di tolleranza, la replica Aurora viene chiusa ed eliminata.

Per i cluster database Aurora, l'eliminazione di un'istanza database non comporta necessariamente l'eliminazione dell'intero cluster. Puoi eliminare un'istanza database in un cluster Aurora per ridurre la capacità di calcolo e i costi associati quando il cluster non è occupato. Per informazioni sulle circostanze speciali per i cluster Aurora che dispongono di un'istanza database o zero istanze database, consulta Eliminazione di un cluster Aurora con una singola istanza database e Eliminazione di un cluster Aurora vuoto.

> **Note**
>
> Non puoi eliminare un'istanza database se è abilitata la protezione dall'eliminazione. Per ulteriori informazioni, consulta Protezione dall'eliminazione per i cluster Aurora.

Puoi disabilitare la protezione dall'eliminazione modificando l'istanza database. Per ulteriori informazioni, consulta Modifica di un cluster database Amazon Aurora.

Console

Per eliminare un'istanza database in un cluster database

1. Accedi a Console di gestione AWS e apri la console Amazon RDS all'indirizzo https://console.aws.amazon.com/rds/.
2. Nel riquadro di navigazione, scegliere Databases (Database) e selezionare l'istanza database da eliminare.
3. In Actions (Azioni), selezionare Delete (Elimina).
4. Immettere **`delete me`** nella casella.
5. Scegliere Delete (Elimina).

AWS CLI

Per eliminare un'istanza DB utilizzando il AWS CLI, chiama il delete-db-instancecomando e specifica il `--db-instance-identifier` valore.

Example

Per Linux, macOS o Unix:

```
aws rds delete-db-instance \
    --db-instance-identifier mydbinstance
```

Per Windows:

```
aws rds delete-db-instance ^
    --db-instance-identifier mydbinstance
```

API RDS

Per eliminare un'istanza DB utilizzando l'API Amazon RDS, chiama l'DBInstanceoperazione Delete e specifica il `DBInstanceIdentifier` parametro.

> **Note**
>
> Quando lo stato di un'istanza DB è`deleting`, il relativo valore del certificato CA non viene visualizzato nella console RDS o nell'output dei AWS CLI comandi o delle operazioni dell'API RDS. Per ulteriori informazioni sui certificati CA, consulta Utilizzo SSL/TLS per crittografare una connessione a un'.

Applicazione di tag alle risorse Amazon Aurora e Amazon RDS

Un tag Amazon RDS è una coppia nome-valore definita e associata a una risorsa Amazon RDS, ad esempio un'istanza database o uno snapshot database. Il nome viene definito chiave. Facoltativamente, è possibile fornire un valore per la chiave.

Puoi utilizzare l'Console di gestione AWSAPI Amazon RDS o Amazon RDS per aggiungere, elencare ed eliminare tag sulle risorse Amazon RDS. AWS CLI Quando usi l'interfaccia della linea di comando o l'API, devi fornire il nome della risorsa Amazon (ARN) della risorsa RDS che vuoi utilizzare. Per ulteriori informazioni sulla creazione di un ARN, consultare Costruzione di un ARN per Amazon RDS.

È possibile usare i tag per aggiungere metadati alle risorse Aurora e Amazon RDS. Puoi utilizzare i tag per aggiungere notazioni personalizzate su istanze di database, snapshot, cluster Aurora e così via. In questo modo puoi documentare le risorse Aurora e Amazon RDS. I tag possono inoltre essere utilizzati con procedure di manutenzione automatizzate.

In particolare, puoi usare questi tag con le policy IAM per gestire l'accesso alle risorse Aurora e Amazon RDS e per controllare le azioni che è possibile applicare a tali risorse. Puoi utilizzare questi tag anche per tenere traccia dei costi raggruppando le spese per risorse con tag simili.

Puoi applicare i tag alle seguenti risorse Aurora e Amazon RDS:

- Istanze DB
- Cluster database
- Cluster globali Aurora
- Endpoint del cluster di database.
- Repliche di lettura
- Snapshot DB
- Snapshot cluster database
- Istanze database riservate
- Abbonamenti a eventi
- Gruppi di opzioni database
- Gruppi di parametri database
- Gruppi di parametri di cluster database
- Gruppi di sottoreti database

- Proxy RDS
- Endpoint RDS Proxy
- Implementazioni blu/verdi
- Integrazioni Zero-ETL
- Backup automatizzati
- Backup automatici in cluster

> **Note**
>
> Quando tagghi un'istanza DB, Aurora applica automaticamente tali tag alle risorse Performance Insights associate. Al momento, non è possibile etichettare proxy RDS e endpoint proxy RDS utilizzando il. Console di gestione AWS

Argomenti

- Perché utilizzare i tag delle risorse Amazon RDS?
- In che modo funzionano i tag delle risorse Amazon RDS
- Best practice per l'utilizzo dei tag delle risorse Amazon RDS
- Copia di tag in snapshot di cluster database
- Etichettatura delle risorse di backup automatizzate
- Aggiunta ed eliminazione di tag in Amazon RDS
- Tutorial: utilizza i tag per specificare i cluster DB Aurora da arrestare

Perché utilizzare i tag delle risorse Amazon RDS?

È possibile utilizzare i tag per le seguenti operazioni:

- Suddividere le risorse RDS in categorie in base all'applicazione, al progetto, al reparto, all'ambiente e così via. Ad esempio, è possibile utilizzare una chiave di tag per definire una categoria, in cui il valore del tag è un elemento della categoria. È possibile creare il tag `environment=prod`. Oppure è possibile definire una chiave di tag `project` e un valore del tag `Salix`, che indica che una risorsa Amazon RDS è assegnata al progetto Salix.
- Automatizzare le attività di gestione delle risorse. Ad esempio, è possibile creare una finestra di manutenzione per le istanze con tag `environment=prod` diversa dalla finestra per le istanze

con tag `environment=test`. È inoltre possibile configurare snapshot database automatici per le istanze con tag `environment=prod`.

- Controllare l'accesso alle risorse RDS in una policy IAM. Per farlo, devi utilizzare la chiave di condizione globale `aws:ResourceTag/tag-key`. Ad esempio, una policy potrebbe consentire solo agli utenti del gruppo `DBAdmin` di modificare le istanze database con tag `environment=prod`. Per informazioni sulla gestione dell'accesso alle risorse con tag con le policy IAM, consulta Gestione accessi e identità per Amazon Aurora Controlling access to AWS resources nella AWSIdentity and Access Management User Guide.
- Monitorare le risorse in base a un tag. Ad esempio, puoi creare una CloudWatch dashboard Amazon per le istanze DB contrassegnate con`environment=prod`.
- Puoi utilizzare questi tag anche per tenere traccia dei costi raggruppando le spese per risorse con tag simili. Ad esempio, se si applica il tag `project=Salix` alle risorse RDS associate al progetto Salix, è possibile generare report sui costi e allocare le spese per questo progetto. Per ulteriori informazioni, consulta Come funziona la AWS fatturazione con i tag in Amazon RDS.

In che modo funzionano i tag delle risorse Amazon RDS

AWSnon applica alcun significato semantico ai tag. I tag sono interpretati prettamente come stringhe di caratteri.

Argomenti

- Set di tag in Amazon RDS
- Struttura di tag in Amazon RDS
- Risorse Amazon RDS idonee all'applicazione di tag
- Come funziona la AWS fatturazione con i tag in Amazon RDS

Set di tag in Amazon RDS

Ogni risorsa Amazon RDS dispone di un container chiamato set di tag. Il container include tutti i tag assegnati alla risorsa. Una risorsa ha esattamente un set di tag.

Un set di tag contiene da 0 a 50 tag. Se aggiungi un tag a una risorsa RDS con la stessa chiave di un tag esistente per la risorsa, il nuovo valore sovrascrive quello precedente.

Struttura di tag in Amazon RDS

La struttura di un tag RDS è la seguente:

Chiave tag

La chiave è il nome obbligatorio del tag. Il valore della stringa deve essere composto da 1 a 128 caratteri Unicode e non può avere il prefisso `aws:` o `rds:`. La stringa può contenere solo il set di lettere, cifre, spazi vuoti, `_`, `.`, `:`, `/`, `=`, `+`, `-` e `@` Unicode. L'espressione regolare Java è `"^([\\p{L}\\p{Z}\\p{N}_.:/=+\\-@]*)$"`. Le chiavi dei tag prevedono una distinzione tra lettere maiuscole e minuscole. Pertanto, `project` e `Project` sono chiavi distinte.

La chiave è univoca per un set di tag. Ad esempio, non è possibile avere una coppia di chiavi in un set di tag con la chiave uguale e valori diversi, come `project=Trinity` e `project=Xanadu`.

Valore di tag

Il valore è un valore di stringa facoltativo del tag. Il valore di stringa deve essere composto da 1 a 256 caratteri Unicode. La stringa può contenere solo il set di lettere, cifre, spazi vuoti, `_`, `.`, `:`, `/`, `=`, `+`, `-` e `@` Unicode. L'espressione regolare Java è `"^([\\p{L}\\p{Z}\\p{N}_.:/=+\\-@]*)$"`. I valori di tag fanno distinzione tra maiuscole e minuscole. Pertanto, `prod` e `Prod` sono valori distinti.

I valori non devono essere necessariamente univoci in un set di tag e possono essere null. Ad esempio, puoi avere una coppia chiave-valore in un set di tag `project=Trinity` e `cost-center=Trinity`.

Risorse Amazon RDS idonee all'applicazione di tag

Puoi contrassegnare con i tag le seguenti risorse Amazon RDS:

- Istanze DB
- Cluster database
- Endpoint del cluster di database.
- Repliche di lettura
- Snapshot DB
- Snapshot cluster database

- Istanze database riservate
- Abbonamenti a eventi
- Gruppi di opzioni database
- Gruppi di parametri database
- Gruppi di parametri di cluster database
- Gruppi di sottoreti database
- Proxy RDS
- Endpoint RDS Proxy

> **Note**
>
> Al momento, non è possibile taggare i proxy RDS e gli endpoint RDS Proxy utilizzando la Console di gestione AWS.

- Implementazioni blu/verdi
- Integrazioni Zero-ETL (anteprima)
- Backup automatizzati
- Backup automatici in cluster

Come funziona la AWS fatturazione con i tag in Amazon RDS

Utilizza i tag per organizzare la AWS fattura in modo da rispecchiare la tua struttura dei costi. A tale scopo, registrati per ricevere la Account AWS fattura con i valori chiave dell'etichetta inclusi. Per visualizzare il costo delle risorse combinate, puoi organizzare le informazioni di fatturazione in base alle risorse con gli stessi valori di chiave di tag. Puoi ad esempio applicare tag a numerose risorse con un nome di applicazione specifico, quindi organizzare le informazioni di fatturazione per visualizzare il costo totale dell'applicazione in più servizi. Per ulteriori informazioni, consulta la pagina sull'utilizzo dei tag per l'allocazione dei costi nella Guida per l'utente di AWS Billing.

In che modo funzionano i tag di allocazione dei costi con snapshotcluster di database

È possibile aggiungere un tag a uno snapshot di cluster di database. Tuttavia, l'addebito non rifletterà questo raggruppamento. Per applicare i tag di allocazione dei costi agli snapshot di cluster di database, è necessario che siano soddisfatte le condizioni elencate di seguito:

- I tag devono essere collegati all’istanza database principale.

- L'istanza DB principale deve esistere nella Account AWS stessa istantanea del cluster DB.
- L'istanza DB principale deve esistere nella Regione AWS stessa istantanea del cluster DB.

Gli snapshot di cluster di database sono considerati orfani se non sono presenti nella stessa Regione dell'istanza database principale o se l'istanza database principale viene eliminata. Gli snapshot DB orfani non supportano i tag di allocazione dei costi. I costi degli snapshot orfani vengono aggregati in un'unica voce senza tag. Gli snapshot di cluster di database multi-account non sono considerati orfani quando vengono soddisfatte le seguenti condizioni:

- Esistono nella stessa Regione dell'istanza database principale.
- L'istanza database principale è di proprietà dell'account di origine.

> **Note**
>
> Se l'istanza database principale è di proprietà di un account diverso, i tag di allocazione dei costi non si applicano agli snapshot multi-account nell'account di destinazione.

Best practice per l'utilizzo dei tag delle risorse Amazon RDS

Quando si utilizzano i tag, è consigliabile attenersi alle seguenti best practice:

- Documentare le convenzioni per l'utilizzo dei tag che devono essere seguite da tutti i team dell'organizzazione. In particolare, è necessario assicurarsi che i nomi siano descrittivi e coerenti. Ad esempio, standardizzare il formato `environment:prod` anziché utilizzare il tag `env:production` per alcune risorse.

> **Important**
>
> Non memorizzare informazioni personali identificabili o altre informazioni riservate o sensibili nei tag.

- Automatizzare l'utilizzo dei tag per garantire la coerenza. Ad esempio, è possibile utilizzare le seguenti tecniche:
  - Includi tag in un CloudFormation modello. Quando le risorse vengono create con il modello, i tag vengono automaticamente applicati.
  - Definisci e applica i tag utilizzando AWS Lambda le funzioni.

  - Creare un documento SSM che includa i passaggi per aggiungere i tag alle risorse RDS.
- Utilizzare i tag solo quando necessario. È possibile aggiungere fino a 50 tag per una singola risorsa RDS, ma è consigliabile evitare la proliferazione e la complessità non necessarie dei tag.
- Controllare periodicamente i tag per verificarne la pertinenza e la precisione. Rimuovere o modificare i tag obsoleti secondo necessità.
- Prendi in considerazione la possibilità di creare AWS tag con il Tag Editor inConsole di gestione AWS. Puoi utilizzare Tag Editor per aggiungere tag a più AWS risorse supportate, incluse le risorse RDS, contemporaneamente. Per ulteriori informazioni, consulta Tag Editor nella Guida per l'utente di AWS Resource Groups.

Copia di tag in snapshot di cluster database

Quando si crea o si ripristina un cluster di database, è possibile specificare che i tag del cluster vengano copiati negli snapshot del cluster di database. La copia dei tag garantisce che i metadati per gli snapshot DB corrispondano a quelli del cluster database di origine. Garantisce inoltre che tutte le policy di accesso per lo snapshot DB corrispondano a quelle del cluster database di origine. I tag non vengono copiati per impostazione predefinita.

È possibile specificare che i tag vengano copiati nelle snapshot DB per le seguenti azioni:

- Creazione di un cluster di database
- Ripristino di un cluster di database
- Creazione di una replica di lettura
- Copia di una snapshot cluster database

> **Note**
>
> In alcuni casi, è possibile includere un valore per il `--tags` parametro del create-db-snapshotAWS CLIcomando. Oppure potresti fornire almeno un tag all'operazione Create DBSnapshot API. In questi casi, RDS non copia i tag dall'istanza database di origine nel nuovo snapshot DB. Questa funzionalità si applica anche se per l'istanza database di origine è stata attivata l'opzione `--copy-tags-to-snapshot` (`CopyTagsToSnapshot`).
> Con questo approccio puoi creare la copia di un'istanza database da uno snapshot DB ed evitare di aggiungere tag che non si applicano alla nuova istanza database. È possibile creare lo snapshot DB utilizzando il AWS CLI `create-db-snapshot` comando (o

l'operazione API `CreateDBSnapshot` RDS). Dopo aver creato uno snapshot DB, puoi aggiungere i tag come descritto più avanti in questo argomento.

Etichettatura delle risorse di backup automatizzate

Le risorse di backup automatizzate vengono create quando si imposta il valore del periodo di conservazione dei backup da 0 a un valore superiore a 0. È possibile etichettare le risorse di backup automatizzate dell'istanza o del cluster durante la creazione utilizzando il `--tag-specifications` parametro.

Parametro Tag-Specifications

APIs che supportano il parametro di `--tag-specifications` richiesta (come restore-db-instance-from-db-snapshot create-db-instancecreate-db-cluster, ecc.) possono etichettare i backup automatici (Resource Type: or) durante la creazione. `auto-backup` `cluster-auto-backup`

Etichettatura dei backup automatici del cluster

Da utilizzare `--tag-specifications` con `ResourceType=cluster-auto-backup` quando si creano cluster DB con backup automatici abilitati.

> **Note**
>
> - I tag di backup automatici sono indipendenti dai tag dell'istanza DB di origine, dai tag del cluster DB o dai tag degli snapshot DB.

Aggiunta ed eliminazione di tag in Amazon RDS

È possibile effettuare le seguenti operazioni:

- Crea tag quando crei una risorsa, ad esempio quando esegui il AWS CLI comando`create-db-instance`.
- Aggiungere tag a una risorsa esistente utilizzando il comando `add-tags-to-resource`.
- Elencare i tag associati a una risorsa specifica utilizzando il comando `list-tags-for-resource`.

- Aggiornare i tag utilizzando il comando `add-tags-to-resource`.
- Rimuovere i tag da una risorsa utilizzando il comando `remove-tags-from-resource`.

Le procedure seguenti illustrano come eseguire operazioni di applicazione di tag tipiche sulle risorse correlate alle istanze database e ai cluster di database Aurora. Tieni presente che i tag sono memorizzati nella cache a fini di autorizzazione. Per questo motivo, quando si aggiungono o si aggiornano tag sulle risorse Amazon RDS, possono trascorrere diversi minuti prima che le modifiche siano disponibili.

Console

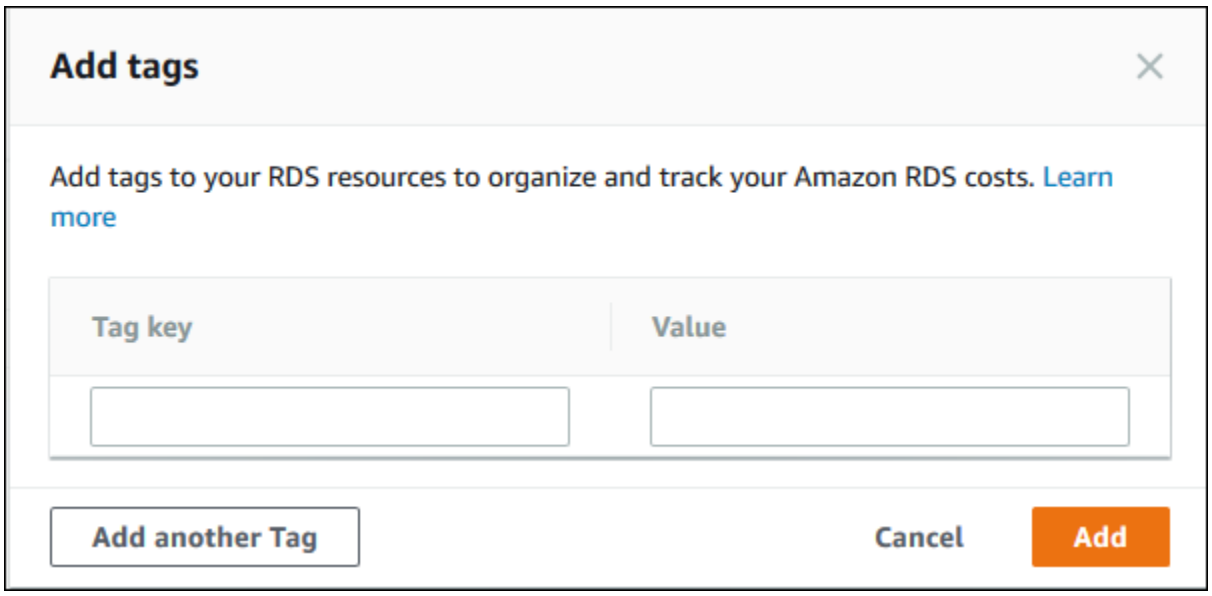
Il processo di applicazione dei tag a una risorsa di Amazon RDS è simile per tutte le risorse. Di seguito viene mostrato come applicare i tag a un'istanza database Amazon RDS.

Aggiunta di un tag a un'istanza database

1. Accedi a Console di gestione AWS e apri la console Amazon RDS all'indirizzo https://console.aws.amazon.com/rds/.
2. Nel pannello di navigazione, seleziona Database.

   > **Note**
   >
   > Per filtrare l'elenco di istanze database nel riquadro Databases (Database), inserire una stringa di testo per Filter instances (Filtra istanze). Vengono visualizzate solo le istanze database che contengono la stringa.

3. Scegliere il nome dell'istanza database a cui si desidera aggiungere tag per visualizzarne i dettagli.
4. Nella sezione dei dettagli, scorrere verso il basso fino alla sezione Tags (Tag).
5. Scegliere Aggiungi. Viene visualizzata la finestra Add tags (Aggiungi tag).



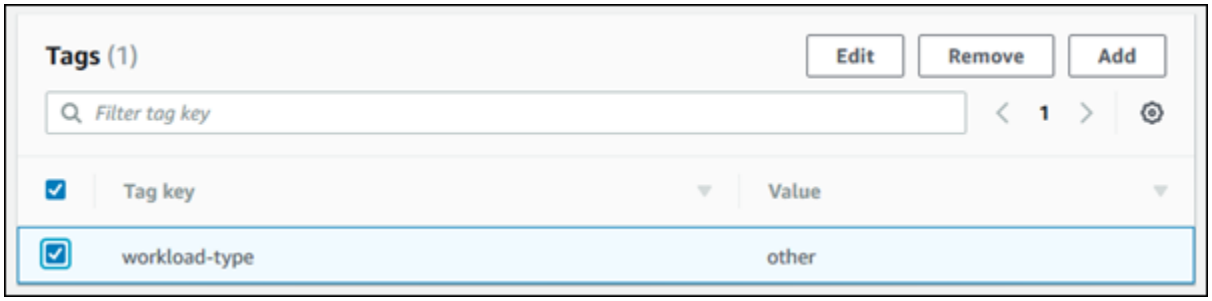
6. Inserire un valore per Tag key (Chiave tag) e Value (Valore).
7. Per aggiungere un altro tag, scegliere Add another Tag (Aggiungi un altro tag) e inserire un valore per Tag key (Chiave tag) e Value (Valore).

   Ripetere questa operazione tutte le volte necessarie.
8. Scegliere Aggiungi.

Eliminazione di un tag da un'istanza database

1. Accedi a Console di gestione AWS e apri la console Amazon RDS all'indirizzo https://console.aws.amazon.com/rds/.
2. Nel pannello di navigazione, seleziona Database.

   > **Note**
   >
   > Per filtrare l'elenco di istanze database nel riquadro Databases (Database), inserire una stringa di testo nella casella Filter instances (Filtra istanze). Vengono visualizzate solo le istanze database che contengono la stringa.
3. Scegliere il nome dell'istanza database per visualizzarne i dettagli.
4. Nella sezione dei dettagli, scorrere verso il basso fino alla sezione Tags (Tag).
5. Scegliere il tag da eliminare.



6. Scegliere Delete (Elimina) e quindi scegliere Delete (Elimina) nella finestra Delete tags (Elimina tag).

AWS CLI

È possibile aggiungere, elencare o rimuovere i tag per un'istanza database utilizzando AWS CLI.

- Per aggiungere uno o più tag a una risorsa Amazon RDS, usa il AWS CLI comando `add-tags-to-resource`.
- Per elencare i tag su una risorsa Amazon RDS, usa il AWS CLI comando `list-tags-for-resource`.
- Per rimuovere uno o più tag da una risorsa Amazon RDS, usa il AWS CLI comando `remove-tags-from-resource`.

Per ulteriori informazioni su come creare l'ARN necessario, consultare Costruzione di un ARN per Amazon RDS.

API RDS

È possibile aggiungere, elencare o rimuovere i tag per un'istanza database utilizzando l'API di Amazon RDS.

- Per aggiungere un tag a una risorsa Amazon RDS, utilizza l'operazione `AddTagsToResource`.
- Per elencare i tag assegnati a una risorsa Amazon RDS, utilizza `ListTagsForResource`.
- Per rimuovere i tag da una risorsa Amazon RDS, utilizza l'operazione `RemoveTagsFromResource`.

Per ulteriori informazioni su come creare l'ARN necessario, consultare Costruzione di un ARN per Amazon RDS.

Quando utilizzi XML con l'API di Amazon RDS i tag seguono questo schema:

```
<Tagging>
      <TagSet>
          <Tag>
              <Key>Project</Key>
              <Value>Trinity</Value>
          </Tag>
          <Tag>
              <Key>User</Key>
              <Value>Jones</Value>
          </Tag>
      </TagSet>
  </Tagging>
```

La tabella riportata di seguito fornisce un elenco dei tag XML consentiti e le relative caratteristiche. I valori per `Key` e `Value` rispettano la distinzione tra maiuscole e minuscole. Ad esempio, `project=Trinity` e `PROJECT=Trinity` sono tag distinti.

Elemento del tagging	Description
TagSet	Un set di tag è un contenitore di tutti i tag assegnati a una risorsa Amazon RDS. Ogni risorsa può disporre di un solo set di tag. Puoi lavorare con un TagSet solo tramite l'API Amazon RDS.
Tag	Un tag è una coppia chiave-valore definita dall'utente. Un set di tag può contenere da 1 a 50 tag.
Chiave	Una chiave corrisponde al nome obbligatorio del tag. Per le restrizioni, consulta Struttura di tag in Amazon RDS. Il valore della stringa può essere composto da 1 a 128 caratteri Unicode e non può avere il prefisso `aws:` o `rds:`. La stringa può contenere solo il set di lettere, cifre, spazi vuoti Unicode, '_', '.', '/', '=', '+', '-' (espressioni regolari Java: "^([\\p{L}\\p{Z}\\p{N}_.:/=+\\-]*)$"). Le chiavi devono essere uniche per un set di tag. Ad esempio, non è possibile avere una coppia di chiavi in un set di tag con la stessa chiave ma con valori diversi, ad esempio Project/Xanadu. project/Trinity

Elemento del tagging	Description
Valore	Un valore è il valore opzione del tag. Per le restrizioni, consulta Struttura di tag in Amazon RDS. Il valore della stringa può essere composto da 1 a 256 caratteri Unicode e non può avere il prefisso `aws:` o `rds:`. La stringa può contenere solo il set di lettere, cifre, spazi vuoti Unicode, '_', '.', '/', '=', '+', '-' (espressioni regolari Java: "^([\\p{L}\\p{Z}\\p{N}_.:/=+\\-]*)$"). I valori non devono essere necessariamente univoci in un set di tag e possono essere Null. Ad esempio, è possibile avere una coppia chiave-valore in un set di tag di Cost-center/Trinity. project/Trinity

Tutorial: utilizza i tag per specificare i cluster DB Aurora da arrestare

Si assuma di creare un numero di cluster database Aurora in un ambiente di sviluppo o di test. È necessario mantenere tutti questi cluster per diversi giorni. Alcuni cluster eseguono test durante la notte. Altri cluster possono essere fermati durante la notte e ricominciare il giorno successivo. L'esempio seguente mostra come assegnare un tag a quei cluster che possono essere arrestati durante la notte. L'esempio mostra come uno script può rilevare i cluster che hanno quel tag e quindi arrestarli. In questo esempio, la parte del valore della coppia chiave-valore non ha importanza. La presenza del tag `stoppable` indica che il cluster dispone di questa proprietà definita dall'utente.

Per specificare quali cluster DB Aurora arrestare

1. Determina l'ARN di un cluster da indicare come arrestabile.

   I comandi e le API per l'assegnazione dei tag funzionano con gli ARN. In questo modo, possono funzionare perfettamente in regioni AWS, account AWS e con tipi di risorse differenti che potrebbero avere nomi brevi identici. È possibile specificare l'ARN anziché l'ID cluster nei comandi della CLI che operano sui cluster. Sostituisci il nome del cluster per *`dev-test-cluster`*. Nei comandi successivi che utilizzano i parametri ARN, sostituisci l'ARN del tuo cluster. L'ARN include il tuo ID account AWS e il nome della regione AWS in cui si trova il cluster.

   ```
   $ aws rds describe-db-clusters --db-cluster-identifier dev-test-cluster \
   ```

```
  --query "*[].{DBClusterArn:DBClusterArn}" --output text
arn:aws:rds:us-east-1:123456789:cluster:dev-test-cluster
```

2. Aggiungi il tag `stoppable` a questo cluster.

   Scegli il nome per il tag. Con questo approccio puoi evitare di definire una convenzione di denominazione che codifichi tutte le informazioni rilevanti nei nomi. Con la convenzione puoi codificare le informazioni nel nome dell'istanza database o nei nomi di altre risorse. Poiché questo esempio considera il tag come un attributo presente o assente, viene omessa la parte `Value=` del parametro `--tags`.

```
$ aws rds add-tags-to-resource \
  --resource-name arn:aws:rds:us-east-1:123456789:cluster:dev-test-cluster \
  --tags Key=stoppable
```

3. Verifica che il tag sia presente nel cluster.

   Questi comandi recuperano le informazioni sui tag per il cluster in formato JSON e in testo semplice separato da tabulazioni.

```
$ aws rds list-tags-for-resource \
  --resource-name arn:aws:rds:us-east-1:123456789:cluster:dev-test-cluster
{
    "TagList": [
        {
            "Key": "stoppable",
            "Value": ""

        }
    ]
}
$ aws rds list-tags-for-resource \
  --resource-name arn:aws:rds:us-east-1:123456789:cluster:dev-test-cluster --output
 text
TAGLIST stoppable
```

4. Per arrestare tutti i cluster designati come `stoppable`, prepara un elenco di tutti i cluster. Scorri l'elenco e verifica se ogni cluster è contrassegnato con l'attributo pertinente.

   In questo esempio di Linux viene utilizzato lo script della shell per salvare l'elenco degli ARN del cluster in un file temporaneo e quindi vengono eseguiti i comandi della CLI per ogni cluster.

```
$ aws rds describe-db-clusters --query "*[].[DBClusterArn]" --output text >/tmp/
cluster_arns.lst
$ for arn in $(cat /tmp/cluster_arns.lst)
do
  match="$(aws rds list-tags-for-resource --resource-name $arn --output text | grep
 'TAGLIST\tstoppable')"
  if [[ ! -z "$match" ]]
  then
      echo "Cluster $arn is tagged as stoppable. Stopping it now."
# Note that you can specify the full ARN value as the parameter instead of the
 short ID 'dev-test-cluster'.
      aws rds stop-db-cluster --db-cluster-identifier $arn
  fi
done

Cluster arn:aws:rds:us-east-1:123456789:cluster:dev-test-cluster is tagged as
 stoppable. Stopping it now.
{
    "DBCluster": {
        "AllocatedStorage": 1,
        "AvailabilityZones": [
            "us-east-1e",
            "us-east-1c",
            "us-east-1d"
        ],
        "BackupRetentionPeriod": 1,
        "DBClusterIdentifier": "dev-test-cluster",
        ...
```

Puoi eseguire uno script come questo alla fine di ogni giorno per assicurarti che i cluster non essenziali vengano arrestati. Puoi inoltre pianificare un processo utilizzando un'utilità come `cron` per eseguire tale controllo ogni notte. Ad esempio, puoi eseguire questa operazione nel caso in cui alcuni cluster venissero lasciati in esecuzione per errore. Puoi quindi ottimizzare il comando che prepara l'elenco di cluster da controllare.

Il comando seguente produce un elenco di cluster che si trovano nello stato `available`. Lo script può ignorare i cluster che sono già stati arrestati, perché avranno valori di stato diversi, ad esempio `stopped` o `stopping`.

```
$ aws rds describe-db-clusters \
```

```
  --query '*[].{DBClusterArn:DBClusterArn,Status:Status}|[?Status == `available`]|[].
{DBClusterArn:DBClusterArn}' \
  --output text
arn:aws:rds:us-east-1:123456789:cluster:cluster-2447
arn:aws:rds:us-east-1:123456789:cluster:cluster-3395
arn:aws:rds:us-east-1:123456789:cluster:dev-test-cluster
arn:aws:rds:us-east-1:123456789:cluster:pg2-cluster
```

Tip

Puoi utilizzare l'assegnazione di tag e la ricerca di cluster con i tag per ridurre i costi in altri modi. Supponi, ad esempio, uno scenario in cui i cluster database Aurora sono utilizzati per lo sviluppo e il test. In questo caso, potresti designare alcuni cluster da eliminare alla fine di ogni giornata o eliminare solo le istanze database del lettore. In alternativa, potresti designarli per modificare le istanze database in classi di istanza database di piccole dimensioni durante i periodi previsti di basso utilizzo.

Nome della risorsa Amazon (ARN) in Amazon RDS

Le risorse create in Amazon Web Services sono identificate in modo univoco con un nome della risorsa Amazon (ARN). Per determinate operazioni Amazon RDS, è necessario identificare in modo univoco una risorsa Amazon RDS specificandone l'ARN. Quando, ad esempio, crei una replica di lettura di un'istanza database di RDS, devi fornire l'ARN dell'istanza database di origine.

Per informazioni sulla costruzione di un ARN e sul recupero di un ARN esistente, consulta i seguenti argomenti.

Argomenti

- Costruzione di un ARN per Amazon RDS
- Recupero di un ARN esistente per Amazon RDS

Costruzione di un ARN per Amazon RDS

Le risorse create in Amazon Web Services sono identificate in modo univoco con un nome della risorsa Amazon (ARN). È possibile creare un ARN per una risorsa Amazon RDS utilizzando la sintassi seguente.

`arn:aws:rds:<region>:<account number>:<resourcetype>:<name>`

Per le risorse di cluster globali, l'ARN non include:. Regione AWS `arn:aws:rds::<account number>:global-cluster:<name>` ARNs perché i cluster globali non compaiono in. Console di gestione AWS

Nome della regione	Regione	Endpoint	Protocollo	
US East (Ohio)	us-east-2	rds.us-east-2.amazonaws.com	HTTPS	
		rds-fips.us-east-2.api.aws	HTTPS	
		rds.us-east-2.api.aws	HTTPS	
		rds-fips.us-east-2.amazonaws.com	HTTPS	

Nome della regione	Regione	Endpoint	Protocollo
US East (N. Virginia)	us-east-1	rds.us-east-1.amazonaws.com	HTTPS
		rds-fips.us-east-1.api.aws	HTTPS
		rds-fips.us-east-1.amazonaws.com	HTTPS
		rds.us-east-1.api.aws	HTTPS
Stati Uniti occidentali (California settentrionale)	us-west-1	rds.us-west-1.amazonaws.com	HTTPS
		rds.us-west-1.api.aws	HTTPS
		rds-fips.us-west-1.amazonaws.com	HTTPS
		rds-fips.us-west-1.api.aws	HTTPS
US West (Oregon)	us-west-2	rds.us-west-2.amazonaws.com	HTTPS
		rds-fips.us-west-2.amazonaws.com	HTTPS
		rds.us-west-2.api.aws	HTTPS
		rds-fips.us-west-2.api.aws	HTTPS
Africa (Cape Town)	af-south-1	rds.af-south-1.amazonaws.com	HTTPS
		rds.af-south-1.api.aws	HTTPS
Asia Pacifico (Hong Kong)	ap-east-1	rds.ap-east-1.amazonaws.com	HTTPS
		rds.ap-east-1.api.aws	HTTPS

Nome della regione	Regione	Endpoint	Protocollo
Asia Pacific (Hyderabad)	ap-south-2	rds.ap-south-2.amazonaws.com rds.ap-south-2.api.aws	HTTPS HTTPS
Asia Pacifico (Giacarta)	ap-southeast-3	rds.ap-southeast-3.amazonaws.com rds.ap-southeast-3.api.aws	HTTPS HTTPS
Asia Pacifico (Malesia)	ap-southeast-5	rds.ap-southeast-5.amazonaws.com	HTTPS
Asia Pacifico (Melbourne)	ap-southeast-4	rds.ap-southeast-4.amazonaws.com rds.ap-southeast-4.api.aws	HTTPS HTTPS
Asia Pacifico (Mumbai)	ap-south-1	rds.ap-south-1.amazonaws.com rds.ap-south-1.api.aws	HTTPS HTTPS
Asia Pacifico (Nuova Zelanda)	ap-southeast-6	rds.ap-southeast-6.amazonaws.com	HTTPS
Asia Pacifico (Osaka-Locale)	ap-northeast-3	rds.ap-northeast-3.amazonaws.com rds.ap-northeast-3.api.aws	HTTPS HTTPS

Nome della regione	Regione	Endpoint	Protocollo
Asia Pacifico (Seoul)	ap-northeast-2	rds.ap-northeast-2.amazonaws.com	HTTPS
		rds.ap-northeast-2.api.aws	HTTPS
Asia Pacifico (Singapore)	ap-southeast-1	rds.ap-southeast-1.amazonaws.com	HTTPS
		rds.ap-southeast-1.api.aws	HTTPS
Asia Pacifico (Sydney)	ap-southeast-2	rds.ap-southeast-2.amazonaws.com	HTTPS
		rds.ap-southeast-2.api.aws	HTTPS
Asia Pacifico (Taipei)	ap-east-2	rds.ap-east-2.amazonaws.com	HTTPS
Asia Pacifico (Thailandia)	ap-southeast-7	rds.ap-southeast-7.amazonaws.com	HTTPS
Asia Pacifico (Tokyo)	ap-northeast-1	rds.ap-northeast-1.amazonaws.com	HTTPS
		rds.ap-northeast-1.api.aws	HTTPS
Canada (Centrale)	ca-central-1	rds.ca-central-1.amazonaws.com	HTTPS
		rds.ca-central-1.api.aws	HTTPS
		rds-fips.ca-central-1.api.aws	HTTPS
		rds-fips.ca-central-1.amazonaws.com	HTTPS

Nome della regione	Regione	Endpoint	Protocollo
Canada occidentale (Calgary)	ca-west-1	rds.ca-west-1.amazonaws.com rds-fips.ca-west-1.amazonaws.com	HTTPS HTTPS
Europa (Francoforte)	eu-central-1	rds.eu-central-1.amazonaws.com rds.eu-central-1.api.aws	HTTPS HTTPS
Europa (Irlanda)	eu-west-1	rds.eu-west-1.amazonaws.com rds.eu-west-1.api.aws	HTTPS HTTPS
Europa (Londra)	eu-west-2	rds.eu-west-2.amazonaws.com rds.eu-west-2.api.aws	HTTPS HTTPS
Europa (Milano)	eu-south-1	rds.eu-south-1.amazonaws.com rds.eu-south-1.api.aws	HTTPS HTTPS
Europa (Parigi)	eu-west-3	rds.eu-west-3.amazonaws.com rds.eu-west-3.api.aws	HTTPS HTTPS
Europa (Spagna)	eu-south-2	rds.eu-south-2.amazonaws.com rds.eu-south-2.api.aws	HTTPS HTTPS
Europa (Stoccolma)	eu-north-1	rds.eu-north-1.amazonaws.com rds.eu-north-1.api.aws	HTTPS HTTPS
Europa (Zurigo)	eu-central-2	rds.eu-central-2.amazonaws.com rds.eu-central-2.api.aws	HTTPS HTTPS

Nome della regione	Regione	Endpoint	Protocollo
Israele (Tel Aviv)	il-central-1	rds.il-central-1.amazonaws.com rds.il-central-1.api.aws	HTTPS HTTPS
Messico (Centrale)	mx-central-1	rds.mx-central-1.amazonaws.com	HTTPS
Medio Oriente (Bahrein)	me-south-1	rds.me-south-1.amazonaws.com rds.me-south-1.api.aws	HTTPS HTTPS
Medio Oriente (Emirati Arabi Uniti)	me-central-1	rds.me-central-1.amazonaws.com rds.me-central-1.api.aws	HTTPS HTTPS
Sud America (São Paulo)	sa-east-1	rds.sa-east-1.amazonaws.com rds.sa-east-1.api.aws	HTTPS HTTPS
AWSGovCloud (Stati Uniti orientali)	us-gov-east-1	rds.us-gov-east-1.amazonaws.com rds.us-gov-east-1.api.aws	HTTPS HTTPS
AWSGovCloud (Stati Uniti occidentali)	us-gov-west-1	rds.us-gov-west-1.amazonaws.com rds.us-gov-west-1.api.aws	HTTPS HTTPS

La tabella riportata di seguito mostra il formato da utilizzare quando si crea un ARN per un determinato tipo di risorsa di Amazon RDS.

Tipo di risorsa	Formato ARN
Istanza database	arn:aws:rds: ::db: *<region>* *<account>* *<name>* Esempio: `arn:aws:rds: us-east-2 :123456789012 :db:my-mysql-instance-1`
Cluster DB	arn:aws:rds: *<region>* :cluster: *<account>* *<name>* Esempio: `arn:aws:rds: us-east-2 :123456789012 :cluster: my-aurora-cluster-1`
Sottoscrizione a eventi	arn:aws:rds: *<region>* :it: *<account>* *<name>* Esempio: `arn:aws:rds: us-east-2 :123456789012 :es:my-subscription`
DB parameter group (Gruppo di parametri database)	arn:aws:rds: *<region>* ::pg: *<account>* *<name>* Esempio: `arn:aws:rds: us-east-2 :123456789012 :pg:my-param-enable-logs`
DB cluster parameter group (Gruppo di parametri del cluster database)	arn:aws:rds: *<region>* :cluster-pg: *<account>* *<name>* Esempio:

Tipo di risorsa	Formato ARN
	`arn:aws:rds: us-east-2 :123456789012 :cluster-pg: my-cluster-param-timezone`
Istanza database riservata	arn:aws:rds: *<region>* ::ri: *<account>* *<name>* Esempio: `arn:aws:rds: us-east-2 :123456789012 :ri:my-reserved-postgresql`
Gruppo di sicurezza DB	arn:aws:rds: *<region>* :secgrp: *<account>* *<name>* Esempio: `arn:aws:rds: us-east-2 :123456789012 :secgrp:my-public`
Snapshot di database automatizzato	arn:aws:rds: *<region>* *<account>* ::snapshot:rds: *<name>* Esempio: `arn:aws:rds: us-east-2 :123456789012 :snapshot:rds: my-mysql-db-2019-07-22-07-23`
Snapshot di cluster database automatizzato	arn:aws:rds: *<region>* *<account>* :cluster-snapshot:rds: *<name>* Esempio: `arn:aws:rds: us-east-2 :123456789012 :cluster-snapshot:rds: my-aurora-cluster-2019-07-22-16-16`

Tipo di risorsa	Formato ARN
Snapshot di database manuale	arn:aws:rds: *<region>* :snapshot: *<account>* *<name>* Esempio: `arn:aws:rds: us-east-2 :123456789012 :snapshot: my-mysql-db-snap`
Snapshot del cluster di database manuale	arn:aws:rds: *<region>* *<account>* :cluster-snapshot: *<name>* Esempio: `arn:aws:rds: us-east-2 :123456789012 :cluster-snapshot: my-aurora-cluster-snap`
Gruppo di sottoreti DB	arn:aws:rds: *<region>* :subgrp: *<account>* *<name>* Esempio: `arn:aws:rds: us-east-2 :123456789012 :subgrp:my-subnet-10`
Cluster globale	arn:aws:rds: ::global-cluster: *<account>* *<name>* Esempio: `arn:aws:rds:: 123456789012 :global-cluster: my-aurora-global-cluster-1`

Recupero di un ARN esistente per Amazon RDS

È possibile ottenere l'ARN di una risorsa RDS utilizzando l'APIConsole di gestione AWS, AWS Command Line Interface (AWS CLI) o RDS.

Console

Per ottenere un ARN daConsole di gestione AWS, vai alla risorsa per cui desideri un ARN e visualizza i dettagli di quella risorsa.

Ad esempio, puoi ottenere l'ARN per un cluster database dalla scheda Configurazione dei dettagli del cluster database.
ARN del cluster database.

AWS CLI

Per ottenere un ARN da AWS CLI per una particolare risorsa RDS, si utilizza il `describe` comando per quella risorsa. La tabella seguente mostra ogni AWS CLI comando e la proprietà ARN utilizzata con il comando per ottenere un ARN.

AWS CLIcomando	Proprietà ARN
describe-event-subscriptions	EventSubscriptionArn
describe-certificates	CertificateArn
describe-db-parameter-groups	DBParameterGroupArn
describe-db-cluster-parameter-gruppi	DBClusterParameterGroupArn
describe-db-instances	DBInstance- Arn
describe-db-security-groups	DBSecurityGroupArn
describe-db-snapshots	DBSnapshotArn
describe-events	SourceArn
describe-reserved-db-instances	Arn riservato DBInstance
describe-db-subnet-groups	DBSubnetGroupArn
describe-db-clusters	DBClusterArn
describe-db-cluster-snapshots	DBClusterSnapshotArn

Ad esempio, il AWS CLI comando seguente ottiene l'ARN per un'istanza DB.

Example

Per Linux, macOS o Unix:

```
aws rds describe-db-instances \
--db-instance-identifier DBInstanceIdentifier \
--region us-west-2 \
--query "*[].{DBInstanceIdentifier:DBInstanceIdentifier,DBInstanceArn:DBInstanceArn}"
```

Per Windows:

```
aws rds describe-db-instances ^
--db-instance-identifier DBInstanceIdentifier ^
--region us-west-2 ^
--query "*[].{DBInstanceIdentifier:DBInstanceIdentifier,DBInstanceArn:DBInstanceArn}"
```

L'output del comando è simile al seguente:

```
[
    {
        "DBInstanceArn": "arn:aws:rds:us-west-2:account_id:db:instance_id",
        "DBInstanceIdentifier": "instance_id"
    }
]
```

API RDS

Per ottenere un ARN per una determinata risorsa RDS, puoi chiamare le operazioni API RDS e le proprietà ARN seguenti.

Operazione API RDS	Proprietà ARN
DescribeEventSubscriptions	EventSubscriptionArn
DescribeCertificates	CertificateArn
Descrivi gruppi DBParameter	DBParameterGroupArn

Operazione API RDS	Proprietà ARN
Descriva DBCluster Parameter Groups	DBClusterParameterGroupArn
Descriva DBInstances	DBInstanceArn
Descrivi gruppi DBSecurity	DBSecurityGroupArn
Descriva DBSnapshots	DBSnapshotArn
DescribeEvents	SourceArn
DescribeReservedDBInstances	Arn riservato DBInstance
Descrivi gruppi DBSubnet	DBSubnetGroupArn
Descriva DBClusters	DBClusterArn
Descrivi le istantanee DBCluster	DBClusterSnapshotArn

Aggiornamenti di Amazon Aurora

Gli aggiornamenti di Amazon Aurora vengono resi disponibili periodicamente. Gli aggiornamenti vengono applicati ai cluster di database di Amazon Aurora durante le finestre di manutenzione del sistema. I tempi per l'applicazione degli aggiornamenti dipendono dalla Regione e dall'impostazione della finestra di manutenzione del cluster di database, ma anche dal tipo di aggiornamento. Gli aggiornamenti richiedono il riavvio del database, pertanto ci sarà un tempo di inattività di 20 - 30 secondi, al termine del quale si potrà ricominciare a usare i cluster di database o i cluster. Puoi visualizzare o modificare le impostazioni della finestra di manutenzione dalla Console di gestione AWS.

> **Note**
>
> Il tempo necessario per riavviare l'istanza DB dipende dal processo di ripristino dell'arresto anomalo, l'attività del database al momento del riavvio e il comportamento del motore DB specifico. Per ottimizzare i tempi di riavvio, è consigliabile ridurre l'attività del database il più possibile durante il processo di riavvio. Riducendo l'attività del database si riduce l'attività di rollback per le transazioni in transito.

Per informazioni sugli aggiornamenti del sistema operativo per Amazon Aurora, consulta Aggiornamenti del sistema operativo per i cluster Aurora DB.

Alcuni aggiornamenti sono specifici di un motore database supportato da Aurora. Per ulteriori informazioni sugli aggiornamenti del motore database, consulta la tabella seguente.

Motore del database	Aggiornamenti
Amazon Aurora MySQL	Per informazioni, consulta Aggiornamenti del motore del database per Amazon Aurora MySQL
Amazon Aurora PostgreSQL	Per informazioni, consulta Aggiornamenti del motore di database Amazon Aurora PostgreSQL per 02/06/2020

Identificare la versione Amazon Aurora

Amazon Aurora include alcune funzionalità generiche per Aurora e disponibili per tutti i cluster database Aurora. Aurora include altre funzionalità specifiche per un particolare motore del database supportato da Aurora. Queste caratteristiche sono disponibili solo per i cluster di database Aurora che utilizzano tale motore di database, ad esempio Aurora PostgreSQL.

Un'istanza database Aurora fornisce due numeri di versione: il numero di versione di Aurora e quello del motore di database di Aurora. Per i numeri di versione di Aurora viene utilizzato il formato seguente.

```
<major version>.<minor version>.<patch version>
```

Per ottenere il numero di versione di Aurora da un'istanza database Aurora che utilizza un motore di database specifico, potrai servirti di una delle query seguenti.

Motore di database	Query
Amazon Aurora MySQL	`SELECT AURORA_VERSION();` `SHOW @@aurora_version;`
Amazon Aurora PostgreSQL	`SELECT AURORA_VERSION();`

Supporto esteso di Amazon RDS con Amazon Aurora

Il supporto esteso di RDS ti consente di continuare a eseguire un database su una versione principale del motore oltre la data di fine del supporto standard di Aurora, a fronte di un costo aggiuntivo.

Puoi registrare un database al supporto esteso di RDS solo abilitando il supporto esteso di RDS al momento della creazione o del ripristino di un'istanza database. Non puoi aggiornare lo stato di registrazione al supporto esteso di RDS su istanze database esistenti, a meno che tu non le stia ripristinando.

Se hai abilitato il supporto esteso di RDS durante la creazione o il ripristino di un'istanza database, allora dopo la data di fine del supporto standard di Aurora, Amazon Aurora registrerà automaticamente l'istanza database al supporto esteso di RDS. La registrazione automatica al supporto esteso di RDS non modifica il motore di database e non influisce sul tempo di attività o sulle prestazioni dell'istanza database.

Il supporto esteso di RDS fornisce i seguenti aggiornamenti e supporto tecnico:

- Aggiornamenti di sicurezza per CVE critiche e a elevata vulnerabilità per l'istanza database o il cluster di database, incluso il motore di database
- Correzioni di bug e patch per problemi critici
- La possibilità di aprire casi di supporto e ricevere assistenza per la risoluzione dei problemi nell'ambito dell'accordo sul livello di servizio di Amazon RDS standard

Questa offerta a pagamento concede più tempo per l'aggiornamento a una versione principale del motore supportata. Ad esempio, la data di fine del supporto standard di Aurora per Aurora MySQL versione 2 è il 31 ottobre 2024. Tuttavia, non sei pronto per l'aggiornamento manuale ad Aurora MySQL versione 3 prima di tale data. In questo caso, Amazon Aurora registra automaticamente i cluster al supporto esteso di RDS il 31 ottobre 2024, permettendoti di continuare a eseguire Aurora MySQL versione 2. A partire dal 1° dicembre 2024, Amazon Aurora addebita automaticamente i costi per il supporto esteso di RDS.

Il supporto esteso per RDS è disponibile fino a 3 anni oltre la data di fine del ciclo vita della versione Community per una versione principale del motore (3 anni e 4 mesi per Aurora MySQL versione 2). Trascorso questo periodo, se non hai aggiornato la versione principale del motore a una versione

supportata, Amazon Aurora aggiornerà automaticamente la versione principale del motore. Ti consigliamo di eseguire l'aggiornamento a una versione del motore principale supportata il prima possibile.

Per ulteriori informazioni sulle date di fine del supporto standard di Aurora e sulle date di fine del supporto esteso di RDS, consulta Calendario dei rilasci per le versioni principali di Aurora MySQL e Calendario dei rilasci per le versioni principali di Aurora PostgreSQL.

Argomenti

- Panoramica del supporto esteso di Amazon RDS
- Costi del supporto esteso di Amazon RDS
- Versioni con supporto esteso di Amazon RDS
- Amazon Aurora e le responsabilità dei clienti con il supporto esteso di Amazon RDS
- Creazione di un cluster di database Aurora o un cluster globale con il supporto esteso per Amazon RDS
- Visualizzazione della registrazione dei cluster di database Aurora o dei cluster globali al supporto esteso per Amazon RDS
- Visualizzazione delle date di supporto per le versioni del motore con supporto esteso di Amazon RDS
- Ripristino di un cluster di database Aurora o un cluster globale con il supporto esteso di Amazon RDS

Panoramica del supporto esteso di Amazon RDS

Dopo la data di fine del supporto standard di Aurora, se il supporto esteso di RDS non è stato disabilitato durante la creazione o il ripristino delle istanze database, Amazon Aurora le registra automaticamente nel supporto esteso di RDS. Aurora aggiorna automaticamente l'istanza database all'ultima versione secondaria rilasciata prima della data di fine del supporto standard di Aurora, se non si utilizza già tale versione. Amazon Aurora non esegue l'aggiornamento della versione secondaria fino a dopo la data di fine del supporto standard di Aurora per la versione principale del motore.

Puoi creare nuovi database con le versioni principali del motore che hanno raggiunto la data di fine del supporto standard di Aurora. Aurora registra automaticamente i nuovi database nel supporto esteso di RDS e addebita il costo di questa offerta.

Se esegui l'aggiornamento a un motore ancora coperto del supporto standard di Aurora prima della data di fine del supporto standard di Aurora, Amazon Aurora non registra il motore nel supporto esteso di RDS.

Se tenti di ripristinare uno snapshot di un database compatibile con una versione del motore precedente alla data di fine del supporto standard di Aurora, ma non registrata nel supporto esteso di RDS, Amazon Aurora tenta di aggiornare lo snapshot in modo che sia compatibile con l'ultima versione del motore ancora coperta dal supporto standard di Aurora. Se il ripristino non riesce, Amazon Aurora registra automaticamente il motore nel supporto esteso di RDS con una versione compatibile con lo snapshot.

Puoi terminare la registrazione al supporto esteso di RDS in qualsiasi momento. Per terminare la registrazione, aggiorna ogni motore registrato a una versione più recente del motore ancora coperta dal supporto standard di Aurora. La fine della registrazione al supporto esteso di RDS entrerà in vigore il giorno in cui completi l'aggiornamento a una versione più recente del motore ancora coperta dal supporto standard di Aurora.

Per ulteriori informazioni sulle date di fine del supporto standard di Aurora e sulle date di fine del supporto esteso di RDS, consulta Calendario dei rilasci per le versioni principali di Aurora MySQL e Calendario dei rilasci per le versioni principali di Aurora PostgreSQL.

Costi del supporto esteso di Amazon RDS

Per tutti i motori registrati nel supporto esteso di RDS, verranno addebitati i costi a partire dal giorno successivo alla fine della data di supporto standard di Aurora. Per la data di fine del supporto standard di Aurora, consulta Versioni principali di Amazon Aurora.

L'addebito dei costi aggiuntivi per il supporto esteso per RDS si interrompe automaticamente quando si esegue una di queste operazioni:

- Aggiornamento a una versione del motore coperta dal supporto standard.
- Eliminazione del database su cui è in esecuzione una versione principale oltre la data di fine del supporto standard di Aurora.

Gli addebiti vengono ripresi se la versione del motore di destinazione sarà coperta dal supporto esteso di RDS in futuro.

Ad esempio, Aurora PostgreSQL 11 sarà coperto dal supporto esteso a partire dal 1° marzo 2024, ma i costi non vengono addebitati fino al 1° aprile 2024. Se aggiorni il database Aurora PostgreSQL

11 a Aurora PostgreSQL 12 il 30 aprile 2024, vengono addebitati i costi per 30 giorni di supporto esteso per Aurora PostgreSQL 11. Se continui a eseguire Aurora PostgreSQL 12 su questa istanza database oltre il 28 febbraio 2025, data di fine del supporto standard di RDS, il database sarà nuovamente soggetto ai costi del supporto esteso di RDS a partire dal 1° marzo 2025.

Per ulteriori informazioni, consulta Prezzi di Amazon Aurora.

Evitare i costi del supporto esteso di Amazon RDS

Puoi evitare l'addebito dei costi del supporto esteso di RDS impedendo ad Aurora di creare o ripristinare un cluster di database Aurora o un cluster globale oltre la data di fine del supporto standard di Aurora. Per eseguire questa operazione, utilizza la AWS CLI o l'API RDS.

Nella AWS CLI, specifica `open-source-rds-extended-support-disabled` per l'opzione `--engine-lifecycle-support`. Nell'API RDS, specifica `open-source-rds-extended-support-disabled` per il parametro `LifeCycleSupport`. Per ulteriori informazioni, consulta Creazione di un cluster di database Aurora o un cluster globale o Ripristino di un cluster di database Aurora o un cluster globale.

Versioni con supporto esteso di Amazon RDS

Il supporto esteso di RDS è disponibile per Aurora MySQL e Aurora PostgreSQL. Per ulteriori informazioni, consulta Versioni principali di Amazon Aurora.

Il supporto esteso di RDS è disponibile solo su determinate versioni secondarie. Le versioni secondarie sono indicate come compatibili con il supporto esteso per RDS solo dopo che le versioni principali hanno raggiunto la data di fine del ciclo di vita della versione Community. Per ulteriori informazioni, consulta Calendario dei rilasci per le versioni di Aurora MySQL nelle Note di rilascio di Aurora MySQL e Calendario dei rilasci per le versioni di Aurora PostgreSQL nelle Note di rilascio di Aurora PostgreSQL.

Il supporto esteso di RDS è disponibile solo su Aurora Serverless v2. Non è disponibile su Aurora Serverless v1.

Puoi anche visualizzare le informazioni sulle date di supporto per le versioni del motore utilizzando la AWS CLI o l'API RDS. Per ulteriori informazioni, consulta Visualizzazione delle date di supporto per le versioni del motore con supporto esteso di Amazon RDS.

Amazon Aurora e le responsabilità dei clienti con il supporto esteso di Amazon RDS

I contenuti seguenti descrivono le responsabilità di Amazon Aurora e dei clienti con il supporto esteso di RDS.

Argomenti

- Responsabilità di Amazon Aurora
- Le tue responsabilità

Responsabilità di Amazon Aurora

Dopo la data di fine del supporto standard di Aurora, Amazon Aurora fornirà patch, correzioni di bug e aggiornamenti per i motori registrati al supporto esteso di RDS. Questo avverrà per un massimo di 3 anni o fino a quando non smetterai di utilizzare i motori, a seconda di quale condizione si verifichi per prima.

Le patch riguarderanno CVE critiche e a elevata vulnerabilità come definito nelle valutazioni di gravità CVSS del National Vulnerability Database (NVD). Per ulteriori informazioni, consulta Vulnerability Metrics (Metriche relative alla vulnerabilità).

Le tue responsabilità

Sei responsabile dell'applicazione delle patch, delle correzioni di bug e degli aggiornamenti forniti per i cluster di database Aurora o i cluster globali registrati al supporto esteso di RDS. Amazon Aurora si riserva il diritto di modificare, sostituire o ritirare tali patch, correzioni di bug e aggiornamenti in qualsiasi momento. Se è necessaria una patch per risolvere problemi di sicurezza o di stabilità critica, Amazon Aurora si riserva il diritto di aggiornare i cluster di database Aurora o i cluster globali con la patch o di richiedere l'installazione della patch da parte tua.

Sei inoltre responsabile dell'aggiornamento del motore a una versione più recente prima della data di fine del supporto esteso di RDS. La data di fine del supporto esteso per RDS è stabilita in genere 3 anni dopo la fine del ciclo di vita della versione Community. . Per la data di fine del supporto esteso per RDS per la versione principale del motore di database, consulta Versioni principali di Amazon Aurora.

Se non aggiorni il motore, dopo la data di fine del supporto esteso di RDS, Amazon Aurora tenterà di aggiornare il motore a una versione più recente ancora coperta dal supporto standard di Aurora. Se

l'aggiornamento fallisce, Amazon Aurora si riserva il diritto di eliminare il cluster di database Aurora o il cluster globale che esegue il motore oltre la data di fine del supporto standard di Aurora. Tuttavia, prima di farlo, Amazon Aurora conserverà i dati di quel motore.

Creazione di un cluster di database Aurora o un cluster globale con il supporto esteso per Amazon RDS

Quando crei un cluster di database Aurora o un cluster globale, seleziona Abilita il supporto esteso RDS nella console oppure utilizza l'opzione Supporto esteso nella AWS CLI o il parametro nell'API RDS. Quando registri un cluster di database Aurora o un cluster globale nel supporto esteso di Amazon RDS, la registrazione nel supporto esteso di RDS è permanente per la durata del cluster di database Aurora o del cluster globale.

Se utilizzi la console, devi selezionare Abilita il supporto esteso RDS. L'impostazione non è selezionata per impostazione predefinita.

Se utilizzi la AWS CLI o l'API RDS e non specifichi l'impostazione Supporto esteso RDS, per impostazione predefinita Amazon RDS abilita il supporto esteso di RDS. Quando automatizzi le operazioni tramite CloudFormation o altri servizi, questo comportamento predefinito mantiene la disponibilità del database oltre la data di fine del supporto standard di Aurora.

Puoi impedire la registrazione al supporto esteso di RDS utilizzando AWS CLI o l'API RDS per creare un cluster di database Aurora o un cluster globale.

Argomenti

- Comportamento del supporto esteso di RDS
- Considerazioni per il supporto esteso di RDS
- Creazione di un cluster di database Aurora o un cluster globale con il supporto esteso di RDS

Comportamento del supporto esteso di RDS

La tabella seguente riassume cosa accade quando una versione principale del motore raggiunge la fine del supporto standard di Aurora.

Stato del supporto esteso di RDS*	Comportamento
Abilitato	Amazon RDS addebita i costi per il supporto esteso di RDS.
Disabilitato	Amazon RDS aggiorna un cluster di database Aurora o un cluster globale a una versione del motore supportata. Questo aggiornamento viene eseguito alla data di fine del supporto standard per Aurora o poco dopo.

* Nella console RDS, lo stato del supporto esteso di RDS viene visualizzato come Sì o No, mentre nell'AWS CLI o nell'API RDS, lo stato viene visualizzato come `open-source-rds-extended-support` o `open-source-rds-extended-support-disabled`.

Considerazioni per il supporto esteso di RDS

Prima di creare un cluster di database Aurora o un cluster globale, considera gli aspetti seguenti:

- Dopo la data di fine del supporto standard di Aurora, puoi impedire la creazione di un nuovo cluster di database Aurora o un cluster globale ed evitare addebiti per il supporto esteso di RDS. Per eseguire questa operazione, utilizza la AWS CLI o l'API RDS. Nella AWS CLI, specifica `open-source-rds-extended-support-disabled` per l'opzione `--engine-lifecycle-support`. Nell'API RDS, specifica `open-source-rds-extended-support-disabled` per il parametro `LifeCycleSupport`. Se si specifica `open-source-rds-extended-support-disabled` e la data di fine del supporto standard per Aurora è trascorsa, la creazione di un cluster di database Aurora o un cluster globale avrà sempre esito negativo.
- Il supporto esteso di RDS viene impostato a livello di cluster. I membri di un cluster avranno sempre la stessa impostazione per il supporto esteso di RDS nella console RDS, `--engine-lifecycle-support` nella AWS CLI e `EngineLifecycleSupport` nell'API RDS.

Per ulteriori informazioni, consulta Versioni di Amazon Aurora.

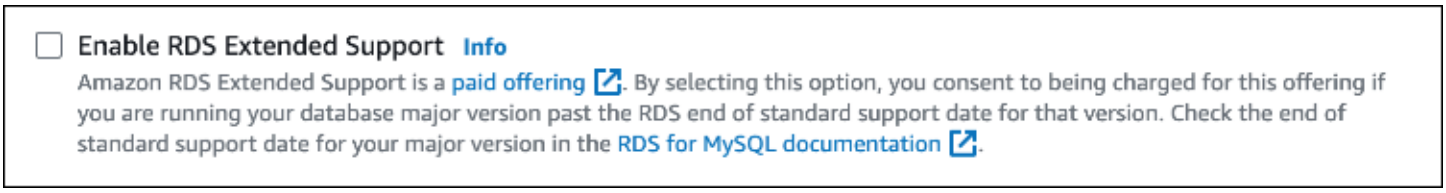
Creazione di un cluster di database Aurora o un cluster globale con il supporto esteso di RDS

Puoi creare un cluster di database Aurora o di un cluster globale con una versione del supporto esteso per RDS utilizzando la Console di gestione AWS, la AWS CLI o l'API RDS.

Console

Quando crei un cluster di database Aurora o un cluster globale, seleziona Abilita il supporto esteso RDS nella sezione Opzioni motore. Questa impostazione non è selezionata per impostazione predefinita.

L'immagine seguente mostra l'impostazione Abilita il supporto esteso RDS:



AWS CLI

Quando esegui il comando create-db-cluster o create-global-clusterAWS CLI, seleziona il supporto esteso di RDS specificando `open-source-rds-extended-support` per l'opzione `--engine-lifecycle-support`. Questa opzione è impostata su `open-source-rds-extended-support` per impostazione predefinita.

Per impedire la creazione di un cluster di database Aurora o un cluster globale dopo la data di fine del supporto standard di Aurora, specifica `open-source-rds-extended-support-disabled` per l'opzione `--engine-lifecycle-support`. In tal modo, eviti i costi associati al supporto esteso di RDS.

API RDS

Quando si utilizza l'operazione dell'API di Amazon RDS CreateDBCluster o CreateGlobalCluster , seleziona il supporto esteso per RDS impostando il parametro `EngineLifecycleSupport` su `open-source-rds-extended-support`. Questo parametro è impostato su `open-source-rds-extended-support` per impostazione predefinita.

Per impedire la creazione di un cluster di database Aurora o un cluster globale dopo la data di fine del supporto standard di Aurora, specifica `open-source-rds-extended-support-disabled`

per il parametro `EngineLifecycleSupport`. In tal modo, eviti i costi associati al supporto esteso di RDS.

Per ulteriori informazioni, consulta i seguenti argomenti:

- Per creare un cluster di database Aurora, segui le istruzioni per il motore di database in Creazione di un cluster database Amazon Aurora.
- Per creare un cluster globale, segui le istruzioni per il motore di database in Creazione di un database globale Amazon Aurora.

Visualizzazione della registrazione dei cluster di database Aurora o dei cluster globali al supporto esteso per Amazon RDS

Puoi visualizzare la registrazione dei cluster di database Aurora o cluster globali al supporto esteso per RDS utilizzando la Console di gestione AWS, la AWS CLI o l'API RDS.

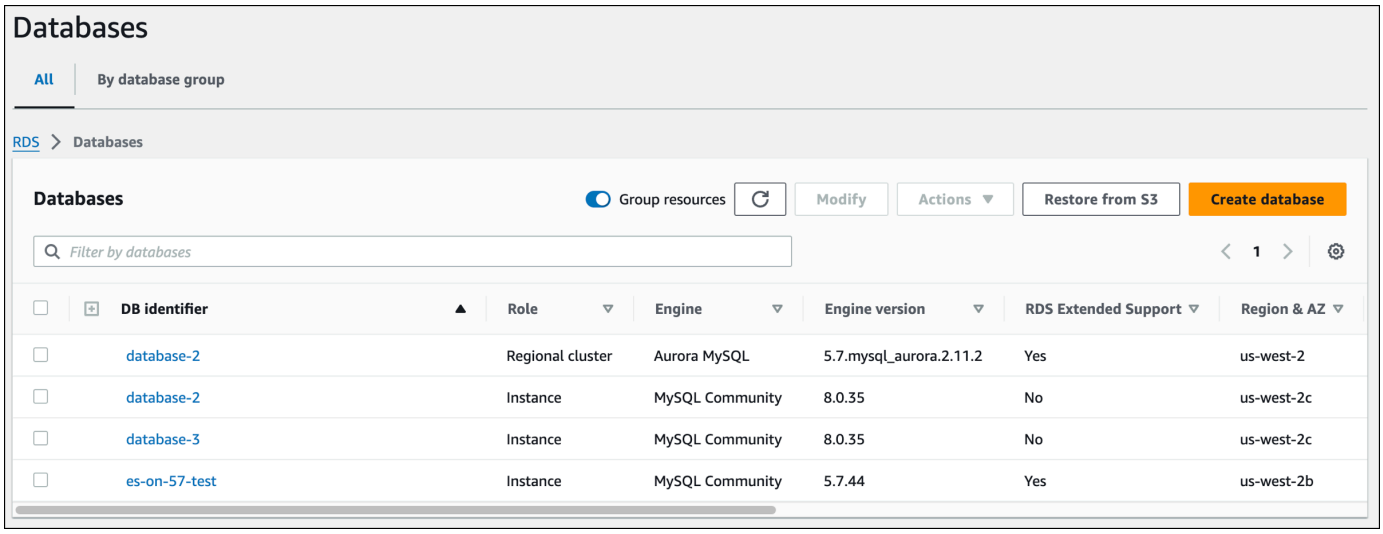
> **Note**
>
> La colonna Supporto esteso per RDS nella console, l'opzione `--engine-lifecycle-support` nella AWS CLI e il parametro `EngineLifecycleSupport` nell'API RDS indicano solo la registrazione al supporto esteso per RDS. I costi per il supporto esteso per RDS iniziano solo quando la versione del motore di database ha raggiunto la fine del supporto standard di Aurora. Per ulteriori informazioni, consulta Versioni principali di Amazon Aurora. Ad esempio, disponi di un database Aurora PostgreSQL 11 registrato nel supporto esteso per RDS. Il 1° aprile 2024, Amazon RDS ha iniziato ad addebitare i costi del supporto esteso per RDS per questo database. Il 31 luglio 2024, hai aggiornato questo database ad Aurora PostgreSQL 12. Lo stato del supporto esteso per RDS per questo database rimane abilitato. Tuttavia, i costi del supporto esteso per RDS per questo database sono stati sospesi perché Aurora PostgreSQL 12 non aveva ancora raggiunto la fine del supporto standard di Aurora. Amazon RDS non addebiterà alcun costo per il supporto esteso per RDS per questo database fino al 1° marzo 2025, data di fine del supporto standard di Aurora per Aurora PostgreSQL 12.

Console

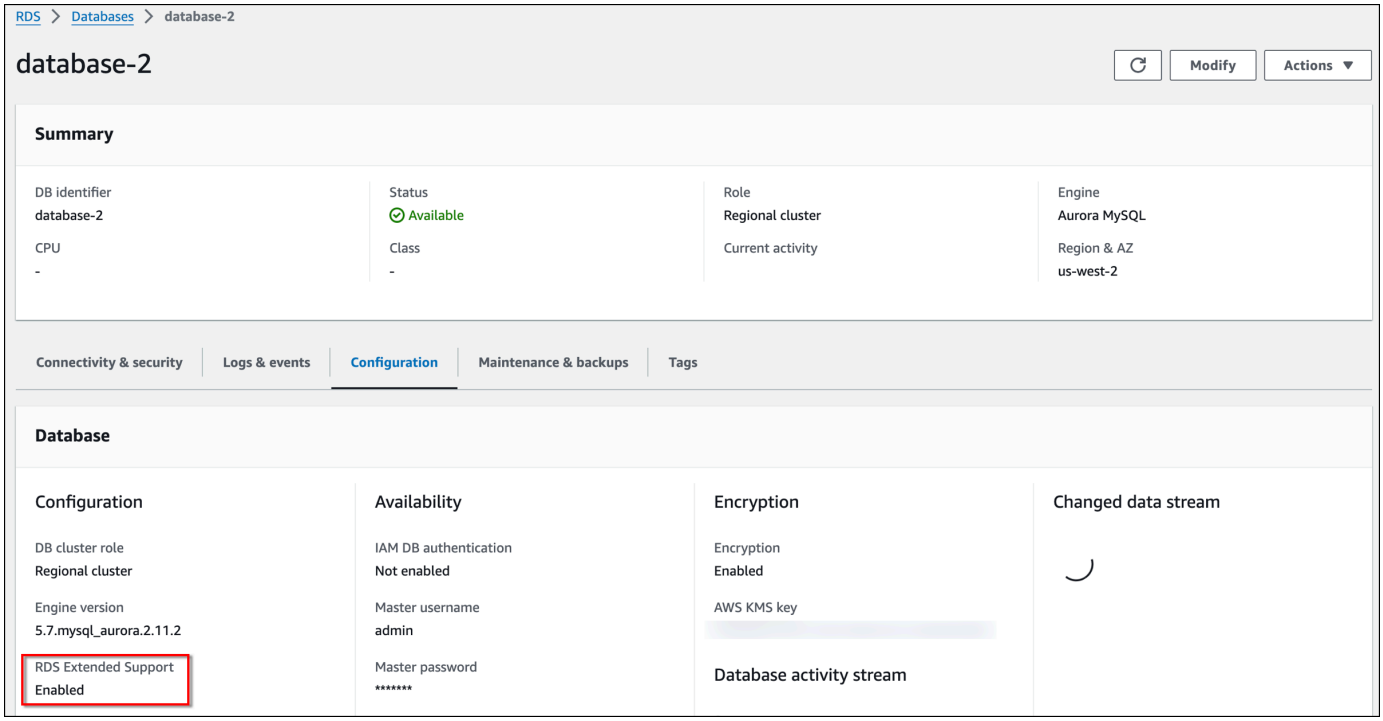
Per visualizzare la registrazione dei cluster di database Aurora o cluster globali al supporto esteso per RDS

1. Accedi alla Console di gestione AWS e apri la console Amazon RDS all'indirizzo https://console.aws.amazon.com/rds/.
2. Nel pannello di navigazione, scegliere Databases (Database). Il valore in Supporto esteso di Amazon RDS indica se è stata effettuata la registrazione di un cluster di database Aurora o cluster globale nel supporto esteso per RDS. Se non viene visualizzato alcun valore, il supporto esteso per RDS non è disponibile per il database.

> **Tip**
>
> Se la colonna Supporto esteso per RDS non viene visualizzata, scegli l'icona Preferenze, quindi attiva Supporto esteso di Amazon RDS.



3. Puoi inoltre visualizzare la registrazione nella scheda Configurazione per ogni database. Scegli un database in Identificatore DB. Nella scheda Configurazione, controlla in Supporto esteso per vedere se il database è registrato o meno.



AWS CLI

Per visualizzare la registrazione dei database nel supporto esteso per RDS utilizzando la AWS CLI, esegui il comando describe-db-clusters o describe-global-clusters.

Se il supporto esteso per RDS è disponibile per un database, la risposta include il parametro `EngineLifecycleSupport`. Il valore `open-source-rds-extended-support` indica che è stata effettuata la registrazione di un cluster di database Aurora o cluster globale nel supporto esteso per RDS. Il valore `open-source-rds-extended-support-disabled` indica che la registrazione del cluster di database Aurora o cluster globale nel supporto esteso per RDS è stata disabilitata.

Esempio

Il comando seguente restituisce informazioni su tutti i cluster di database Aurora:

```
aws rds describe-db-clusters
```

La risposta seguente mostra che un motore Aurora PostgreSQL in esecuzione sul cluster di database Aurora `database-1` è registrato nel supporto esteso per RDS:

```
{
    "DBClusterIdentifier": "database-1",
```

```
    ...
    "Engine": "aurora-postgresql",
    ...
    "EngineLifecycleSupport": "open-source-rds-extended-support"
}
```

API RDS

Per visualizzare la registrazione dei database nel supporto esteso per RDS utilizzando l'API Amazon RDS, utilizza l'operazione DescribeDBClusters o DescribeGlobalClusters.

Se il supporto esteso per RDS è disponibile per un database, la risposta include il parametro `EngineLifecycleSupport`. Il valore `open-source-rds-extended-support` indica che è stata effettuata la registrazione di un cluster di database Aurora o cluster globale nel supporto esteso per RDS. Il valore `open-source-rds-extended-support-disabled` indica che la registrazione del cluster di database Aurora o cluster globale nel supporto esteso per RDS è stata disabilitata.

Visualizzazione delle date di supporto per le versioni del motore con supporto esteso di Amazon RDS

Puoi visualizzare informazioni sulle date di supporto per le versioni del motore dei cluster di database Aurora o dei cluster globali con supporto esteso di Amazon RDS utilizzando la AWS CLI o l'API RDS. Queste informazioni possono aiutarti a pianificare gli aggiornamenti.

I comandi AWS CLI e le operazioni dell'API RDS restituiscono le date di inizio e fine per il supporto standard di Aurora e per il supporto esteso di RDS. Queste date sono disponibili anche nelle tabelle delle versioni principali del motore. Per ulteriori informazioni, consulta Versioni principali di Amazon Aurora.

AWS CLI

Per visualizzare le date di inizio e fine del supporto standard di Aurora e del supporto esteso di RDS per le tue versioni principali del motore utilizzando la AWS CLI, esegui il comando describe-db-major-engine-versions.

Il comando restituisce i seguenti parametri rilevanti:

- `SupportedEngineLifecycles`: questo parametro è un array di oggetti che include `LifecycleSupportName`, `LifecycleSupportStartDate` e `LifecycleSupportEndDate`.

- `LifecycleSupportName`: questo parametro indica il tipo di supporto utilizzato dalla versione del motore: supporto standard Aurora (`open-source-rds-standard-support`) o supporto esteso di RDS (`open-source-rds-extended-support`).
- `LifecycleSupportStartDate`: questo parametro indica la data di inizio del supporto standard di Aurora o del supporto esteso di RDS per la versione principale del motore, a seconda del valore di `LifecycleSupportName`.
- `LifecycleSupportEndDate`: questo parametro indica la data di fine del supporto standard di Aurora o del supporto esteso di RDS per la versione principale del motore, a seconda del valore di `LifecycleSupportName`.

Esempio

L'esempio di risposta mostra le date di inizio e fine dei cicli di vita del motore supportati `open-source-rds-standard-support` e `open-source-rds-extended-support` per Aurora MySQL versione 2 (MySQL 5.7). Il supporto esteso di RDS è disponibile per Aurora MySQL versione 2 (MySQL 5.7).

```
{
    "DBMajorEngineVersions": [
        {
            "Engine": "aurora-mysql",
            "MajorEngineVersion": "5.7",
            "SupportedEngineLifecycles": [
                {
                    "LifecycleSupportName": "open-source-rds-standard-support",
                    "LifecycleSupportStartDate": "2018-02-06T00:00:00+00:00",
                    "LifecycleSupportEndDate": "2024-10-31T23:59:59.999000+00:00"
                },
                {
                    "LifecycleSupportName": "open-source-rds-extended-support",
                    "LifecycleSupportStartDate": "2024-11-01T00:00:00+00:00",
                    "LifecycleSupportEndDate": "2027-02-28T23:59:59.999000+00:00"
                }
            ]
        },
        ...
    ]
}
```

API RDS

Per visualizzare le date di inizio e fine del supporto standard di Aurora e del supporto esteso di RDS per le versioni principali del motore utilizzando l'API RDS, utilizza l'operazione DescribeDBMajorEngineVersions.

Se il supporto esteso di RDS è disponibile per una versione del motore, la risposta include il parametro `SupportedEngineLifeCycles` come array con due oggetti. Un oggetto include le date di inizio e fine del supporto standard di Aurora. Il secondo oggetto include le date di inizio e fine del supporto esteso di RDS.

Se il supporto esteso di RDS non è disponibile per una versione del motore, la risposta include solo il parametro `SupportedEngineLifeCycles` come array con un singolo oggetto. Questo oggetto include le date di inizio e fine del supporto standard di Aurora.

Ripristino di un cluster di database Aurora o un cluster globale con il supporto esteso di Amazon RDS

Quando ripristini un cluster di database Aurora o un cluster globale, seleziona Abilita il supporto esteso RDS nella console oppure utilizza l'opzione Supporto esteso nell'AWS CLI o il parametro nell'API RDS. Quando registri un cluster di database Aurora o un cluster globale nel supporto esteso di RDS, la registrazione nel supporto esteso di RDS è permanente per la durata del cluster di database Aurora o del cluster globale.

Il valore predefinito per l'impostazione Supporto esteso RDS dipende dal fatto che si utilizzi la console, l'AWS CLI o l'API RDS per ripristinare il database. Se utilizzi la console, non selezionare Abilita il supporto esteso RDS e la versione principale del motore in fase di ripristino ha superato la data di fine del supporto standard di Aurora, Amazon Aurora esegue automaticamente l'aggiornamento dell'istanza database a una versione del motore più recente. Se utilizzi l'AWS CLI o l'API RDS e non specifichi l'impostazione Supporto esteso RDS, per impostazione predefinita Amazon RDS abilita il supporto esteso RDS. Quando automatizzi le operazioni tramite CloudFormation o altri servizi, questo comportamento predefinito mantiene la disponibilità del database oltre la data di fine del supporto standard di Aurora. Puoi disattivare il supporto esteso di RDS utilizzando l'AWS CLI o l'API RDS.

Argomenti

- Comportamento del supporto esteso di RDS
- Considerazioni per il supporto esteso di RDS

- Ripristino di un cluster di database Aurora o un cluster globale con il supporto esteso di RDS

Comportamento del supporto esteso di RDS

La tabella seguente riepiloga cosa accade quando una versione principale del motore di un cluster di database Aurora o un cluster globale in fase di ripristino ha raggiungo la fine del supporto standard Aurora.

Stato del supporto esteso di RDS*	Comportamento
Abilitato	Amazon RDS addebita i costi per il supporto esteso di RDS.
Disabilitato	Al termine del ripristino, Amazon RDS aggiorna automaticamente il cluster di database Aurora o il cluster globale a una versione del motore più recente (in una finestra di manutenzione futura).

* Nella console RDS, lo stato del supporto esteso di RDS viene visualizzato come Sì o No, mentre nell'AWS CLI o nell'API RDS, lo stato viene visualizzato come `open-source-rds-extended-support` o `open-source-rds-extended-support-disabled`.

Considerazioni per il supporto esteso di RDS

Prima di ripristinare un cluster di database Aurora o un cluster globale, considera gli aspetti seguenti:

- Dopo che la data di fine del supporto standard di Aurora è trascorsa, se desideri ripristinare un cluster di database Aurora o un cluster globale da Amazon S3, puoi farlo solo tramite l'AWS CLI o l'API RDS. Utilizza l'opzione `--engine-lifecycle-support` nel comando restore-db-cluster-from-s3 AWS CLI oppure il parametro `EngineLifecycleSupport` nell'operazione RestoreDBClusterFromS3 dell'API RDS.
- Se desideri impedire all'istanza di Aurora di ripristinare i database nelle versione del supporto esteso di RDS, specifica `open-source-rds-extended-support-disabled` nell'AWS CLI o nell'API RDS. In tal modo, eviti i costi associati al supporto esteso di RDS.

  Se specifichi questa impostazione, Amazon Aurora aggiornerà automaticamente il database ripristinato a una versione principale più recente e supportata. Se l'aggiornamento non supera i controlli preliminari, Amazon Aurora eseguirà un rollback in modo sicuro alla versione del motore

del supporto esteso di RDS. Questo database rimarrà in modalità supporto esteso di RDS e Amazon Aurora ti addebiterà i costi relativi al supporto esteso di RDS fino all'aggiornamento manuale del database.

- Il supporto esteso di RDS viene impostato a livello di cluster. I membri di un cluster avranno sempre la stessa impostazione per il supporto esteso di RDS nella console RDS, `--engine-lifecycle-support` nella AWS CLI e `EngineLifecycleSupport` nell'API RDS.

Per ulteriori informazioni, consulta Versioni di Amazon Aurora.

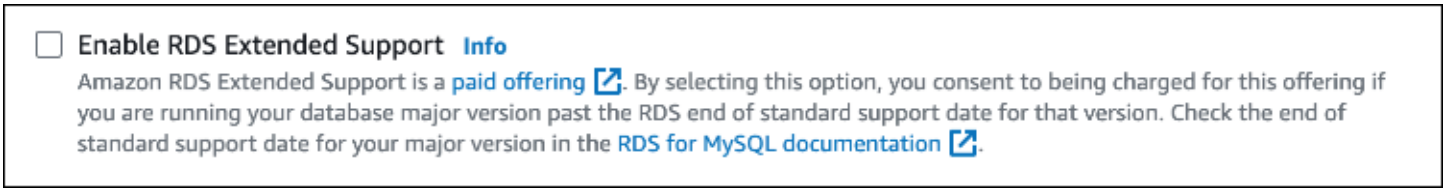
Ripristino di un cluster di database Aurora o un cluster globale con il supporto esteso di RDS

Puoi ripristinare un cluster di database Aurora o un cluster globale con una versione del supporto esteso di RDS utilizzando la Console di gestione AWS, la AWS CLI o l'API RDS.

Console

Quando ripristini un cluster di database Aurora o un cluster globale, seleziona Abilita il supporto esteso RDS nella sezione Opzioni motore. Se non selezioni questa opzione e la versione principale del motore che stai ripristinando ha superato la fine del periodo di supporto standard di Aurora, Amazon Aurora aggiornerà automaticamente il cluster di database Aurora o il cluster globale a una versione ancora coperta dal supporto standard di Aurora.

L'immagine seguente mostra l'impostazione Abilita il supporto esteso RDS:



AWS CLI

Quando esegui il comando restore-db-cluster-from-snapshot AWS CLI, seleziona il supporto esteso di RDS specificando `open-source-rds-extended-support` per l'opzione `--engine-lifecycle-support`.

Se desideri evitare i costi associati al supporto esteso di RDS, imposta `--engine-lifecycle-support`opzione su `open-source-rds-extended-support-disabled`. Questa opzione è impostata su `open-source-rds-extended-support` per impostazione predefinita.

Puoi anche specificare questo valore utilizzando i seguenti comandi AWS CLI:

- restore-db-cluster-from-s3
- restore-db-cluster-to-point-in-time

API RDS

Quando utilizzi l'operazione API di Amazon RDS come RestoreDBClusterFromSnapshot , seleziona il supporto esteso di RDS impostando il parametro `EngineLifecycleSupport` su `open-source-rds-extended-support`.

Per evitare i costi associati al supporto esteso di RDS, imposta il `EngineLifecycleSupport` parametro su `open-source-rds-extended-support-disabled`. Questo parametro è impostato su `open-source-rds-extended-support` per impostazione predefinita.

Puoi anche specificare questo valore utilizzando le seguenti operazioni API RDS:

- RestoreDBClusterFromS3
- RestoreDBClusterToPointInTime

Per ulteriori informazioni sul ripristino di un cluster di database Aurora, consulta le istruzioni relative al motore di database in Backup e ripristino di un cluster DB Amazon Aurora.

Utilizzo di Amazon Blue/Green Aurora Deployments per gli aggiornamenti del database

Una blue/green distribuzione copia un ambiente di database di produzione in un ambiente di staging separato e sincronizzato. Utilizzando Blue/Green Aurora Deployments, puoi apportare modifiche al database nell'ambiente di staging senza influire sull'ambiente di produzione. Ad esempio, è possibile aggiornare la versione principale o secondaria del motore di database o modificare i parametri del database nell'ambiente di staging. Quando è tutto pronto, è possibile promuovere l'ambiente di staging nel nuovo ambiente di database di produzione, con un tempo di inattività in genere inferiore al minuto.

Amazon Aurora crea l'ambiente di staging clonando il volume di archiviazione di Aurora sottostante nell'ambiente di produzione. Il volume del cluster nell'ambiente di staging memorizza solo le modifiche incrementali apportate all'ambiente.

> **Note**
>
> Attualmente, Blue/Green le implementazioni sono supportate solo per Aurora MySQL, Aurora PostgreSQL e Aurora Global Database. Per la disponibilità del motore Amazon RDS, consulta Using Amazon RDS Blue/Green Deployments for database updates nella Guida per l'utente di Amazon RDS.

Argomenti

- Panoramica delle implementazioni di Amazon Aurora Blue/Green
- Creazione di una blue/green distribuzione in
- Visualizzazione di un'implementazione blu/verde in Amazon Aurora
- Cambiare una blue/green distribuzione in
- Eliminazione di un'implementazione blu/verde in Amazon Aurora

Panoramica delle implementazioni di Amazon Aurora Blue/Green

Utilizzando Blue/Green Aurora Deployments, puoi apportare e testare modifiche al database prima di implementarle in un ambiente di produzione. Un'implementazione blu/verde crea un ambiente di gestione temporanea che copia l'ambiente di produzione. In un'implementazione blu/verde, l'ambiente blu è l'ambiente di produzione corrente. L'ambiente verde è l'ambiente di staging e rimane sincronizzato con l'attuale ambiente di produzione.

È possibile apportare modifiche al cluster di database Aurora nell'ambiente verde senza influire sui carichi di lavoro di produzione. Ad esempio, è possibile aggiornare la versione principale o secondaria del motore di database o modificare i parametri di database nell'ambiente di gestione temporanea. È possibile testare le modifiche nell'ambiente verde. Quando è tutto pronto, è possibile eseguire lo switchover degli ambienti per passare l'ambiente verde nel nuovo ambiente di produzione. Lo switchover richiede in genere meno di un minuto senza perdita di dati e senza la necessità di modificare l'applicazione.

Poiché l'ambiente verde è una copia della topologia dell'ambiente di produzione, il cluster di database e tutte le relative istanze database vengono copiati nell'implementazione. L'ambiente verde include anche le funzionalità utilizzate dal cluster di database, come snapshot del cluster di database, approfondimenti sulle prestazioni, monitoraggio avanzato e Aurora Serverless v2.

> **Note**
>
> Le implementazioni Blue/Green sono supportate per Aurora MySQL, Aurora PostgreSQL e Aurora Global Database. Per la disponibilità di Amazon RDS, consulta Panoramica delle Blue/Green distribuzioni di Amazon RDS nella Amazon RDS User Guide.

Argomenti

- Disponibilità di regioni e versioni
- Vantaggi dell'utilizzo di Amazon RDS Blue/Green Deployments
- Flusso di lavoro di una blue/green distribuzione
- Autorizzazione dell'accesso alle operazioni di Amazon blue/green Aurora
- Limitazioni e considerazioni per le distribuzioni di Amazon Aurora blue/green
- Best practice per le implementazioni di Amazon Aurora blue/green

Disponibilità di regioni e versioni

Il supporto varia a seconda delle versioni specifiche di ciascun motore di database e a seconda delle Regioni AWS. Per ulteriori informazioni, consulta the section called “Distribuzioni blu/verdi”.

Vantaggi dell'utilizzo di Amazon RDS Blue/Green Deployments

Utilizzando Amazon RDS Blue/Green Deployments, puoi rimanere aggiornato sulle patch di sicurezza, migliorare le prestazioni del database e adottare nuove funzionalità di database con tempi di inattività brevi e prevedibili. Blue/green le implementazioni riducono i rischi e i tempi di inattività per gli aggiornamenti dei database, ad esempio gli aggiornamenti principali o secondari delle versioni del motore.

Le implementazioni blu/verde offrono i seguenti vantaggi:

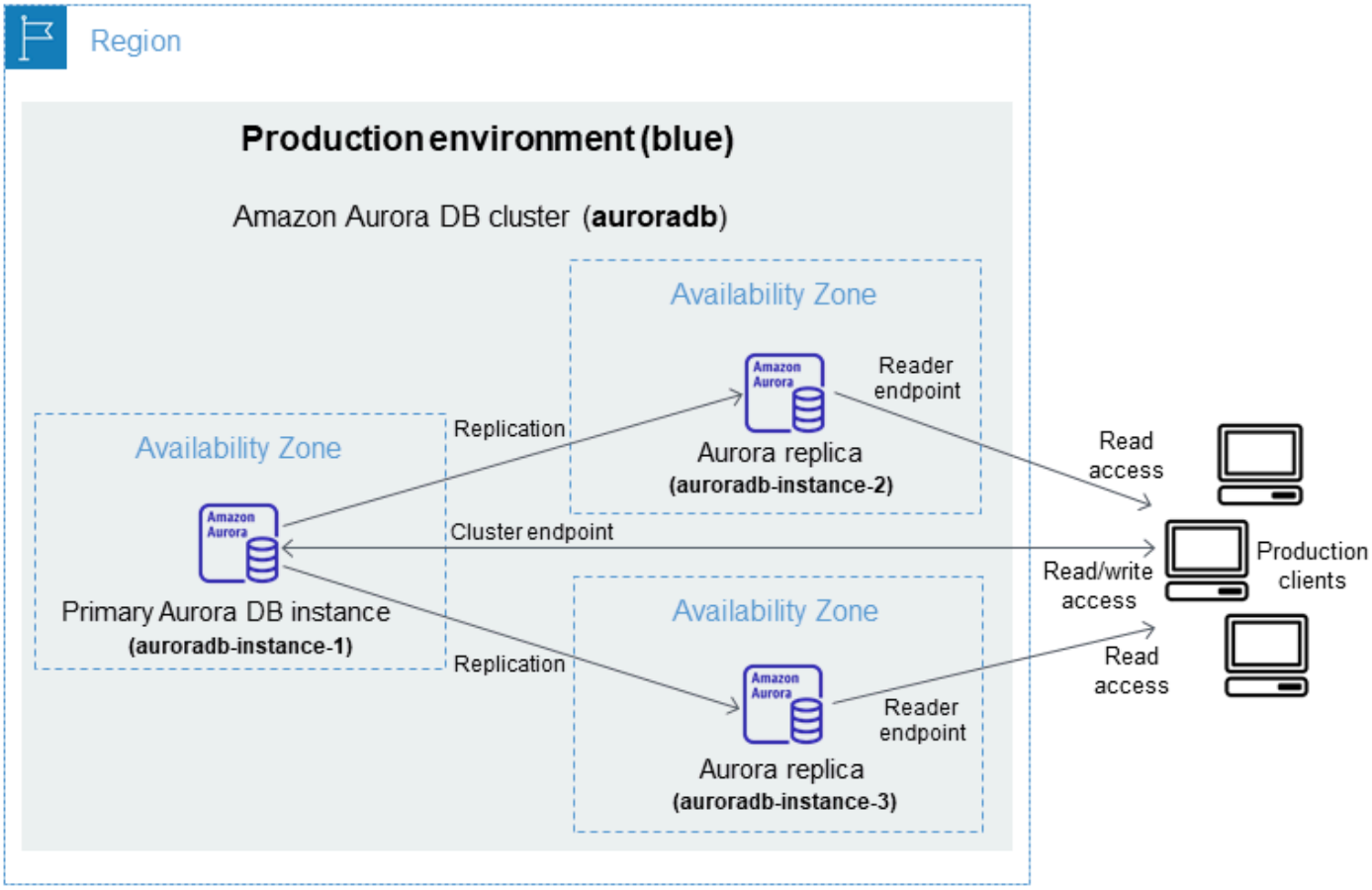
- Crea facilmente un ambiente di gestione temporanea pronto per la produzione.
- Replica automaticamente le modifiche del database dall'ambiente di produzione all'ambiente di gestione temporanea.
- Esegui il test delle modifiche del database in un ambiente di gestione temporanea sicuro, senza influire sull'ambiente di produzione.
- Rimani aggiornato con le patch del database e gli aggiornamenti di sistema.
- Implementa ed esegui il test delle nuove funzionalità del database.
- Esegui lo switchover dell'ambiente di gestione temporanea in un nuovo ambiente di produzione senza modificare l'applicazione.
- Esegui lo switchover in sicurezza usando i guardrail di switchover integrati.
- Elimina la perdita di dati durante lo switchover.
- Esegui lo switchover rapidamente, in genere in meno di un minuto a seconda del carico di lavoro.

Flusso di lavoro di una blue/green distribuzione

Completa i seguenti passaggi principali quando utilizzi una blue/green distribuzione per gli aggiornamenti del cluster Aurora DB.

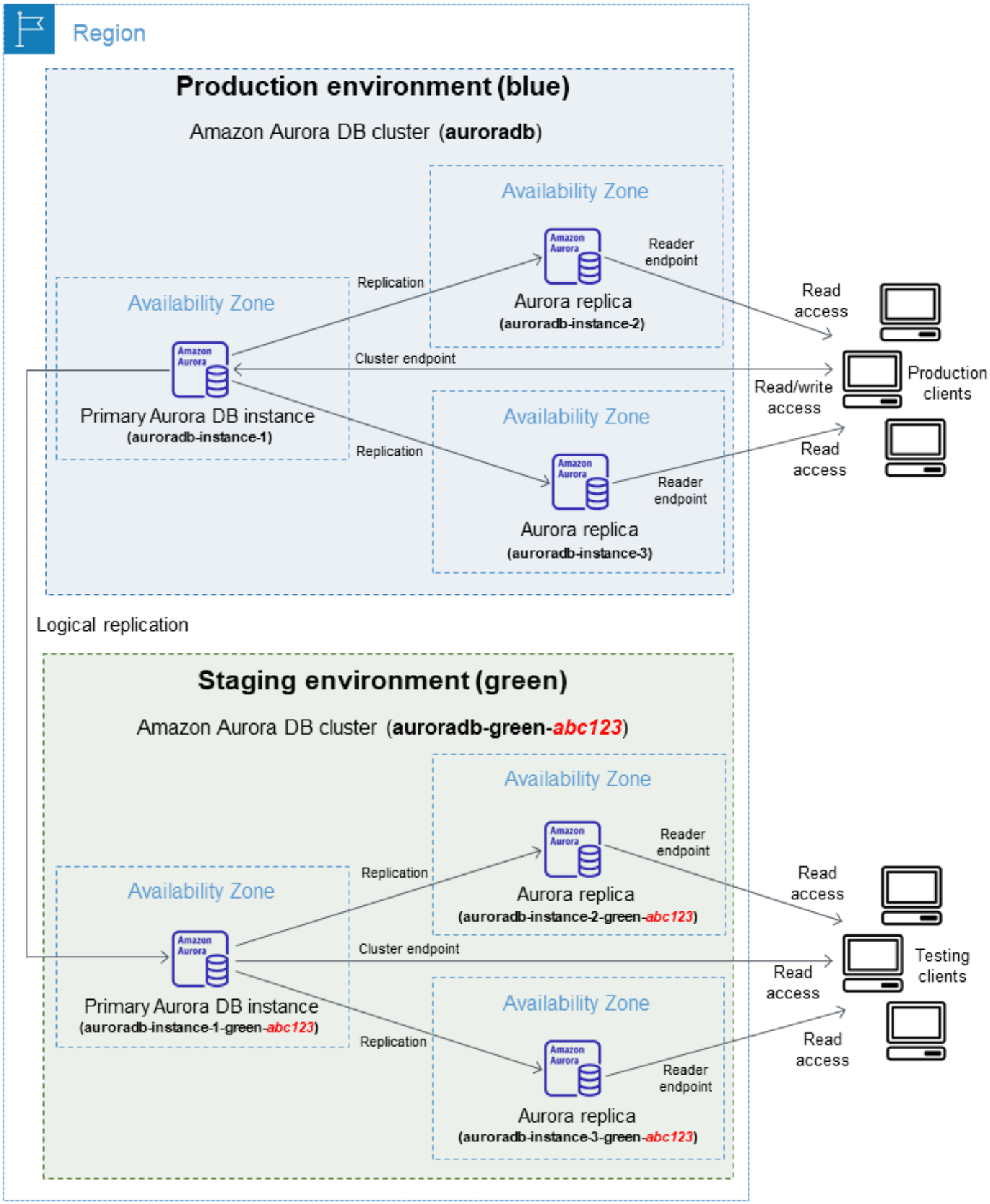
1. Identifica un cluster di database di produzione che richieda aggiornamenti.

   L'immagine seguente mostra un esempio di cluster di database di produzione.



2. Crea la blue/green distribuzione. Per istruzioni, consulta Creazione di una blue/green distribuzione in .

   L'immagine seguente mostra un esempio di blue/green implementazione dell'ambiente di produzione a partire dalla fase 1. Durante la creazione della blue/green distribuzione, RDS copia la topologia e la configurazione complete del cluster Aurora DB per creare l'ambiente verde. Ai nomi del cluster di database e delle istanze database copiati viene aggiunto `-green-`*`random-characters`*. L'ambiente di staging nell'immagine contiene il cluster DB (auroradb-green-). *`abc123`* Contiene anche le tre istanze DB nel cluster DB (auroradb-instance1-green-, auroradb-instance2-green- e auroradb-instance3-green-). *`abc123`* *`abc123`* *`abc123`*


Region
Production environment (blue)
Amazon Aurora DB cluster (auroradb)
Availability Zone
Amazon Aurora
Reader endpoint
Aurora replica
(auroradb-instance-2)
Replication
Availability Zone
Read access
Cluster endpoint
Production clients
Read/write access
Primary Aurora DB instance
(auroradb-instance-1)
Availability Zone
Replication
Read access
Reader endpoint
Aurora replica
(auroradb-instance-3)
Logical replication
Staging environment (green)
Amazon Aurora DB cluster (auroradb-green-abc123)
Availability Zone
Amazon Aurora
Reader endpoint
Aurora replica
(auroradb-instance-2-green-abc123)
Replication
Availability Zone
Read access
Cluster endpoint
Testing clients
Read access
Primary Aurora DB instance
(auroradb-instance-1-green-abc123)
Availability Zone
Replication
Read access
Reader endpoint
Aurora replica
(auroradb-instance-3-green-abc123)

Quando si crea la blue/green distribuzione, è possibile specificare una versione superiore del motore DB e un gruppo di parametri del cluster DB diverso per il cluster DB nell'ambiente verde. È anche possibile specificare un gruppo di parametri database diverso per le istanze database nel cluster di database.

RDS configura anche la replica dall'istanza database primaria dell'ambiente blu all'istanza database primaria dell'ambiente verde.

> **Important**
>
> Per Aurora MySQL versione 3, dopo aver creato la blue/green distribuzione, il cluster DB nell'ambiente verde non consente le operazioni di scrittura per impostazione predefinita. Tuttavia, questo approccio non si applica agli utenti che dispongono del privilegio `CONNECTION_ADMIN`, incluso l'utente master di Aurora. Gli utenti con questo privilegio possono ignorare il comportamento `read_only`. Per ulteriori informazioni, consulta Privilegio basato sui ruoli.

3. Apporta le modifiche all'ambiente di gestione temporanea.

   Ad esempio, è possibile modificare la classe di istanza database utilizzata da una o più istanze database nell'ambiente verde.

   Per ulteriori informazioni sulla modifica di un cluster di database, consulta Modifica di un cluster database Amazon Aurora.

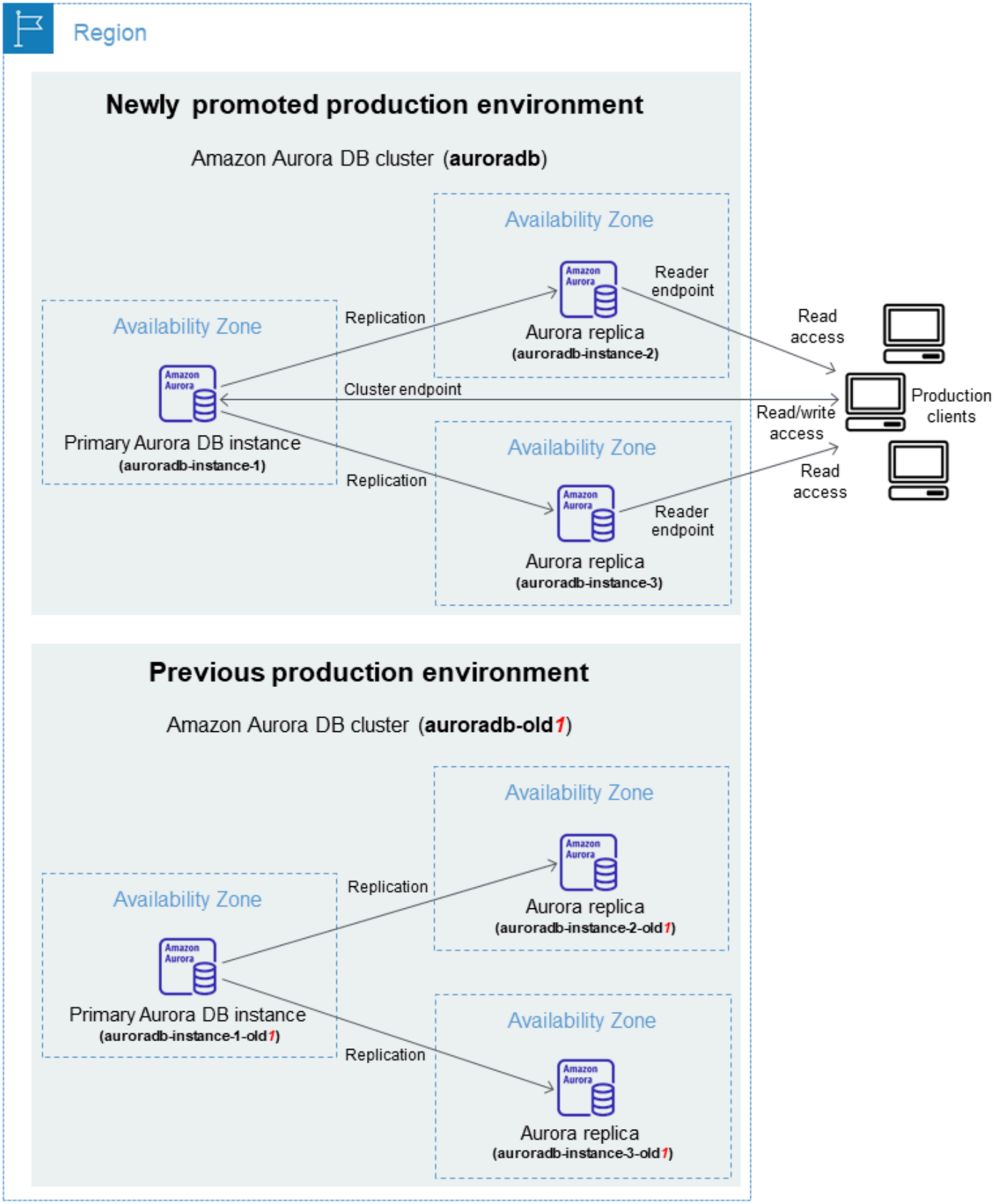
4. Esegui il test dell'ambiente di gestione temporanea.

   Durante i test, ti consigliamo di mantenere i database in un ambiente verde di sola lettura. Abilita le operazioni di scrittura nell'ambiente verde con cautela perché possono causare conflitti di replica nell'ambiente verde. Possono inoltre generare dati non previsti nei database di produzione dopo lo switchover. Per abilitare le operazioni di scrittura per Aurora MySQL, imposta il parametro `read_only` su `0`, quindi riavvia l'istanza database. Per Aurora PostgreSQL, imposta il parametro `default_transaction_read_only` su `off` a livello di sessione.

5. Quando è tutto pronto, è possibile eseguire lo switchover per passare l'ambiente di staging come nuovo ambiente di produzione. Per istruzioni, consulta Cambiare una blue/green distribuzione in .

   Lo switchover comporta tempi di inattività. I tempi di inattività sono in genere inferiori al minuto, ma possono essere più lunghi a seconda del carico di lavoro.

L'immagine seguente mostra i cluster di database dopo lo switchover.


Region
Newly promoted production environment
Amazon Aurora DB cluster (auroradb)
Availability Zone
Amazon Aurora
Reader endpoint
Availability Zone
Replication
Aurora replica
(auroradb-instance-2)
Read access
Cluster endpoint
Production clients
Read/write access
Primary Aurora DB instance
(auroradb-instance-1)
Availability Zone
Replication
Read access
Reader endpoint
Aurora replica
(auroradb-instance-3)
Previous production environment
Amazon Aurora DB cluster (auroradb-old1)
Availability Zone
Availability Zone
Replication
Aurora replica
(auroradb-instance-2-old1)
Primary Aurora DB instance
(auroradb-instance-1-old1)
Availability Zone
Replication
Aurora replica
(auroradb-instance-3-old1)

Dopo lo switchover, il cluster di database Aurora dell'ambiente verde diventa il nuovo cluster di database di produzione. I nomi e gli endpoint dell'ambiente di produzione corrente vengono assegnati all'ambiente di produzione appena sottoposto allo switchover e non sono richieste modifiche all'applicazione. Di conseguenza, il traffico di produzione ora viene indirizzato al nuovo ambiente di produzione. Il cluster di database e le istanze database nell'ambiente blu vengono rinominati aggiungendo `-old`*n* al nome corrente, dove *n* è un numero. Ad esempio, supponi che il nome dell'istanza database nell'ambiente blu sia `auroradb-instance-1`. Dopo lo switchover, il nome dell'istanza database diventa `auroradb-instance-1-old1`.

Nell'esempio dell'immagine, durante lo switchover si verificano le seguenti modifiche:

- Il cluster di database dell'ambiente verde `auroradb-green-abc123` diventa il cluster di database di produzione denominato `auroradb`.
- L'istanza database dell'ambiente verde denominata `auroradb-instance1-green-abc123` diventa l'istanza database di produzione `auroradb-instance1`.
- L'istanza database dell'ambiente verde denominata `auroradb-instance2-green-abc123` diventa l'istanza database di produzione `auroradb-instance2`.
- L'istanza database dell'ambiente verde denominata `auroradb-instance3-green-abc123` diventa l'istanza database di produzione `auroradb-instance3`.
- Il cluster di database dell'ambiente blu denominato `auroradb` diventa `auroradb-old1`.
- L'istanza database dell'ambiente blu denominata `auroradb-instance1` diventa `auroradb-instance1-old1`.
- L'istanza database dell'ambiente blu denominata `auroradb-instance2` diventa `auroradb-instance2-old1`.
- L'istanza database dell'ambiente blu denominata `auroradb-instance3` diventa `auroradb-instance3-old1`.

6. Se non hai più bisogno di una blue/green distribuzione, puoi eliminarla. Per istruzioni, consulta Eliminazione di un'implementazione blu/verde in Amazon Aurora.

   Dopo lo switchover, l'ambiente di produzione precedente non viene eliminato, quindi è possibile utilizzarlo per i test di regressione, se necessario.

Autorizzazione dell'accesso alle operazioni di Amazon blue/green Aurora

Gli utenti devono disporre delle autorizzazioni necessarie per eseguire operazioni relative alle implementazioni blu/verde. Puoi creare le policy IAM che concedono a utenti e ruoli l'autorizzazione per eseguire operazioni API specifiche sulle risorse indicate di cui hanno bisogno. Puoi quindi collegare tali policy ai ruoli o ai set di autorizzazioni IAM che richiedono le autorizzazioni. Per ulteriori informazioni, consulta Gestione accessi e identità per Amazon Aurora.

L'utente che crea una blue/green distribuzione deve disporre delle autorizzazioni per eseguire le seguenti operazioni RDS:

- `rds:CreateBlueGreenDeployment`
- `rds:AddTagsToResource`
- `rds:CreateDBCluster`
- `rds:CreateDBInstance`
- `rds:CreateDBClusterEndpoint`

L'utente che passa a una blue/green distribuzione deve disporre delle autorizzazioni per eseguire le seguenti operazioni RDS:

- `rds:SwitchoverBlueGreenDeployment`
- `rds:ModifyDBCluster`
- `rds:PromoteReadReplicaDBCluster`

L'utente che elimina una blue/green distribuzione deve disporre delle autorizzazioni per eseguire le seguenti operazioni RDS:

- `rds:DeleteBlueGreenDeployment`
- `rds:DeleteDBCluster`
- `rds:DeleteDBInstance`
- `rds:DeleteDBClusterEndpoint`

Aurora alloca e modifica le risorse nell'ambiente di staging per conto dell'utente. Queste risorse includono istanze database che utilizzano una convenzione di denominazione definita internamente. Pertanto, le policy IAM allegate non possono contenere modelli di nomi di risorse parziali come `my-`

`db-prefix-*`. Solo i caratteri jolly (*) sono supportati. In generale, è consigliabile utilizzare tag di risorsa e altri attributi supportati per controllare l'accesso a queste risorse, anziché i caratteri jolly. Per ulteriori informazioni, consulta Operazioni, risorse e chiavi di condizione per Amazon RDS.

Autorizzazioni aggiuntive per Aurora Global Database Deployments Blue/Green

Quando si creano blue/green distribuzioni per i cluster Aurora Global Database, oltre alle autorizzazioni sopra elencate, gli utenti necessitano delle seguenti autorizzazioni per eseguire operazioni di gestione della topologia globale del cluster.

L'utente che crea una blue/green distribuzione deve disporre delle autorizzazioni per eseguire le seguenti operazioni RDS:

- `rds:CreateGlobalCluster`

L'utente che passa a una blue/green distribuzione deve disporre delle autorizzazioni per eseguire le seguenti operazioni RDS:

- `rds:ModifyGlobalCluster`
- `rds:PromoteReadReplicaDBCluster`

L'utente che elimina una blue/green distribuzione deve disporre delle autorizzazioni per eseguire le seguenti operazioni RDS:

- `rds:DeleteGlobalCluster`

Limitazioni e considerazioni per le distribuzioni di Amazon Aurora blue/green

Le implementazioni blu/verde in Amazon RDS richiedono un'attenta considerazione dei fattori quali gli slot di replica, la gestione delle risorse, il dimensionamento delle istanze e i potenziali impatti sulle prestazioni del database. Le sezioni seguenti forniscono indicazioni per ottimizzare la strategia di implementazione e garantire un tempo di inattività minimo, transizioni fluide e una gestione efficace dell'ambiente di database.

Argomenti

- blue/green Limitazioni per le distribuzioni

- Limitazioni di Aurora Global Database per le distribuzioni blue/green
- Considerazioni per le implementazioni blue/green

blue/green Limitazioni per le distribuzioni

Le seguenti limitazioni si applicano alle blue/green distribuzioni.

Argomenti

- Limitazioni generali per le implementazioni blu/verde
- Aurora MySQL for MySQL: limitazioni per le implementazioni blue/green
-

Limitazioni generali per le implementazioni blu/verde

Le seguenti limitazioni generali si applicano alle blue/green distribuzioni:

- Non è possibile interrompere e avviare un cluster che fa parte di una blue/green distribuzione.
- Le implementazioni blu/verde non supportano la gestione delle password degli utenti master con Gestione dei segreti AWS.
- Se si tenta di forzare un backtrack sul cluster DB blu, la blue/green distribuzione si interrompe e lo switchover viene bloccato.
- Durante il passaggio, gli ambienti blu e verdi non possono avere integrazioni Zero-ETL con Amazon Redshift. Occorre prima eliminare l'integrazione ed eseguire il passaggio, quindi ricreare l'integrazione.
- L'Event Scheduler (`event_scheduler`parametro) deve essere disabilitato nell'ambiente verde quando si crea una distribuzione. blue/green Ciò impedisce che si generino eventi nell'ambiente verde e conseguentemente incongruenze.
- Le policy di dimensionamento automatico configurate sul cluster di database blu non vengono copiate nell'ambiente verde. È necessario riconfigurarle dopo lo switchover, indipendentemente dal fatto che siano state inizialmente configurate nell'ambiente blu o verde.
- Non è possibile modificare un cluster di database decrittografato in un cluster di database crittografato. Inoltre, non è possibile modificare un cluster di database crittografato in un cluster di database decrittografato.
- Non è possibile modificare un cluster di database dell'ambiente blu con una versione successiva del motore rispetto al cluster di database dell'ambiente verde.

- Le risorse nell'ambiente blu e nell'ambiente verde devono trovarsi nello stesso Account AWS.
- Le implementazioni blu/verde non sono supportate per le seguenti funzionalità:
  - Server proxy per Amazon RDS
  - Repliche di lettura tra regioni diverse
  - Cluster di database Aurora Serverless v1
  - CloudFormation

Aurora MySQL for MySQL: limitazioni per le implementazioni blue/green

blue/green

- Il cluster di database di origine non può contenere database denominati `tmp`. I database con questo nome non verranno copiati nell'ambiente verde.
- Il cluster di database dell'ambiente blu non può essere una replica di log binari esterna.
- Se il cluster di database di origine ha il backtrack abilitato, il cluster di database dell'ambiente verde viene creato senza supporto per il backtrack. Questo perché il backtracking non funziona con la replica dei log binari (binlog), necessaria per le distribuzioni. blue/green Per ulteriori informazioni, consulta the section called "Backtrack di un cluster database".
- Le distribuzioni blu/green non supportano il driver AWS JDBC per MySQL. Per ulteriori informazioni, consulta Limitazioni note su. GitHub

Le seguenti limitazioni si applicano alle implementazioni blu/verde di Aurora PostgreSQL blue/green deployments.

- Le tabelle non registrate nei log non vengono replicate nell'ambiente verde a meno che il parametro `rds.logically_replicate_unlogged_tables` non sia impostato su `1` nel cluster di database blu. Non modificare il valore di questo parametro dopo aver creato una blue/green distribuzione per evitare possibili errori di replica su tabelle non registrate.
- Il cluster di database blu non può essere un'origine logica (publisher) o una replica (subscriber).
- Se il cluster di database blu è configurato come server esterno di un'estensione FDW (Foreign Data Wrapper), è necessario utilizzare il nome dell'endpoint del cluster anziché gli indirizzi IP. Ciò consente alla configurazione di rimanere funzionale dopo lo switchover.
- In una blue/green distribuzione, ogni database richiede uno slot di replica logico. Con l'aumentare del numero di database, incrementa il sovraccarico delle risorse che può potenzialmente causare

ritardi nella replica, soprattutto se il dimensionamento del cluster di database non è sufficiente. L'impatto dipende da fattori quali il carico di lavoro del database e il numero di connessioni. Per mitigare questo problema, è possibile considerare la possibilità di aumentare verticalmente la classe dell'istanza database o di ridurre il numero di database del cluster.

- Le implementazioni blu/verde sono supportate in Babelfish per Aurora PostgreSQL solo per la versione 15.7 e le versioni successive alla 15 e per la versione 16.3 e le versioni successive alla 16.
- Se desideri acquisire i piani di esecuzione nelle repliche di Aurora, devi fornire l'endpoint blu del cluster di database quando chiami la funzione `apg_plan_mgmt.create_replica_plan_capture`. Ciò garantisce che le acquisizioni dei piani continuino a funzionare dopo lo switchover. Per ulteriori informazioni, consulta the section called "Acquisizione dei piani di esecuzione Aurora PostgreSQL nelle repliche".
- Il processo di applicazione della replica logica nell'ambiente verde è a thread singolo. Se l'ambiente blu genera un volume elevato di traffico di scrittura, l'ambiente verde potrebbe non essere in grado di tenere il passo. Ciò può causare ritardi o errori di replica, in particolare per i carichi di lavoro che producono un throughput di scrittura elevato. È necessario testare accuratamente i carichi di lavoro. Per gli scenari che richiedono aggiornamenti della versione principale e la gestione di carichi di lavoro di scrittura a volume elevato, si possono prendere in considerazione approcci alternativi come l'utilizzo di AWS Database Migration Service (AWS DMS) o la replica logica autogestita.
- La creazione di nuove partizioni sulle tabelle partizionate non è supportata durante le implementazioni blu/verde per Aurora. La creazione di nuove partizioni implica operazioni DDL (Data Definition Language) come `CREATE TABLE`, che non vengono replicate dall'ambiente blu nell'ambiente verde. Tuttavia, le tabelle partizionate esistenti e i relativi dati verranno replicati nell'ambiente verde.
- Alle estensioni PostgreSQL si applicano le seguenti limitazioni:
  - L'`pg_partman`estensione deve essere disabilitata nell'ambiente blu quando si crea una distribuzione. blue/green L'estensione esegue operazioni DDL come `CREATE TABLE` che interrompono la replica logica dall'ambiente blu all'ambiente verde.
  - L'`pg_cron`estensione deve rimanere disabilitata su tutti i database verdi dopo la creazione della blue/green distribuzione. L'estensione dispone di worker in background che vengono eseguiti come superutente e aggirano l'impostazione di sola lettura dell'ambiente verde, il che potrebbe causare conflitti di replica.
  - L'estensione `apg_plan_mgmt` deve avere il parametro `apg_plan_mgmt.capture_plan_baselines` impostato su `off` in tutti i database verdi per

evitare conflitti di chiave primaria se un piano identico viene acquisito nell'ambiente blu. Per ulteriori informazioni, consulta the section called “Panoramica della gestione del piano di query per Aurora PostgreSQL”.

- Le `pgactive` estensioni `pglogical` e devono essere disabilitate nell'ambiente blu quando si crea una blue/green distribuzione. Dopo aver eseguito lo switchover dell’ambiente verde nel nuovo ambiente di produzione, si possono abilitare nuovamente le estensioni. Inoltre, il database blu non può essere un abbonato logico di un'istanza esterna.
- Se utilizzata, l’estensione `pgAudit` deve rimanere nelle librerie condivise (`shared_preload_libraries`) nei gruppi di parametri di database personalizzati sia per le istanze database blu sia per quelle verdi. Per ulteriori informazioni, consulta the section called “Configurazione dell'estensione pgAudit”.

Limitazioni specifiche della replica logica per le distribuzioni blue/green

La tabella seguente descrive le limitazioni della replica logica che si applicano alle implementazioni blu/verde per Aurora PostgreSQL. Per ulteriori informazioni, consulta Restrictions nella documentazione della replica logica di PostgreSQL.

Limitazione	Spiegazione
Le istruzioni DDL (Data Definitio n Language), come `CREATE TABLE` e `CREATE SCHEMA`, non vengono replicate dall'ambiente blu nell'ambiente verde.	Se Aurora rileva una modifica DDL nell'ambiente blu, i database verdi entrano nello stato di replica degradata. È necessario eliminare la blue/green distribuzione e tutti i database ecologici, quindi ricrearli.
Le istruzioni DCL (Data Control Language), come `GRANT` e `REVOKE`, non vengono	Se Aurora rileva un tentativo di eseguire un’istruzione DCL nell’ambiente blu, verrà visualizzato un messaggio di avviso. Non è disponibile alcuna configura zione o API per modificare questo comportamento, poiché si tratta di una limitazione del processo di blue/green distribuzione.

Limitazione	Spiegazione
replicate dall'ambiente blu nell'ambiente verde.	
Le operazioni `NEXTVAL` sugli oggetti della sequenza non sono sincronizzate tra l'ambiente blu e l'ambiente verde.	Durante lo switchover, Aurora incrementa i valori di sequenza nell'ambiente verde in modo che corrispondano a quelli nell'ambiente blu. Se hai migliaia di sequenze, lo switchover può subire un ritardo.
Gli oggetti di grandi dimensioni nell'ambiente blu non vengono replicati nell'ambiente verde. Ciò include sia gli oggetti di grandi dimensioni esistenti che gli oggetti di grandi dimensioni appena creati o modificati durante il processo di blue/green distribuzione.	Se Aurora rileva la creazione o la modifica di oggetti di grandi dimensioni nell'ambiente blu archiviati nella tabella di sistema `pg_largeobject` , i database verdi entrano nello stato di replica degradata. È necessario eliminare la blue/green distribuzione e tutti i database verdi, quindi ricrearli.

Limitazione	Spiegazione
L’aggiornamento delle viste materializzate interrompe la replica.	L’aggiornamento delle viste materializzate nell’ambiente blu interrompe la replica nell’ambiente verde. È consigliabile evitare di aggiornare le viste materializzate nell’ambiente blu. Dopo uno switchover, è possibile aggiornar le manualmente utilizzando il comando REFRESH MATERIALIZED VIEW o pianificare un aggiornamento.
Le operazion i UPDATE e DELETE non sono consentite nelle tabelle che non dispongon o di una chiave primaria.	Prima di creare una blue/green distribuzione, assicurati che tutte le tabelle abbiano una chiave o un utilizzo `REPLICA IDENTITY FULL` primari. Tuttavia, `REPLICA IDENTITY FULL` si utilizza solo se non esiste una chiave primaria o univoca, poiché influisce sulle prestazioni di replica. Per ulteriori informazioni, consultare la documentazione di PostgreSQL.

Limitazioni di Aurora Global Database per le distribuzioni blue/green

Oltre alle limitazioni generali e specifiche del motore sopra indicate, si applicano le seguenti limitazioni per le blue/green distribuzioni per Aurora Global Database:

- Tutte le operazioni devono essere avviate dalla stessa regione del cluster di scrittura del Global Database.
- L'esecuzione di uno switchover globale o di un failover globale renderà non valida la blue/green distribuzione attiva. La distribuzione blu-verde deve essere eliminata e ricreata dalla nuova regione primaria.
- Per Aurora PostgreSQL, se hai abilitato l'inoltro globale di scrittura nell'ambiente di produzione e crei una blue/green distribuzione, l'inoltro di scrittura è disabilitato nel cluster verde. È abilitato nell'ambiente verde solo dopo il blue/green passaggio, quando l'ambiente verde diventa il nuovo ambiente di produzione. Dopo il passaggio, l'inoltro della scrittura è disabilitato nel cluster. `-old1`
- La modifica della topologia del database globale dopo la creazione della blue/green distribuzione renderà non valida la distribuzione attiva. blue/green La distribuzione blu-verde dovrebbe essere eliminata e ricreata dalla nuova regione primaria.

- Le istantanee automatizzate vengono conservate per i giorni di conservazione dei backup originariamente configurati nel vecchio ambiente blu. Le istantanee automatiche del vecchio cluster blu non vengono copiate in verde.
- Il failover globale è supportato durante uno blue/green switchover, ma uno switchover globale non è supportato durante uno switchover. blue/green
- Assicurati che il cluster DB e i gruppi di parametri DB per l'ambiente verde esistano in tutte le aree secondarie con nomi identici. Se il gruppo di parametri in qualsiasi regione non è disponibile, viene utilizzato il gruppo di parametri predefinito nelle regioni.
- Evita di utilizzare il proxy RDS su qualsiasi membro del database globale durante il passaggio blue/green alla distribuzione.

Considerazioni per le implementazioni blue/green

Amazon RDS tiene traccia delle risorse nelle blue/green distribuzioni con l'ausilio `DbClusterResourceId` di `DbiResourceId` ogni risorsa. Questo ID di risorsa è un identificatore Regione AWS univoco e immutabile per la risorsa.

L'ID della risorsa è separato dall'ID del cluster database. Ciascuno di essi è elencato nella configurazione del database nella console RDS.

Il nome (ID del cluster) di una risorsa cambia quando si passa a una blue/green distribuzione, ma ogni risorsa mantiene lo stesso ID di risorsa. Ad esempio, un identificatore di cluster di database potrebbe essere `mycluster` nell'ambiente blu. Dopo lo switchover, lo stesso cluster di database potrebbe essere rinominato in `mycluster-old1`. Tuttavia, l'ID risorsa del cluster di database non viene modificato durante lo switchover. Pertanto, quando si sostituiscono le risorse verdi con le nuove risorse di produzione, le relative risorse IDs non corrispondono alla risorsa IDs blu precedentemente in produzione.

Dopo il passaggio a una blue/green distribuzione, valuta la possibilità di aggiornare la risorsa IDs con quelle delle risorse di produzione appena trasferite per le funzionalità e i servizi integrati che hai utilizzato con le risorse di produzione. In particolare, considera i seguenti aggiornamenti:

- Se esegui il filtraggio utilizzando l'API e la risorsa RDS IDs, modifica la risorsa IDs utilizzata nel filtraggio dopo il passaggio.
- Se le utilizzate CloudTrail per il controllo delle risorse, impostate i consumatori della risorsa in modo che tengano traccia della nuova risorsa dopo il CloudTrail passaggio al digitale. IDs

Per ulteriori informazioni, consulta Monitoraggio delle chamate API di Amazon Aurora in AWS CloudTrail.

- Se utilizzi i flussi di attività del database per le risorse dell'ambiente blu, regola l'applicazione per monitorare gli eventi del database per il nuovo flusso dopo lo switchover. Per ulteriori informazioni, consulta Regioni e motori di database Aurora supportati per flussi di attività del database.
- Se utilizzi l'API Performance Insights, modifica la risorsa IDs nelle chiamate all'API dopo lo switchover. Per ulteriori informazioni, consulta Monitoraggio del carico DB con Performance Insights su Amazon Aurora.

  Dopo lo switchover è possibile monitorare un database con lo stesso nome, ma non contiene i dati precedenti allo lo switchover.
- Se utilizzi una risorsa IDs nelle policy IAM, assicurati di aggiungere la risorsa IDs delle risorse appena trasferite quando necessario. Per ulteriori informazioni, consulta Gestione accessi e identità per Amazon Aurora.
- Se sono disponibili ruoli IAM associati al cluster di database, è necessario riassociarli dopo lo switchover. I ruoli collegati non vengono copiati automaticamente nell'ambiente verde.
- Se si esegue l'autenticazione nel cluster database utilizzando l'autenticazione del database IAM, assicurarsi che la policy IAM utilizzata per l'accesso al database contenga sia i database blu che quelli verdi elencati sotto l'elemento `Resource` della policy. Ciò è necessario per connettersi al database verde dopo il passaggio. Per ulteriori informazioni, consulta the section called "Creazione e utilizzo di una policy IAM per l'accesso al database IAM".
- Se desideri ripristinare uno snapshot manuale del cluster DB per un cluster DB che faceva parte di una blue/green distribuzione, assicurati di ripristinare lo snapshot corretto del cluster DB esaminando l'ora in cui è stata scattata la snapshot. Per ulteriori informazioni, consulta Ripristino da uno snapshot cluster database.
- Dopo il passaggio, le attività di replica AWS Database Migration Service (AWS DMS) non possono riprendere perché il checkpoint dall'ambiente blu non è valido nell'ambiente verde. È necessario ricreare l'attività DMS con un nuovo checkpoint per continuare la replica.
- Amazon Aurora crea l'ambiente verde clonando il volume di archiviazione di Aurora sottostante nell'ambiente blu. Il volume del cluster verde memorizza solo le modifiche incrementali apportate all'ambiente verde. Se elimini il cluster di database nell'ambiente blu, la dimensione del volume di archiviazione di Aurora sottostante nell'ambiente verde aumenta fino a raggiungere la dimensione completa. Per ulteriori informazioni, consulta the section called "Clonazione di un volume per un cluster database Aurora".

- Quando aggiungi un'istanza database al cluster di database nell'ambiente verde di un'implementazione blu/verde, la nuova istanza database non sostituisce un'istanza database nell'ambiente blu al momento dello switchover. Tuttavia, la nuova istanza database viene mantenuta nel cluster di database e diventa un'istanza database nel nuovo ambiente di produzione.
- Quando elimini un'istanza DB nel cluster DB nell'ambiente verde di una blue/green deployment, you can't create a new DB instance to replace it in the blue/green distribuzione.

  Se crei una nuova istanza database con lo stesso nome e nome della risorsa Amazon (ARN) dell'istanza database eliminata, ha un `DbiResourceId` diverso e quindi non fa parte dell'ambiente verde.

  Se elimini un'istanza database nel cluster di database nell'ambiente verde, si verifica il seguente comportamento:

  - Se l'istanza database con lo stesso nome esiste nell'ambiente blu, non verrà eseguito lo switchover all'istanza database dell'ambiente verde. Questa istanza database non verrà rinominata aggiungendo `-old`*n* al suo nome.
  - Qualsiasi applicazione che punti all'istanza database nell'ambiente blu continua a utilizzare la stessa istanza database dopo lo switchover.
- Se si utilizzano i tag delle risorse per il controllo degli accessi o la gestione operativa, è necessario comprendere che le modifiche ai tag non vengono sincronizzate tra ambienti blu e verdi fino al passaggio. Quando crei una blue/green distribuzione, i tag dall'ambiente blu vengono copiati nell'ambiente verde. Dopo la creazione, tutte le modifiche ai tag apportate a entrambi gli ambienti non vengono sincronizzate automaticamente. Durante lo switchover, i tag di ambiente blu sostituiscono tutti i tag nell'ambiente verde. Applica tutti i tag necessari all'ambiente blu prima di creare la blue/green distribuzione o riapplica i tag richiesti al nuovo ambiente di produzione dopo il passaggio. Per ulteriori informazioni sui tag, consulta Applicazione di tag alle risorse Amazon Aurora e Amazon RDS.

Best practice per le implementazioni di Amazon Aurora blue/green

Di seguito sono riportate le best practice per le distribuzioni. blue/green

Argomenti

- Procedure consigliate generali per le implementazioni blue/green
- per le implementazioni blue/green

- Best practice di Aurora PostgreSQL per le implementazioni blue/green
- Aurora Global Database: best practice per le implementazioni blue/green

Procedure consigliate generali per le implementazioni blue/green

Le seguenti best practice generali si utilizzano quando si crea un'implementazione blu/verde.

- Esegui accuratamente il test del cluster di database Aurora nell'ambiente verde prima di effettuare lo switchover.
- Mantieni i tuoi database nell'ambiente verde di sola lettura. Si consiglia di abilitare le operazioni di scrittura nell'ambiente verde con cautela perché possono causare conflitti di replica nell'ambiente verde. Possono inoltre generare dati non previsti nei database di produzione dopo lo switchover.
- Se utilizzi una blue/green distribuzione per implementare modifiche allo schema, apporta solo modifiche compatibili con la replica.

  Ad esempio, è possibile aggiungere nuove colonne alla fine di una tabella senza interrompere la replica dall'implementazione blu all'implementazione verde. Tuttavia, le modifiche dello schema, come la ridenominazione delle colonne o delle tabelle, interrompono la replica nell'implementazione verde.

  Per ulteriori informazioni sulle modifiche compatibili con la replica, consulta Replication with Differing Table Definitions on Source and Replica nella documentazione MySQL e Restrictions nella documentazione della replica logica PostgreSQL.
- Usa l'endpoint del cluster, l'endpoint di lettura o l'endpoint personalizzato per tutte le connessioni in entrambi gli ambienti. Non utilizzare endpoint di istanza o endpoint personalizzati con gli elenchi statici o di esclusione.
- Quando passi da una blue/green distribuzione all'altra, segui le migliori pratiche di commutazione. Per ulteriori informazioni, consulta the section called "Best practice per lo switchover".

per le implementazioni blue/green

Prendi in considerazione le seguenti best practice quando crei una blue/green distribuzione da un cluster .

- Se l'ambiente verde presenta un ritardo di replica, considera i seguenti requisiti:

- Disabilita la conservazione dei log binari se non è necessaria o disabilitala temporaneamente fino al recupero della replica. A tale scopo, reimposta il parametro del cluster di database `binlog_format` su `0` e riavvia l'istanza database di scrittura verde.
- Imposta temporaneamente il parametro `innodb_flush_log_at_trx_commit` su `0` nel gruppo di parametri di database verde. Una volta recuperata la replica, ripristina il valore predefinito di `1` prima dello switchover. Se si verifica un arresto imprevisto o anomalo con il valore temporaneo del parametro, ricrea l'ambiente verde per evitare il danneggiamento dei dati non rilevati. Per ulteriori informazioni, consulta the section called "Configurazione della frequenza di svuotamento del buffer dei registri".

Best practice di Aurora PostgreSQL per le implementazioni blue/green

Prendi in considerazione le seguenti best practice quando crei una blue/green distribuzione da un cluster Aurora PostgreSQL DB.

- Monitora la cache write-through della replica logica di Aurora PostgreSQL e apporta modifiche al buffer della cache, se necessario. Per ulteriori informazioni, consulta the section called "Monitoraggio della cache write-through della replica logica".
- Aumenta il valore del parametro di database `logical_decoding_work_mem` nell'ambiente blu. In questo modo si riduce la decodifica su disco e si utilizza invece la memoria. Per ulteriori informazioni, consulta the section called "Regolazione della memoria di lavoro per la decodifica logica".
  - È possibile monitorare l'overflow delle transazioni scritte su disco utilizzando la metrica. `ReplicationSlotDiskUsage` CloudWatch Questa metrica offre approfondimenti sull'utilizzo del disco degli slot di replica, che consentono di individuare quando i dati delle transazioni superano la capacità di memoria e vengono archiviati su disco. È possibile monitorare la memoria liberabile con la metrica. `FreeableMemory` CloudWatch Per ulteriori informazioni, consulta the section called "Parametri a livello di istanza per Amazon Aurora".
  - In Aurora PostgreSQL 14 e versioni successive, è possibile monitorare la dimensione dei file di overflow logico utilizzando la visualizzazione di sistema `pg_stat_replication_slots`.
- Aggiorna tutte le estensioni PostgreSQL alla versione più recente prima di creare una distribuzione. blue/green Per ulteriori informazioni, consulta the section called "Aggiornamento estensioni PostgreSQL".
- Se utilizzi l'estensione `aws_s3`, assicurati di fornire al cluster di database verde l'accesso ad Amazon S3 tramite un ruolo IAM dopo la creazione dell'ambiente verde. In tal modo i comandi di

importazione ed esportazione continuano a funzionare dopo lo switchover. Per istruzioni, consulta the section called “Configurazione dell'accesso a un bucket Amazon S3”.

- Se specifichi una versione del motore superiore per l’ambiente verde, esegui l’operazione `ANALYZE` su tutti i database per aggiornare la tabella `pg_statistic`. Le statistiche di ottimizzazione non vengono trasferite durante un aggiornamento della versione principale, quindi è necessario rigenerare tutte le statistiche per evitare problemi di prestazioni. Per ulteriori best practice per gli aggiornamenti della versione principale, consulta the section called “Esecuzione di un aggiornamento alla versione principale”.
- Evita di configurare i trigger come `ENABLE REPLICA` o `ENABLE ALWAYS` se il trigger viene utilizzato sull’origine per manipolare i dati. In caso contrario, il sistema di replica propaga le modifiche ed esegue il trigger, che porta a una duplicazione.
- Le transazioni di lunga durata possono causare un notevole ritardo di replica. Per ridurre il ritardo di replica, valuta le seguenti operazioni:
  - Riduci le transazioni di lunga durata e le transazioni secondarie che possono essere ritardate fino a quando l’ambiente verde non raggiunge il livello dell’ambiente blu.
  - Riduci le operazioni di massa sull’ambiente blu fino a quando l’ambiente verde non raggiungerà il livello dell’ambiente blu.
  - Avvia un'operazione manuale di congelamento sottovuoto su tabelle occupate prima di creare la distribuzione. blue/green Per istruzioni, consulta Esecuzione di un congelamento manuale del vacuum.
  - In PostgreSQL 12 e versioni successive, disabilita il parametro `index_cleanup` sulle tabelle di grandi dimensioni o particolarmente utilizzate per migliorare l’efficienza della manutenzione regolare dei database blu. Per ulteriori informazioni, consulta Vacuum di una tabella il più rapidamente possibile.

    > **Note**
    >
    > Saltare regolarmente la pulizia dell’indice durante il vacuum può causare un aumento dell’indice, che potrebbe compromettere le prestazioni di scansione. Come best practice, utilizzate questo approccio solo durante l'utilizzo di una blue/green distribuzione. Una volta completata l’implementazione, è consigliabile riprendere la manutenzione e la pulizia regolari dell’indice.

  - Il ritardo di replica può verificarsi se le istanze database blu e verdi sono sottodimensionate per il carico di lavoro. Assicurati che le istanze database non raggiungano i limiti di risorse per il tipo di

istanza. Per ulteriori informazioni, consulta the section called “Analizza l'utilizzo delle risorse con i parametri CloudWatch ”.

- Una replica lenta può causare il riavvio frequente di mittenti e destinatari, il che ritarda la sincronizzazione. Per assicurare che rimangano attivi, disabilita i timeout impostando il parametro `wal_sender_timeout` su `0` nell’ambiente blu e il parametro `wal_receiver_timeout` su `0` nell’ambiente verde.
- Esamina le prestazioni delle istruzioni UPDATE e DELETE e valuta se la creazione di un indice sulla colonna utilizzata nella clausola WHERE può ottimizzare queste query. Può migliorare le prestazioni quando le operazioni vengono ripetute nell’ambiente verde. Per ulteriori informazioni, consulta the section called “Controlla i filtri predicati per le query che generano attese”.
- Se utilizzi i trigger, assicurati che non interferiscano con la creazione, l’aggiornamento e l’eliminazione di oggetti `pg_catalog.pg_publication`, `pg_catalog.pg_subscription` e `pg_catalog.pg_replication_slots` i cui nomi iniziano con “rds”.
- Se Babelfish è abilitato sul cluster di database di origine, i seguenti parametri devono avere nel gruppo di parametri del cluster di database di destinazione per l’ambiente verde le stesse impostazioni del gruppo di parametri del cluster di database di origine:
  - rds.babelfish_status
  - babelfishpg_tds.tds_default_numeric_precision
  - babelfishpg_tds.tds_default_numeric_scale
  - babelfishpg_tsql.default_locale
  - babelfishpg_tsql.migration_mode
  - babelfishpg_tsql.server_collation_name

  Per ulteriori informazioni su questi parametri, consultare the section called “Impostazioni del gruppo di parametri del cluster database per Babelfish ”.

Aurora Global Database: best practice per le implementazioni blue/green

Oltre alle best practice generali e specifiche del motore sopra elencate, prendi in considerazione le seguenti best practice per l'istanza Aurora Global Database.

- Monitora le seguenti CloudWatch metriche per identificare i periodi di scarsa attività nell'ambiente di produzione:
  - `DatabaseConnections`
  - `ActiveTransactions`

Pianifica il blue/green passaggio al digitale durante la finestra di manutenzione pianificata o durante un periodo di bassa attività.

- Blue/Green switchover duration varies based on your workload and the number of secondary regions. When you initiate a blue/greenswitchover, il servizio attende che il ritardo della replica raggiunga lo zero prima di procedere. Si consiglia di verificare il ritardo della replica prima di iniziare uno switchover.
- Se intendi utilizzare un parametro DB o un gruppo di parametri DB Cluster diverso da quello predefinito per il tuo ambiente verde, crea il gruppo di parametri desiderato con lo stesso nome in tutte le regioni secondarie prima di iniziare la distribuzione. blue/green

Creazione di una blue/green distribuzione in

RDS copia la topologia e le funzionalità dell'ambiente blu in un'area di staging. Quando l'istanza database blu ha delle repliche di lettura, queste vengono copiate come repliche dell'istanza verde. Lo spazio di archiviazione allocato di tutte le repliche verdi corrisponde all'istanza primaria verde, mentre gli altri parametri di archiviazione vengono ereditati dalle repliche blu.

Quando si crea una blue/green distribuzione, si specifica il cluster DB da copiare nella distribuzione. Il cluster scelto è il cluster di database di produzione e diventa il cluster di database dell'ambiente blu. RDS copia la topologia dell'ambiente blu in un'area di gestione temporanea, insieme alle funzionalità configurate. Il cluster di database viene copiato nell'ambiente verde e RDS configura la replica dal cluster di database dell'ambiente blu nel cluster di database dell'ambiente verde. RDS copia anche tutte le istanze database del cluster di database.

Argomenti

- Preparazione di una implementazione blu/verde
- Specificazione delle modifiche durante la creazione di una distribuzione blue/green
- Creazione di un'implementazione blu/verde
- Impostazioni per la creazione di implementazioni blu/verde

Preparazione di una implementazione blu/verde

Esistono alcuni passaggi da eseguire prima di creare una blue/green distribuzione, a seconda del motore su cui è in esecuzione l'.

Argomenti

- Preparazione di un cluster Aurora MySQL DB per una distribuzione blue/green
- Preparazione di un cluster Aurora PostgreSQL DB per una distribuzione blue/green
- Preparazione di un cluster Aurora Global Database DB per una distribuzione blue/green

Preparazione di un cluster Aurora MySQL DB per una distribuzione blue/green

Prima di creare una blue/green distribuzione per un cluster Aurora MySQL DB, il cluster deve essere associato a un gruppo di parametri del cluster DB personalizzato con logging binario () attivato. `binlog_format` Il log binario è necessario per la replica dall'ambiente blu nell'ambiente verde. Sebbene qualsiasi formato binlog funzioni, consigliamo `ROW` per ridurre il rischio di incongruenze di replica. Per informazioni sulla creazione di un gruppo di parametri del·cluster di database personalizzato e sull'impostazione dei parametri, consulta the section called “Gruppi di parametri di cluster database”.

> **Note**
>
> L'abilitazione della registrazione binaria aumenta il numero di I/O operazioni su disco di scrittura nel cluster DB. È possibile monitorare l'utilizzo degli IOPS con la `VolumeWriteIOPs` CloudWatch metrica.

Dopo aver abilitato la registrazione binaria, assicuratevi di riavviare il cluster DB in modo che le modifiche abbiano effetto. Blue/green le distribuzioni richiedono che l'istanza writer sia sincronizzata con il gruppo di parametri del cluster DB, altrimenti la creazione fallisce. Per ulteriori informazioni, consulta Riavvio di un'istanza database in un cluster Aurora.

Inoltre, è consigliabile modificare il periodo di conservazione dei log binari su un valore diverso da `NULL` per impedire l’eliminazione dei file di log binari. Per ulteriori informazioni, consulta the section called “Impostazione e visualizzazione della configurazione dei log binari”.

Preparazione di un cluster Aurora PostgreSQL DB per una distribuzione blue/green

Prima di creare una blue/green distribuzione per un cluster Aurora PostgreSQL DB, assicurati di fare quanto segue.

- Associa il cluster a un gruppo di parametri del·cluster di database personalizzato con la replica logica (`rds.logical_replication`) abilitata. La replica logica è necessaria per la replica dall'ambiente blu nell'ambiente verde.

  Quando abiliti la replica logica, dovrai anche ottimizzare alcuni parametri del cluster, ad esempio `max_replication_slots`, `max_logical_replication_workers` e `max_worker_processes`. Per istruzioni su come abilitare la replica logica e ottimizzare questi parametri, consulta the section called “Configurazione della replica logica”.

  Inoltre, assicurati che il parametro `synchronous_commit` sia impostato su on.

  Dopo aver configurato i parametri richiesti, riavvia il cluster DB in modo che le modifiche abbiano effetto. Blue/green le distribuzioni richiedono che l'istanza writer sia sincronizzata con il gruppo di parametri del cluster DB, altrimenti la creazione fallisce. Per ulteriori informazioni, consulta Riavvio di un'istanza database in un cluster Aurora.
- Verifica che il tuo cluster DB stia eseguendo una versione di Aurora PostgreSQL compatibile con Deployments. Blue/Green Per l'elenco delle versioni compatibili, consulta the section called “Implementazioni blu/verde con Aurora PostgreSQL”.
- Assicurati che tutte le tabelle del cluster di database abbiano una chiave primaria. La replica logica di PostgreSQL non consente operazioni UPDATE o DELETE su tabelle che non dispongono di una chiave primaria.

Preparazione di un cluster Aurora Global Database DB per una distribuzione blue/green

Prima di creare una blue/green distribuzione per il cluster Aurora Global Database DB, tieni presente i seguenti punti:

- Tutte le operazioni devono essere avviate dalla stessa regione del cluster di scrittura del Global Database.
- Configurazione del gruppo di parametri:
  - L'ambiente Green utilizza un nuovo gruppo di parametri specificato dall'utente o lo stesso gruppo di parametri del cluster blu (impostazione predefinita).
  - I gruppi di parametri personalizzati vengono copiati nell'ambiente verde.
  - Se un gruppo di parametri specificato non esiste nella regione secondaria, per l'ambiente verde viene utilizzato il gruppo di parametri predefinito nell'area secondaria.

Specificazione delle modifiche durante la creazione di una distribuzione blue/green

È possibile apportare le seguenti modifiche al cluster di DB nell'ambiente verde quando si crea la blue/green distribuzione.

È possibile apportare altre modifiche al cluster di database e alle relative istanze database nell'ambiente verde dopo l'implementazione. Ad esempio, è possibile specificare una versione del motore superiore o un gruppo di parametri diverso.

Per informazioni sulla modifica di un cluster di database, consulta Modifica di un cluster database Amazon Aurora.

Argomenti

- Specifica di una versione successiva del motore
- Specifica di un gruppo di parametri di database

Specifica di una versione successiva del motore

È possibile specificare una versione superiore del motore se si desidera testare un aggiornamento del motore di database. Al momento dello switchover, il database viene aggiornato alla versione principale o secondaria specificata del motore di database.

Specifica di un gruppo di parametri di database

Specifica un gruppo di parametri del·cluster di database diverso da quello utilizzato dal cluster di database. È possibile verificare in che modo le modifiche ai parametri influiscono sul cluster di database nell'ambiente verde o specificare un gruppo di parametri per una nuova versione principale del motore di database in caso di aggiornamento.

Se si specifica un gruppo di parametri del·cluster di database diverso, il gruppo specificato viene associato al cluster di database nell'ambiente verde. Se non si specifica un gruppo di parametri del·cluster di database diverso, il cluster dell'ambiente verde è associato allo stesso gruppo di parametri del·cluster di database dell'ambiente blu.

Creazione di un'implementazione blu/verde

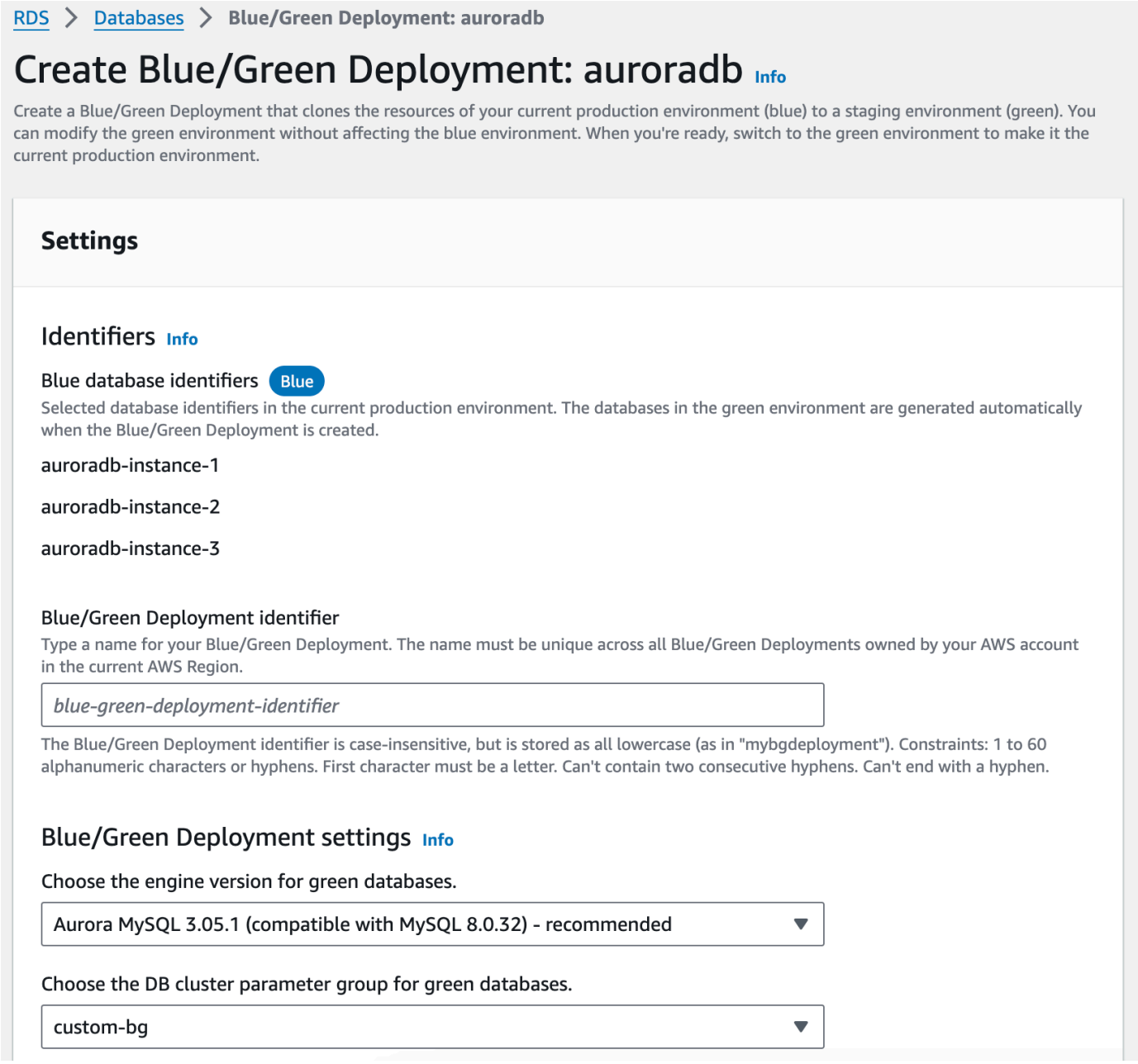
Puoi creare una blue/green distribuzione utilizzando l' Console di gestione AWS, la o l'API AWS CLI RDS.

Console

Per creare una distribuzione blue/green

1. Accedi a Console di gestione AWS e apri la console Amazon RDS all'indirizzo https://console.aws.amazon.com/rds/.
2. Nel riquadro di navigazione, scegli Databases (Database) quindi seleziona il cluster di database da copiare nell'ambiente verde.
3. Scegli Operazioni, Crea implementazione blu/verde.

   Viene visualizzata la pagina Crea blue/green distribuzione.



4. Esamina gli identificatori del database blu. Assicurati che corrispondano alle istanze database che ti aspetti nell'ambiente blu. In caso contrario, scegli Cancel (Annulla).
5. Per il nome della distribuzione blu/verde, inserisci un nome per la blue/green distribuzione.
6. Nelle restanti sezioni, specifica le impostazioni per l'ambiente verde. Per informazioni su ciascuna impostazione, consulta the section called "Impostazioni disponibili".

   È possibile apportare altre modifiche ai database nell'ambiente verde dopo che è stato implementato.
7. Scegli Create (Crea).

AWS CLI

Per creare una blue/green distribuzione utilizzando AWS CLI, usa il create-blue-green-deploymentcomando. Per informazioni su tutte le opzioni disponibili, consulta the section called “Impostazioni disponibili”.

Example

Per Linux, macOS o Unix:

```
aws rds create-blue-green-deployment \
    --blue-green-deployment-name aurora-blue-green-deployment \
    --source arn:aws:rds:us-east-2:123456789012:cluster:auroradb \
    --target-engine-version 8.0 \
    --target-db-cluster-parameter-group-name mydbclusterparametergroup
```

Per Windows:

```
aws rds create-blue-green-deployment ^
    --blue-green-deployment-name aurora-blue-green-deployment ^
    --source arn:aws:rds:us-east-2:123456789012:cluster:auroradb ^
    --target-engine-version 8.0 ^
    --target-db-cluster-parameter-group-name mydbclusterparametergroup
```

API RDS

Per creare una blue/green distribuzione utilizzando l'API Amazon RDS, utilizza l'`CreateBlueGreenDeployment`operazione. Per ulteriori informazioni su ciascuna opzione, consulta the section called “Impostazioni disponibili”.

Impostazioni per la creazione di implementazioni blu/verde

La tabella seguente spiega le impostazioni che puoi scegliere quando crei una blue/green distribuzione. Per ulteriori informazioni sulle AWS CLI opzioni, vedere create-blue-green-deployment. Per ulteriori informazioni sui parametri dell'API RDS, vedere CreateBlueGreenDeployment.

Impostazione della console	Descrizione impostazione	Opzione CLI e parametro API di RDS

Identificatore di implement azione blu/verde	Un nome per la blue/green distribuz ione.	Opzione CLI:
		`--blue-green-deployment-nam e`
		Parametro API:
		`BlueGreenDeploymentName`

Identificatore di database blu	L'identificatore dell'del cluster da copiare nell'ambiente verde. Quando si utilizza la CLI o l'API, si specifica il nome della risorsa Amazon (ARN) del cluster.	Opzione CLI:
		`--source`
		Parametro API:
		`Source`

Gruppo di parametri del cluster di database per i database verdi	Un gruppo di parametri da associare ai database dell'ambiente verde.	Opzione CLI:
		`--target-db-cluster-paramet er-group-name`
		Parametro API:
		`TargetDBClusterParameterGro upName`

Versione del motore per database verdi	Aggiorna il cluster nell'ambiente verde alla versione del motore di database specificata.
	Se scegli un cluster di database Aurora PostgreSQL, esamina e verifica i limiti della replica logica. Per ulteriori informazioni, consulta the section called "Limitazioni
		Opzione CLI:
		`--target-engine-version`
		Parametro API RDS:
		`TargetEngineVersion`

Impostazione della console	Descrizione impostazione	Opzione CLI e parametro API di RDS
	specifiche della replica logica per le distribuzioni blue/green ”.	

Visualizzazione di un’implementazione blu/verde in Amazon Aurora

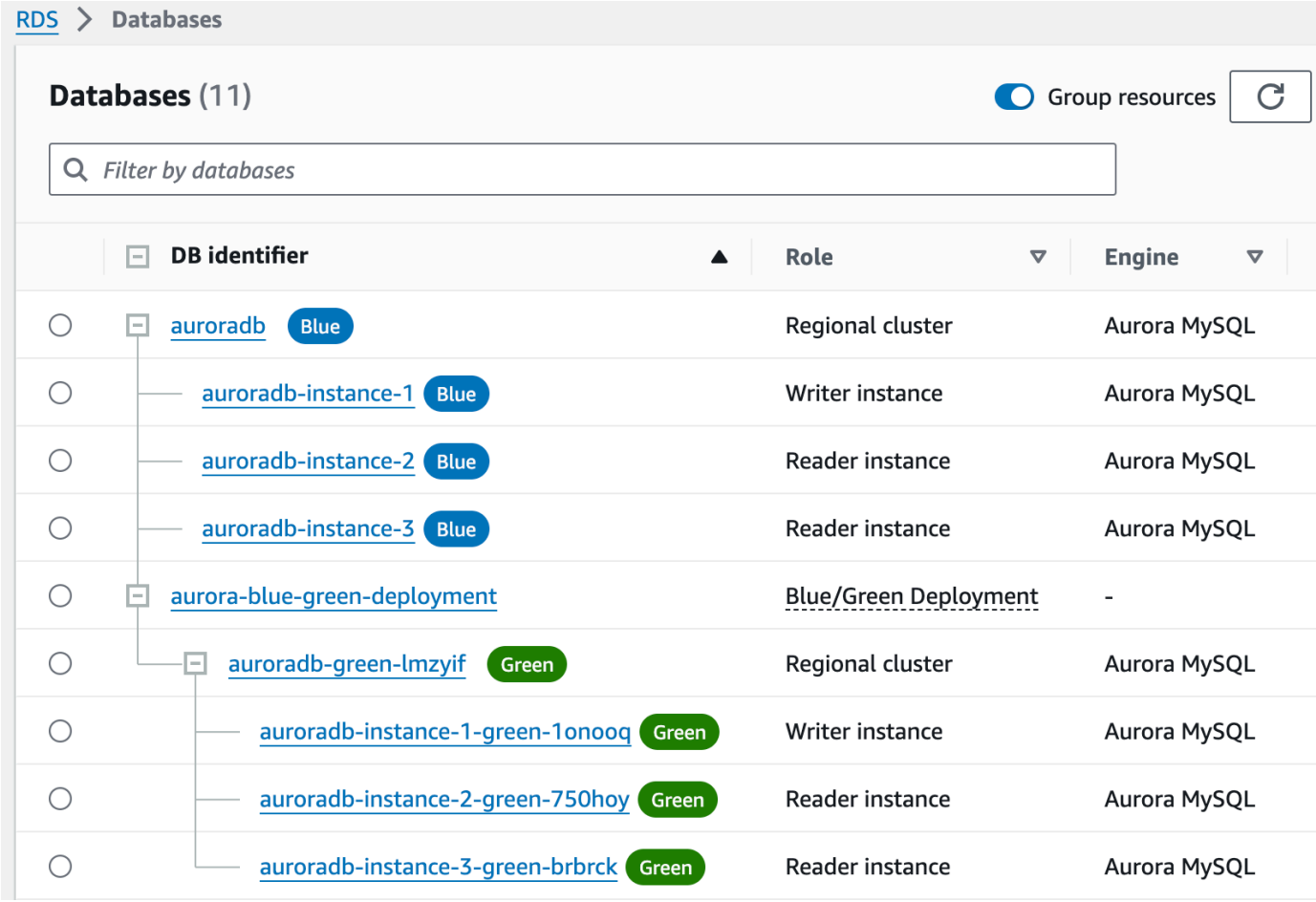
È possibile visualizzare i dettagli di un'implementazione blu/verde utilizzando la Console di gestione AWS, AWS CLI o l'API RDS.

È anche possibile visualizzare e sottoscrivere gli eventi per informazioni su un'implementazione blu/verde. Per ulteriori informazioni, consulta Eventi di implementazione blu/verde.

Console

Per visualizzare i dettagli di un'implementazione blu/verde

1. Accedi alla Console di gestione AWS e apri la console Amazon RDS all'indirizzo https://console.aws.amazon.com/rds/.
2. Nel riquadro di navigazione, scegli Databases (Database), quindi trova l'implementazione blu/verde nell'elenco.

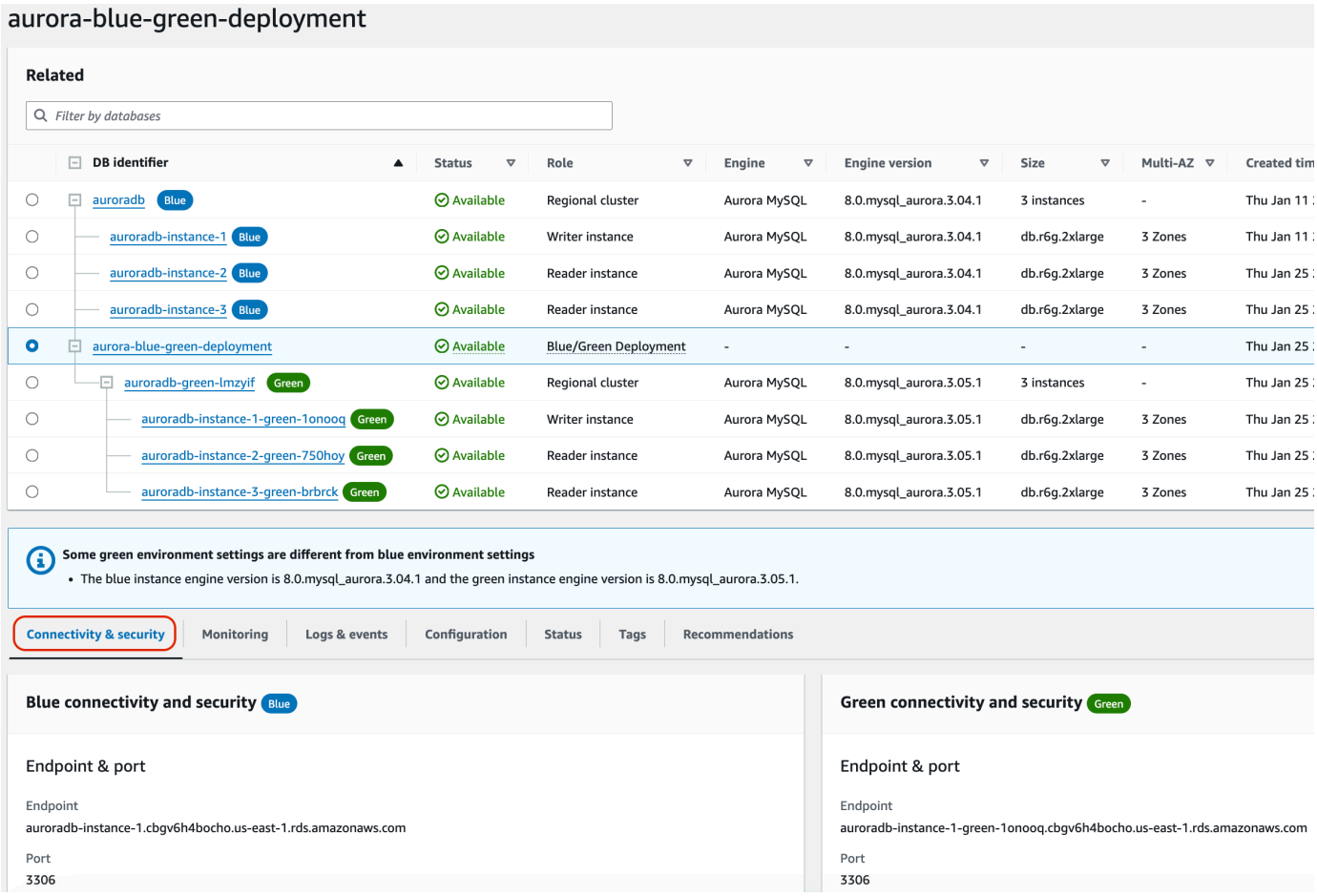


Il valore Role (Ruolo) per l'implementazione blu/verde è Blue/Green Deployment (Implementazione blu/verde).

3. Scegli il nome dell'implementazione blu/verde di cui desideri visualizzarne i dettagli.

   Ogni scheda ha una sezione per l'implementazione blu e una sezione per l'implementazione verde. Ad esempio, nella scheda Configurazione, la versione del motore di database potrebbe essere diversa nell'ambiente blu e nell'ambiente verde, se si aggiorna la versione del motore di database nell'ambiente verde.

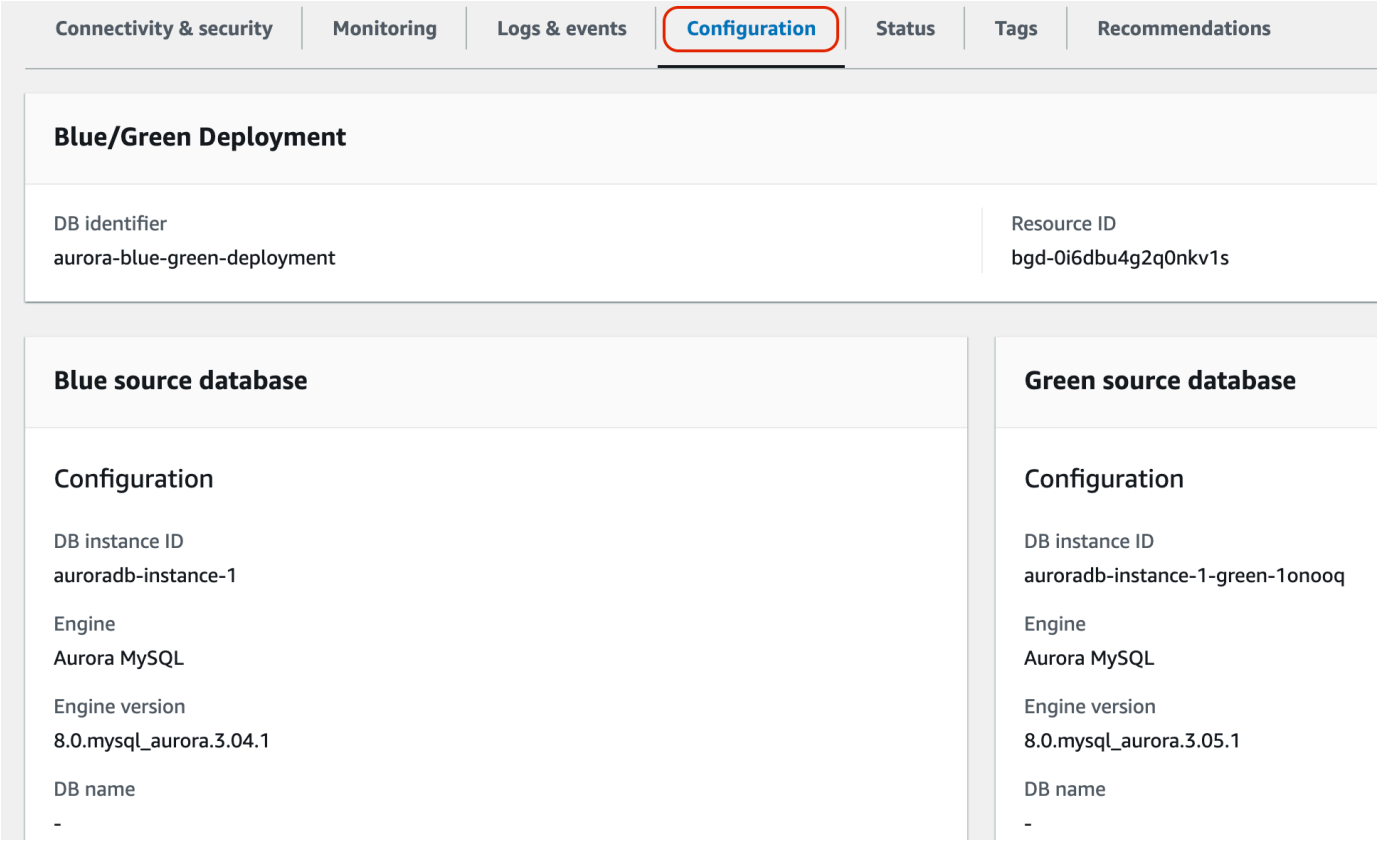
   L'immagine seguente mostra un esempio della scheda Connettività e sicurezza.



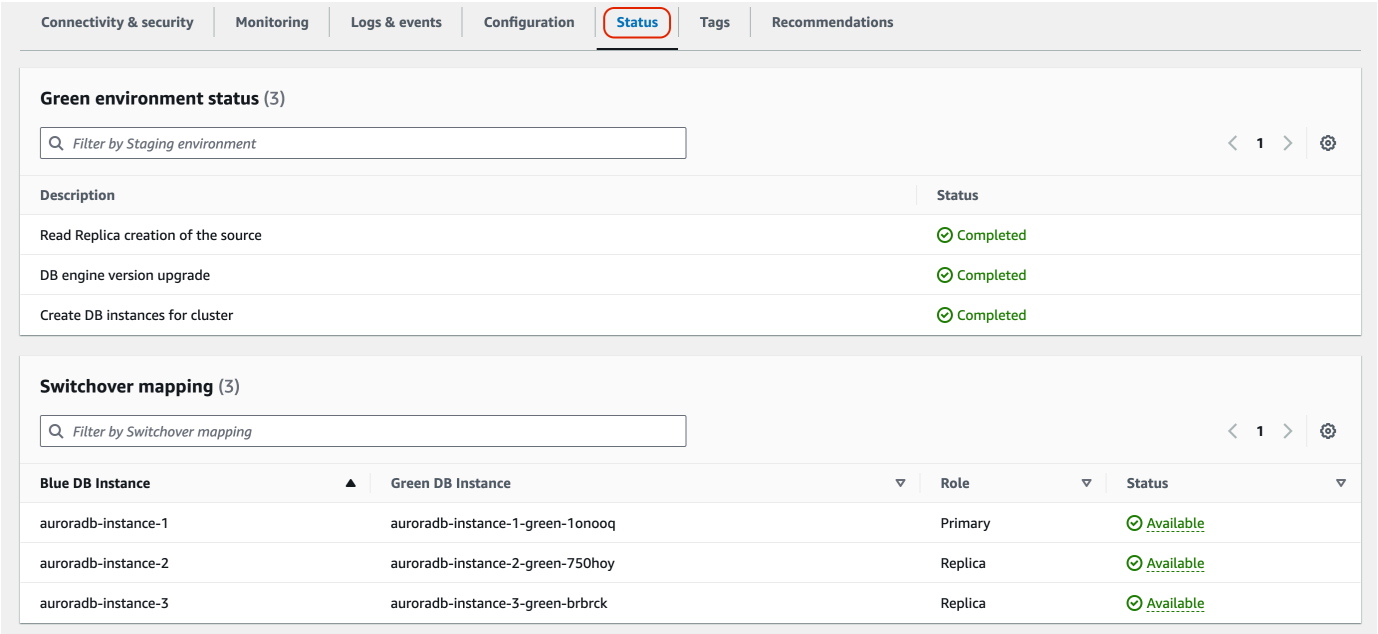
La scheda Connettività e sicurezza include anche una sezione chiamata Replica che mostra lo stato attuale della replica e il ritardo della replica tra gli ambienti blu e verde. Se lo stato di replica è `Replicating`, l'implementazione blu/verde viene replicata correttamente.

Per le implementazioni blu/verde Aurora PostgreSQL, lo stato di replica può cambiare in `Replication degraded` se si apportano modifiche DDL non supportate o a oggetti di grandi dimensioni nell'ambiente blu. Per ulteriori informazioni, consulta the section called "Limitazioni specifiche della replica logica per le distribuzioni blue/green ".

L'immagine seguente mostra un esempio della scheda Configurazione:



L'immagine seguente mostra un esempio della scheda Stato:



AWS CLI

Per visualizzare i dettagli di un'implementazione blu/verde utilizzando AWS CLI, usa il comando describe-blue-green-deployments.

Example Visualizzazione dei dettagli di un'implementazione blu/verde filtrando per nome

Quando si usa il comando describe-blue-green-deployments, è possibile filtrare in base a `--blue-green-deployment-name`.

L'esempio seguente mostra i dettagli per un'implementazione blu/verde denominata *my-blue-green-deployment*.

```
aws rds describe-blue-green-deployments \
  --filters Name=blue-green-deployment-name,Values=my-blue-green-deployment
```

Example Visualizzazione dei dettagli di un'implementazione blu/verde specificando l'identificatore

Quando si utilizza il comando describe-blue-green-deployments, è possibile specificare l'opzione `--blue-green-deployment-identifier`.

L'esempio seguente mostra i dettagli per un'implementazione blu/verde con l'identificatore *bgd-1234567890abcdef*.

```
aws rds describe-blue-green-deployments \
  --blue-green-deployment-identifier bgd-1234567890abcdef
```

API RDS

Per visualizzare i dettagli su un'implementazione blu/verde utilizzando l'API Amazon RDS, utilizza l'operazione `DescribeBlueGreenDeployments` e specifica `BlueGreenDeploymentIdentifier`.

Cambiare una blue/green distribuzione in

Uno switchover passa il cluster di database, comprese le relative istanze database, nell'ambiente verde come cluster di database di produzione. Prima dello switchover, il traffico di produzione viene indirizzato al cluster nell'ambiente blu. Dopo lo switchover, il traffico di produzione viene indirizzato al cluster di database nell'ambiente verde.

Cambiare una blue/green distribuzione non equivale a promuovere il cluster DB all'interno della blue/green distribuzione. Se si promuove manualmente il cluster DB scegliendo Promuovi dal menu Azioni, la replica tra gli ambienti blu e verde si interrompe e la blue/green distribuzione entra in uno stato di configurazione non valida.

Argomenti

- Timeout dello switchover
- Guardrail dello switchover
- Azioni dello switchover
- Best practice per lo switchover
- CloudWatch Verifica delle metriche prima del passaggio al digitale
- Monitoraggio del ritardo della replica prima dello switchover
- Passaggio blue/green da una distribuzione all'altra
- Dopo lo switchover

Timeout dello switchover

È possibile specificare un timeout per lo switchover compreso tra 30 secondi e 3.600 secondi (un'ora). Se lo switchover richiede più tempo della durata specificata, viene eseguito il rollback di tutte le modifiche e non viene apportata alcuna modifica agli ambienti. L'impostazione predefinita del timeout è 300 secondi (cinque minuti).

Guardrail dello switchover

Quando avvii uno switchover, Amazon RDS esegue alcuni controlli di base per verificare la preparazione degli ambienti blu e verdi per lo switchover. Questi controlli sono noti come guardrail dello switchover e impediscono lo switchover se gli ambienti non sono pronti per farlo. Pertanto, evitano tempi di inattività più lunghi del previsto e impediscono la perdita di dati tra gli ambienti blu e quelli verdi che potrebbe verificarsi se lo switchover venisse avviato.

Amazon RDS esegue i seguenti controlli guardrail sull'ambiente verde:

- Integrità della replica: verifica se lo stato della replica del cluster di database verde è integro. Il cluster di database verde è una replica del cluster di database blu.
- Ritardo della replica: verifica se il ritardo della replica del cluster di database verde rientra nei limiti consentiti per lo switchover. I limiti consentiti si basano sul periodo di timeout specificato. Il ritardo

della replica indica il ritardo del cluster di database verde rispetto al cluster di database blu. Per ulteriori informazioni, consulta the section called “Monitoraggio del ritardo della replica prima dello switchover”.

- Scritture attive: assicura che non vi siano scritture attive nel cluster di database verde.

Amazon RDS esegue i seguenti controlli guardrail sull'ambiente blu:

- Replica esterna: per Aurora PostgreSQL, assicura che l’ambiente blu non sia un’origine logica autogestita (publisher) o una replica (subscriber). In caso contrario, è consigliabile eliminare gli slot e gli abbonamenti di replica autogestiti su tutti i database nell’ambiente blu, procedere con lo switchover e quindi ricrearli per riprendere la replica. Per Aurora MySQL, verifica che il database blu non sia una replica di log binari esterna. In caso contrario, è necessario assicurarsi che la replica non sia attiva.
- Scritture attive di lunga durata: assicura che non vi siano scritture attive di lunga durata nel cluster di database blu perché possono aumentare il ritardo della replica.
- Istruzioni DDL di lunga durata: assicura che non vi siano istruzioni DLL di lunga durata nel cluster di database blu perché possono aumentare il ritardo della replica.
- Modifiche PostgreSQL non supportate: per le di oggetti di grandi dimensioni nell'ambiente blu. Per ulteriori informazioni, consulta the section called “Limitazioni specifiche della replica logica per le distribuzioni blue/green ”.

  Se Amazon RDS rileva modifiche PostgreSQL non supportate, modifica lo stato di replica e notifica che lo switchover non è disponibile `Replication degraded` per la distribuzione. blue/green Per procedere con il passaggio al digitale, ti consigliamo di eliminare e ricreare la distribuzione e tutti i database ecologici. blue/green A tale scopo, scegli Operazioni, Elimina con database verdi.

Azioni dello switchover

Quando si passa a una blue/green distribuzione, RDS esegue le seguenti azioni:

1. Esegue controlli guardrail per verificare se gli ambienti blu e verdi sono pronti per lo switchover.
2. Interrompe le nuove operazioni di scrittura nel cluster di database in entrambi gli ambienti.
3. Elimina le connessioni alle istanze database in entrambi gli ambienti e non consente nuove connessioni.
4. Attende che la replica recuperi l'ambiente verde in modo che sia sincronizzato con l'ambiente blu.

5. Rinomina il cluster di database e le istanze database in entrambi gli ambienti.

   RDS rinomina il cluster di database e le istanze database nell'ambiente verde in modo che corrispondano al cluster di database e alle istanze database nell'ambiente blu. Ad esempio, supponi che il nome dell'istanza database nell'ambiente blu sia `mydb`. Supponi anche che il nome dell'istanza database corrispondente nell'ambiente verde sia `mydb-green-abc123`. Durante lo switchover, il nome dell'istanza database nell'ambiente verde viene modificato in `mydb`.

   RDS rinomina il cluster di database e le istanze database nell'ambiente blu aggiungendo `-old`*n* al nome corrente, dove *n* è un numero. Ad esempio, supponi che il nome dell'istanza database nell'ambiente blu sia `mydb`. Dopo lo switchover, il nome dell'istanza database diventa `mydb-old1`.

   RDS rinomina anche gli endpoint nell'ambiente verde in modo che corrispondano agli endpoint nell'ambiente blu, per non apportare modifiche all'applicazione.
6. Consente le connessioni ai database in entrambi gli ambienti.
7. Consente le operazioni di scrittura nel cluster di database nel nuovo ambiente di produzione.

   Dopo lo switchover, il cluster di database di produzione precedente consente operazioni di lettura solo. Anche se abiliti le scritture sul cluster DB, questo rimane di sola lettura finché non elimini la distribuzione. blue/green

Puoi monitorare lo stato di uno switchover utilizzando Amazon. EventBridge Per ulteriori informazioni, consulta the section called “Eventi di implementazione blu/verde”.

Durante il passaggio, i tag dell'ambiente blu sostituiscono tutti i tag sulle risorse nell'ambiente verde. Tutti i tag aggiunti direttamente alle risorse ambientali verdi vengono sovrascritti durante questo processo. Per ulteriori informazioni sui tag, consulta Applicazione di tag alle risorse Amazon Aurora e Amazon RDS.

Se lo switchover inizia e poi si interrompe prima del termine per un qualsiasi motivo, viene eseguito il rollback di tutte le modifiche e non viene apportata alcuna modifica agli ambienti.

Best practice per lo switchover

Prima di effettuare lo switchover, ti consigliamo vivamente di seguire le best practice completando le seguenti attività:

- Esegui accuratamente il test delle risorse nell'ambiente verde. Assicurati che funzionino correttamente ed efficacemente.

- Monitora i CloudWatch parametri Amazon pertinenti. Per ulteriori informazioni, consulta the section called “ CloudWatch Verifica delle metriche prima del passaggio al digitale”.
- Identifica il momento migliore per lo switchover.

  Durante lo switchover, le scritture dei database vengono interrotte in entrambi gli ambienti. Identifica il momento in cui il traffico è più basso nell'ambiente di produzione. Le transazioni a lunga durata, come quelle attive DDLs, possono aumentare i tempi di passaggio al digitale, con conseguenti tempi di inattività più lunghi per i carichi di lavoro di produzione.

  Se è presente un numero elevato di connessioni sulle istanze DB, valuta la possibilità di ridurle manualmente al minimo necessario per l'applicazione prima di passare alla distribuzione. blue/green Un modo per raggiungere questo obiettivo è creare uno script che monitori lo stato della blue/green distribuzione e inizi a ripulire le connessioni quando rileva che lo stato è cambiato in. `SWITCHOVER_IN_PROGRESS`
- Assicurati che il cluster di database e le istanze database siano nello stato `Available` in entrambi gli ambienti.
- Assicurati che il cluster di database nell'ambiente verde sia nello stato integro e in grado di replicare.
-

  Altrimenti, le applicazioni continueranno a inviare traffico di scrittura all'ambiente blu dopo lo switchover.
- Per le distribuzioni Aurora PostgreSQL, le blue/green implementazioni che utilizzano la replica logica, procedi come segue:
  - Esaminare i limiti della replica logica e intraprendere le azioni necessarie prima dello switchover. Per ulteriori informazioni, consulta the section called “Limitazioni specifiche della replica logica per le distribuzioni blue/green ”.
  - Eseguire l'operazione `ANALYZE` per aggiornare la tabella `pg_statistics`. Ciò riduce il rischio di problemi di prestazioni dopo lo switchover.

> **Note**
>
> Durante uno switchover non è possibile modificare il cluster di database incluso nello switchover.

CloudWatch Verifica delle metriche prima del passaggio al digitale

Prima di passare a una blue/green distribuzione, ti consigliamo di verificare i valori delle seguenti metriche all'interno di Amazon CloudWatch.

- `DatabaseConnections`— Utilizza questa metrica per stimare il livello di attività della blue/green distribuzione e assicurati che il valore sia a un livello accettabile per la tua implementazione prima di passare alla versione successiva. Se Approfondimenti sulle prestazioni è attivato, `DBLoad` è un parametro più accurato.
- `ActiveTransactions`: se `innodb_monitor_enable` è impostato su `all` nel gruppo di parametri del database per una qualsiasi istanza database, usa questo parametro per vedere se c'è un numero elevato di transazioni attive che possono bloccare lo switchover.

Per ulteriori informazioni, consulta the section called “CloudWatch metriche per Aurora”.

Monitoraggio del ritardo della replica prima dello switchover

Prima di passare a un' blue/green implementazione, assicurati che il ritardo di replica sia vicino allo zero per ridurre i tempi di inattività.

Aurora MySQL

Per le implementazioni blue/green MariadB, `AuroraBinlogReplicaLag` CloudWatch controlla la metrica nell'ambiente verde per identificare l'attuale ritardo della replica. Per ulteriori informazioni, consulta the section called “Diagnosi e risoluzione del ritardo tra repliche di lettura”.

Aurora PostgreSQL

Per le `OldestReplicationSlotLag` CloudWatch l'attuale ritardo della replica. Per ulteriori informazioni, consulta the section called “Parametri a livello di istanza per Amazon Aurora”.

Inoltre, è possibile eseguire la seguente query SQL nell’ambiente blu:

```
SELECT slot_name,
       confirmed_flush_lsn as flushed,
       pg_current_wal_lsn(),
       (pg_current_wal_lsn() - confirmed_flush_lsn) AS lsn_distance
FROM pg_catalog.pg_replication_slots
```

```
WHERE slot_type = 'logical';

slot_name        |    flushed    | pg_current_wal_lsn | lsn_distance
-----------------+---------------+--------------------+------------
logical_replica1 | 47D97/CF32980 | 47D97/CF3BAC8      | 37192
```

`confirmed_flush_lsn` rappresenta l'ultimo numero di sequenza di log (LSN) inviato alla replica. `pg_current_wal_lsn` rappresenta la posizione attuale del database. Un valore `lsn_distance` pari a 0 significa che la replica è stata recuperata.

Passaggio blue/green da una distribuzione all'altra

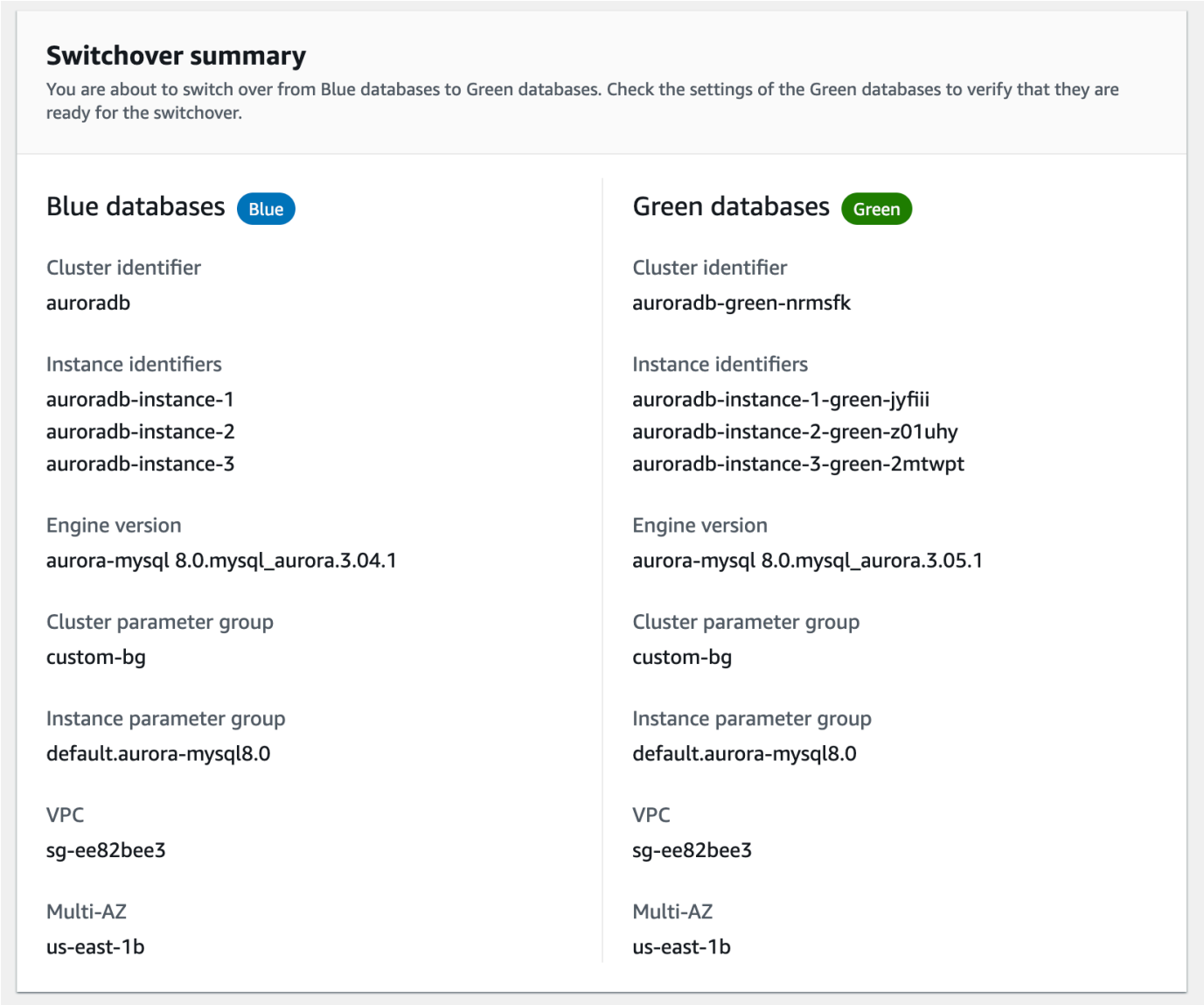
È possibile passare da una blue/green distribuzione all'altra utilizzando Console di gestione AWS l' AWS CLI API RDS o l'API RDS.

Console

Per passare da una blue/green distribuzione all'altra

1. Accedi a Console di gestione AWS e apri la console Amazon RDS all'indirizzo https://console.aws.amazon.com/rds/.
2. Nel riquadro di navigazione, scegli Database, quindi scegli la blue/green distribuzione che desideri trasferire.
3. In Actions (Operazioni), scegli Switch over (Esegui switchover).

   Viene visualizzata la pagina Switch over (Switchover).



4. Nella pagina Switch over (Switchover), consulta il riepilogo dello switchover. Assicurati che le risorse in entrambi gli ambienti corrispondano a quelle previste. In caso contrario, scegli Cancel (Annulla).
5. In Impostazioni di timeout, inserisci il limite di tempo per lo switchover.
6. Se sul cluster è in esecuzione Aurora PostgreSQL, esamina e verifica i suggerimenti prima dello switchover. Per ulteriori informazioni, consulta the section called “Limitazioni specifiche della replica logica per le distribuzioni blue/green ”.
7. Seleziona Switch over (Switchover).

AWS CLI

Per cambiare una blue/green distribuzione utilizzando il AWS CLI, usa il switchover-blue-green-deploymentcomando con le seguenti opzioni:

- `--blue-green-deployment-identifier`— Specificare l'ID della risorsa della blue/green distribuzione.
- `--switchover-timeout`: specifica il limite di tempo per lo switchover, in secondi. Il valore predefinito è 300.

Example Passare da una blue/green distribuzione all'altra

Per Linux, macOS o Unix:

```
aws rds switchover-blue-green-deployment \
    --blue-green-deployment-identifier bgd-1234567890abcdef \
    --switchover-timeout 600
```

Per Windows:

```
aws rds switchover-blue-green-deployment ^
    --blue-green-deployment-identifier bgd-1234567890abcdef ^
    --switchover-timeout 600
```

API RDS

Per passare da una blue/green distribuzione utilizzando l'API Amazon RDS, utilizza l'SwitchoverBlueGreenDeploymentoperazione con i seguenti parametri:

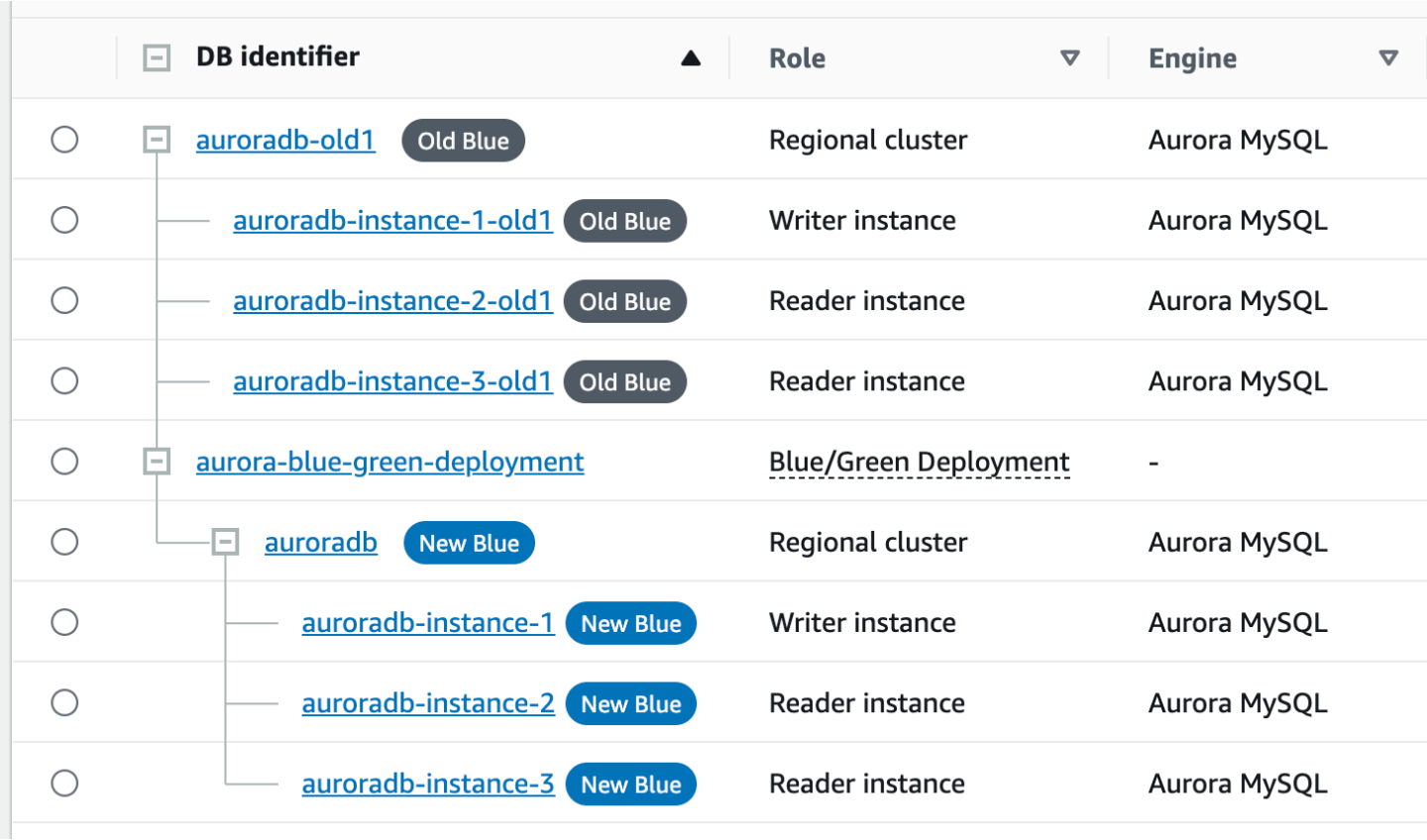
- `BlueGreenDeploymentIdentifier`— Specificare l'ID della risorsa della blue/green distribuzione.
- `SwitchoverTimeout`: specifica il limite di tempo per lo switchover, in secondi. Il valore predefinito è 300.

Dopo lo switchover

Dopo uno switchover, il cluster di database e le istanze database vengono mantenuti nell'ambiente blu precedente. A queste risorse si applicano i costi standard. La replica e il log binario tra gli ambienti blu e verde vengono arrestati.

RDS rinomina il cluster di database e le istanze database nell'ambiente blu aggiungendo `-old`*n* al nome corrente, dove *n* è un numero. Il cluster di database blu viene forzato nello stato di sola lettura. Anche se abiliti le operazioni di scrittura, rimane di sola lettura fino a quando non elimini la blue/green distribuzione. RDS rinomina il cluster di database e le istanze database nell'ambiente verde `-new`*n*.

Se si elimina la risorsa blue/green di distribuzione, RDS conserva le risorse and. `-old`*n* `-new`*n*



Aggiornamento del nodo principale per i consumer

RDS offre repliche di lettura completamente gestite. Tuttavia, fornisce anche la possibilità di configurare repliche autogestite, note anche come repliche esterne. Le repliche esterne consentono di utilizzare risorse di terze parti come destinazioni della replica.

Dopo aver cambiato una distribuzione , se il cluster DB di istanze DB passaggio, è necessario aggiornare il nodo principale dopo il passaggio per mantenere la continuità della replica.

Per aggiornare il nodo principale

1. Dopo lo switchover, l'istanza database di scrittura che si trovava precedentemente nell'ambiente verde emette un evento che contiene il nome del file di log principale e la posizione del log

principale. Per individuare l'evento, accedere alla console RDS e scegliere Eventi nel riquadro di navigazione a sinistra.

2. Filtrare per eventi la cui origine è il nome della vecchia istanza database di scrittura verde, prima dello switchover.
3. Individuare l'evento che contiene le coordinate del log binario. Il messaggio dell'evento è simile a: `Binary log coordinates in green environment after switchover: file mysql-bin-changelog.000003 and position 40134574`.
4. Assicurarsi che il consumer o la replica abbiano applicato tutti i log binari del vecchio ambiente blu. Quindi, utilizzare le coordinate del log binario fornite per riprendere la replica sui consumer. Ad esempio, se stai eseguendo una replica MySQL EC2 su, puoi usare i seguenti comandi:

   MySQL 8.0.22 e versioni principali e secondarie precedenti

   ```
   CHANGE MASTER TO MASTER_HOST='{new-writer-endpoint}', MASTER_LOG_FILE='mysql-bin-changelog.000003', MASTER_LOG_POS=40134574;
   ```

   MySQL 8.0.23 e versioni principali e secondarie successive

   ```
   CHANGE REPLICATION SOURCE TO SOURCE_HOST='{new-writer-endpoint}',
    SOURCE_LOG_FILE='mysql-bin-changelog.000003', SOURCE_LOG_POS=40134574;
   ```

Eliminazione di un'implementazione blu/verde in Amazon Aurora

È possibile eliminare l'implementazione blu/verde prima o dopo lo switchover.

Quando elimini un'implementazione blu/verde prima dello switchover, Amazon RDS elimina facoltativamente il cluster di database nell'ambiente verde:

- Se scegli di eliminare il cluster di database nell'ambiente verde (`--delete-target`), per tale cluster la protezione dall'eliminazione deve essere disattivata.
- Se non elimini il cluster di database nell'ambiente verde (`--no-delete-target`), il cluster viene mantenuto ma non fa più parte di un'implementazione blu/verde. Per Aurora MySQL, la replica continua tra gli ambienti. Per Aurora PostgreSQL, l'ambiente verde viene promosso ad ambiente autonomo, quindi la replica si arresta.

L'opzione per eliminare i database verdi non è disponibile nella console dopo lo switchover. Quando si eliminano le implementazioni blu/verde utilizzando la AWS CLI, non è possibile specificare l'opzione `--delete-target` se lo stato dell'implementazione è `SWITCHOVER_COMPLETED`.

> **Important**
>
> Dopo aver eliminato un'implementazione blu/verde, RDS rimuove le protezioni di sola lettura dal precedente cluster di database di produzione. Se il parametro `read_only` è disabilitato per il cluster di database, consente di nuovo le operazioni di scrittura.

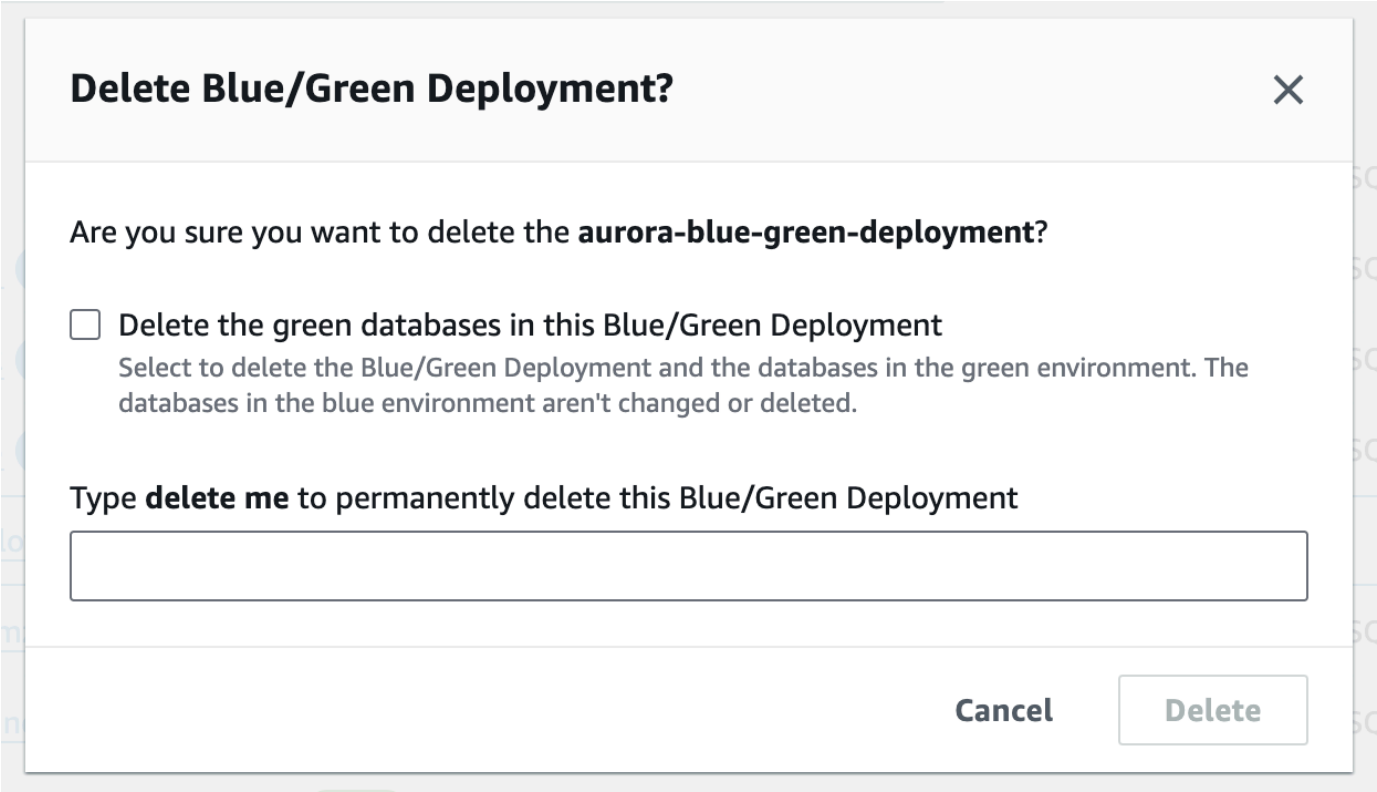
È possibile eliminare un'implementazione blu/verde utilizzando la Console di gestione AWS, AWS CLI o l'API RDS.

Console

Per eliminare un'implementazione blu/verde

1. Accedi alla Console di gestione AWS e apri la console Amazon RDS all'indirizzo https://console.aws.amazon.com/rds/.
2. Nel riquadro di navigazione, scegli Databases (Database) e seleziona l'implementazione blu/verde da eliminare.
3. In Actions (Azioni), scegliere Delete (Elimina).

   Viene visualizzata una finestra Eliminare l'implementazione blu/verde?



Per eliminare i database verdi, seleziona Elimina i database verdi in questa implementazione blu/verde.

4. Immettere **`delete me`** nella casella.
5. Scegliere Delete (Elimina).

AWS CLI

Per eliminare un'implementazione blu/verde utilizzando l'AWS CLI, usa il comando delete-blue-green-deployment con le seguenti opzioni:

- `--blue-green-deployment-identifier`: l'ID risorsa dell'implementazione blu/verde da eliminare.
- `--delete-target`: specifica che il cluster di database nell'ambiente verde viene eliminato. Non è possibile specificare questa opzione se lo stato dell'implementazione blu/verde è `SWITCHOVER_COMPLETED`.
- `--no-delete-target`: specifica che il cluster di database nell'ambiente verde viene mantenuto.

Example Eliminazione di un'implementazione blu/verde e del cluster di database nell'ambiente verde

Per Linux, macOS o Unix:

```
aws rds delete-blue-green-deployment \
    --blue-green-deployment-identifier bgd-1234567890abcdef \
    --delete-target
```

Per Windows:

```
aws rds delete-blue-green-deployment ^
    --blue-green-deployment-identifier bgd-1234567890abcdef ^
    --delete-target
```

Example Eliminazione di un'implementazione blu/verde, mantenendo il cluster di database nell'ambiente verde

Per Linux, macOS o Unix:

```
aws rds delete-blue-green-deployment \
    --blue-green-deployment-identifier bgd-1234567890abcdef \
    --no-delete-target
```

Per Windows:

```
aws rds delete-blue-green-deployment ^
    --blue-green-deployment-identifier bgd-1234567890abcdef ^
    --no-delete-target
```

API RDS

Per eliminare un'implementazione blu/verde con l'API Amazon RDS, utilizza l'operazione `DeleteBlueGreenDeployment` con i seguenti parametri:

- `BlueGreenDeploymentIdentifier`: l'ID risorsa dell'implementazione blu/verde da eliminare.
- `DeleteTarget`: specifica `TRUE` per eliminare il cluster di database nell'ambiente verde o `FALSE` per mantenerlo. Non può essere `TRUE` se lo stato dell'implementazione blu/verde è `SWITCHOVER_COMPLETED`.

Backup e ripristino di un cluster DB Amazon Aurora

Questi argomenti forniscono informazioni sul backup e sul ripristino dei cluster database Amazon Aurora.

> **Tip**
>
> Le funzioni di Aurora a elevata disponibilità e le funzionalità di backup automatico consentono di proteggere i dati senza richiedere una configurazione completa. Prima di implementare una strategia di backup, scopri come Aurora gestisce più copie dei dati e ti aiuta ad accedervi tra più istanze database e regioni AWS. Per informazioni dettagliate, consulta Elevata disponibilità di Amazon Aurora.

Argomenti

- Panoramica di backup e ripristino di un cluster di database Aurora
- Mantenimento dei backup automatici
- Informazioni sull'utilizzo dello storage di backup Amazon Aurora
- Creazione di uno snapshot del cluster database
- Ripristino da uno snapshot cluster database
- Copia di uno snapshot del cluster di database
- Condivisione di uno snapshot cluster database
- Esportazione dei dati del cluster database in Amazon S3
- Esportazione dei dati dello snapshot del cluster di database in Amazon S3
- Ripristino di un cluster di database a un determinato momento
- Eliminazione di una snapshot del cluster di database
- Tutorial: Ripristino di un cluster di database Amazon Aurora da uno snapshot del cluster di database

Panoramica di backup e ripristino di un cluster di database Aurora

Negli argomenti seguenti vengono descritti i backup Aurora e come ripristinare il cluster database Aurora.

Indice

- Backup
  - Usando AWS Backup
- Backup window (Finestra di backup)
- Ripristino dei dati
- Clonazione dei database per Aurora
- Backtrack

Backup

Aurora esegue automaticamente il backup del volume del cluster e conserva i dati per la durata del tempo di conservazione del backup. I backup automatizzati Aurora sono continui e incrementali, in modo da poter eseguire rapidamente un ripristino da qualsiasi point-in-time incluso nel periodo di conservazione del backup. Durante la scrittura dei dati di backup, non si verifica alcun impatto sulle prestazioni o interruzione del funzionamento del servizio del database. È possibile specificare un periodo di conservazione del backup (da 1 a 35 giorni) quando un cluster database viene creato o modificato. I backup automatizzati Aurora sono archiviati in Amazon S3. Per ulteriori informazioni sulla conservazione dei backup automatici, consulta Mantenimento dei backup automatici.

Per mantenere i dati oltre il tempo di conservazione del backup, è possibile eseguire uno snapshot dei dati del volume del cluster. Gli snapshot dei cluster Aurora non scadono. Puoi creare un nuovo cluster database dallo snapshot. Per ulteriori informazioni, consulta Creazione di uno snapshot del cluster database.

> **Note**
>
> - Per i cluster di database Amazon Aurora, il periodo di retention dei backup predefinito è di un giorno indipendentemente dal modo in cui viene creato il cluster DB.
> - Non puoi disabilitare i backup automatici su Aurora. Il periodo di retention dei backup per Aurora viene gestito dal cluster di database.

I costi dello storage di backup dipendono dalla quantità di dati di backup e snapshot Aurora mantenuti e dalla durata di mantenimento. Per informazioni sullo storage associato ai backup e agli snapshot Aurora, consulta Informazioni sull'utilizzo dello storage di backup Amazon Aurora. Per informazioni sui prezzi dello storage di backup di Aurora, consulta Prezzi di Amazon RDS for Aurora. Dopo che un cluster Aurora associato a uno snapshot viene eliminato, per l'archiviazione di tale snapshot verranno applicati i costi dello storage standard di Aurora.

Usando AWS Backup

Puoi utilizzarlo AWS Backup per gestire i backup dei cluster Amazon Aurora DB.

Le istantanee gestite da AWS Backup sono considerate istantanee manuali del cluster DB, ma non vengono conteggiate ai fini della quota di snapshot del cluster DB per Aurora. Le istantanee create con hanno nomi con. AWS Backup `awsbackup:job-`*`AWS-Backup-job-number`* Per ulteriori informazioni AWS Backup, consulta la AWS Backup Developer Guide.

Puoi anche utilizzarlo AWS Backup per gestire i backup automatici dei cluster Amazon Aurora DB. Se il tuo cluster DB è associato a un piano di backup in AWS Backup, puoi utilizzare quel piano di backup per il ripristino. point-in-time I nomi dei backup automatici (continui) gestiti da AWS Backup includono `continuous:cluster-`*`AWS-Backup-job-number`*. Per ulteriori informazioni, consulta Ripristino di un cluster database a un'ora specificata mediante AWS Backup.

Backup window (Finestra di backup)

I backup automatici vengono effettuati quotidianamente durante la finestra di backup scelta. Se il backup richiede più tempo rispetto alla finestra di backup prevista, il backup continua dopo il termine della finestra, finché non viene completato. La finestra di backup non può sovrapporsi con la finestra di manutenzione settimanale per il cluster database.

I backup automatici di Aurora sono continui e incrementali, tuttavia si utilizza una finestra di backup per creare un backup giornaliero del sistema che viene conservato per tutto il periodo di conservazione del backup. È possibile copiare il backup per conservarlo oltre il periodo di conservazione.

> **Note**
>
> Quando si crea un cluster DB utilizzando Console di gestione AWS, non è possibile specificare una finestra di backup. Tuttavia, è possibile specificare una finestra di backup quando si crea un cluster database utilizzando la AWS CLI o l'API RDS.

Se non si specifica una finestra di backup preferita al momento della creazione del cluster, Aurora assegna una finestra di backup predefinita di 30 minuti. Questa finestra viene selezionata a caso da un blocco di tempo di 8 ore per ciascuna Regione AWS. La tabella seguente elenca i blocchi temporali per ciascuno Regione AWS a cui sono assegnate le finestre di backup predefinite.

Nome della regione	Regione	Periodo di tempo
US East (N. Virginia)	us-east-1	03:00 - 11:00 UTC
Stati Uniti orientali (Ohio)	us-east-2	03:00 - 11:00 UTC
US West (N. Californi a)	us-west-1	06:00 - 14:00 UTC
US West (Oregon)	us-west-2	06:00 - 14:00 UTC
Africa (Cape Town)	af-south-1	03:00 - 11:00 UTC
Asia Pacific (Hong Kong)	ap-east-1	06:00 - 14:00 UTC
Asia Pacific (Hyderabad)	ap-south-2	06:30 - 14:30 UTC
Asia Pacifico (Giacarta)	ap-southeast-3	08:00 - 16:00 UTC
Asia Pacifico (Malesia)	ap-southeast-5	09:00 - 17:00 UTC
Asia Pacifico (Melbourne)	ap-southeast-4	11:00 - 19:00 UTC
Asia Pacific (Mumbai)	ap-south-1	16:30 - 00:30 UTC
Asia Pacifico (Nuova Zelanda)	ap-southeast-6	13:00 - 21:00 UTC

Nome della regione	Regione	Periodo di tempo
Asia Pacifico (Osaka-Locale)	ap-northeast-3	00:00 - 08:00 UTC
Asia Pacific (Seoul)	ap-northeast-2	13:00 - 21:00 UTC
Asia Pacific (Singapore)	ap-southeast-1	14:00 - 22:00 UTC
Asia Pacific (Sydney)	ap-southeast-2	12:00 - 20:00 UTC
Asia Pacifico (Taipei)	ap-east-2	9:00 - 17:00 UTC
Asia Pacifico (Thailandia)	ap-southeast-7	8:00 - 16:00 UTC
Asia Pacifico (Tokyo)	ap-northeast-1	13:00 - 21:00 UTC
Canada (Central)	ca-central-1	03:00 - 11:00 UTC
Canada occidentale (Calgary)	ca-west-1	18:00 - 02:00 UTC
China (Beijing)	cn-north-1	06:00 - 14:00 UTC
China (Ningxia)	cn-northwest-1	06:00 - 14:00 UTC
Europe (Frankfurt)	eu-central-1	20:00 - 04:00 UTC
Europe (Ireland)	eu-west-1	22:00 - 06:00 UTC
Europe (London)	eu-west-2	22:00 - 06:00 UTC
Europe (Milan)	eu-south-1	02:00 - 10:00 UTC
Europe (Paris)	eu-west-3	07:29 - 14:29 UTC
Europa (Spagna)	eu-south-2	02:00 - 10:00 UTC
Europe (Stockholm)	eu-north-1	23:00 - 07:00 UTC

Nome della regione	Regione	Periodo di tempo
Europa (Zurigo)	eu-central-2	02:00 - 10:00 UTC
Israele (Tel Aviv)	il-central-1	03:00 - 11:00 UTC
Messico (Centrale)	mx-central-1	19:00 - 03:00 UTC
Medio Oriente (Bahrein)	me-south-1	06:00 - 14:00 UTC
Medio Oriente (Emirati Arabi Uniti)	me-central-1	05:00–13:00 UTC
Sud America (São Paulo)	sa-east-1	23:00 - 07:00 UTC
AWS GovCloud (Stati Uniti orientali)	us-gov-east-1	17:00 - 01:00 UTC
AWS GovCloud (Stati Uniti occidentali)	us-gov-west-1	06:00 - 14:00 UTC

Ripristino dei dati

È possibile recuperare i dati creando un nuovo cluster database Aurora dai dati di backup mantenuti da Aurora, da uno snapshot del cluster database salvato o da un backup automatico mantenuto. Puoi ripristinare rapidamente una nuova copia di un cluster DB creato dai dati di backup a un momento qualsiasi del periodo di retention dei backup. Data la natura continua e incrementale dei backup Aurora durante il tempo di conservazione del backup, non sarà necessario effettuare snapshot frequenti dei dati per migliorare i tempi di ripristino.

L'ultimo orario di ripristino di un cluster database corrisponde al punto più recente in corrispondenza del quale è possibile ripristinare il cluster database. In genere, ciò avviene entro 5 minuti dall'ora corrente per un cluster database attivo o 5 minuti dall'orario di eliminazione del cluster per un backup automatico mantenuto.

Il primo orario di ripristino indica il punto all'interno del periodo di conservazione dei backup dai cui è possibile ripristinare il volume del cluster.

Per determinare l'ora più recente o meno recente disponibile per il ripristino di un cluster database, individuare i valori `Latest restorable time` o `Earliest restorable time` nella console RDS. Per informazioni sulla visualizzazione di tali valori consulta Visualizzazione dei backup automatici mantenuti per Amazon Aurora.

Puoi stabilire il completamento del ripristino di un cluster DB verificando i valori `Latest restorable time` ed `Earliest restorable time`. Questi valori restituiranno NULL fino al termine dell'operazione di ripristino. Non è possibile richiedere l'esecuzione di un'operazione di backup o ripristino se `Latest restorable time` o `Earliest restorable time` restituisce NULL.

Per informazioni sul ripristino di un cluster di database a un punto temporale specifico, consulta Ripristino di un cluster di database a un determinato momento.

Clonazione dei database per Aurora

Anziché ripristinare la snapshot di un cluster DB, puoi anche clonare i database del cluster DB Aurora in un nuovo cluster DB. I database dei cloni utilizzano un ridottissimo spazio aggiuntivo al momento della creazione. I dati vengono copiati solo come modifiche nel database di origine o in quelli clone. Puoi effettuare più cloni dello stesso cluster DB oppure creare cloni aggiuntivi da altri cloni. Per ulteriori informazioni, consulta Clonazione di un volume per un cluster di database Amazon Aurora.

Backtrack

Aurora MySQL ora supporta il "riavvolgimento" di un cluster database a un'ora specifica, senza ripristinare i dati di un backup. Per ulteriori informazioni, consulta Backtrack di un cluster database Aurora.

Mantenimento dei backup automatici

Quando si elimina un cluster di database Aurora Serverless v2 o con provisioning, è possibile mantenere i backup automatici. Ciò consente di ripristinare un cluster database in un punto nel tempo specifico all'interno del periodo di conservazione del backup, anche dopo l'eliminazione del cluster.

I backup automatici mantenuti contengono snapshot di sistema e log delle transazioni di un cluster database. Includono anche le proprietà del cluster database, come la classe di istanza database, necessarie per ripristinarlo in un cluster attivo.

Puoi ripristinare o rimuovere i backup automatici mantenuti tramite la Console di gestione AWS, l'API RDS e AWS CLI.

> **Note**
>
> Non è possibile mantenere i backup automatici per i cluster di database Aurora Serverless v1.

Argomenti

- Periodo di conservazione
- Costi di retention
- Come impedire l’eliminazione dei backup automatici
- Limitazioni
- Visualizzazione dei backup automatici mantenuti per Amazon Aurora
- Eliminazione dei backup automatici mantenuti per Amazon Aurora

Periodo di conservazione

Gli snapshot di sistema e i log delle transazioni di un backup automatico mantenuto scadono allo stesso modo del cluster database di origine. Le impostazioni per il periodo di conservazione del cluster di origine si applicano anche ai backup automatici. Poiché non vengono creati nuovi snapshot o log per questo cluster, i backup automatici mantenuti scadranno. Al termine del periodo di conservazione, si continua a mantenere gli snapshot del cluster database manuali, ma tutti i backup automatici scadranno.

È possibile rimuovere backup automatici mantenuti mediante la console, AWS CLI o l'API RDS. Per ulteriori informazioni, consulta Eliminazione dei backup automatici mantenuti per Amazon Aurora.

A differenza di un backup automatico mantenuto, uno snapshot finale non scade. Ti suggeriamo di acquisire uno snapshot finale anche se mantieni i backup automatici, perché i backup automatici mantenuti prima o poi scadono.

Costi di retention

Per lo storage di backup non vengono addebitati costi aggiuntivi fino al 100% dello storage di database Aurora totale per ogni cluster database Aurora. Inoltre, non sono previsti costi aggiuntivi fino a un giorno se si mantengono i backup automatici dopo l'eliminazione di un cluster database. I backup mantenuti per più di un giorno vengono addebitati.

Non sono previsti costi aggiuntivi per i log delle transazioni o i metadati dell'istanza. Tutte le altre regole di prezzo per i backup si applicano ai cluster ripristinabili. Per ulteriori informazioni, consulta la pagina Prezzi di Amazon Aurora.

Come impedire l'eliminazione dei backup automatici

Amazon RDS elimina i backup automatici in diverse situazioni:

- Al termine del periodo di conservazione.
- Quando si elimina un cluster di database.

Se si desidera mantenere uno snapshot automatico per un periodo più lungo, è possbile copiarlo e creare uno snapshot manuale che sarà conservato finché non lo si elimina personalmente. I costi di archiviazione di Amazon RDS potrebbero applicarsi agli snapshot manuali se superano lo spazio di archiviazione predefinito.

Per ulteriori informazioni su come copiare uno snapshot di un cluster di database, consulta Copia di uno snapshot del cluster di database.

Per ulteriori informazioni sui costi di storage dei backup, consulta Prezzi di Amazon RDS.

Limitazioni

Le seguenti limitazioni si applicano ai backup automatici mantenuti:

- Il numero massimo di backup automatici mantenuti in una Regione AWS è pari a 40. Non è incluso nella quota per i cluster database. È possibile avere fino a 40 cluster database in esecuzione, 40 istanze database in esecuzione e 40 backup automatici mantenuti per i cluster database contemporaneamente.

Per ulteriori informazioni, consulta Quote in Amazon Aurora.

- I backup automatici mantenuti non contengono informazioni relative ai parametri o ai gruppi di opzioni.
- È possibile ripristinare un cluster eliminato in un punto temporale che si trova all'interno del periodo di conservazione al momento dell'eliminazione.
- Non è possibile modificare un backup automatico mantenuto perché è costituito da backup di sistema, log delle transazioni e proprietà del cluster database presenti al momento dell'eliminazione del cluster di origine.
- La replica dei backup automatici tra Regioni non è supportata per i cluster di database Aurora. Aurora non supporta la replica automatica di snapshot e log delle transazioni in un'altra Regione AWS. Per il disaster recovery tra Regioni, è necessario copiare manualmente gli snapshot di Aurora nella Regione di destinazione desiderata.

Visualizzazione dei backup automatici mantenuti per Amazon Aurora

Per visualizzare i backup automatici mantenuti nella console RDS, scegli Backup automatici nel pannello di navigazione, quindi seleziona Conservato. Per visualizzare singoli snapshot associati a un backup automatico mantenuto, scegli Snapshot nel pannello di navigazione. In alternativa, puoi descrivere singoli snapshot associati a un backup automatico mantenuto. Da qui, puoi ripristinare un'istanza database direttamente da uno di tali snapshot.

Per descrivere i backup automatizzati conservati utilizzando il AWS CLI, utilizza uno dei seguenti comandi:

```
aws rds describe-db-cluster-automated-backups --db-cluster-resource-
id DB_cluster_resource_ID
```

Per descrivere i backup automatici mantenuti tramite l'API RDS, chiama l'operazione `DescribeDBClusterAutomatedBackups` con il parametro `DbClusterResourceId`.

Eliminazione dei backup automatici mantenuti per Amazon Aurora

Puoi eliminare i backup automatici mantenuti quando non servono più. Per eliminare un backup automatico mantenuto utilizzando la Console di gestione AWS, l'interfaccia AWS CLI o l'API Amazon RDS, utilizza le seguenti procedure.

Console

Per eliminare i backup automatici mantenuti

1. Accedi alla Console di gestione AWS e apri la console Amazon RDS all'indirizzo https://console.aws.amazon.com/rds/.
2. Nel riquadro di navigazione, selezionare Automated backups (Backup automatici).
3. Scegli la scheda Conservato.

4. Scegliere il backup automatico mantenuto da eliminare.
5. In Actions (Azioni), selezionare Delete (Elimina).
6. Nella pagina di conferma, immetti **`delete me`** e seleziona Elimina.

AWS CLI

È possibile eliminare un backup automatico mantenuto mediante il comando AWS CLI delete-db-cluster-automated-backup con la seguente opzione:

- `--db-cluster-resource-id`: l'identificatore della risorsa per il cluster database di origine.

  È possibile trovare identificatore della risorsa per il cluster database di origine di un backup automatico mantenuto eseguendo il comando AWS CLI describe-db-cluster-automated-backups.

Example

Questo esempio elimina il backup automatico mantenuto per il cluster database di origine con l'ID risorsa `cluster-123ABCEXAMPLE`.

Per Linux, macOS o Unix:

```
aws rds delete-db-cluster-automated-backup \
    --db-cluster-resource-id cluster-123ABCEXAMPLE
```

Per Windows:

```
aws rds delete-db-cluster-automated-backup ^
    --db-cluster-resource-id cluster-123ABCEXAMPLE
```

API RDS

È possibile eliminare un backup automatico mantenuto mediante l'operazione API Amazon RDS DeleteDBClusterAutomatedBackup con il seguente parametro:

- `DbClusterResourceId`: l'identificatore della risorsa per il cluster database di origine.

  È possibile individuare l'identificatore della risorsa per l'istanza database di origine di un backup automatico mantenuto mediante l'operazione API Amazon RDS DescribeDBClusterAutomatedBackups.

Informazioni sull'utilizzo dello storage di backup Amazon Aurora

Amazon Aurora gestisce due tipi di backup: backup automatici (continui) e istantanee.

Storage di backup automatici

Il backup automatico (continuo) per un cluster archivia in modo incrementale tutte le modifiche apportate al database in un periodo di conservazione specificato in modo da poter eseguire il ripristino in qualsiasi momento entro il periodo di conservazione definito. I periodi di conservazione possono variare da 1 a 35 giorni. I backup automatici sono incrementali e il relativo costo viene addebitato in base alla quantità di storage necessaria per il ripristino in qualsiasi momento entro il periodo di conservazione.

Aurora offre anche una quantità di utilizzo gratuito del backup. Questa quantità di utilizzo gratuito è pari alla dimensione del volume del cluster più recente (rappresentata dalla metrica Amazon CloudWatch `VolumeBytesUsed`). Questa quantità viene sottratta dall'utilizzo del backup automatico calcolato. Inoltre, non è previsto alcun costo per backup automatici il cui periodo di conservazione è solo di 1 giorno.

Ad esempio, il backup automatico ha un periodo di conservazione di 7 giorni e quindi puoi ripristinare il cluster allo stato di quattro giorni fa. Aurora utilizza i dati incrementali archiviati nel backup automatico per ricreare lo stato del cluster all'ora esatta di quattro giorni fa.

Il backup automatico archivia tutte le informazioni necessarie per poter ripristinare il cluster in qualsiasi momento compreso nella finestra di conservazione. Ciò significa che archivia tutte le modifiche durante il periodo di conservazione, incluse le operazioni di scrittura di nuove informazioni o di eliminazione delle informazioni esistenti. Per i database a cui vengono apportate molte modifiche, la dimensione del backup automatico aumenta nel tempo. Non appena un database non viene più modificato, puoi aspettarti che la dimensione del backup automatico diminuisca, man mano che le modifiche precedentemente archiviate non rientrano più nella finestra di conservazione.

L'utilizzo totale fatturato per il backup automatico non supera mai la dimensione cumulativa del volume del cluster durante il periodo di conservazione. Ad esempio, se il periodo di conservazione è di 7 giorni e il volume del cluster è di 100 GB al giorno, l'utilizzo fatturato del backup automatico non supera mai 700 GB (100 GB*7).

Archiviazione delle istantanee

Gli snapshot dei cluster di database sono sempre backup completi che catturano le dimensioni del volume dei cluster al momento della creazione. Sia che tu acquisisca snapshot manualmente o tramite un piano di AWS Backup, Aurora le li considera snapshot manuali. Aurora offre spazio di archiviazione gratuito illimitato per gli snapshot che rientrano nel periodo di conservazione del backup automatico. Una volta trascorso tale periodo, uno snapshot manuale prevede l'addebito di costi per GB al mese. Gli snapshot di sistema automatici rimangono gratuiti a meno che non vengano copiati. Poiché i backup automatici non coprono le copie degli snapshot, AWS addebita sempre costi per il relativo utilizzo.

Per informazioni generali sui backup Aurora, consulta Backup. Per informazioni sui prezzi dello spazio di archiviazione per i backup Aurora, consulta la pagina Prezzi di Amazon Aurora.

Metriche di Amazon CloudWatch per lo storage di backup Aurora

Puoi monitorare i cluster Aurora e creare rapporti utilizzando metriche di Amazon CloudWatch tramite la console CloudWatch. Puoi utilizzare le seguenti metriche di CloudWatch per rivedere e monitorare la quantità di archiviazione utilizzata dai backup Aurora. Questi parametri vengono calcolati in modo indipendente per ogni cluster DB Aurora.

- `BackupRetentionPeriodStorageUsed`: rappresenta la quantità di spazio di archiviazione di backup, in byte, utilizzato al momento per l'archiviazione dei backup automatizzati.
  - Il valore dipende dalla dimensione del volume del cluster e dal numero di modifiche (scritture e aggiornamenti) apportate al cluster database durante il periodo di conservazione. Questo è dovuto al fatto che il backup automatico deve archiviare tutte le modifiche incrementali apportate al cluster per poterle ripristinare in qualsiasi momento.
  - Il valore di questa metrica non viene sottratto dal livello gratuito di utilizzo del backup fornito da Aurora.
  - Questa metrica restituisce un unico punto dati giornaliero per l'utilizzo del backup automatico registrato in un giorno specifico.
- `SnapshotStorageUsed`: rappresenta la quantità di archiviazione dei backup utilizzata, in byte, per l'archiviazione di istantanee manuali oltre il periodo di conservazione dei backup.
  - Il valore dipende dal numero di istantanee conservate oltre il periodo di conservazione del backup automatico e dalle dimensioni di ciascuna istantanea.
  - Ogni snapshot corrisponde, per dimensioni, al volume del cluster al momento della sua acquisizione.

- Gli snapshot sono backup completi, non incrementali.
- Questa metrica restituisce un punto dati giornaliero per ogni istantanea addebitata. Per recuperare l'utilizzo totale giornaliero delle istantanee, calcola la somma di questa metrica in un periodo di 1 giorno.

- `TotalBackupStorageBilled`: rappresenta le metriche per tutti gli utilizzi di backup fatturati, in byte, per il cluster specificato:

  ```
  BackupRetentionPeriodStorageUsed + SnapshotStorageUsed - free tier
  ```

  - Questa metrica restituisce un punto dati giornaliero per il valore `BackupRetentionPeriodStorageUsed` meno il livello gratuito di utilizzo del backup fornito da Aurora. Questo livello gratuito corrisponde all'ultima dimensione registrata del volume del cluster database. Questo punto dati rappresenta l'utilizzo effettivo fatturato per il backup automatico.
  - Questa metrica restituisce singoli punti dati giornalieri per tutti i valori `SnapshotStorageUsed`.
  - Per recuperare l'utilizzo totale giornaliero fatturato del backup, calcola la somma di questa metrica in un periodo di 1 giorno. L'utilizzo fatturato delle istantanee viene sommato all'utilizzo del backup automatico fatturato per ottenere l'utilizzo totale del backup fatturato.

Per ulteriori informazioni su come utilizzare i parametri di CloudWatch, consulta Disponibilità dei parametri Aurora nella console Amazon RDS.

Calcolo dell'utilizzo dello storage dei backup

L'utilizzo di un backup automatico viene calcolato facendo riferimento a tutti i record incrementali che devono essere archiviati, per poterli ripristinare in qualsiasi momento durante il periodo di conservazione del backup. Queste modifiche includono non solo il numero di operazioni di scrittura, ma anche le dimensioni e la portata delle modifiche ai dati. Ogni tipo di operazione (INSERT, UPDATE, DELETE) crea record di modifiche che devono essere conservati per il recupero point-in-time. Pertanto, sebbene due database possano avere lo stesso numero di operazioni di scrittura (IOPS), i relativi requisiti di spazio archiviazione per i backup potrebbero differire in modo significativo in base al volume di dati modificati in ciascuna transazione.

Si supponga, ad esempio, di avere un backup automatico con periodo di conservazione di 7 giorni. La dimensione del volume del cluster prima del periodo di conservazione è pari a 100 GB, quindi questo valore è la quantità minima che Aurora deve archiviare. Nei successivi 7 giorni viene eseguita la seguente attività, in cui la dimensione incrementale dei record corrisponde alla quantità

di storage necessaria per archiviare i record delle modifiche provenienti dalle operazioni di scrittura e aggiornamento del database.

Day (Giorno)	Dimensioni incrementali dei record (GB)
1	10
2	15
3	25
4	20
5	10
6	25
7	30
Totale	135

Questi dati indicano che l'utilizzo del backup automatico calcolato per il backup è il seguente:

```
100 GB (volume size before retention period) + 135 GB (size of incremental records) =
 235 GB total backup usage
```

Dall'utilizzo fatturato viene quindi sottratto il livello di utilizzo gratuito. Supponiamo che la dimensione più recente del volume sia 200 GB:

```
235 GB total backup usage - 200 GB (latest volume size) = 35 GB billed backup usage
```

Domande frequenti

Quando vengono fatturate le istantanee?

Vengono fatturate le istantanee manuali che non rientrano nel periodo di conservazione del backup automatico (ovvero precedenti al periodo indicato).

Cos'è un'istantanea manuale?

Un'istantanea manuale è un'istantanea per la quale è valida una delle seguenti condizioni:

- Viene richiesta manualmente dall'utente
- Viene acquisita da un servizio di backup automatico, ad esempio Backup AWS
- Viene copiata da un'istantanea di sistema automatica per essere conservata oltre il periodo di conservazione

Cosa succede alle istantanee manuali se elimino il cluster database?

Le istantanee manuali non scadono finché non scegli di eliminarle.

Quando elimini il cluster di database, le istantanee manuali acquisite in precedenza continuano a esistere. Se in precedenza queste istantanee non venivano fatturate perché rientravano nel periodo di conservazione dei backup automatici, ora non sono più coperte e iniziano a essere fatturate per intero in base al loro utilizzo.

Come posso ridurre i costi dell'archiviazione dei backup?

Esistono alcuni modi per ridurre i costi relativi all'utilizzo dei backup:

- Elimina le istantanee manuali che non rientrano nel periodo di conservazione dei backup automatici. Ciò include le istantanee acquisite e le istantanee potenzialmente acquisite dal piano di Backup AWS. Controlla il piano di Backup AWS per assicurarti che non conservi le istantanee al di fuori del periodo di conservazione previsto.
- Valuta le operazioni di scrittura e aggiornamento eseguite sul database per vedere se puoi ridurre il numero di modifiche che stai apportando. Poiché il backup automatico archivia tutte le modifiche incrementali nel il periodo di conservazione definito, la riduzione del numero di aggiornamenti da effettuare riduce anche i costi del backup automatico.
- Valuta se è ragionevole ridurre il periodo di conservazione del backup automatico. La riduzione del periodo di conservazione significa che il backup archivia meno giorni di dati incrementali, il che potrebbe ridurre il costo complessivo del backup. Tuttavia, la riduzione di questo periodo potrebbe anche comportare la fatturazione di alcune istantanee in quanto non rientrano più nel periodo di conservazione. Assicurati di controllare tutti i costi aggiuntivi per le istantanee che potresti dover sostenere prima di decidere se questa è la linea d'azione più idonea.

Come viene fatturato lo storage dei backup?

Lo storage di backup è fatturato per GB/mese.

Ciò significa che l'utilizzo dello storage di backup viene addebitato come la media ponderata dell'utilizzo durante il mese specificato. Ecco alcuni esempi riferiti a un mese di 30 giorni:

- L'utilizzo fatturato del backup è pari a 100 GB per tutti i 30 giorni del mese. Il costo viene addebitato nel seguente modo:

```
(100 GB * 30) / 30 = 100 GB-month
```

- L'utilizzo fatturato del backup è pari a 100 GB per i primi 15 giorni del mese, quindi a 0 GB per gli ultimi 15. Il costo viene addebitato nel seguente modo:

```
(100 GB * 15 + 0 GB * 15) / 30 = 50 GB-month
```

- L'utilizzo fatturato del backup è pari a 50 GB per i primi 10 giorni del mese, a 100 GB per i successivi 10 giorni, quindi a 150 GB per gli ultimi 10. Il costo viene addebitato nel seguente modo:

```
(50 GB * 10 + 100 GB * 10 + 150 GB * 10) / 30 = 100 GB-month
```

In che modo l'impostazione del backtracking per il cluster database influisce sull'utilizzo dello spazio di archiviazione del backup?

L'impostazione di backtrack per un cluster DB Aurora non influenza il volume dei dati di backup per tale cluster. Aurora fattura separatamente l'archiviazione per dati di backtracking. Per informazioni sui prezzi del backtracking Aurora, consulta la pagina Prezzi di Amazon Aurora.

Come si applicano i costi di archiviazione alle istantanee condivise?

Se si condivide uno snapshot con un altro utente, si continua a essere il proprietario di tale snapshot. I costi dello storage spettano al proprietario della snapshot. Se elimini una snapshot condivisa di tua proprietà, nessuno potrà accedervi.

Per mantenere l'accesso a una snapshot condivisa di proprietà di un altro utente, puoi copiarla. In questo modo diventerai il proprietario della nuova snapshot. Eventuali costi di storage per la snapshot copiata si applicano al tuo account.

Per ulteriori informazioni sugli snapshot di condivisione, consulta Condivisione di uno snapshot cluster database. Per ulteriori informazioni sulla copia di snapshot, consulta Copia di uno snapshot del cluster di database.

Creazione di uno snapshot del cluster database

Amazon RDS crea una snapshot dei volumi di storage del cluster di database eseguendo il backup dell'intero cluster anziché dei singoli database. Quando crei uno snapshot di cluster database è necessario identificare qual è il cluster database di cui stai effettuando il backup e dare un nome alla snapshot di cluster database in modo da poterlo usare successivamente per il ripristino. La quantità di tempo necessaria per creare una snapshot di cluster di database varia a seconda della dimensione dei database. Poiché lo snapshot include l'intero volume d'archiviazione, la dimensione dei file, come i file temporanei, influisce sulla quantità di tempo necessaria per creare lo snapshot.

> **Note**
>
> Il tuo cluster database deve essere nello stato `available` per poter acquisire uno snapshot del cluster database.

A differenza dei backup automatizzati, gli snapshot manuali non sono soggetti al periodo di retention dei backup. Gli snapshot non scadono.

Per i backup a lungo termine, si consiglia di esportare i dati delle snapshot in Amazon S3. Se la versione principale del motore DB non è più supportata, non è possibile ripristinare tale versione da un'istantanea. Per ulteriori informazioni, consulta Esportazione dei dati dello snapshot del cluster di database in Amazon S3.

È possibile creare uno snapshot del cluster DB utilizzando l'Console di gestione AWSAPIAWS CLI, the o RDS.

Console

Come creare uno snapshot del cluster di database

1. Accedi a Console di gestione AWS e apri la console Amazon RDS all'indirizzo https://console.aws.amazon.com/rds/.
2. Nel riquadro di navigazione, selezionare Snapshots (Snapshot).

   Viene visualizzato l'elenco Snapshot manuali.
3. Seleziona Acquisisci snapshot.

   Viene visualizzata la finestra Acquisizione di snapshot DB.

4. Per il Tipo di snapshot, seleziona Cluster DB.
   Acquisizione di uno snapshot del database
5. Scegli il cluster di database per cui desideri acquisire uno snapshot.
6. Immetti il Nome snapshot.
7. Seleziona Acquisisci snapshot.

   Viene visualizzato l'elenco Snapshot manuali, con lo stato del nuovo snapshot cluster di database che indica `Creating`. Dopo che lo stato è diventato `Available`, potrai vedere il tempo di creazione.

AWS CLI

Quando crei uno snapshot del cluster DB utilizzando ilAWS CLI, devi identificare il cluster di DB di cui vuoi eseguire il backup e quindi assegnare un nome allo snapshot del cluster DB in modo da poterlo ripristinare in un secondo momento. È possibile eseguire questa operazione utilizzando il AWS CLI `create-db-cluster-snapshot`comando con i seguenti parametri:

- `--db-cluster-identifier`
- `--db-cluster-snapshot-identifier`

In questo esempio, si crea uno snapshot del cluster DB denominato *`mydbclustersnapshot`* per un cluster DB chiamato*`mydbcluster`*.

Example

Per Linux, macOS o Unix:

```
aws rds create-db-cluster-snapshot \
    --db-cluster-identifier mydbcluster \
    --db-cluster-snapshot-identifier mydbclustersnapshot
```

Per Windows:

```
aws rds create-db-cluster-snapshot ^
    --db-cluster-identifier mydbcluster ^
    --db-cluster-snapshot-identifier mydbclustersnapshot
```

API RDS

Quando crei una snapshot del cluster di database con l'API Amazon RDS, devi specificare il cluster di database di cui intendi eseguire il backup e dare un nome alla snapshot del cluster di database in modo da poterla utilizzare successivamente per il ripristino. Puoi eseguire questa operazione utilizzando il comando API Amazon RDS `CreateDBClusterSnapshot` con i seguenti parametri:

- DBClusterIdentificatore
- DBClusterSnapshotIdentifier

Determinare se lo snapshot del cluster di database è disponibile

È possibile verificare che lo snapshot del cluster DB sia disponibile esaminando la sezione Istantanee nella scheda Manutenzione e backup nella pagina dei dettagli del cluster inConsole di gestione AWS, utilizzando il comando `describe-db-cluster-snapshots`CLI o utilizzando l'azione API. `DescribeDBClusterSnapshots`

Puoi inoltre utilizzare il comando CLI `wait db-cluster-snapshot-available` per eseguire il polling dell'API ogni 30 secondi fino a quando lo snapshot è disponibile.

Ripristino da uno snapshot cluster database

Amazon RDS crea una snapshot dei volumi di storage del cluster di database eseguendo il backup dell'intero cluster anziché dei singoli database. È possibile creare un nuovo cluster di database eseguendo il ripristino da uno snapshot di database. Devi fornire il nome dello snapshot del cluster di database da cui eseguire il ripristino e quindi un nome per il nuovo cluster di database che viene creato dal ripristino. Non è possibile eseguire il ripristino da una snapshot di cluster di database in un cluster di database esistente. Al momento del ripristino, viene creato un nuovo cluster di database.

> **Important**
>
> Non è possibile ripristinare un'istantanea su una versione del motore DB che ha superato la data di fine del supporto standard di Aurora. È possibile accedere a un database solo dopo che è stato eseguito correttamente l'aggiornamento a una versione supportata. Per ulteriori informazioni sulle versioni supportate del motore Aurora DB, consulta. Supporto esteso di Amazon RDS con Amazon Aurora
> Se l'aggiornamento a una versione supportata per il cluster fallisce, lo stato del cluster cambia in `upgrade_failed` e Aurora crea un'istantanea finale con il prefisso. `rds-final` Per accedere al database ripristinato sulla versione obsoleta dopo un errore di aggiornamento, contattate il supporto. AWS

È possibile utilizzare l'istanza del cluster di database ripristinato non appena lo stato diventa `available`.

È possibile utilizzare CloudFormation per ripristinare un cluster DB da un'istantanea del cluster DB. Per ulteriori informazioni, consulta AWS: :RDS:: DBCluster nella Guida per l'AWS CloudFormation utente.

> **Note**
>
> La condivisione manuale di un'istantanea del cluster DB, crittografata o non crittografata, consente agli AWS account autorizzati di ripristinare direttamente un cluster DB dalla snapshot invece di prenderne una copia ed eseguire il ripristino da quella. Per ulteriori informazioni, consulta Condivisione di uno snapshot cluster database.

Per informazioni sul ripristino di un cluster di database Aurora o di un cluster globale con una versione Supporto esteso di RDS, consulta Ripristino di un cluster di database Aurora o un cluster globale con il supporto esteso di Amazon RDS.

Considerazioni sui gruppi di parametri

È consigliabile mantenere il gruppo parametri del database e il gruppo di parametri del cluster di database per tutti gli snapshot del cluster di database creati, in modo da poter associare il cluster di database ripristinato ai gruppi di parametri corretti.

Il gruppo parametri del database di default e il gruppo di parametri del cluster di database sono associati al cluster ripristinato, a meno che non si scelgano gruppi diversi. Non sono disponibili impostazioni personalizzate per i parametri nei gruppi di parametri di default.

È possibile specificare i gruppi di parametri al momento del ripristino del cluster di database.

Per ulteriori informazioni sui gruppi di parametri del database e i gruppi di parametri del cluster di database, consulta Gruppi di parametri per Amazon Aurora.

Considerazioni relative al gruppo di sicurezza

Quando ripristini un cluster di database, il cloud privato virtuale (VPC) di default, il gruppo di sottoreti di database e il gruppo di sicurezza VPC sono associati all'istanza ripristinata, a meno che non si scelgano altri gruppi.

- Se utilizzi la console Amazon RDS, puoi specificare un gruppo di sicurezza VPC personalizzato da associare al cluster o creare un nuovo gruppo di sicurezza VPC.
- Se utilizzi il AWS CLI, puoi specificare un gruppo di sicurezza VPC personalizzato da associare al cluster includendo l'`--vpc-security-group-ids`opzione nel comando. `restore-db-cluster-from-snapshot`
- Se utilizzi l'API di Amazon RDS, puoi includere il parametro `VpcSecurityGroupIds.VpcSecurityGroupId.N` nell'operazione `RestoreDBClusterFromSnapshot`.

Non appena il ripristino è completo e il nuovo cluster di database è disponibile, puoi anche cambiare le impostazioni del VPC modificando il cluster di database. Per ulteriori informazioni, consulta Modifica di un cluster database Amazon Aurora.

Considerazioni Amazon Aurora

Con Aurora, si ripristina uno snapshot cluster DB in un cluster DB.

Con Aurora MySQL e Aurora PostgreSQL, è anche possibile ripristinare uno snapshot cluster database in un cluster database Aurora Serverless. Per ulteriori informazioni, consulta Ripristino di un cluster database Aurora Serverless v1.

Con Aurora MySQL, è anche possibile ripristinare uno snapshot cluster database da un cluster senza query parallela a un cluster con query parallela. Poiché la query parallela viene usata in genere con tabelle di dimensioni molto grandi, il meccanismo di snapshot è il più veloce per inserire volumi elevati di dati in un cluster Aurora MySQL abilitato per la query parallela. Per ulteriori informazioni, consulta Query parallela per Amazon Aurora MySQL.

Ripristino da uno snapshot

È possibile ripristinare un cluster di database da uno snapshot del cluster di database tramite la Console di gestione AWS, la AWS CLI o l'API RDS.

Console

Per ripristinare un cluster database da una snapshot cluster DB

1. Accedi a Console di gestione AWS e apri la console Amazon RDS all'indirizzo https://console.aws.amazon.com/rds/.
2. Nel riquadro di navigazione, selezionare Snapshots (Snapshot).
3. Scegliere lo snapshot cluster database da cui eseguire il ripristino.
4. Per Actions (Operazioni), selezionare Restore Snapshot (Ripristina snapshot).

   Viene visualizzata la pagina Ripristina snapshot.
5. Scegli la versione del motore del database in cui desideri ripristinare il cluster database.

   Per impostazione predefinita, lo snapshot viene ripristinato nella stessa versione del motore del database del cluster database di origine, se tale versione è disponibile.
6. Per DB Instance Identifier (Identificatore di istanza database), immetti il nome dell'istanza database ripristinata. È importante ricordare che Amazon RDS ricava l’identificatore del cluster di database dall’identificatore dell’istanza database specificato dall’utente.
7. Specificare altre impostazioni, ad esempio la configurazione dell'archiviazione del cluster database.

Per informazioni su ciascuna impostazione, consulta Impostazioni per cluster di database Aurora.

8. Scegliere Restore DB Cluster (Ripristina cluster di database).

AWS CLI

Per ripristinare un cluster DB da un'istantanea del cluster DB, usa il AWS CLI comando restore-db-cluster-from -snapshot.

In questo esempio il ripristino avviene da uno snapshot del cluster DB creato precedentemente e denominato `mydbclustersnapshot`. Viene ripristinato un nuovo cluster DB denominato `mynewdbcluster`.

Puoi specificare altre impostazioni, come la versione del motore del database. Se non specifichi una versione del motore, il cluster database viene ripristinato alla versione del motore predefinita.

Per informazioni su ciascuna impostazione, consulta Impostazioni per cluster di database Aurora.

Example

Per Linux, macOS o Unix:

```
aws rds restore-db-cluster-from-snapshot \
    --db-cluster-identifier mynewdbcluster \
    --snapshot-identifier mydbclustersnapshot \
    --engine aurora-mysql|aurora-postgresql
```

Per Windows:

```
aws rds restore-db-cluster-from-snapshot ^
    --db-cluster-identifier mynewdbcluster ^
    --snapshot-identifier mydbclustersnapshot ^
    --engine aurora-mysql|aurora-postgresql
```

Dopo il ripristino del cluster di database, devi aggiungere quest'ultimo al gruppo di sicurezza utilizzato dal cluster di database per creare la snapshot cluster di database, se desideri le stesse funzionalità del cluster di database precedente.

> **Important**
>
> Se si utilizza la console per ripristinare un cluster database, Amazon RDS crea automaticamente l'istanza DB primaria (istanza di scrittura) per il cluster database. Se si utilizza AWS CLI per ripristinare un cluster di database, è necessario creare in modo esplicito l'istanza principale per il cluster di database. L'istanza primaria è la prima istanza creata in un cluster di database. Se non si crea l'istanza DB primaria, lo stato degli endpoint del cluster database rimane impostato su `creating`.
> Chiama il create-db-instance AWS CLI comando per creare l'istanza principale per il tuo cluster DB. Includi il nome del cluster di database come valore dell'opzione `--db-cluster-identifier`.

API RDS

Per ripristinare un cluster DB da un'istantanea del cluster DB, richiama l'operazione Restore dell'API RDS DBCluster FromSnapshot con i seguenti parametri:

- `DBClusterIdentifier`
- `SnapshotIdentifier`

> **Important**
>
> Se si utilizza la console per ripristinare un cluster database, Amazon RDS crea automaticamente l'istanza DB primaria (istanza di scrittura) per il cluster database. Se si utilizza API RDS per ripristinare un cluster di database, è necessario creare in modo esplicito l'istanza principale per il cluster di database. L'istanza primaria è la prima istanza creata in un cluster di database. Se non si crea l'istanza DB primaria, lo stato degli endpoint del cluster database rimane impostato su `creating`.
> Chiama l'operazione Create dell'API RDS DBInstance per creare l'istanza principale per il cluster DB. Includi il nome del cluster di database come valore del parametro `DBClusterIdentifier`.

Copia di uno snapshot del cluster di database

Con Amazon Aurora è possibile copiare backup automatici o snapshot manuali del cluster di database. Dopo aver copiato uno snapshot, la copia è uno snapshot manuale. È possibile creare più copie di un backup automatico o di uno snapshot manuale, ma ogni copia deve avere un identificatore univoco.

È possibile copiare un'istantanea all'interno della stessa Regione AWS, copiare un'istantanea dall'altra Regioni AWS e copiare istantanee condivise. È possibile copiare le istantanee su un altro account in Regione AWS un unico passaggio.

> **Note**
>
> Amazon ti invia una fattura in base alla quantità dei dati di backup e snapshot Amazon Aurora mantenuti e alla durata di mantenimento. Per informazioni sullo storage associato ai backup e agli snapshot Aurora, consulta Informazioni sull'utilizzo dello storage di backup Amazon Aurora. Per informazioni sui prezzi relativi allo storage Aurora, consulta prezzi di Amazon RDS for Aurora.

Esamina i limiti e le considerazioni per la copia di snapshot del cluster di database. Per copiare snapshot del cluster di database, consulta uno dei seguenti argomenti.

- Copia di uno snapshot di cluster di database tramite Console di gestione AWS
- Copia di uno snapshot del cluster database non crittografato tramite AWS CLI o l'API di Amazon RDS
- Copia di uno snapshot del cluster database crittografato tramite AWS CLI o l'API di Amazon RDS
- Copia di uno snapshot cluster database tra account

Copia di uno snapshot di cluster di database tramite Console di gestione AWS

Utilizza le procedure in questo argomento, per copiare uno snapshot cluster database. Se il motore di database di origine è Aurora, lo snapshot è uno snapshot cluster database.

Per ogni account AWS, puoi copiare fino a cinque snapshot del cluster di database alla volta, da una Regione AWS all'altra. Viene supportata la copia di snapshot cluster database crittografato e

non crittografato. Se copi uno snapshot del cluster di database in un'altra Regione AWS, crei uno snapshot del cluster di database manuale che viene conservato in tale Regione AWS. La copia di uno snapshot del cluster di database dalla Regione AWS di origine comporta dei costi di trasferimento dei dati Amazon RDS.

Per ulteriori informazioni sui prezzi del trasferimento dati, consulta Prezzi di Amazon RDS.

Dopo che la copia dello snapshot del cluster di database è stata creata nella nuova Regione AWS, tale copia si comporta allo stesso modo degli altri snapshot del cluster di database in tale Regione AWS.

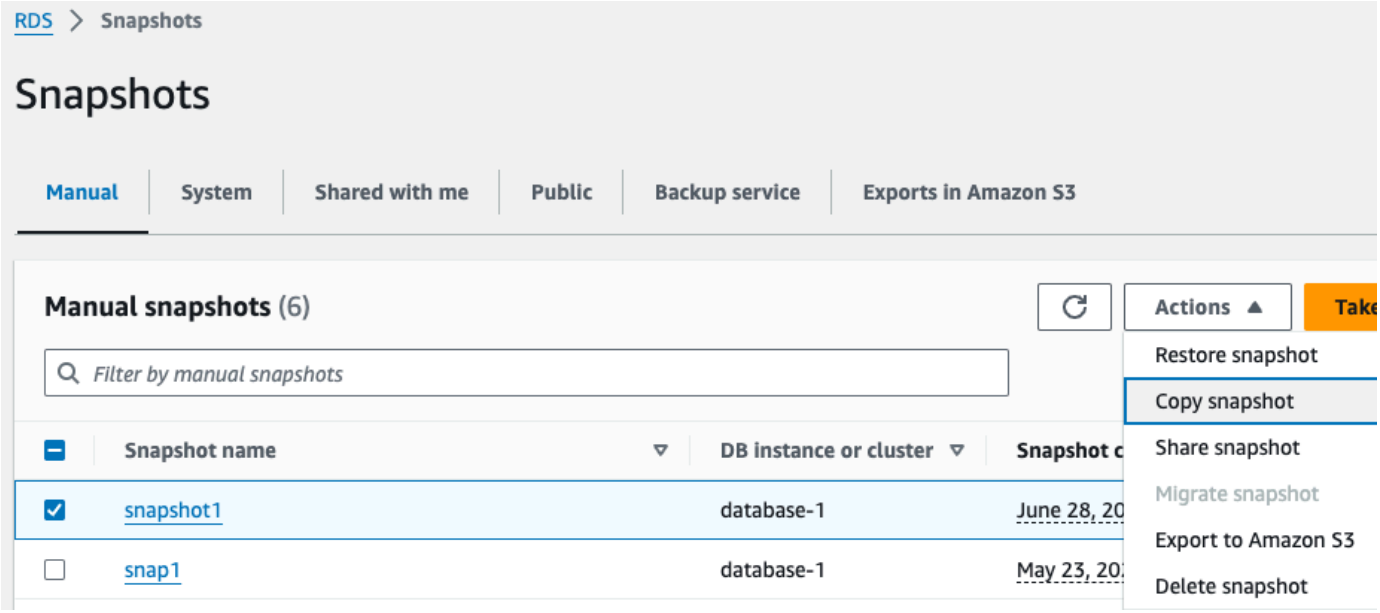
Questa procedura funziona per snapshot del cluster di database crittografati o non crittografati, nella stessa Regione AWS o tra regioni.

Per cancellare un'operazione di copia quando è in corso, elimina la snapshot cluster database target mentre la snapshot cluster database è nello stato copying (copia in corso).

Prima di copiare uno snapshot del cluster di database, esamina Limitazioni e Considerazioni sulla copia di snapshot.

Per copiare uno snapshot cluster database

1. Accedi alla Console di gestione AWS e apri la console Amazon RDS all'indirizzo https://console.aws.amazon.com/rds/.
2. Nel pannello di navigazione, scegli Snapshot.
3. Seleziona lo snapshot del cluster di database che desideri copiare.
4. Scegli Operazioni, quindi Copia snapshot.



5. (Facoltativo) Per copiare lo snapshot del cluster di database in una Regione AWS diversa, scegli quella Regione AWS in Regione di destinazione.
6. Digita il nome della copia di snapshot di cluster di database in Nuovo identificatore snapshot del database.
7. Per copiare i tag e i valori dalla snapshot alla copia della snapshot, seleziona Copy Tags (Copia tag).
8. Seleziona Copy Snapshot (Copia snapshot).

Limitazioni

Di seguito sono riportate alcune limitazioni che si applicano quando si copiano le snapshot:

- Non puoi copiare un'istantanea da o verso i seguenti siti: Regioni AWS
  - Cina (Pechino)
  - Cina (Ningxia)
- È possibile copiare un'istantanea tra AWS GovCloud (Stati Uniti orientali) e AWS GovCloud (Stati Uniti occidentali). Tuttavia, non puoi copiare un'istantanea tra queste AWS GovCloud (US) regioni e quelle commerciali. Regioni AWS
- Se elimini una snapshot origine prima che la snapshot target diventi disponibile, la copia della snapshot potrebbe non riuscire. Verifica che la snapshot target abbia lo stato di `AVAILABLE` prima di eliminare una snapshot origine.

- Puoi avere un massimo di cinque richieste di copia di snapshot in corso in una singola regione di destinazione per account
- Quando vengono richieste più copie di snapshot per la stessa istanza database di origine, vengono accodate internamente. Le copie richieste in seguito non verranno avviate fino al completamento delle copie di snapshot precedenti. Per ulteriori informazioni, consulta Perché la creazione di snapshot EC2 AMI o EBS è lenta? nel AWS Knowledge Center.
- A seconda del soggetto Regioni AWS coinvolto e della quantità di dati da copiare, il completamento di una copia istantanea tra diverse regioni può richiedere ore. In alcuni casi, potrebbe esserci un numero elevato di richieste di copia di snapshot tra regioni da una determinata regione di origine. In questi casi, Amazon RDS potrebbe mettere in coda le richieste di copia tra regioni provenienti dalla regione di origine fino al completamento di alcune copie in corso. Nessuna informazione di progresso viene visualizzata sulle richieste di copia mentre sono in coda. Le informazioni sul progresso vengono visualizzate quando inizia la copia.
- Aurora non supporta gli snapshot incrementali. Le copie degli snapshot del cluster di database Aurora vengono sempre archiviate come copie complete. La copia di uno snapshot completa contiene tutti i dati e i metadati necessari per archiviare l'istanza database.

Considerazioni sulla copia di snapshot

Di seguito sono riportate alcune considerazioni relative alla copia di snapshot.

Argomenti

- Considerazioni sulla copia di snapshot condivisi
- Considerazioni sulla copia di snapshot crittografati dei cluster di database
- Considerazioni sulla copia di snapshot tra Regioni diverse
- Considerazioni sui gruppi di parametri

Considerazioni sulla copia di snapshot condivisi

Puoi copiare istantanee condivise con te da altri account. AWS In alcuni casi, è possibile copiare un'istantanea crittografata che è stata condivisa da un altro AWS account. In questi casi, è necessario avere accesso allo AWS KMS key strumento utilizzato per crittografare l'istantanea. Per ulteriori informazioni, consulta Condivisione di snapshot crittografati.

Copia tra regioni e più account in un unico passaggio

Per copiare un'istantanea tra regioni e più account in un'unica operazione, devi prima condividere l'istantanea con l'account di destinazione. AWS Se l'istantanea è crittografata, devi anche condividere la AWS KMS chiave con l'account di destinazione. AWS Se l'istantanea è crittografata con la AWS KMS chiave predefinita, è necessario prima copiarla per crittografarla nuovamente con una chiave gestita dal cliente prima di condividerla con l'account di destinazione. Una volta condivisa, puoi iniziarne una copia su quell'account (interno o interregionale) dall'account di destinazione.

Considerazioni sulla copia di snapshot crittografati dei cluster di database

Puoi copiare una snapshot che è stata crittografata utilizzando una chiave KMS. Se la copia di una snapshot crittografata, la copia della snapshot deve anche essere crittografata. Se copi un'istantanea crittografata all'interno dello stesso Regione AWS, puoi crittografare la copia con la stessa chiave KMS dell'istantanea originale. Oppure puoi specificare una chiave KMS diversa.

Se copi uno snapshot crittografato tra regioni, devi specificare una chiave KMS valida nella Regione AWS di destinazione. Può essere una chiave KMS specifica per la regione o una chiave multi-regione. Per ulteriori informazioni sulle chiavi multi-regione, consulta Utilizzo delle chiavi multi-regione in AWS KMS.

Per ulteriori informazioni sulla gestione delle AWS KMS chiavi per Amazon RDS, consultaGestione di AWS KMS key.

La snapshot di origine resta crittografata nel processo di copia. Per ulteriori informazioni, consulta Limiti relativi a istanze database crittografate Amazon Aurora.

> **Note**
>
> Per le snapshot di cluster di database Amazon Aurora, non puoi crittografare una snapshot di cluster di database quando copi la snapshot.

Per copiare snapshot crittografati dei cluster di database, consulta i seguenti argomenti.

- Copia di uno snapshot del cluster database crittografato tramite AWS CLI o l'API di Amazon RDS
- Copia di uno snapshot cluster database tra account

Considerazioni sulla copia di snapshot tra Regioni diverse

È possibile copiare snapshot di cluster di database tra Regioni AWS. Tuttavia, esistono alcuni vincoli e considerazioni per la copia di snapshot tra le regioni.

A seconda del soggetto Regioni AWS coinvolto e della quantità di dati da copiare, il completamento di una copia istantanea tra diverse regioni può richiedere ore.

In alcuni casi, potrebbe esserci un numero elevato di richieste di copia di snapshot tra regioni da una determinata Regione AWS di origine. In questi casi, Amazon RDS potrebbe mettere in coda nuove richieste di copia interregionali provenienti da tale fonte fino al completamento di alcune copie Regione AWS in corso. Nessuna informazione di progresso viene visualizzata sulle richieste di copia mentre sono in coda. Le informazioni sul progresso vengono visualizzate quando inizia la copia.

Per la copia di istantanee in più regioni si applicano i costi di trasferimento dei dati. La copia di snapshot tra Regioni crea copie complete dei dati di destinazione, ma i costi di trasferimento dei dati sono incrementali. I dati incrementali includono sia i nuovi dati che sono stati aggiunti al database di un cliente dopo l'ultima copia, sia tutte le modifiche apportate ai dati esistenti. Per ulteriori informazioni, consulta Creazione di copie di backup tra Regioni AWS nella Guida per gli sviluppatori di AWS Backup .

> **Note**
>
> Aurora copia la quantità minima di dati richiesta per creare una copia completa di uno snapshot nella Regione di destinazione. Le tariffe per il trasferimento dei dati si applicano quando si copiano snapshot tra Regioni.

Considerazioni sui gruppi di parametri

Quando copi uno snapshot tra regioni, la copia non include il gruppo di parametri usato dal cluster di database originale. Quando si ripristina un'istantanea per creare un nuovo cluster DB, tale cluster DB ottiene il gruppo di parametri predefinito in cui Regione AWS è stato creato. Per assegnare al nuovo cluster di database gli stessi parametri dell'originale, completa la seguente procedura:

1. Nella destinazione Regione AWS, crea un gruppo di parametri del cluster DB con le stesse impostazioni del cluster DB originale. Se ne esiste già uno nel nuovo Regione AWS, puoi usare quello.

2. Dopo aver ripristinato l'istantanea nella destinazione Regione AWS, modifica il nuovo cluster DB e aggiungi il gruppo di parametri nuovo o esistente del passaggio precedente.

Copia di uno snapshot del cluster database non crittografato tramite AWS CLI o l'API di Amazon RDS

Usa le procedure presentate nelle sezioni seguenti per copiare uno snapshot cluster di database non crittografato tramite Console di gestione AWS, AWS CLI o l'API Amazon RDS.

Per cancellare un'operazione di copia quando è in corso, elimina lo snapshot del cluster di database target identificato da `--target-db-cluster-snapshot-identifier` o `TargetDBClusterSnapshotIdentifier` mentre lo stato dello snapshot è Copia in corso.

Console

Copia di uno snapshot di cluster di database tramite Console di gestione AWS, consulta Copia di uno snapshot di cluster di database tramite Console di gestione AWS.

AWS CLI

Puoi copiare uno snapshot di database tramite il comando AWS CLI copy-db-cluster-snapshot. Se stai copiando lo snapshot in un'altra Regione AWS, esegui il comando nella Regione AWS nella quale verrà copiato lo snapshot.

Le seguenti opzioni vengono utilizzate per copiare una snapshot cluster database non crittografata:

- `--source-db-cluster-snapshot-identifier` – Identificatore per lo snapshot cluster database da copiare. Se stai copiando lo snapshot in un'altra Regione AWS, l'identificatore deve essere nel formato ARN valido per la Regione AWS di origine.
- `--target-db-cluster-snapshot-identifier` – Identificatore per la nuova copia dello snapshot cluster database.

Il codice seguente crea una copia dello snapshot del cluster di database `arn:aws:rds:us-east-1:123456789012:cluster-snapshot:aurora-cluster1-snapshot-20130805` denominata `myclustersnapshotcopy` nella Regione AWS nella quale viene eseguito il comando. Quando viene creata la copia, tutti i tag della snapshot originale vengono copiati nella copia della snapshot.

Example

Per Linux, macOS o Unix:

```
aws rds copy-db-cluster-snapshot \
  --source-db-cluster-snapshot-identifier arn:aws:rds:us-east-1:123456789012:cluster-snapshot:aurora-cluster1-snapshot-20130805 \
  --target-db-cluster-snapshot-identifier myclustersnapshotcopy \
  --copy-tags
```

Per Windows:

```
aws rds copy-db-cluster-snapshot ^
  --source-db-cluster-snapshot-identifier arn:aws:rds:us-east-1:123456789012:cluster-snapshot:aurora-cluster1-snapshot-20130805 ^
  --target-db-cluster-snapshot-identifier myclustersnapshotcopy ^
  --copy-tags
```

API RDS

Per copiare uno snapshot del cluster di database, utilizzare l'operazione API Amazon RDS CopyDBClusterSnapshot. Se stai copiando lo snapshot in un'altra Regione AWS, esegui l'operazione nella Regione AWS nella quale verrà copiato lo snapshot.

I seguenti parametri vengono utilizzati per copiare una snapshot cluster database non crittografata:

- `SourceDBClusterSnapshotIdentifier` – Identificatore per lo snapshot cluster database da copiare. Se stai copiando lo snapshot in un'altra Regione AWS, l'identificatore deve essere nel formato ARN valido per la Regione AWS di origine.
- `TargetDBClusterSnapshotIdentifier` – Identificatore per la nuova copia dello snapshot cluster database.

Il seguente codice crea una copia di uno snapshot `arn:aws:rds:us-east-1:123456789012:cluster-snapshot:aurora-cluster1-snapshot-20130805` denominata `myclustersnapshotcopy` nella regione Stati Uniti occidentali (California settentrionale). Quando viene creata la copia, tutti i tag della snapshot originale vengono copiati nella copia della snapshot.

Example

```
https://rds.us-west-1.amazonaws.com/
   ?Action=CopyDBClusterSnapshot
   &CopyTags=true
   &SignatureMethod=HmacSHA256
   &SignatureVersion=4
   &SourceDBSnapshotIdentifier=arn%3Aaws%3Ards%3Aus-east-1%3A123456789012%3Acluster-
snapshot%3Aaurora-cluster1-snapshot-20130805
   &TargetDBSnapshotIdentifier=myclustersnapshotcopy
   &Version=2013-09-09
   &X-Amz-Algorithm=AWS4-HMAC-SHA256
   &X-Amz-Credential=AKIADQKE4SARGYLE/20140429/us-west-1/rds/aws4_request
   &X-Amz-Date=20140429T175351Z
   &X-Amz-SignedHeaders=content-type;host;user-agent;x-amz-content-sha256;x-amz-date
   &X-Amz-Signature=9164337efa99caf850e874a1cb7ef62f3cea29d0b448b9e0e7c53b288ddffed2
```

Copia di uno snapshot del cluster database crittografato tramite AWS CLI o l'API di Amazon RDS

Usa le procedure presentate nelle sezioni seguenti per copiare uno snapshot cluster di database crittografato tramite Console di gestione AWS, AWS CLI o l'API di Amazon RDS.

Per cancellare un'operazione di copia quando è in corso, elimina lo snapshot del cluster di database target identificato da `--target-db-cluster-snapshot-identifier` o `TargetDBClusterSnapshotIdentifier` mentre lo stato dello snapshot è Copia in corso.

Console

Copia di uno snapshot di cluster di database tramite Console di gestione AWS, consulta Copia di uno snapshot di cluster di database tramite Console di gestione AWS.

AWS CLI

Puoi copiare uno snapshot di database tramite il comando AWS CLI copy-db-cluster-snapshot. Se stai copiando lo snapshot in un'altra Regione AWS, esegui il comando nella Regione AWS nella quale verrà copiato lo snapshot.

Le seguenti opzioni vengono utilizzate per copiare uno snapshot cluster database crittografato:

- `--source-db-cluster-snapshot-identifier` – Identificatore per lo snapshot cluster database crittografato da copiare. Se stai copiando lo snapshot in un'altra Regione AWS, l'identificatore deve essere nel formato ARN valido per la Regione AWS di origine.
- `--target-db-cluster-snapshot-identifier` – Identificatore per la nuova copia dello snapshot cluster database crittografato.
- `--kms-key-id`: l'identificatore della chiave KMS per la chiave da usare per crittografare la copia dello snapshot cluster database.

  Facoltativamente puoi utilizzare questa opzione se lo snapshot del cluster di database è crittografato, se si copia lo snapshot nella stessa Regione AWS e se desideri specificare una nuova chiave KMS da utilizzare per crittografare la copia. Altrimenti, la copia dello snapshot cluster database viene crittografata con la stessa chiave KMS della snapshot cluster database origine.

  Devi utilizzare questa opzione se lo snapshot del cluster di database è crittografato e lo stai copiando in un'altra Regione AWS. In questo caso, devi specificare una chiave KMS per la Regione AWS di destinazione.

Il seguente esempio di codice copia lo snapshot del cluster di database crittografato dalla regione Stati Uniti occidentali (Oregon) alla regione US East (N. Virginia). Il comando viene chiamato nella regione US East (N. Virginia).

Example

Per Linux, macOS o Unix:

```
aws rds copy-db-cluster-snapshot \
  --source-db-cluster-snapshot-identifier arn:aws:rds:us-west-2:123456789012:cluster-snapshot:aurora-cluster1-snapshot-20161115 \
  --target-db-cluster-snapshot-identifier myclustersnapshotcopy \
  --kms-key-id my-us-east-1-key
```

Per Windows:

```
aws rds copy-db-cluster-snapshot ^
  --source-db-cluster-snapshot-identifier arn:aws:rds:us-west-2:123456789012:cluster-snapshot:aurora-cluster1-snapshot-20161115 ^
  --target-db-cluster-snapshot-identifier myclustersnapshotcopy ^
  --kms-key-id my-us-east-1-key
```

Il parametro `--source-region` è obbligatorio quando si copia uno snapshot del cluster di database crittografato tra le regioni AWS GovCloud (Stati Uniti orientali) e AWS GovCloud (Stati Uniti occidentali). Per `--source-region`, specifica la Regione AWS dell'istanza database di origine. La Regione AWS specificata in `source-db-cluster-snapshot-identifier` deve corrispondere alla Regione AWS specificata in `--source-region`.

Se non viene specificato `--source-region`, è necessario specificare un valore per `--pre-signed-url`. Un URL prefirmato è un URL che contiene una richiesta firmata Signature Versione 4 per il comando `copy-db-cluster-snapshot` chiamato nella Regione AWS di origine. Per ulteriori informazioni sull'opzione `pre-signed-url`, consulta copy-db-cluster-snapshot nel Riferimento ai comandi AWS CLI.

API RDS

Per copiare uno snapshot del cluster di database, utilizzare l'operazione API Amazon RDS CopyDBClusterSnapshot. Se stai copiando lo snapshot in un'altra Regione AWS, esegui l'operazione nella Regione AWS nella quale verrà copiato lo snapshot.

I seguenti parametri vengono utilizzati per copiare uno snapshot cluster database crittografato:

- `SourceDBClusterSnapshotIdentifier` – Identificatore per lo snapshot cluster database crittografato da copiare. Se stai copiando lo snapshot in un'altra Regione AWS, l'identificatore deve essere nel formato ARN valido per la Regione AWS di origine.
- `TargetDBClusterSnapshotIdentifier` – Identificatore per la nuova copia dello snapshot cluster database crittografato.
- `KmsKeyId`: l'identificatore della chiave KMS per la chiave da usare per crittografare la copia dello snapshot cluster database.

  Facoltativamente puoi utilizzare questo parametro se lo snapshot del cluster di database è crittografato, se si copia lo snapshot nella stessa Regione AWS e se desideri specificare una nuova chiave KMS da utilizzare per crittografare la copia. Altrimenti, la copia dello snapshot cluster database viene crittografata con la stessa chiave KMS della snapshot cluster database origine.

  Devi utilizzare questo parametro se lo snapshot del cluster di database è crittografato e lo stai copiando in un'altra Regione AWS. In questo caso, devi specificare una chiave KMS per la Regione AWS di destinazione.
- `PreSignedUrl`: se stai copiando lo snapshot in un'altra Regione AWS, devi specificare il parametro `PreSignedUrl`. Il valore `PreSignedUrl` deve essere un URL che contiene una

richiesta firmata Signature Version 4 per l'operazione `CopyDBClusterSnapshot` da chiamare nella Regione AWS di origine dalla quale lo snapshot del cluster di database viene copiato. Per ulteriori informazioni sull'utilizzo di un URL prefirmato, consulta CopyDBClusterSnapshot.

Il seguente esempio di codice copia lo snapshot del cluster di database crittografato dalla regione Stati Uniti occidentali (Oregon) alla regione US East (N. Virginia). L'operazione viene chiamata nella regione US East (N. Virginia).

Example

```
https://rds.us-east-1.amazonaws.com/
    ?Action=CopyDBClusterSnapshot
    &KmsKeyId=my-us-east-1-key
    &PreSignedUrl=https%253A%252F%252Frds.us-west-2.amazonaws.com%252F
         %253FAction%253DCopyDBClusterSnapshot
         %2526DestinationRegion%253Dus-east-1
         %2526KmsKeyId%253Dmy-us-east-1-key
         %2526SourceDBClusterSnapshotIdentifier%253Darn%25253Aaws%25253Ards
%25253Aus-west-2%25253A123456789012%25253Acluster-snapshot%25253Aaurora-cluster1-
snapshot-20161115
         %2526SignatureMethod%253DHmacSHA256
         %2526SignatureVersion%253D4
         %2526Version%253D2014-10-31
         %2526X-Amz-Algorithm%253DAWS4-HMAC-SHA256
         %2526X-Amz-Credential%253DAKIADQKE4SARGYLE%252F20161117%252Fus-west-2%252Frds
%252Faws4_request
         %2526X-Amz-Date%253D20161117T215409Z
         %2526X-Amz-Expires%253D3600
         %2526X-Amz-SignedHeaders%253Dcontent-type%253Bhost%253Buser-agent%253Bx-amz-
content-sha256%253Bx-amz-date
         %2526X-Amz-Signature
%253D255a0f17b4e717d3b67fad163c3ec26573b882c03a65523522cf890a67fca613
    &SignatureMethod=HmacSHA256
    &SignatureVersion=4
    &SourceDBClusterSnapshotIdentifier=arn%3Aaws%3Ards%3Aus-
west-2%3A123456789012%3Acluster-snapshot%3Aaurora-cluster1-snapshot-20161115
    &TargetDBClusterSnapshotIdentifier=myclustersnapshotcopy
    &Version=2014-10-31
    &X-Amz-Algorithm=AWS4-HMAC-SHA256
    &X-Amz-Credential=AKIADQKE4SARGYLE/20161117/us-east-1/rds/aws4_request
    &X-Amz-Date=20161117T221704Z
    &X-Amz-SignedHeaders=content-type;host;user-agent;x-amz-content-sha256;x-amz-date
```

```
&X-Amz-Signature=da4f2da66739d2e722c85fcfd225dc27bba7e2b8dbea8d8612434378e52adccf
```

Il parametro `PreSignedUrl` è obbligatorio quando si copia uno snapshot del cluster di database crittografato tra le regioni AWS GovCloud (Stati Uniti orientali) e AWS GovCloud (Stati Uniti occidentali). Il valore `PreSignedUrl` deve essere un URL che contiene una richiesta firmata Signature Version 4 per l'operazione `CopyDBClusterSnapshot` da chiamare nella Regione AWS di origine dalla quale lo snapshot del cluster di database viene copiato. Per ulteriori informazioni sull'utilizzo di un URL prefirmato, consulta CopyDBClusterSnapshot nella documentazione di riferimento dell'API Amazon RDS.

Per generare automaticamente anziché manualmente un URL prefirmato, utilizza il comando AWS CLI copy-db-cluster-snapshot con l'opzione `--source-region`.

Copia di uno snapshot cluster database tra account

Puoi abilitare altri account AWS per copiare snapshot di cluster database da te specificati usando le operazioni `ModifyDBClusterSnapshotAttribute` e `CopyDBClusterSnapshot` dell'API Amazon RDS. Puoi copiare gli snapshot del cluster di database solo tra account nella stessa Regione AWS. Il processo di copia tra account funziona come segue, dove l'Account A rende disponibile la snapshot da copiare e l'Account B la copia.

1. Usando l'account A, chiama `ModifyDBClusterSnapshotAttribute`, specificando **`restore`** per il parametro `AttributeName` e l'ID per l'account B per il parametro `ValuesToAdd`.
2. (Se la snapshot è crittografata) Quando utilizzi l'Account A, aggiorna la policy delle chiavi per la chiave KMS, prima aggiungendo l'ARN dell'Account B come un `Principal` e in seguito consenti l'operazione `kms:CreateGrant`.
3. Se lo snapshot è crittografato: usando l'account B, scegli o crea un utente e collega all'utente una policy IAM che permette di copiare uno snapshot cluster database crittografato tramite la chiave KMS.
4. Quando utilizzi l'Account B, chiama `CopyDBClusterSnapshot` e utilizza il parametro `SourceDBClusterSnapshotIdentifier` per specificare l'ARN della snapshot cluster database da copiare, che deve includere l'ID per l'Account A.

Per elencare tutti gli account AWS cui è permesso ripristinare uno snapshot del cluster DB, utilizza l'operazione API DescribeDBSnapshotAttributes o DescribeDBClusterSnapshotAttributes.

Per rimuovere l'autorizzazione di condivisione per un account AWS, utilizza l'operazione `ModifyDBSnapshotAttribute` o `ModifyDBClusterSnapshotAttribute` con `AttributeName` impostato su `restore` e l'ID dell'account da rimuovere nel parametro `ValuesToRemove`.

Copia di una snapshot cluster database non crittografata in un altro account

Utilizza la procedura seguente per copiare uno snapshot del cluster di database non crittografato in un altro account nella stessa Regione AWS.

1. Nell'account di origine per lo snapshot cluster database chiama `ModifyDBClusterSnapshotAttribute`, specificando **`restore`** per il parametro `AttributeName` e l'ID per l'account di destinazione per il parametro `ValuesToAdd`.

   L'esecuzione dell'esempio seguente tramite l'account `987654321` permette a due identificatori di account AWS, `123451234512` e `123456789012`, di ripristinare lo snapshot del cluster di database denominato `manual-snapshot1`.

   ```
   https://rds.us-west-2.amazonaws.com/
    ?Action=ModifyDBClusterSnapshotAttribute
    &AttributeName=restore
    &DBClusterSnapshotIdentifier=manual-snapshot1
    &SignatureMethod=HmacSHA256&SignatureVersion=4
    &ValuesToAdd.member.1=123451234512
    &ValuesToAdd.member.2=123456789012
    &Version=2014-10-31
    &X-Amz-Algorithm=AWS4-HMAC-SHA256
    &X-Amz-Credential=AKIADQKE4SARGYLE/20150922/us-west-2/rds/aws4_request
    &X-Amz-Date=20150922T220515Z
    &X-Amz-SignedHeaders=content-type;host;user-agent;x-amz-content-sha256;x-amz-date
    &X-Amz-Signature=ef38f1ce3dab4e1dbf113d8d2a265c67d17ece1999ffd36be85714ed36dddbb3
   ```

2. Nell'account target, chiama `CopyDBClusterSnapshot` e utilizza il parametro `SourceDBClusterSnapshotIdentifier` per specificare l'ARN della snapshot cluster database da copiare, che deve includere l'ID per l'account origine.

   L'esecuzione dell'esempio seguente utilizzando l'account `123451234512` copia la snapshot cluster database `aurora-cluster1-snapshot-20130805` dall'account `987654321` e crea una snapshot cluster database denominata `dbclustersnapshot1`.

   ```
   https://rds.us-west-2.amazonaws.com/
   ```

```
   ?Action=CopyDBClusterSnapshot
   &CopyTags=true
   &SignatureMethod=HmacSHA256
   &SignatureVersion=4
   &SourceDBClusterSnapshotIdentifier=arn:aws:rds:us-west-2:987654321:cluster-
snapshot:aurora-cluster1-snapshot-20130805
   &TargetDBClusterSnapshotIdentifier=dbclustersnapshot1
   &Version=2013-09-09
   &X-Amz-Algorithm=AWS4-HMAC-SHA256
   &X-Amz-Credential=AKIADQKE4SARGYLE/20150922/us-west-2/rds/aws4_request
   &X-Amz-Date=20140429T175351Z
   &X-Amz-SignedHeaders=content-type;host;user-agent;x-amz-content-sha256;x-amz-
date
   &X-Amz-
Signature=9164337efa99caf850e874a1cb7ef62f3cea29d0b448b9e0e7c53b288ddffed2
```

Copia di una snapshot cluster database crittografata in un altro account

Utilizza la procedura seguente per copiare uno snapshot del cluster di database crittografato in un altro account nella stessa Regione AWS.

1. Nell'account di origine per lo snapshot cluster database chiama `ModifyDBClusterSnapshotAttribute`, specificando **`restore`** per il parametro `AttributeName` e l'ID per l'account di destinazione per il parametro `ValuesToAdd`.

   L'esecuzione dell'esempio seguente tramite l'account `987654321` permette a due identificatori di account AWS, `123451234512` e `123456789012`, di ripristinare lo snapshot del cluster di database denominato `manual-snapshot1`.

   ```
   https://rds.us-west-2.amazonaws.com/
    ?Action=ModifyDBClusterSnapshotAttribute
    &AttributeName=restore
    &DBClusterSnapshotIdentifier=manual-snapshot1
    &SignatureMethod=HmacSHA256&SignatureVersion=4
    &ValuesToAdd.member.1=123451234512
    &ValuesToAdd.member.2=123456789012
    &Version=2014-10-31
    &X-Amz-Algorithm=AWS4-HMAC-SHA256
    &X-Amz-Credential=AKIADQKE4SARGYLE/20150922/us-west-2/rds/aws4_request
    &X-Amz-Date=20150922T220515Z
    &X-Amz-SignedHeaders=content-type;host;user-agent;x-amz-content-sha256;x-amz-date
   ```

```
   &X-Amz-Signature=ef38f1ce3dab4e1dbf113d8d2a265c67d17ece1999ffd36be85714ed36dddbb3
```

2. Nell'account di origine per lo snapshot cluster di database crea una chiave KMS personalizzata nella stessa Regione AWS dello snapshot di cluster di database crittografata. Durante la creazione della chiave gestita dal cliente, concedi l'accesso ad essa per la destinazione. Account AWS Per ulteriori informazioni, consulta Crea una chiave gestita dal cliente e concedine l'accesso.
3. Copia e condividi lo snapshot sulla destinazione Account AWS. Per ulteriori informazioni, consulta Copia e condividi l'istantanea dall'account di origine.
4. Nell'account target, chiama `CopyDBClusterSnapshot` e utilizza il parametro `SourceDBClusterSnapshotIdentifier` per specificare l'ARN della snapshot cluster database da copiare, che deve includere l'ID per l'account origine.

   L'esecuzione dell'esempio seguente utilizzando l'account `123451234512` copia la snapshot cluster database `aurora-cluster1-snapshot-20130805` dall'account `987654321` e crea una snapshot cluster database denominata `dbclustersnapshot1`.

```
https://rds.us-west-2.amazonaws.com/
   ?Action=CopyDBClusterSnapshot
   &CopyTags=true
   &SignatureMethod=HmacSHA256
   &SignatureVersion=4
   &SourceDBClusterSnapshotIdentifier=arn:aws:rds:us-west-2:987654321:cluster-
snapshot:aurora-cluster1-snapshot-20130805
   &TargetDBClusterSnapshotIdentifier=dbclustersnapshot1
   &Version=2013-09-09
   &X-Amz-Algorithm=AWS4-HMAC-SHA256
   &X-Amz-Credential=AKIADQKE4SARGYLE/20150922/us-west-2/rds/aws4_request
   &X-Amz-Date=20140429T175351Z
   &X-Amz-SignedHeaders=content-type;host;user-agent;x-amz-content-sha256;x-amz-
date
   &X-Amz-
Signature=9164337efa99caf850e874a1cb7ef62f3cea29d0b448b9e0e7c53b288ddffed2
```

Condivisione di uno snapshot cluster database

Con Amazon RDS puoi condividere uno snapshot cluster database manuale nei modi seguenti:

- La condivisione di uno snapshot del cluster di database manuale, crittografato o non crittografato permette agli account AWS autorizzati di copiare lo snapshot.
- La condivisione di uno snapshot cluster database manuale, crittografato o non crittografato, permette agli account AWS autorizzati di ripristinare direttamente un cluster database dallo snapshot, anziché eseguirne una copia da cui effettuare il ripristino.

> **Note**
>
> Per condividere uno snapshot cluster database automatico, crea uno snapshot cluster database manuale copiando lo snapshot automatico e quindi condividi la copia. Questo processo si applica anche alle risorse generate da AWS Backup.

Per ulteriori informazioni sulla creazione di una copia di una snapshot, consulta Copia di uno snapshot del cluster di database. Per ulteriori informazioni sul ripristino di un'istanza database da uno snapshot del cluster di database, consulta Ripristino da uno snapshot cluster database.

Per ulteriori informazioni sul ripristino di un cluster DB da una snapshot cluster DB, consulta Panoramica di backup e ripristino di un cluster di database Aurora.

È possibile condividere uno snapshot manuale con altri 20 Account AWS al massimo.

Nella condivisione di snapshot manuali con altri Account AWS, si applicano le seguenti restrizioni:

- Quando ripristini un cluster di database da uno snapshot condiviso usando AWS Command Line Interface (AWS CLI) o l'API di Amazon RDS, devi specificare il nome della risorsa Amazon (ARN) dello snapshot condiviso come identificatore dello snapshot.

Scopri come condividere snapshot, snapshot pubblici e snapshot crittografati nelle seguenti sezioni. Puoi anche scoprire come interrompere la condivisione di snapshot.

Argomenti

- Condivisione di uno snapshot

- Condivisione di snapshot pubblici
- Condivisione di snapshot crittografati
- Interruzione della condivisione degli snapshot

Condivisione di uno snapshot

Puoi condividere uno snapshot di cluster di database usando la Console di gestione AWS, la AWS CLI o l'API RDS.

Console

Utilizzando la console Amazon RDS, puoi condividere uno snapshot del cluster di database manuale con un massimo di 20 Account AWS. Puoi anche utilizzare la console per interrompere la condivisione di una snapshot manuale con uno o più account.

Per condividere uno snapshot cluster database manuale usando la console Amazon RDS

1. Accedi alla Console di gestione AWS e apri la console Amazon RDS all'indirizzo https://console.aws.amazon.com/rds/.
2. Nel riquadro di navigazione, selezionare Snapshots (Snapshot).
3. Selezionare lo snapshot manuale da condividere.
4. Per Actions (Operazioni), seleziona Share snapshot (Condividi snapshot).
5. Scegliere una delle opzioni seguenti per DB snapshot visibility (Visibilità snapshot DB).

    - Se l'origine non è crittografata, scegli Pubblico per consentire a tutti gli Account AWS di ripristinare un cluster di database dallo snapshot del cluster di database manuale oppure scegli Privato per consentire solo agli Account AWS di specificare il ripristino di un cluster di database dallo snapshot del cluster di database manuale.

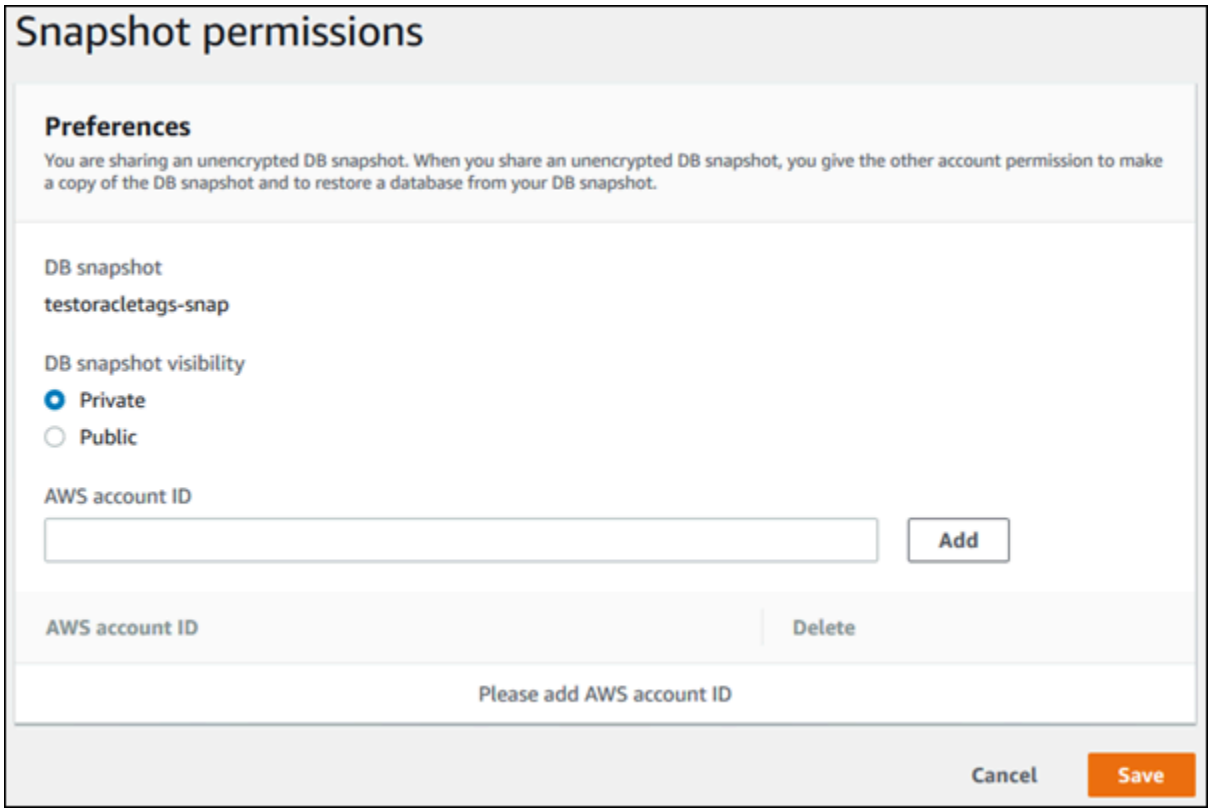
    > **Warning**
    >
    > Se imposti Visibilità snapshot DB su Pubblico, tutti gli Account AWS possono ripristinare un cluster di database dallo snapshot del cluster di database manuale e accedere ai dati. Non condividere snapshot di cluster di database manuali che contengono informazioni private impostandoli come Public (Pubblico).
    > Per ulteriori informazioni, consulta Condivisione di snapshot pubblici.

- Se l'origine è crittografata, l'opzione DB snapshot visibility (Visibilità snapshot DB) è impostata su Private (Privato), perché gli snapshot crittografati non possono essere condivisi come pubblici.

> **Note**
>
> Non è possibile condividere gli snapshot che sono stati crittografati con la AWS KMS key predefinita. Per informazioni su come ovviare al problema, consulta Condivisione di snapshot crittografati.

6. Per ID account AWS, immetti l'identificatore Account AWS per un account a cui si desidera permettere di ripristinare un cluster di database dallo snapshot manuale, quindi scegli Aggiungi. Ripeti questa operazione per includere altri identificatori di Account AWS, per un massimo di 20 Account AWS.

   In caso di errori durante l'aggiunta di un identificatore di Account AWS all'elenco di account consentiti, puoi eliminare l'account in questione dall'elenco selezionando Elimina a destra dell'identificatore di Account AWS errato.



7. Dopo aver aggiunto gli identificatori per tutti gli Account AWS a cui permettere il ripristino dello snapshot manuale, scegli Salva per salvare le modifiche.

AWS CLI

Per condividere uno snapshot del cluster di database, usa il comando `aws rds modify-db-cluster-snapshot-attribute`. Utilizza il parametro `--values-to-add` per aggiungere un elenco di ID per gli Account AWS autorizzati a ripristinare lo snapshot manuale.

Example di condividere uno snapshot con un singolo account

L'esempio seguente abilita l'identificatore di Account AWS `123456789012` per ripristinare lo snapshot del cluster di database denominato `cluster-3-snapshot`.

Per Linux, macOS o Unix:

```
aws rds modify-db-cluster-snapshot-attribute \
--db-cluster-snapshot-identifier cluster-3-snapshot \
--attribute-name restore \
--values-to-add 123456789012
```

Per Windows:

```
aws rds modify-db-cluster-snapshot-attribute ^
--db-cluster-snapshot-identifier cluster-3-snapshot ^
--attribute-name restore ^
--values-to-add 123456789012
```

Example di condividere uno snapshot con più account

L'esempio seguente abilita due identificatori di Account AWS, `111122223333` e `444455556666`, per ripristinare lo snapshot del cluster di database denominato `manual-cluster-snapshot1`.

Per Linux, macOS o Unix:

```
aws rds modify-db-cluster-snapshot-attribute \
--db-cluster-snapshot-identifier manual-cluster-snapshot1 \
--attribute-name restore \
--values-to-add {"111122223333","444455556666"}
```

Per Windows:

```
aws rds modify-db-cluster-snapshot-attribute ^
--db-cluster-snapshot-identifier manual-cluster-snapshot1 ^
```

```
--attribute-name restore ^
--values-to-add "[\"111122223333\",\"444455556666\"]"
```

> **Note**
>
> Quando usi il prompt comandi di Windows, non devi inserire le doppie virgolette (") nel codice JSON precedendole con il backslash (\).

Per elencare gli Account AWS abilitati a ripristinare uno snapshot, utilizza il comando `describe-db-cluster-snapshot-attributes` dell'interfaccia AWS CLI.

API RDS

Puoi anche condividere uno snapshot del cluster di database manuale con altri Account AWS utilizzando l'API Amazon RDS. Per eseguire questa operazione, chiama l'operazione `ModifyDBClusterSnapshotAttribute`. Specifica `restore` per `AttributeName` e utilizza il parametro `ValuesToAdd` per aggiungere l'elenco degli ID per gli Account AWS che sono autorizzati a ripristinare lo snapshot manuale.

Per rendere uno snapshot manuale pubblico e ripristinabile da tutti gli Account AWS, utilizza il valore `all`. Tuttavia, non aggiungere il valore `all` agli snapshot manuali contenenti informazioni private che non devono essere disponibili a tutti gli Account AWS. Inoltre, non è necessario specificare `all` per le snapshot crittografate, perché l'operazione di rendere pubbliche queste snapshot non è supportata.

Per elencare tutti gli Account AWS autorizzati a ripristinare uno snapshot, utilizza l'operazione API `DescribeDBClusterSnapshotAttributes`.

Condivisione di snapshot pubblici

È possibile condividere uno snapshot manuale non crittografato come pubblico, rendendolo disponibile in tutti gli Account AWS. Asicurati, quando condividi uno snapshot come pubblico, che non siano incluse nessuna delle tue informazioni personali.

Quando uno snapshot è condiviso pubblicamente, fornisce a tutti gli Account AWS l'autorizzazione sia per copiare lo snapshot sia per creare dei cluster di database da esso.

Non ti viene addebitata l'archiviazione di backup degli snapshot pubblici di proprietà di altri account. Ti verranno addebitate solo gli snapshot di tua proprietà.

Se si copia uno snapshot pubblico, si è proprietari della copia. Viene addebitato l'archiviazione di backup dello snapshot istantaneo. Se si crea un cluster di database da uno snapshot pubblico, ti viene addebitato di tale cluster di database. Per informazioni sui prezzi di Amazon Aurora, consulta la pagina dei prezzi di Aurora.

È possibile eliminare solo gli snapshot pubblici di tua proprietà. Per eliminare uno snapshot condiviso o pubblico, è necessario accedere all'Account AWS proprietario dello snapshot.

Visualizzazione di snapshot pubblici di proprietà di altri Account AWS

È possibile visualizzare snapshot pubblici di proprietà di altri account in una particolare Regione AWS nella scheda Pubblico della pagina Snapshot della console Amazon RDS. Gli snapshot (quelli di proprietà del tuo account) non vengono visualizzati in questa scheda.

Per visualizzare gli snapshot pubblici

1. Apri la console di Amazon RDS all'indirizzo https://console.aws.amazon.com/rds/.
2. Nel pannello di navigazione, scegli Snapshots (Snapshot).
3. Scegliere la scheda Public (Pubblico).

   Vengono visualizzati gli snapshot pubblici. È possibile vedere quale account possiede uno snapshot pubblico nella colonna Owner (Proprietario).

   > **Note**
   >
   > Potrebbe essere necessario modificare le preferenze della pagina, selezionando l'icona a forma di ingranaggio in alto a destra dell'elenco Public snapshots (Snapshot pubblici), per visualizzare questa colonna.

Visualizzazione degli snapshot pubblici

È possibile utilizzare il seguente comando AWS CLI (solo Unix) per visualizzare gli snapshot pubblici di proprietà dell'Account AWS in una determinata Regione AWS.

```
aws rds describe-db-cluster-snapshots --snapshot-type public --include-public |
 grep account_number
```

Se si dispone di snapshot pubblici, l'output restituito è simile all'esempio seguente.

```
"DBClusterSnapshotArn": "arn:aws:rds:us-west-2:123456789012:cluster-
snapshot:myclustersnapshot1",
"DBClusterSnapshotArn": "arn:aws:rds:us-west-2:123456789012:cluster-
snapshot:myclustersnapshot2",
```

Condivisione di snapshot pubblici da versioni obsolete del motore di database

I processi di ripristino o copia di snapshot pubblici da versioni obsolete del motore di database non sono supportati. I seguenti passaggi consentono di rendere lo snapshot pubblico esistente non supportato disponibile per il ripristino o la copia:

1. Contrassegnare lo snapshot come privato.
2. Ripristinare lo snapshot:
3. Aggiornare il cluster di database ripristinato a una versione supportata del motore.
4. Creare un nuovo snapshot.
5. Condividere nuovamente lo snapshot pubblicamente.

Condivisione di snapshot crittografati

Puoi condividere snapshot di cluster di database che sono state crittografati in stato inattivo usando l'algoritmo di crittografia AES-256, come descritto in Crittografia delle risorse Amazon Aurora.

Per la condivisione di snapshot crittografati vigono le seguenti restrizioni:

- Non puoi condividere snapshot crittografati come pubblici.
- Non è possibile condividere uno snapshot crittografato con la chiave KMS predefinita dell'Account AWS che l'ha condiviso.

  Per ulteriori informazioni sulla gestione delle chiavi AWS KMS di Amazon RDS, consulta Gestione di AWS KMS key.

Per risolvere il problema della chiave KMS predefinita, esegui le seguenti attività:

1. Crea una chiave gestita dal cliente e concedine l'accesso.
2. Copia e condividi l'istantanea dall'account di origine.
3. Copia lo snapshot condiviso nell'account di destinazione.

Crea una chiave gestita dal cliente e concedine l’accesso

Per prima cosa crea una chiave KMS personalizzata nella stessa istantanea crittografata Regione AWS del cluster di database. Durante la creazione della chiave gestita dal cliente, concedi l’accesso ad essa per un’altra Account AWS.

Per creare una chiave gestita dal cliente e concederne l’accesso

1. Accedi a Console di gestione AWS dall’origine Account AWS.
2. Aprire la console AWS KMS all'indirizzo https://console.aws.amazon.com/kms.
3. Per modificare la Regione AWS, utilizza il Selettore di regione nell'angolo in alto a destra della pagina.
4. Nel riquadro di navigazione, scegli Chiavi gestite dal cliente.
5. Scegli Crea chiave.
6. Nella pagina Configura chiave:
    a. In Tipo di chiave, seleziona Simmetrico.
    b. In Utilizzo delle chiavi, seleziona Crittografa e decrittografa.
    c. Espandi Opzioni avanzate.
    d. In Origine materiale chiave, scegli KMS.
    e. In Regionalità, seleziona Chiave a Regione singola.
    f. Scegli Next (Successivo).
7. Sulla pagina Aggiungi etichette:
    a. In Alias, inserisci un nome visualizzato per la tua chiave KMS, ad esempio **`share-snapshot`**.
    b. (Facoltativo) Inserisci una descrizione per la chiave KMS.
    c. (Facoltativo) Aggiungi tag alla chiave KMS.
    d. Scegli Next (Successivo).
8. Nella pagina Definisci le autorizzazioni per gestire la chiave scegli Avanti.
9. Nella pagina Definisci le autorizzazioni per utilizzare la chiave:
    a. In Altro Account AWS, scegli Aggiungi un’altra Account AWS.
    b. Inserisci l’ID del Account AWS cui desideri concedere l’accesso.

Puoi concedere l’accesso a più Account AWS.

c. Scegli Next (Successivo).

10. Controlla la tua chiave KMS, quindi scegli Fine.

Copia e condividi l’istantanea dall’account di origine

Successivamente si copia lo snapshot del cluster di database di origine in un nuovo snapshot utilizzando la chiave gestita dal cliente. Quindi lo condividi con il destinatario Account AWS.

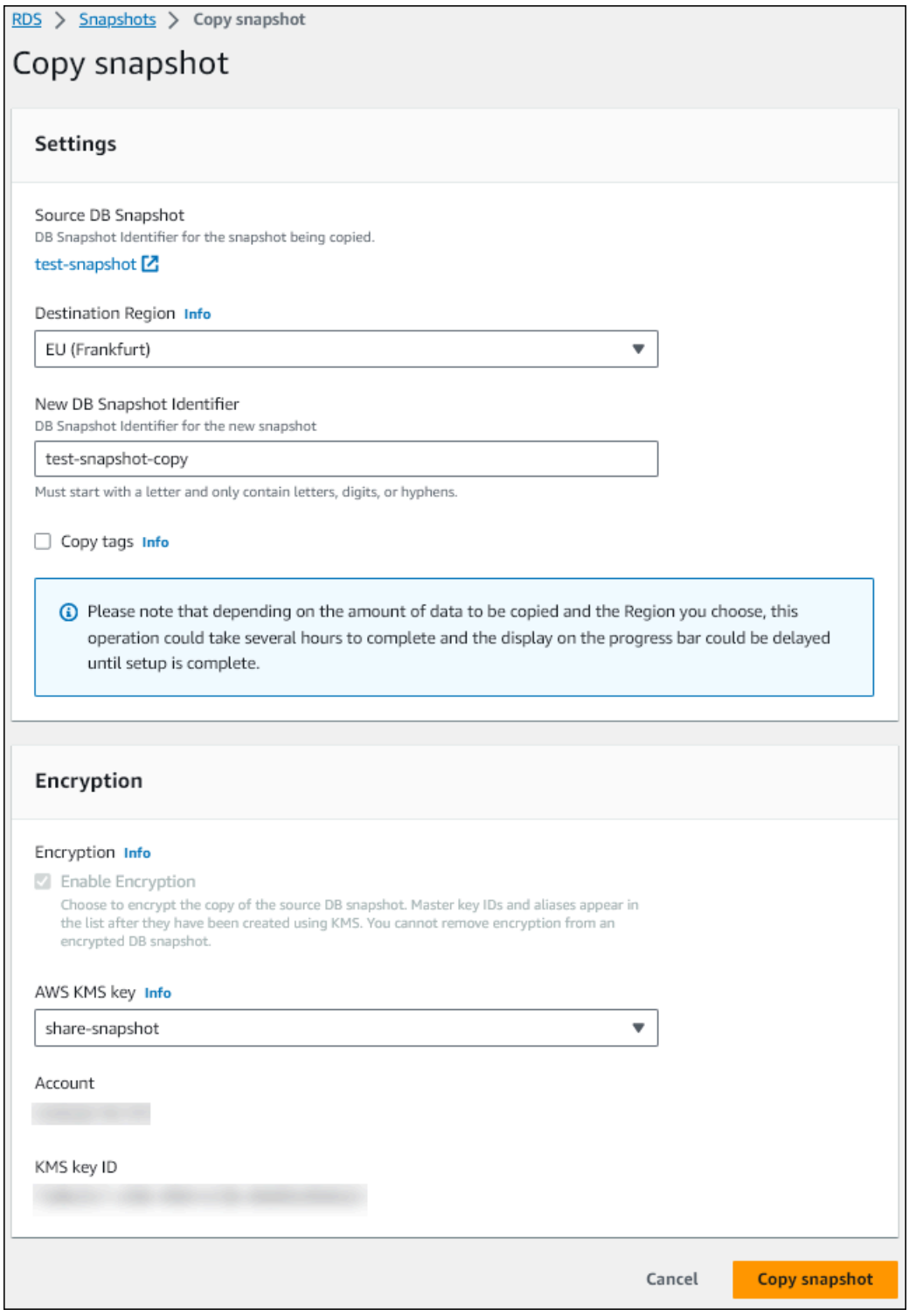
Per copiare e condividere lo snapshot

1. Accedi a Console di gestione AWS dall’origine Account AWS.
2. Apri la console di Amazon RDS all’indirizzo https://console.aws.amazon.com/rds/
3. Nel pannello di navigazione, scegli Snapshot.
4. Seleziona lo snapshot del cluster di database che desideri copiare.
5. In Operazioni, seleziona Copia snapshot.
6. Nella pagina Copia snapshot:

   a. In Regione di destinazione, scegli Regione AWS dove hai creato la chiave gestita dal cliente nella procedura precedente.

   b. Digita il nome della copia di snapshot di cluster di database in Nuovo identificatore snapshot del database.

   c. In AWS KMS key, scegli la chiave gestita dal cliente che hai creato.



d. Seleziona Copia snapshot.

7. Quando la copia dello snapshot è disponibile, selezionalo.
8. In Operazioni, seleziona Condividi snapshot.
9. Nella pagina Autorizzazioni snapshot:

a. Inserisci l'Account AWSID con cui vuoi condividere la copia dello snapshot, quindi scegli Aggiungi.
b. Scegli Salva.

Lo snapshot è condiviso.

Copia lo snapshot condiviso nell'account di destinazione

Ora puoi copiare lo snapshot condiviso nella destinazione Account AWS.

Per copiare lo snapshot condiviso

1. Accedi a Console di gestione AWS dalla destinazione Account AWS.
2. Apri la console di Amazon RDS all'indirizzo https://console.aws.amazon.com/rds/
3. Nel pannello di navigazione, scegli Snapshot.
4. Scegli la scheda Condivisi con me.
5. Seleziona lo snapshot condiviso.
6. In Operazioni, seleziona Copia snapshot.
7. Scegli le impostazioni per copiare lo snapshot come nella procedura precedente, ma utilizza una AWS KMS key che appartenga all'account di destinazione.

   Seleziona Copia snapshot.

Interruzione della condivisione degli snapshot

Per interrompere la condivisione di uno snapshot del cluster di database, rimuovi l'autorizzazione dalla destinazione Account AWS.

Console

Per interrompere la condivisione di uno snapshot del cluster di database manuale con un Account AWS

1. Accedi alla Console di gestione AWS e apri la console Amazon RDS all'indirizzo https://console.aws.amazon.com/rds/.
2. Nel pannello di navigazione, scegli Snapshot.

3. Selezionare lo snapshot manuale di cui interrompere la condivisione.
4. Scegli Actions (Operazioni) e quindi Share Snapshot (Condividi snapshot).
5. Per rimuovere l'autorizzazione per un Account AWS, scegli Elimina per l'identificatore account AWS per quell'account dall'elenco di account autorizzati.
6. Scegli Salva per salvare le modifiche.

CLI

Per rimuovere un identificatore Account AWS dall'elenco, utilizza il parametro `--values-to-remove`.

Example di interruzione della condivisione degli snapshot

L'esempio seguente impedisce all'ID Account AWS 444455556666 di ripristinare lo snapshot.

Per Linux, macOS o Unix:

```
aws rds modify-db-cluster-snapshot-attribute \
--db-cluster-snapshot-identifier manual-cluster-snapshot1 \
--attribute-name restore \
--values-to-remove 444455556666
```

Per Windows:

```
aws rds modify-db-cluster-snapshot-attribute ^
--db-cluster-snapshot-identifier manual-cluster-snapshot1 ^
--attribute-name restore ^
--values-to-remove 444455556666
```

API RDS

Per rimuovere l'autorizzazione di condivisione di un Account AWS, utilizza l'operazione `ModifyDBClusterSnapshotAttribute` con `AttributeName` impostato su `restore` e il parametro `ValuesToRemove`. Per contrassegnare uno snapshot manuale come privata, rimuovi il valore `all` dall'elenco dei valori per l'attributo `restore`.

Esportazione dei dati del cluster database in Amazon S3

È possibile esportare i dati da un cluster database Amazon Aurora in tempo reale in un bucket Amazon S3. Il processo di esportazione viene eseguito in background e non ha ripercussioni sulle prestazioni dell'istanza del cluster di database attivo.

Per impostazione predefinita, vengono esportati tutti i dati nel cluster database. Tuttavia, è possibile scegliere di esportare set specifici di database, schemi o tabelle.

Amazon Aurora clona il cluster database, estrae i dati dal clone e li archivia in un bucket Amazon S3. I dati vengono archiviati in un formato Apache Parquet compresso e coerente. I singoli file Parquet hanno in genere una dimensione compresa tra 1 e 10 MB.

Le prestazioni più veloci che si possono ottenere con l'esportazione dei dati di shapshot per Aurora MySQL versione 2 e versione 3 non si applicano all'esportazione dei dati del cluster database. Per ulteriori informazioni, consulta Esportazione dei dati dello snapshot del cluster di database in Amazon S3.

Ti viene addebitato il costo dell'esportazione dell'intero cluster di database, indipendentemente dal fatto che esporti tutti o parte dei dati. Per ulteriori informazioni, consulta la pagina Prezzi di Amazon Aurora.

Dopo l'esportazione dei dati, è possibile analizzare i dati esportati direttamente mediante strumenti quali Amazon Athena o Amazon Redshift Spectrum. Per ulteriori informazioni sull'utilizzo Athena per leggere i dati Parquet, consultaParquet SerDe in Guida per l'utente di Amazon Athena. Per ulteriori informazioni sull'utilizzo Redshift Spectrum per leggere i dati Parquet, consulta COPY da formati di dati a colonna nella Guida per gli sviluppatori di database di Amazon Redshift.

Il supporto varia a seconda delle versioni specifiche di ciascun motore di database e a seconda delle Regioni AWS. Per ulteriori informazioni sulla disponibilità di versioni e regioni per l'esportazione dei dati del cluster database in S3, consulta Regioni e motori di database Aurora supportati per l'esportazione dei dati del cluster in Amazon S3.

Per esportare i dati del cluster database in un bucket Amazon S3 puoi utilizzare il processo riportato di seguito. Per ulteriori dettagli, consulta le seguenti sezioni:

Panoramica sull'esportazione dei dati di un cluster database

1. Identifica il cluster database di cui desideri esportare i dati.
2. Configurare l'accesso al bucket Simple Storage Service (Amazon S3).

Un bucket è un container per oggetti o file Simple Storage Service (Amazon S3). Per fornire le informazioni per accedere a un bucket, attenersi alla seguente procedura:

a. Identifica il bucket S3 in cui devono essere esportati i dati del cluster database. Il bucket S3 deve trovarsi nella stessa Regione AWS del cluster database. Per ulteriori informazioni, consulta Identificazione del bucket Simple Storage Service (Amazon S3) in cui esportare.

b. Crea un ruolo AWS Identity and Access Management (IAM) che conceda all'attività di esportazione del cluster database l'accesso al bucket S3. Per ulteriori informazioni, consulta Fornire l'accesso a un bucket Simple Storage Service (Amazon S3) utilizzando un ruolo IAM.

3. Crea una AWS KMS key di crittografia simmetrica per la crittografia lato server. La chiave KMS viene utilizzata dall'attività di esportazione del cluster per impostare la crittografia lato server AWS KMS per la scrittura dei dati dell'esportazione in S3.

   La policy della chiave KMS deve includere entrambe le autorizzazioni `kms:CreateGrant` e `kms:DescribeKey`. Per ulteriori informazioni sull'uso delle chiavi KMS in Amazon Aurora, consulta Gestione di AWS KMS key.

   Inoltre, se nella policy della chiave KMS è presente un'istruzione di negazione, assicurati di escludere esplicitamente il principale del servizio AWS `export.rds.amazonaws.com`.

   È possibile utilizzare una chiave KMS all'interno dell'account AWS oppure puoi usare una chiave KMS cross-account. Per ulteriori informazioni, consulta Utilizzo di un account multiplo AWS KMS key.

4. Esporta il cluster database in Amazon S3 utilizzando la console o il comando dell'interfaccia della linea di comando `start-export-task`. Per ulteriori informazioni, consulta Creazione di attività di esportazione del cluster di database.

5. Per accedere ai dati esportati nel bucket Simple Storage Service (Amazon S3), consulta Caricamento, download e gestione di oggetti nella Guida per l'utente di Amazon Simple Storage Service.

Nelle sezioni seguenti sono fornite informazioni su come configurare, esportare, monitorare, annullare e risolvere i problemi relativi alle attività di esportazione di cluster di database.

Argomenti

- Considerazioni per le esportazioni del cluster di database

- Configurazione dell'accesso a un bucket Simple Storage Service (Amazon S3)
- Creazione di attività di esportazione del cluster di database
- Monitoraggio delle attività di esportazione del cluster database
- Annullamento di un'attività di esportazione del cluster database
- Risoluzione dei problemi delle esportazioni del cluster di database

Considerazioni per le esportazioni del cluster di database

Utilizza le sezioni seguenti per scoprire le limitazioni, le convenzioni di denominazione dei file e la conversione e l'archiviazione dei dati durante l'esportazione dei dati dei cluster di database in Amazon S3.

Argomenti

- Limitazioni
- Convenzione di denominazione file
- Formato di conversione e archiviazione dei dati

Limitazioni

L'esportazione dei dati del cluster database in Amazon S3 presenta le seguenti limitazioni:

- Non è possibile eseguire contemporaneamente più attività di esportazione per lo stesso cluster database. Ciò è valido sia per le esportazioni totali sia per le esportazioni parziali.
- È possibile avere fino a cinque attività di esportazione di snapshot di database simultanee in esecuzione per ogni Account AWS.
- I cluster database Aurora Serverless v1 non supportano le esportazioni in S3.
- Aurora MySQL e Aurora PostgreSQL supportano le esportazioni in S3 solo per la modalità del motore allocato.
- Le esportazioni in S3 non supportano i prefissi S3 contenenti i due punti (:).
- I seguenti caratteri nel percorso del file S3 vengono convertiti in caratteri di sottolineatura (_) durante l'esportazione:

```
\ ` " (space)
```

- Se un database, uno schema o una tabella contiene caratteri diversi da quelli riportati di seguito, l'esportazione parziale non è supportata. Tuttavia, è possibile esportare l'intero cluster database.
  - Lettere latine (A–Z)
  - Numeri (0–9)
  - Simbolo del dollaro ($)
  - Carattere di sottolineatura (_)
- Gli spazi ( ) e alcuni caratteri non sono supportati nei nomi delle colonne delle tabelle del database. Le tabelle con i seguenti caratteri nei nomi delle colonne vengono ignorate durante l'esportazione:

  ```
  , ; { } ( ) \n \t = (space)
  ```

- Le tabelle con barre (/) nei rispettivi nomi vengono ignorate durante l'esportazione.
- Le tabelle temporanee e non registrate di Aurora PostgreSQL vengono ignorate durante l'esportazione.
- Se i dati contengono un oggetto di grandi dimensioni, ad esempio un BLOB o un CLOB, vicino o superiore a 500 MB, l'esportazione non riesce.
- Se una tabella contiene una riga di grandi dimensioni, vicine o superiori a 2 GB, la tabella viene ignorata durante l'esportazione.
- Per le esportazioni parziali, l'elenco `ExportOnly` ha una dimensione massima di 200 KB.
- Si consiglia vivamente di utilizzare un nome univoco per ogni attività di esportazione. Se non utilizzi un nome di attività univoco, potresti ricevere il seguente messaggio di errore:

  ExportTaskAlreadyExistsFault: si è verificato un errore (ExportTaskAlreadyExists) durante la chiamata all'operazione StartExportTask: l'attività di esportazione con ID *xxxxx* esiste già.
- Poiché alcune tabelle potrebbero essere ignorate, ti consigliamo di verificare il numero di righe e tabelle nei dati dopo l'esportazione.

Convenzione di denominazione file

I dati esportati per tabelle specifiche vengono memorizzati nel formato *base_prefix*/*files*, dove il prefisso di base è il seguente:

```
export_identifier/database_name/schema_name.table_name/
```

Per esempio:

```
export-1234567890123-459/rdststcluster/mycluster.DataInsert_7ADB5D19965123A2/
```

I file di output utilizzano la seguente convenzione di denominazione, in cui *partition_index* è alfanumerico:

```
partition_index/part-00000-random_uuid.format-based_extension
```

Per esempio:

```
1/part-00000-c5a881bb-58ff-4ee6-1111-b41ecff340a3-c000.gz.parquet
    a/part-00000-d7a881cc-88cc-5ab7-2222-c41ecab340a4-c000.gz.parquet
```

La convenzione di denominazione file è soggetta a modifiche. Pertanto, quando usi le tabelle di destinazione ti consigliamo di leggere tutto quanto riportato all'interno del prefisso di base della tabella.

Formato di conversione e archiviazione dei dati

Quando esporti un cluster di database in un bucket Amazon S3, Amazon Aurora converte, esporta e archivia i dati nel formato Parquet. Per ulteriori informazioni, consulta Conversione dei dati durante l'esportazione in un bucket Simple Storage Service (Amazon S3).

Configurazione dell'accesso a un bucket Simple Storage Service (Amazon S3)

Individua il bucket Amazon S3 e fornisci all'attività di esportazione del cluster database l'autorizzazione per accedervi.

Argomenti

- Identificazione del bucket Simple Storage Service (Amazon S3) in cui esportare
- Fornire l'accesso a un bucket Simple Storage Service (Amazon S3) utilizzando un ruolo IAM
- Utilizzo di un bucket Simple Storage Service (Amazon S3) multiaccount

Identificazione del bucket Simple Storage Service (Amazon S3) in cui esportare

Identifica il bucket Amazon S3 in cui esportare il cluster database. Utilizzare un bucket S3 esistente o crearne uno nuovo.

> **Note**
>
> Il bucket S3 deve trovarsi nella stessa AWS regione del cluster DB.

Per ulteriori informazioni sull'utilizzo dei bucket Simple Storage Service (Amazon S3), vedere quanto segue in Guida per l'utente di Amazon Simple Storage Service:

- Come visualizzare le proprietà di un bucket S3?
- In che modo si abilita la crittografia di default per un bucket Amazon S3?
- Come creare un bucket S3?

Fornire l'accesso a un bucket Simple Storage Service (Amazon S3) utilizzando un ruolo IAM

Prima di esportare i dati del cluster database in Amazon S3, fornisci l'autorizzazione di accesso in scrittura alle attività di esportazione al bucket Amazon S3.

Per concedere l'autorizzazione, crea una policy IAM che fornisca accesso al bucket, crea un ruolo IAM e collega la policy al ruolo. Successivamente assegna il ruolo IAM all'attività di esportazione del cluster database.

> **Important**
>
> Se prevedi di utilizzare il Console di gestione AWS per esportare il tuo cluster DB, puoi scegliere di creare automaticamente la policy IAM e il ruolo quando esporti il cluster DB. Per istruzioni, consulta Creazione di attività di esportazione del cluster di database.

Per fornire alle attività l'accesso ad Amazon S3

1. Creare una policy IAM Questa policy fornisce le autorizzazioni per bucket e oggetti che consentono all'attività di esportazione del cluster database l'accesso ad Amazon S3.

Includi nella policy le seguenti operazioni obbligatorie per consentire il trasferimento dei file da Amazon Aurora a un bucket S3:

- `s3:PutObject*`
- `s3:GetObject*`
- `s3:ListBucket`
- `s3:DeleteObject*`
- `s3:GetBucketLocation`

Includi nella policy le seguenti risorse per identificare il bucket S3 e gli oggetti nel bucket. Il seguente elenco di risorse mostra il formato Amazon Resource Name (ARN) per l'accesso a Amazon S3.

- `arn:aws:s3:::amzn-s3-demo-bucket`
- `arn:aws:s3:::amzn-s3-demo-bucket/*`

Per ulteriori informazioni sulla creazione di una policy IAM per Amazon Aurora, consulta Creazione e utilizzo di una policy IAM per l'accesso al database IAM. Consulta anche il Tutorial: Creare e collegare la prima policy gestita dal cliente nella Guida per l'utente di IAM.

Il AWS CLI comando seguente crea una policy IAM denominata `ExportPolicy` con queste opzioni. Concede l'accesso a un bucket denominato *amzn-s3-demo-bucket*.

> **Note**
>
> Dopo aver creato la policy, prendere nota del relativo ARN. Per la fase successiva, in cui si associa la policy a un ruolo IAM, è necessario l'ARN.

```
aws iam create-policy  --policy-name ExportPolicy --policy-document '{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "ExportPolicy",
            "Effect": "Allow",
            "Action": [
```

```
                "s3:PutObject*",
                "s3:ListBucket",
                "s3:GetObject*",
                "s3:DeleteObject*",
                "s3:GetBucketLocation"
            ],
            "Resource": [
                "arn:aws:s3:::amzn-s3-demo-bucket",
                "arn:aws:s3:::amzn-s3-demo-bucket/*"
            ]
        }
    ]
}'
```

2. Crea un ruolo IAM in modo che Aurora possa assumere questo ruolo IAM per tuo conto per accedere ai bucket Amazon S3. Per ulteriori informazioni, consulta la pagina relativa alla creazione di un ruolo per delegare le autorizzazioni a un utente IAM nella Guida per l'utente IAM.

   L'esempio seguente mostra l'utilizzo del AWS CLI comando per creare un ruolo denominato`rds-s3-export-role`.

```
aws iam create-role  --role-name rds-s3-export-role  --assume-role-policy-document
 '{
    "Version": "2012-10-17",
    "Statement": [
      {
        "Effect": "Allow",
        "Principal": {
           "Service": "export.rds.amazonaws.com"
         },
        "Action": "sts:AssumeRole"
      }
    ]
  }'
```

3. Collegare la policy IAM al ruolo IAM creato.

   Il AWS CLI comando seguente collega la politica creata in precedenza al ruolo denominato`rds-s3-export-role`. Sostituire *`your-policy-arn`* con l'ARN della policy annotato nella fase precedente.

```
aws iam attach-role-policy  --policy-arn your-policy-arn  --role-name rds-s3-
export-role
```

Utilizzo di un bucket Simple Storage Service (Amazon S3) multiaccount

Puoi utilizzare i bucket S3 su più account. AWS Per ulteriori informazioni, consulta Utilizzo di un bucket Simple Storage Service (Amazon S3) multiaccount.

Creazione di attività di esportazione del cluster di database

Crea attività di esportazione per esportare i dati del cluster di database Aurora in un bucket Amazon S3. È possibile avere fino a cinque attività di esportazione di cluster database simultanee in esecuzione per ogni Account AWS.

> **Note**
>
> L'esportazione dei dati del cluster database può richiedere qualche minuto a seconda del tipo e delle dimensioni del database. L'attività di esportazione clona e ridimensiona innanzitutto l'intero database prima di estrarre i dati su Amazon S3. Lo stato di avanzamento dell'attività durante questa fase viene visualizzato come Avvio. Quando l'attività passa all'esportazione dei dati in S3, lo stato di avanzamento diventa In progress (In corso).
> Il tempo necessario per completare l'esportazione dipende dai dati memorizzati nel database. Ad esempio, le tabelle con chiave primaria numerica o colonne indice ben distribuite esporteranno più velocemente. Le tabelle che non contengono una colonna adatta al partizionamento e le tabelle con un solo indice su una colonna basata su stringhe richiedono più tempo perché l'esportazione utilizza un processo a thread singolo più lento.

È possibile esportare i dati del cluster database in Amazon S3 utilizzando la Console di gestione AWS, AWS CLI o l'API RDS.

Se usi una funzione Lambda per esportare i dati del cluster database, aggiungi l'operazione `kms:DescribeKey` alla policy della funzione Lambda. Per ulteriori informazioni, consulta Autorizzazioni di AWS Lambda.

Console

L'opzione Export to Amazon S3 (Esporta in Amazon S3) viene visualizzata solo per i cluster database che possono essere esportati in Amazon S3. Un cluster database potrebbe non essere disponibile per l'esportazione a causa dei seguenti motivi:

- Il motore del database non è supportato per l'esportazione S3.
- La versione del cluster database non è supportata per l'esportazione in S3.
- L'esportazione in S3 non è supportata nella Regione AWS in cui è stato creato il cluster database.

Per esportare i dati del cluster database

1. Accedi alla Console di gestione AWS e apri la console Amazon RDS all'indirizzo https://console.aws.amazon.com/rds/.
2. Nel pannello di navigazione, scegliere Databases (Database).
3. Scegli il cluster database di cui desideri esportare i dati.
4. Per Actions (Operazioni), scegli Export to Amazon S3 (Esporta in Simple Storage Service (Amazon S3).

   Viene visualizzata la finestra Export to Amazon S3 (Esporta in Simple Storage Service (Amazon S3)).
5. Per Export identifier (Identificatore di esportazione), immettere un nome per identificare l'attività di esportazione. Questo valore viene utilizzato anche per il nome del file creato nel bucket S3.
6. Scegli i dati da esportare:

   - Scegli All (Tutto) per esportare tutti i dati del cluster database.
   - Scegli Partial (Parziale) per esportare parti specifiche del cluster database. Per identificare le parti del cluster da esportare, immetti uno o più database, schemi o tabelle per Identifiers (Identificatori), separati da spazi.

     Utilizza il seguente formato:

     ```
     database[.schema][.table] database2[.schema2][.table2] ... databasen[.schemaN]
     [.tablen]
     ```

     Per esempio:

```
mydatabase mydatabase2.myschema1 mydatabase2.myschema2.mytable1
 mydatabase2.myschema2.mytable2
```

7. Per S3 bucket (Bucket S3), scegliere il bucket in cui esportare.

   Per assegnare i dati esportati a un percorso di cartella nel bucket S3, immettere il percorso opzionale per S3 prefix (Prefisso S3).

8. Per il ruolo IAM, scegliere un ruolo che conceda l'accesso in scrittura al bucket S3 scelto o creare un nuovo ruolo.

   - Se è stato creato un ruolo seguendo le fasi in Fornire l'accesso a un bucket Simple Storage Service (Amazon S3) utilizzando un ruolo IAM, scegliere tale ruolo.
   - Se non è stato creato un ruolo che fornisce l'accesso in scrittura al bucket S3 scelto, scegli Create a new role (Crea un nuovo ruolo) per creare automaticamente il ruolo. Immettere quindi un nome per il ruolo nel nome del ruolo IAM.

9. Per KMS key (Chiave KMS), immetti l'ARN per la chiave da utilizzare per crittografare i dati esportati.
10. Scegliere Export to Amazon S3 (Esporta in Simple Storage Service (Amazon S3)).

AWS CLI

Per esportare i dati del cluster database in Amazon S3 tramite AWS CLI, utilizza il comando start-export-task con le seguenti opzioni richieste:

- `--export-task-identifier`
- `--source-arn` - Il nome della risorsa Amazon (ARN) del cluster database
- `--s3-bucket-name`
- `--iam-role-arn`
- `--kms-key-id`

Negli esempi seguenti, l'attività di esportazione è denominata *`my-cluster-export`*, che esporta i dati in un bucket S3 chiamato *`amzn-s3-demo-destination-bucket`*.

Example

Per Linux, macOS o Unix:

```
aws rds start-export-task \
    --export-task-identifier my-cluster-export \
    --source-arn arn:aws:rds:us-west-2:123456789012:cluster:my-cluster \
    --s3-bucket-name amzn-s3-demo-destination-bucket \
    --iam-role-arn iam-role \
    --kms-key-id my-key
```

Per Windows:

```
aws rds start-export-task ^
    --export-task-identifier my-DB-cluster-export ^
    --source-arn arn:aws:rds:us-west-2:123456789012:cluster:my-cluster ^
    --s3-bucket-name amzn-s3-demo-destination-bucket ^
    --iam-role-arn iam-role ^
    --kms-key-id my-key
```

Di seguito è riportato un output di esempio.

```
{
    "ExportTaskIdentifier": "my-cluster-export",
    "SourceArn": "arn:aws:rds:us-west-2:123456789012:cluster:my-cluster",
    "S3Bucket": "amzn-s3-demo-destination-bucket",
    "IamRoleArn": "arn:aws:iam:123456789012:role/ExportTest",
    "KmsKeyId": "my-key",
    "Status": "STARTING",
    "PercentProgress": 0,
    "TotalExtractedDataInGB": 0,
}
```

Per fornire un percorso di cartella nel bucket S3 per l'esportazione del cluster database, includi l'opzione `--s3-prefix` nel comando start-export-task.

API RDS

Per esportare i dati del cluster database in Amazon S3 utilizzando l'API Amazon RDS, usa l'operazione StartExportTask con i seguenti parametri richiesti:

- `ExportTaskIdentifier`
- `SourceArn` - Il nome della risorsa Amazon del cluster di database
- `S3BucketName`
- `IamRoleArn`

- `KmsKeyId`

Monitoraggio delle attività di esportazione del cluster database

È possibile monitorare le esportazioni del cluster database utilizzando la Console di gestione AWS, AWS CLI o l'API RDS.

Console

Per monitorare le esportazioni del cluster database

1. Accedi alla Console di gestione AWS e apri la console Amazon RDS all'indirizzo https://console.aws.amazon.com/rds/.
2. Nel riquadro di navigazione scegli Exports in Amazon S3 (Esporta in Amazon S3).

   Le esportazioni del cluster database sono indicate nella colonna Source type (Tipo di origine). Lo stato dell'esportazione è visualizzato nella colonna Status (Stato).
3. Per visualizzare informazioni dettagliate su un'esportazione del cluster database specifica, scegli l'attività di esportazione.

AWS CLI

Per monitorare le attività di esportazione del cluster database utilizzando AWS CLI, usa il comando describe-export-tasks.

Nell'esempio seguente viene illustrato come visualizzare le informazioni correnti su tutte le esportazioni del cluster database.

Example

```
aws rds describe-export-tasks

{
    "ExportTasks": [
        {
            "Status": "CANCELED",
            "TaskEndTime": "2022-11-01T17:36:46.961Z",
            "S3Prefix": "something",
            "S3Bucket": "amzn-s3-demo-bucket",
            "PercentProgress": 0,
```

```
            "KmsKeyId": "arn:aws:kms:us-west-2:123456789012:key/K7MDENG/
bPxRfiCYEXAMPLEKEY",
            "ExportTaskIdentifier": "anewtest",
            "IamRoleArn": "arn:aws:iam::123456789012:role/export-to-s3",
            "TotalExtractedDataInGB": 0,
            "SourceArn": "arn:aws:rds:us-west-2:123456789012:cluster:parameter-groups-
test"
        },
{
            "Status": "COMPLETE",
            "TaskStartTime": "2022-10-31T20:58:06.998Z",
            "TaskEndTime": "2022-10-31T21:37:28.312Z",
            "WarningMessage": "{\"skippedTables\":[],\"skippedObjectives\":[],\"general
\":[{\"reason\":\"FAILED_TO_EXTRACT_TABLES_LIST_FOR_DATABASE\"}]}",
            "S3Prefix": "",
            "S3Bucket": "amzn-s3-demo-bucket1",
            "PercentProgress": 100,
            "KmsKeyId": "arn:aws:kms:us-west-2:123456789012:key/2Zp9Utk/
h3yCo8nvbEXAMPLEKEY",
            "ExportTaskIdentifier": "thursday-events-test",
            "IamRoleArn": "arn:aws:iam::123456789012:role/export-to-s3",
            "TotalExtractedDataInGB": 263,
            "SourceArn": "arn:aws:rds:us-
west-2:123456789012:cluster:example-1-2019-10-31-06-44"
        },
        {
            "Status": "FAILED",
            "TaskEndTime": "2022-10-31T02:12:36.409Z",
            "FailureCause": "The S3 bucket amzn-s3-demo-bucket2 isn't located in the
 current AWS Region. Please, review your S3 bucket name and retry the export.",
            "S3Prefix": "",
            "S3Bucket": "amzn-s3-demo-bucket2",
            "PercentProgress": 0,
            "KmsKeyId": "arn:aws:kms:us-west-2:123456789012:key/2Zp9Utk/
h3yCo8nvbEXAMPLEKEY",
            "ExportTaskIdentifier": "wednesday-afternoon-test",
            "IamRoleArn": "arn:aws:iam::123456789012:role/export-to-s3",
            "TotalExtractedDataInGB": 0,
            "SourceArn": "arn:aws:rds:us-
west-2:123456789012:cluster:example-1-2019-10-30-06-45"
        }
    ]
}
```

Per visualizzare informazioni su un'attività di esportazione specifica, includi l'opzione `--export-task-identifier` nel comando `describe-export-tasks`. Per filtrare l'output, includere l'opzione `--Filters`. Per ulteriori opzioni, consultare il comando describe-export-tasks.

API RDS

Per visualizzare informazioni sulle esportazioni del cluster database utilizzando l'API Amazon RDS, usa l'operazione DescribeExportTasks.

Per tenere traccia del completamento del flusso di lavoro di esportazione o per attivare un altro flusso di lavoro, è possibile sottoscrivere gli argomenti del Servizio di notifica semplice Amazon. Per ulteriori informazioni su Amazon SNS, consulta Utilizzo della notifica degli eventi di Amazon RDS.

Annullamento di un'attività di esportazione del cluster database

È possibile annullare un'attività di esportazione del cluster database utilizzando la Console di gestione AWS, AWS CLI o l'API RDS.

> **Note**
>
> L'annullamento di un'attività di esportazione non rimuove i dati esportati in Amazon S3. Per informazioni su come eliminare i dati utilizzando la console, consultare Come eliminare oggetti da un bucket S3? Per eliminare i dati utilizzando l'interfaccia della riga di comando (CLI), utilizzare il comando delete-object.

Console

Per annullare un'attività di esportazione del cluster database

1. Accedi alla Console di gestione AWS e apri la console Amazon RDS all'indirizzo https://console.aws.amazon.com/rds/.
2. Nel riquadro di navigazione scegli Exports in Amazon S3 (Esporta in Amazon S3).

   Le esportazioni del cluster database sono indicate nella colonna Source type (Tipo di origine). Lo stato dell'esportazione è visualizzato nella colonna Status (Stato).
3. Scegli l'attività di esportazione che vuoi annullare.
4. Seleziona Cancel (Annulla).
5. Scegli Cancel export task (Annulla attività di esportazione) nella pagina di conferma.

AWS CLI

Per annullare un'attività di esportazione utilizzando l'AWS CLI, usa il comando cancel-export-task. Il comando richiede l'opzione `--export-task-identifier`.

Example

```
aws rds cancel-export-task --export-task-identifier my-export
{
    "Status": "CANCELING",
    "S3Prefix": "",
    "S3Bucket": "amzn-s3-demo-bucket",
    "PercentProgress": 0,
    "KmsKeyId": "arn:aws:kms:us-west-2:123456789012:key/K7MDENG/bPxRfiCYEXAMPLEKEY",
    "ExportTaskIdentifier": "my-export",
    "IamRoleArn": "arn:aws:iam::123456789012:role/export-to-s3",
    "TotalExtractedDataInGB": 0,
    "SourceArn": "arn:aws:rds:us-west-2:123456789012:cluster:export-example-1"
}
```

API RDS

Per annullare un'attività di esportazione utilizzando l'API Amazon RDS, usa l'operazione CancelExportTask con il parametro `ExportTaskIdentifier`.

Risoluzione dei problemi delle esportazioni del cluster di database

Utilizza le seguenti sezioni per la risoluzione dei problemi relativi ai messaggi di errore e agli errori di autorizzazione PostgreSQL per le attività di esportazione di cluster di database in Amazon S3.

Messaggi di errore per le attività di esportazione di Amazon S3

Nella tabella seguente vengono descritti i messaggi restituiti quando le attività di esportazione di Amazon S3 non riescono.

Messaggio di errore	Descrizione
Impossibile trovare o accedere al cluster database di origine: [nome cluster]	Il cluster database di origine non può essere clonato.

Messaggio di errore	Descrizione
Si è verificato un errore interno sconosciuto.	L'elaborazione della richiesta non è riuscita a causa di un errore, un'eccezione o un guasto interno sconosciuto.
Si è verificato un errore interno sconosciuto durante la scrittura dei metadati dell'attività di esportazione nel bucket S3 [nome bucket].	L'elaborazione della richiesta non è riuscita a causa di un errore, un'eccezione o un guasto interno sconosciuto.
L'esportazione RDS non è riuscita a scrivere i metadati dell'attività di esportazione perché non può assumere il ruolo IAM [ruolo ARN].	L'attività di esportazione assume il ruolo IAM per verificare se è consentito scrivere metadati nel bucket S3. Se l'attività non può assumere il ruolo IAM, non riesce.
L'esportazione RDS non è riuscita a scrivere i metadati dell'attività di esportazione nel bucket S3 [nome bucket] utilizzando il ruolo IAM [ruolo ARN] con la chiave KMS [ID chiave]. Codice di errore: [codice di errore]	Mancano una o più autorizzazioni, quindi l'attività di esportazione non può accedere al bucket S3. Questo messaggio di errore viene generato quando si riceve uno dei seguenti codici di errore: • `AWSSecurityTokenServiceException` con il codice di errore `AccessDenied` • `AmazonS3Exception` con il codice di errore`NoSuchBucket` ,`AccessDenied` ,`KMS.KMSInvalidStateException` ,`403 Forbidden` , oppure `KMS.DisabledException` Questi codici di errore indicano che le impostazioni non sono configurate correttamente per il ruolo IAM, il bucket S3 o la chiave KMS.
Il ruolo IAM [ruolo ARN] non è autorizzato a chiamare [azione S3] sul bucket S3 [nome bucket]. Controlla le tue autorizzazioni e riprova l'esportazione.	La policy IAM è configurata in modo errato. L'autorizzazione per l'azione S3 specifica sul bucket S3 è mancante e questa condizione causa l'esito negativo dell'attività di esportazione.

Messaggio di errore	Descrizione
Controllo chiave KMS non riuscito. Controlla le credenziali sulla tua chiave KMS e riprova.	Controllo delle credenziali della chiave KMS non riuscito.
Controllo delle credenziali S3 non riuscito. Controlla le autorizzazioni per il bucket S3 e la policy IAM.	Il controllo delle credenziali S3 non è riuscito.
Il bucket S3 [nome bucket] non è valido. Non si trova nella corrente Regione AWS oppure non esiste. Esamina il nome del bucket S3 e riprova l'esporta zione.	Bucket S3 non è valido.
Il bucket S3 [nome bucket] non si trova nella corrente Regione AWS. Esamina il nome del bucket S3 e riprova l'esporta zione.	Il bucket S3 è nella Regione AWS sbagliata.

Risoluzione degli errori di autorizzazione PostgreSQL

Quando si esportano i database PostgreSQL in Simple Storage Service (Amazon S3), è possibile che venga visualizzato un errore `PERMISSIONS_DO_NOT_EXIST` che indica che alcune tabelle sono state ignorate. Questo errore si verifica in genere quando l'utente con privilegi avanzati che hai specificato durante la creazione del cluster database, non dispone delle autorizzazioni per accedere alle tabelle.

Per risolvere questo errore, eseguire il comando seguente:

```
GRANT ALL PRIVILEGES ON ALL TABLES IN SCHEMA schema_name TO superuser_name
```

Per ulteriori informazioni sui privilegi utente con privilegi avanzati, vedere Privilegi dell'account utente master.

Esportazione dei dati dello snapshot del cluster di database in Amazon S3

È possibile esportare i dati dello snapshot del cluster di database in un bucket Amazon S3. Il processo di esportazione viene eseguito in background e non ha ripercussioni sulle prestazioni dell'istanza del cluster di database attivo.

Quando si esporta uno snapshot del cluster di database, Amazon Aurora estrae i dati dallo snapshot e li archivia in un bucket Amazon S3. È possibile esportare snapshot manuali e snapshot automatici di sistema. Per impostazione predefinita, vengono esportati tutti i dati nello snapshot. Tuttavia, è possibile scegliere di esportare set specifici di database, schemi o tabelle.

> **Note**
>
> L'esportazione di dati da uno snapshot di cluster di database richiede il ripristino dello snapshot. I tempi di ripristino sono influenzati da vari fattori, come la quantità di traffico di rete che una Regione AWS riceve rispetto alla larghezza di banda disponibile. In caso di aumento improvviso del traffico, è possibile che si verifichino tempi di completamento più lunghi del previsto.
> Un'alternativa per ridurre i tempi di esportazione S3 per i database Aurora è l'esportazione in tempo reale di cluster di database in S3. L'esportazione di cluster di database ha tempi di avvio più brevi rispetto all'esportazione di snapshot di database, perché non è necessario ripristinare uno snapshot. Per ulteriori informazioni, consulta Esportazione dei dati del cluster database in Amazon S3.

I dati vengono archiviati in un formato Apache Parquet compresso e coerente. I singoli file Parquet hanno in genere una dimensione compresa tra 1 e 10 MB.

Dopo l'esportazione dei dati, è possibile analizzare i dati esportati direttamente mediante strumenti quali Amazon Athena o Amazon Redshift Spectrum. Per ulteriori informazioni sull'utilizzo Athena per leggere i dati Parquet, consultaParquet SerDe in Guida per l'utente di Amazon Athena. Per ulteriori informazioni sull'utilizzo Redshift Spectrum per leggere i dati Parquet, consulta COPY da formati di dati a colonna nella Guida per gli sviluppatori di database di Amazon Redshift.

Il supporto varia a seconda delle versioni specifiche di ciascun motore di database e a seconda delle Regioni AWS. Per ulteriori informazioni sulla disponibilità di versioni e regioni in caso di esportazione

dei dati di snapshot di cluster di database in S3, consulta Regioni e motori di database Aurora supportati per l'esportazione dei dati dello snapshot in Amazon S3.

Per esportare i dati dello snapshot DB in un bucket Simple Storage Service (Amazon S3) puoi utilizzare il processo riportato di seguito. Per ulteriori dettagli, consulta le seguenti sezioni:

Panoramica sull'esportazione dei dati degli snapshot

1. Identificare lo snapshot da esportare.

   Utilizzare uno snapshot automatico o manuale esistente oppure creare uno snapshot manuale di un'istanza database.

2. Configurare l'accesso al bucket Simple Storage Service (Amazon S3).

   Un bucket è un container per oggetti o file Simple Storage Service (Amazon S3). Per fornire le informazioni per accedere a un bucket, attenersi alla seguente procedura:

   a. Identificare il bucket S3 in cui deve essere esportato lo snapshot. Il bucket S3 deve trovarsi nella stessa regione AWS dello snapshot. Per ulteriori informazioni, consulta Identificazione del bucket Simple Storage Service (Amazon S3) in cui esportare.

   b. Creare un ruolo AWS Identity and Access Management (IAM) che conceda all'attività di esportazione dello snapshot l'accesso al bucket S3. Per ulteriori informazioni, consulta Fornire l'accesso a un bucket Simple Storage Service (Amazon S3) utilizzando un ruolo IAM.

3. Crea una AWS KMS key di crittografia simmetrica per la crittografia lato server. La chiave KMS viene utilizzata dall'attività di esportazione dello snapshot per impostare la crittografia lato server AWS KMS durante la scrittura dei dati di esportazione in S3.

   La policy della chiave KMS deve includere entrambe le autorizzazioni `kms:CreateGrant` e `kms:DescribeKey`. Per ulteriori informazioni sull'uso delle chiavi KMS in Amazon Aurora, consulta Gestione di AWS KMS key.

   Inoltre, se nella policy della chiave KMS è presente un'istruzione di negazione, assicurati di escludere esplicitamente il principale del servizio AWS `export.rds.amazonaws.com`.

   È possibile utilizzare una chiave KMS all'interno dell'account AWS oppure puoi usare una chiave KMS cross-account. Per ulteriori informazioni, consulta Utilizzo di un account multiplo AWS KMS key.

4. Esportare lo snapshot in Simple Storage Service (Amazon S3) utilizzando la console o il comando CLI `start-export-task`. Per ulteriori informazioni, consulta Creazione di attività di esportazione di snapshot.
5. Per accedere ai dati esportati nel bucket Simple Storage Service (Amazon S3), consulta Caricamento, download e gestione di oggetti nella Guida per l'utente di Amazon Simple Storage Service.

Nelle sezioni seguenti sono fornite informazioni su come configurare, esportare, monitorare, annullare e risolvere i problemi relativi alle attività di esportazione di snapshot di cluster di database.

Argomenti

- Considerazioni sulle esportazioni di snapshot di cluster di database
- Configurazione dell'accesso a un bucket Simple Storage Service (Amazon S3)
- Creazione di attività di esportazione di snapshot
- Monitoraggio delle esportazioni di snapshot
- Annullamento di un'attività di esportazione di snapshot
- Prestazioni di esportazione in Aurora MySQL
- Risoluzione dei problemi relativi agli snapshot

Considerazioni sulle esportazioni di snapshot di cluster di database

Limitazioni

L'esportazione dei dati snapshot DB in Simple Storage Service (Amazon S3) presenta le seguenti limitazioni:

- Non è possibile eseguire contemporaneamente più attività di esportazione per lo stesso snapshot del cluster database. Ciò è valido sia per le esportazioni totali sia per le esportazioni parziali.
- È possibile avere fino a cinque attività di esportazione di snapshot di database simultanee in esecuzione per ogni Account AWS.
- Non è possibile esportare i dati degli snapshot da cluster di database Aurora Serverless v1 a S3.
- Le esportazioni in S3 non supportano i prefissi S3 contenenti i due punti (:).
- I seguenti caratteri nel percorso del file S3 vengono convertiti in caratteri di sottolineatura (_) durante l'esportazione:

```
\ ` " (space)
```

- Se un database, uno schema o una tabella contiene caratteri diversi da quelli riportati di seguito, l'esportazione parziale non è supportata. Tuttavia, è possibile esportare l'intero snapshot DB.
  - Lettere latine (A–Z)
  - Numeri (0–9)
  - Simbolo del dollaro ($)
  - Carattere di sottolineatura (_)
- Gli spazi ( ) e alcuni caratteri non sono supportati nei nomi delle colonne delle tabelle del database. Le tabelle con i seguenti caratteri nei nomi delle colonne vengono ignorate durante l'esportazione:

```
, ; { } ( ) \n \t = (space)
```

- Le tabelle con barre (/) nei rispettivi nomi vengono ignorate durante l'esportazione.
- Le tabelle temporanee e non registrate di Aurora PostgreSQL vengono ignorate durante l'esportazione.
- Se i dati contengono un oggetto di grandi dimensioni, ad esempio un BLOB o un CLOB, vicino o superiore a 500 MB, l'esportazione non riesce.
- Se una tabella contiene una riga di grandi dimensioni, vicine o superiori a 2 GB, la tabella viene ignorata durante l'esportazione.
- Per le esportazioni parziali, l'elenco `ExportOnly` ha una dimensione massima di 200 KB.
- Si consiglia vivamente di utilizzare un nome univoco per ogni attività di esportazione. Se non utilizzi un nome di attività univoco, potresti ricevere il seguente messaggio di errore:

  ExportTaskAlreadyExistsFault: si è verificato un errore (ExportTaskAlreadyExists) durante la chiamata all'operazione StartExportTask: l'attività di esportazione con ID *xxxxx* esiste già.
- È possibile eliminare uno snapshot durante l'esportazione dei suoi dati in S3, ma vengono comunque addebitati i costi di storage per tale snapshot fino al completamento dell'attività di esportazione.
- Non puoi ripristinare i dati degli snapshot esportati da S3 in un nuovo cluster di database.

Convenzione di denominazione file

I dati esportati per tabelle specifiche vengono memorizzati nel formato *base_prefix*/*files*, dove il prefisso di base è il seguente:

```
export_identifier/database_name/schema_name.table_name/
```

Ad esempio:

```
export-1234567890123-459/rdststdb/rdststdb.DataInsert_7ADB5D19965123A2/
```

Esistono due convenzioni di denominazione per i file.

- Convenzione attuale:

  ```
  batch_index/part-partition_index-random_uuid.format-based_extension
  ```

  L'indice batch è un numero sequenziale che rappresenta un batch di dati letti dalla tabella. Se la tabella non può essere partizionata in piccoli blocchi da esportare in parallelo, saranno presenti più indici batch. La stessa cosa accade se la tabella è partizionata in più tabelle. Saranno presenti più indici batch, uno per ciascuna delle partizioni di tabella della tabella principale.

  Se la tabella può essere partizionata in piccoli blocchi da leggere in parallelo, sarà presente solo la cartella di indice batch `1`.

  All'interno della cartella dell'indice batch sono presenti più file Parquet che contengono i dati della tabella. Il prefisso del nome file Parquet è `part-`*`partition_index`*. Se la tabella è partizionata, saranno presenti più file che iniziano con l'indice di partizione `00000`.

  La sequenza dell'indice di partizione può presentare delle lacune. Ciò accade perché ogni partizione è ottenuta da una query a intervalli nella tabella. Se l'intervallo di quella partizione non contiene dati, quel numero di sequenza viene ignorato.

  Ad esempio, supponiamo che la colonna `id` sia la chiave primaria della tabella e che i suoi valori minimo e massimo siano `100` e `1000`. Quando proviamo a esportare questa tabella con nove partizioni, la leggiamo con query parallele come le seguenti:

  ```
  SELECT * FROM table WHERE id <= 100 AND id < 200
   SELECT * FROM table WHERE id <= 200 AND id < 300
  ```

Questo dovrebbe generare nove file, da `part-00000-`*`random_uuid`*`.gz.parquet` a `part-00008-`*`random_uuid`*`.gz.parquet`. Tuttavia, se non ci sono righe con ID compresi tra `200` e `350`, una delle partizioni completate è vuota e non viene creato alcun file per essa. Nell'esempio precedente, `part-00001-`*`random_uuid`*`.gz.parquet` non viene creato.

- Convenzione precedente:

```
part-partition_index-random_uuid.format-based_extension
```

È uguale alla convenzione attuale, ma senza il prefisso *`batch_index`*, ad esempio:

```
part-00000-c5a881bb-58ff-4ee6-1111-b41ecff340a3-c000.gz.parquet
 part-00001-d7a881cc-88cc-5ab7-2222-c41ecab340a4-c000.gz.parquet
 part-00002-f5a991ab-59aa-7fa6-3333-d41eccd340a7-c000.gz.parquet
```

La convenzione di denominazione file è soggetta a modifiche. Pertanto, quando usi le tabelle di destinazione ti consigliamo di leggere tutto quanto riportato all'interno del prefisso di base della tabella.

Conversione dei dati durante l'esportazione in un bucket Simple Storage Service (Amazon S3)

Quando si esporta uno snapshot di database in un bucket Amazon S3, Amazon Aurora converte, esporta e memorizza i dati nel formato Parquet. Per ulteriori informazioni su Parquet, consultare il sito Web Apache Parquet.

Parquet archivia tutti i dati in uno dei seguenti tipi primitivi:

- BOOLEAN
- INT32
- INT64
- INT96
- FLOAT
- DOUBLE
- BYTE_ARRAY: un array di byte a lunghezza variabile, noto anche come binario

- FIXED_LEN_BYTE_ARRAY: un array di byte a lunghezza fissa utilizzato quando i valori hanno una dimensione costante

I tipi di dati Parquet sono pochi per ridurre la complessità di lettura e scrittura del formato. Parquet fornisce tipi logici per estendere i tipi primitivi. Un tipo logico viene implementato come annotazione con i dati in un campo di metadati `LogicalType`. L'annotazione di tipo logico spiega come interpretare il tipo primitivo.

Quando il tipo logico `STRING` annota un tipo `BYTE_ARRAY`, indica che l'array di byte deve essere interpretato come una stringa di caratteri con codifica UTF-8. Al termine di un'attività di esportazione, Amazon Aurora notifica all'utente se si è verificata una conversione di stringa. I dati sottostanti esportati sono sempre uguali ai dati provenienti dall'origine. Tuttavia, a causa della differenza di codifica in UTF-8, alcuni caratteri potrebbero apparire diversi dall'origine quando vengono letti in strumenti come Athena.

Per ulteriori informazioni, consultare la sezione relativa alle definizioni dei tipi logici di Parquet nella documentazione di Parquet.

Argomenti

- Mappatura dei tipi di dati MySQL e MariaDB al formato Parquet
- Mappatura dei tipi di dati PostgreSQL su Parquet

Mappatura dei tipi di dati MySQL e MariaDB al formato Parquet

La tabella seguente mostra la mappature tra i tipi di dati MySQL e i tipi di dati Parquet quando i dati vengono convertiti ed esportati in Amazon S3.

Tipo di dati origine	Tipo Parquet primitivo	Annotazione del tipo logico	Note di conversione
Tipi di dati numerici			
BIGINT	INT64		
BIGINT UNSIGNED	FIXED_LEN _BYTE_ARRAY(9)	DECIMAL(20,0)	Parquet supporta solo tipi firmati, quindi la mappatura richiede un byte aggiuntivo (8

Tipo di dati origine	Tipo Parquet primitivo	Annotazione del tipo logico	Note di conversione
			più 1) per memorizzare il tipo BIGINT_UNSIGNED.
BIT	BYTE_ARRAY		
DECIMAL	INT32	DECIMAL(p,s)	Se il valore di origine è inferiore a 2^{31}, viene archiviato come INT32.
	INT64	DECIMAL(p,s)	Se il valore di origine è 2^{31} o superiore, ma inferiore a 2^{63}, viene archiviato come INT64.
	FIXED_LEN_BYTE_ARRAY(N)	DECIMAL(p,s)	Se il valore di origine è 2^{63} o superiore, viene archiviato come FIXED_LEN_BYTE_ARRAY(N).
	BYTE_ARRAY	STRING	Parquet non supporta la precisione decimale superiore a 38. Il valore decimale viene convertito in una stringa di tipo BYTE_ARRAY e codificato come UTF8.
DOUBLE	DOUBLE		
FLOAT	DOUBLE		

Tipo di dati origine	Tipo Parquet primitivo	Annotazione del tipo logico	Note di conversione
INT	INT32		
INT UNSIGNED	INT64		
MEDIUMINT	INT32		
MEDIUMINT UNSIGNED	INT64		
NUMERIC	INT32	DECIMAL(p,s)	Se il valore di origine è inferiore a 2^{31}, viene archiviato come INT32.
	INT64	DECIMAL(p,s)	Se il valore di origine è 2^{31} o superiore, ma inferiore a 2^{63}, viene archiviato come INT64.
	FIXED_LEN _ARRAY(N)	DECIMAL(p,s)	Se il valore di origine è 2^{63} o superiore , viene archiviato come FIXED_LEN _BYTE_ARRAY(N).

Tipo di dati origine	Tipo Parquet primitivo	Annotazione del tipo logico	Note di conversione
	BYTE_ARRAY	STRING	Parquet non supporta la precisione numerica superiore a 38. Questo valore numerico viene convertito in una stringa di tipo BYTE_ARRAY e codificato come UTF8.
SMALLINT	INT32		
SMALLINT UNSIGNED	INT32		
TINYINT	INT32		
TINYINT UNSIGNED	INT32	INT(16, true)	
Tipi di dati stringa			
BINARY	BYTE_ARRAY		
BLOB	BYTE_ARRAY		
CHAR	BYTE_ARRAY		
ENUM	BYTE_ARRAY	STRING	
LINESTRING	BYTE_ARRAY		
LONGBLOB	BYTE_ARRAY		
LONGTEXT	BYTE_ARRAY	STRING	
MEDIUMBLOB	BYTE_ARRAY		

Tipo di dati origine	Tipo Parquet primitivo	Annotazione del tipo logico	Note di conversione
MEDIUMTEXT	BYTE_ARRAY	STRING	
MULTILINESTRING	BYTE_ARRAY		
SET	BYTE_ARRAY	STRING	
TEXT	BYTE_ARRAY	STRING	
TINYBLOB	BYTE_ARRAY		
TINYTEXT	BYTE_ARRAY	STRING	
VARBINARY	BYTE_ARRAY		
VARCHAR	BYTE_ARRAY	STRING	
Tipi di dati data e ora			
DATE	BYTE_ARRAY	STRING	Una data viene convertita in una stringa di tipo BYTE_ARRAY e codificata come UTF8.
DATETIME	INT64	TIMESTAMP_MICROS	
TIME	BYTE_ARRAY	STRING	Un tipo TIME viene convertito in una stringa di tipo BYTE_ARRAY e codificato come UTF8.
TIMESTAMP	INT64	TIMESTAMP_MICROS	

Tipo di dati origine	Tipo Parquet primitivo	Annotazione del tipo logico	Note di conversione
YEAR	INT32		
Tipi di dati geometrici			
GEOMETRY	BYTE_ARRAY		
GEOMETRYCOLLECTION	BYTE_ARRAY		
MULTIPOINT	BYTE_ARRAY		
MULTIPOLYGON	BYTE_ARRAY		
POINT	BYTE_ARRAY		
POLYGON	BYTE_ARRAY		
Tipo di dati JSON			
JSON	BYTE_ARRAY	STRING	

Mappatura dei tipi di dati PostgreSQL su Parquet

Nella tabella seguente viene illustrata la mappatura dai tipi di dati PostgreSQL ai tipi di dati Parquet quando i dati vengono convertiti ed esportati in Simple Storage Service (Amazon S3).

Tipo di dati PostgreSQL	Tipo Parquet primitivo	Annotazione del tipo logico	Note relative alla mappatura
Tipi di dati numerici			
BIGINT	INT64		
BIGSERIAL	INT64		
DECIMAL	BYTE_ARRAY	STRING	Un tipo DECIMAL viene convertito in

Tipo di dati PostgreSQL	Tipo Parquet primitivo	Annotazione del tipo logico	Note relative alla mappatura
			una stringa di tipo BYTE_ARRAY e codificato come UTF8. Questa conversione serve a evitare complicazioni dovute alla precisione dei dati e ai valori di dati che non sono un numero (NaN).
DOUBLE PRECISION	DOUBLE		
INTEGER	INT32		
MONEY	BYTE_ARRAY	STRING	
REAL	FLOAT		
SERIAL	INT32		
SMALLINT	INT32	INT(16, true)	
SMALLSERIAL	INT32	INT(16, true)	
Tipi di dati stringa e correlati			

Tipo di dati PostgreSQL	Tipo Parquet primitivo	Annotazione del tipo logico	Note relative alla mappatura
ARRAY	BYTE_ARRAY	STRING	Un array viene convertito in una stringa e codificato come BINARY (UTF8). Questa conversione serve a evitare complicazioni dovute alla precisione dei dati, ai valori di dati che non sono un numero (NaN) e ai valori di dati temporali.
BIT	BYTE_ARRAY	STRING	
BIT VARYING	BYTE_ARRAY	STRING	
BYTEA	BINARY		
CHAR	BYTE_ARRAY	STRING	
CHAR(N)	BYTE_ARRAY	STRING	
ENUM	BYTE_ARRAY	STRING	
NAME	BYTE_ARRAY	STRING	
TEXT	BYTE_ARRAY	STRING	
TEXT SEARCH	BYTE_ARRAY	STRING	
VARCHAR(N)	BYTE_ARRAY	STRING	

Tipo di dati PostgreSQL	Tipo Parquet primitivo	Annotazione del tipo logico	Note relative alla mappatura
XML	BYTE_ARRAY	STRING	
Tipi di dati data e ora			
DATE	BYTE_ARRAY	STRING	
INTERVAL	BYTE_ARRAY	STRING	
TIME	BYTE_ARRAY	STRING	
TIME WITH TIME ZONE	BYTE_ARRAY	STRING	
TIMESTAMP	BYTE_ARRAY	STRING	
TIMESTAMP WITH TIME ZONE	BYTE_ARRAY	STRING	
Tipi di dati geometrici			
BOX	BYTE_ARRAY	STRING	
CIRCLE	BYTE_ARRAY	STRING	
LINE	BYTE_ARRAY	STRING	
LINESEGMENT	BYTE_ARRAY	STRING	
PATH	BYTE_ARRAY	STRING	
POINT	BYTE_ARRAY	STRING	
POLYGON	BYTE_ARRAY	STRING	
Tipi di dati JSON			
JSON	BYTE_ARRAY	STRING	
JSONB	BYTE_ARRAY	STRING	

Tipo di dati PostgreSQL	Tipo Parquet primitivo	Annotazione del tipo logico	Note relative alla mappatura
Altri tipi di dati			
BOOLEAN	BOOLEAN		
CIDR	BYTE_ARRAY	STRING	Tipo di dati di rete
COMPOSITE	BYTE_ARRAY	STRING	
DOMAIN	BYTE_ARRAY	STRING	
INET	BYTE_ARRAY	STRING	Tipo di dati di rete
MACADDR	BYTE_ARRAY	STRING	
OBJECT IDENTIFIER	N/A		
PG_LSN	BYTE_ARRAY	STRING	
RANGE	BYTE_ARRAY	STRING	
UUID	BYTE_ARRAY	STRING	

Configurazione dell'accesso a un bucket Simple Storage Service (Amazon S3)

Individua il bucket Amazon S3 e fornisci allo snapshot l'autorizzazione per accedervi.

Argomenti

- Identificazione del bucket Simple Storage Service (Amazon S3) in cui esportare
- Fornire l'accesso a un bucket Simple Storage Service (Amazon S3) utilizzando un ruolo IAM
- Utilizzo di un bucket Simple Storage Service (Amazon S3) multiaccount
- Utilizzo di un account multiplo AWS KMS key

Identificazione del bucket Simple Storage Service (Amazon S3) in cui esportare

Identificare il bucket Simple Storage Service (Amazon S3) in cui esportare lo snapshot DB. Utilizzare un bucket S3 esistente o crearne uno nuovo.

> **Note**
>
> Il bucket S3 in cui esportare deve trovarsi nella stessa AWS regione dell'istantanea.

Per ulteriori informazioni sull'utilizzo dei bucket Simple Storage Service (Amazon S3), vedere quanto segue in Guida per l'utente di Amazon Simple Storage Service:

- Come visualizzare le proprietà di un bucket S3?
- In che modo si abilita la crittografia di default per un bucket Amazon S3?
- Come creare un bucket S3?

Fornire l'accesso a un bucket Simple Storage Service (Amazon S3) utilizzando un ruolo IAM

Prima di esportare i dati dello snapshot DB in Simple Storage Service (Amazon S3), fornire l'autorizzazione di accesso in scrittura alle attività di esportazione dello snapshot al bucket Simple Storage Service (Amazon S3).

Per concedere l'autorizzazione, crea una policy IAM che fornisca accesso al bucket, crea un ruolo IAM e collega la policy al ruolo. Successivamente assegna il ruolo IAM all'attività di esportazione dello snapshot.

> **Important**
>
> Se prevedi di utilizzare il Console di gestione AWS per esportare la tua istantanea, puoi scegliere di creare automaticamente la policy IAM e il ruolo quando esporti la snapshot. Per istruzioni, consulta Creazione di attività di esportazione di snapshot.

Per fornire alle attività dello snapshot DB l'accesso a Amazon S3

1. Creare una policy IAM Questa policy fornisce le autorizzazioni al bucket e all'oggetto che consentono all'attività di esportazione snapshot l'accesso a Amazon S3.

Includi nella policy le seguenti operazioni obbligatorie per consentire il trasferimento dei file da Amazon Aurora a un bucket S3:

- `s3:PutObject*`
- `s3:GetObject*`
- `s3:ListBucket`
- `s3:DeleteObject*`
- `s3:GetBucketLocation`

Includi nella policy le seguenti risorse per identificare il bucket S3 e gli oggetti nel bucket. Il seguente elenco di risorse mostra il formato Amazon Resource Name (ARN) per l'accesso a Amazon S3.

- `arn:aws:s3:::amzn-s3-demo-bucket`
- `arn:aws:s3:::amzn-s3-demo-bucket/*`

Per ulteriori informazioni sulla creazione di una policy IAM per Amazon Aurora, consultare Creazione e utilizzo di una policy IAM per l'accesso al database IAM. Consulta anche il Tutorial: Creare e collegare la prima policy gestita dal cliente nella Guida per l'utente di IAM.

Il AWS CLI comando seguente crea una policy IAM denominata `ExportPolicy` con queste opzioni. Concede l'accesso a un bucket denominato *`amzn-s3-demo-bucket`*.

> **Note**
>
> Dopo aver creato la policy, prendere nota del relativo ARN. Per la fase successiva, in cui si associa la policy a un ruolo IAM, è necessario l'ARN.

```
aws iam create-policy  --policy-name ExportPolicy --policy-document '{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "ExportPolicy",
            "Effect": "Allow",
            "Action": [
```

```
                "s3:PutObject*",
                "s3:ListBucket",
                "s3:GetObject*",
                "s3:DeleteObject*",
                "s3:GetBucketLocation"
            ],
            "Resource": [
                "arn:aws:s3:::amzn-s3-demo-bucket",
                "arn:aws:s3:::amzn-s3-demo-bucket/*"
            ]
        }
    ]
}'
```

2. Crea un ruolo IAM in modo che Aurora possa assumere questo ruolo IAM per tuo conto per accedere ai bucket Amazon S3. Per ulteriori informazioni, consulta la pagina relativa alla creazione di un ruolo per delegare le autorizzazioni a un utente IAM nella Guida per l'utente IAM.

   L'esempio seguente mostra l'utilizzo del AWS CLI comando per creare un ruolo denominato`rds-s3-export-role`.

```
aws iam create-role  --role-name rds-s3-export-role  --assume-role-policy-document
 '{
    "Version": "2012-10-17",
    "Statement": [
      {
        "Effect": "Allow",
        "Principal": {
           "Service": "export.rds.amazonaws.com"
         },
        "Action": "sts:AssumeRole"
      }
    ]
  }'
```

3. Collegare la policy IAM al ruolo IAM creato.

   Il AWS CLI comando seguente collega la politica creata in precedenza al ruolo denominato`rds-s3-export-role`. Sostituire *`your-policy-arn`* con l'ARN della policy annotato nella fase precedente.

```
aws iam attach-role-policy  --policy-arn your-policy-arn  --role-name rds-s3-
export-role
```

Utilizzo di un bucket Simple Storage Service (Amazon S3) multiaccount

Puoi utilizzare i bucket Amazon S3 su più account. AWS Per utilizzare un bucket tra account, aggiungi un criterio bucket per consentire l'accesso al ruolo IAM utilizzato per le esportazioni S3. Per informazioni, consulta Esempio 2: il proprietario del bucket concede autorizzazioni per il bucket multiaccount.

- Allega una policy di bucket al bucket, come mostrato nell'esempio riportato di seguito.

  JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Principal": {
                "AWS": "arn:aws:iam::123456789012:role/Admin"
            },
            "Action": [
                "s3:PutObject*",
                "s3:ListBucket",
                "s3:GetObject*",
                "s3:DeleteObject*",
                "s3:GetBucketLocation"
            ],
            "Resource": [
                "arn:aws:s3:::amzn-s3-demo-destination-bucket",
                "arn:aws:s3:::amzn-s3-demo-destination-bucket/*"
            ]
        }
    ]
}
```

Utilizzo di un account multiplo AWS KMS key

Puoi utilizzare un account multiplo AWS KMS key per crittografare le esportazioni Amazon S3. Innanzitutto, aggiungi una policy chiave all'account locale, quindi aggiungi le policy IAM nell'account esterno. Per ulteriori informazioni, consulta Autorizzazione per gli utenti in altri account a utilizzare una chiave KMS.

Per utilizzare una chiave KMS multiaccount

1. Aggiungi una policy di chiave all'account locale.

   Il seguente esempio fornisce `ExampleRole` e `ExampleUser` nell'account esterno 444455556666 autorizzazioni nell'account locale 123456789012.

```
{
    "Sid": "Allow an external account to use this KMS key",
    "Effect": "Allow",
    "Principal": {
        "AWS": [
            "arn:aws:iam::444455556666:role/ExampleRole",
            "arn:aws:iam::444455556666:user/ExampleUser"
        ]
    },
    "Action": [
        "kms:Encrypt",
        "kms:Decrypt",
        "kms:ReEncrypt*",
        "kms:GenerateDataKey*",
        "kms:CreateGrant",
        "kms:DescribeKey",
        "kms:RetireGrant"
    ],
    "Resource": "*"
}
```

2. Aggiungere le policy IAM nell'account esterno

   La policy IAM dell'esempio seguente consente al principale di utilizzare la chiave KMS nell'account 123456789012 per le operazioni di crittografia. Per concedere questa autorizzazione a `ExampleRole` e `ExampleUser` nell'account 444455556666, collega la policy ad essi nell'account.

```
{
    "Sid": "Allow use of KMS key in account 123456789012",
    "Effect": "Allow",
    "Action": [
        "kms:Encrypt",
        "kms:Decrypt",
        "kms:ReEncrypt*",
        "kms:GenerateDataKey*",
        "kms:CreateGrant",
        "kms:DescribeKey",
        "kms:RetireGrant"
    ],
    "Resource": "arn:aws:kms:us-
west-2:123456789012:key/1234abcd-12ab-34cd-56ef-1234567890ab"
}
```

Creazione di attività di esportazione di snapshot

Crea attività di esportazione di snapshot per esportare dati uno snapshot a un bucket Amazon S3. È possibile avere fino a cinque attività di esportazione di snapshot di database simultanee in esecuzione per ogni Account AWS.

> **Note**
>
> L'esportazione di snapshot RDS può richiedere qualche minuto a seconda del tipo e delle dimensioni del database. L'attività di esportazione ripristina e ridimensiona innanzitutto l'intero database prima di estrarre i dati su Simple Storage Service (Amazon S3). Lo stato di avanzamento dell'attività durante questa fase viene visualizzato come Avvio. Quando l'attività passa all'esportazione dei dati in S3, lo stato di avanzamento diventa In progress (In corso). Il tempo necessario per completare l'esportazione dipende dai dati memorizzati nel database. Ad esempio, le tabelle con chiave primaria numerica o colonne indice ben distribuite esporteranno più velocemente. Le tabelle che non contengono una colonna adatta al partizionamento e le tabelle con un solo indice su una colonna basata su stringhe richiedono più tempo. Questo tempo di esportazione più lungo si verifica perché l'esportazione utilizza un processo a thread singolo più lento.

È possibile esportare uno snapshot DB in Simple Storage Service (Amazon S3) utilizzando la Console di gestione AWS, AWS CLI o l'API RDS.

Se si utilizza una funzione Lambda per esportare uno snapshot, aggiungere l'operazione `kms:DescribeKey` alla policy della funzione Lambda. Per ulteriori informazioni, consulta Autorizzazioni di AWS Lambda.

Console

L'opzione Export to Amazon S3 (Esporta in Simple Storage Service (Amazon S3)) viene visualizzata solo per gli snapshot che possono essere esportati in Simple Storage Service (Amazon S3). Uno snapshot potrebbe non essere disponibile per l'esportazione a causa dei seguenti motivi:

- Il motore del database non è supportato per l'esportazione S3.
- La versione dell'istanza database non è supportata per l'esportazione S3.
- L'esportazione S3 non è supportata nella regione AWS in cui è stato creato lo snapshot.

Per esportare uno snapshot DB

1. Accedi alla Console di gestione AWS e apri la console Amazon RDS all'indirizzo https://console.aws.amazon.com/rds/.
2. Nel pannello di navigazione, selezionare Snapshots (Snapshot).
3. Dalle schede, scegliere il tipo di snapshot che si desidera esportare.
4. Nell'elenco degli snapshot, scegliere lo snapshot che si desidera esportare.
5. Per Actions (Operazioni), scegli Export to Amazon S3 (Esporta in Simple Storage Service (Amazon S3).

   Viene visualizzata la finestra Export to Amazon S3 (Esporta in Simple Storage Service (Amazon S3)).
6. Per Export identifier (Identificatore di esportazione), immettere un nome per identificare l'attività di esportazione. Questo valore viene utilizzato anche per il nome del file creato nel bucket S3.
7. Scegli i dati da esportare:

   - Scegliere All (Tutti) per esportare tutti i dati nello snapshot.
   - Scegliere Partial (Parziali) per esportare parti specifiche dello snapshot. Per identificare le parti dello snapshot da esportare, immettere uno o più database, schemi o tabelle per Identifiers (Identificatori), separati da spazi.

Utilizza il seguente formato:

```
database[.schema][.table] database2[.schema2][.table2] ... databasen[.schemaN]
[.tablen]
```

Ad esempio:

```
mydatabase mydatabase2.myschema1 mydatabase2.myschema2.mytable1
 mydatabase2.myschema2.mytable2
```

8. Per S3 bucket (Bucket S3), scegliere il bucket in cui esportare.

   Per assegnare i dati esportati a un percorso di cartella nel bucket S3, immettere il percorso opzionale per S3 prefix (Prefisso S3).

9. Per il ruolo IAM, scegliere un ruolo che conceda l'accesso in scrittura al bucket S3 scelto o creare un nuovo ruolo.

   - Se è stato creato un ruolo seguendo le fasi in Fornire l'accesso a un bucket Simple Storage Service (Amazon S3) utilizzando un ruolo IAM, scegliere tale ruolo.
   - Se non è stato creato un ruolo che fornisce l'accesso in scrittura al bucket S3 scelto, scegli Create a new role (Crea un nuovo ruolo) per creare automaticamente il ruolo. Immettere quindi un nome per il ruolo nel nome del ruolo IAM.

10. Per AWS KMS key, immettere l'ARN per la chiave da utilizzare per crittografare i dati esportati.
11. Scegliere Export to Amazon S3 (Esporta in Simple Storage Service (Amazon S3)).

AWS CLI

Per esportare uno snapshot DB in Simple Storage Service (Amazon S3) tramite AWS CLI, utilizzare il comando start-export-task con le seguenti opzioni richieste:

- `--export-task-identifier`
- `--source-arn`
- `--s3-bucket-name`
- `--iam-role-arn`
- `--kms-key-id`

Negli esempi seguenti, l'attività di esportazione snapshot è denominata *my-snapshot-export* ed esporta uno snapshot in un bucket S3 denominato *amzn-s3-demo-destination-bucket*.

Example

Per Linux, macOS o Unix:

```
aws rds start-export-task \
    --export-task-identifier my-snapshot-export \
    --source-arn arn:aws:rds:AWS_Region:123456789012:snapshot:snapshot-name \
    --s3-bucket-name amzn-s3-demo-destination-bucket \
    --iam-role-arn iam-role \
    --kms-key-id my-key
```

Per Windows:

```
aws rds start-export-task ^
    --export-task-identifier my-snapshot-export ^
    --source-arn arn:aws:rds:AWS_Region:123456789012:snapshot:snapshot-name ^
    --s3-bucket-name amzn-s3-demo-destination-bucket ^
    --iam-role-arn iam-role ^
    --kms-key-id my-key
```

Di seguito è riportato un output di esempio.

```
{
    "Status": "STARTING",
    "IamRoleArn": "iam-role",
    "ExportTime": "2019-08-12T01:23:53.109Z",
    "S3Bucket": "amzn-s3-demo-destination-bucket",
    "PercentProgress": 0,
    "KmsKeyId": "my-key",
    "ExportTaskIdentifier": "my-snapshot-export",
    "TotalExtractedDataInGB": 0,
    "TaskStartTime": "2019-11-13T19:46:00.173Z",
    "SourceArn": "arn:aws:rds:AWS_Region:123456789012:snapshot:snapshot-name"
}
```

Per fornire un percorso di cartella nel bucket S3 per l'esportazione dello snapshot, includere l'opzione `--s3-prefix` nel comando start-export-task.

API RDS

Per esportare uno snapshot DB Amazon S3 utilizzando l'API Amazon RDS, utilizzare l'operazione StartExportTask con i seguenti parametri richiesti:

- `ExportTaskIdentifier`
- `SourceArn`
- `S3BucketName`
- `IamRoleArn`
- `KmsKeyId`

Monitoraggio delle esportazioni di snapshot

È possibile monitorare le esportazioni di snapshot DB utilizzando la Console di gestione AWS, l'AWS CLI o l'API RDS.

Console

Per monitorare le esportazioni di snapshot DB

1. Accedi alla Console di gestione AWS e apri la console Amazon RDS all'indirizzo https://console.aws.amazon.com/rds/.
2. Nel riquadro di navigazione scegli Exports in Amazon S3 (Esporta in Amazon S3).

   Le esportazioni di snapshot di database sono indicate nella colonna Source type (Tipo di origine). Lo stato dell'esportazione è visualizzato nella colonna Status (Stato).
3. Per visualizzare informazioni dettagliate su un'esportazione di snapshot specifica, scegli l'attività di esportazione.

AWS CLI

Per monitorare le esportazioni di snapshot DB utilizzando l'AWS CLI, utilizzare il comando describe-export-tasks.

Nell'esempio seguente viene illustrato come visualizzare le informazioni correnti su tutte le esportazioni di snapshot.

Example

```
aws rds describe-export-tasks

{
    "ExportTasks": [
        {
            "Status": "CANCELED",
            "TaskEndTime": "2019-11-01T17:36:46.961Z",
            "S3Prefix": "something",
            "ExportTime": "2019-10-24T20:23:48.364Z",
            "S3Bucket": "amzn-s3-demo-bucket",
            "PercentProgress": 0,
            "KmsKeyId": "arn:aws:kms:AWS_Region:123456789012:key/K7MDENG/
bPxRfiCYEXAMPLEKEY",
            "ExportTaskIdentifier": "anewtest",
            "IamRoleArn": "arn:aws:iam::123456789012:role/export-to-s3",
            "TotalExtractedDataInGB": 0,
            "TaskStartTime": "2019-10-25T19:10:58.885Z",
            "SourceArn": "arn:aws:rds:AWS_Region:123456789012:snapshot:parameter-
groups-test"
        },
{
            "Status": "COMPLETE",
            "TaskEndTime": "2019-10-31T21:37:28.312Z",
            "WarningMessage": "{\"skippedTables\":[],\"skippedObjectives\":[],\"general
\":[{\"reason\":\"FAILED_TO_EXTRACT_TABLES_LIST_FOR_DATABASE\"}]}",
            "S3Prefix": "",
            "ExportTime": "2019-10-31T06:44:53.452Z",
            "S3Bucket": "amzn-s3-demo-bucket1",
            "PercentProgress": 100,
            "KmsKeyId": "arn:aws:kms:AWS_Region:123456789012:key/2Zp9Utk/
h3yCo8nvbEXAMPLEKEY",
            "ExportTaskIdentifier": "thursday-events-test",
            "IamRoleArn": "arn:aws:iam::123456789012:role/export-to-s3",
            "TotalExtractedDataInGB": 263,
            "TaskStartTime": "2019-10-31T20:58:06.998Z",
            "SourceArn":
 "arn:aws:rds:AWS_Region:123456789012:snapshot:rds:example-1-2019-10-31-06-44"
        },
        {
            "Status": "FAILED",
            "TaskEndTime": "2019-10-31T02:12:36.409Z",
```

```
            "FailureCause": "The S3 bucket my-exports isn't located in the current AWS
 Region. Please, review your S3 bucket name and retry the export.",
            "S3Prefix": "",
            "ExportTime": "2019-10-30T06:45:04.526Z",
            "S3Bucket": "amzn-s3-demo-bucket2",
            "PercentProgress": 0,
            "KmsKeyId": "arn:aws:kms:AWS_Region:123456789012:key/2Zp9Utk/
h3yCo8nvbEXAMPLEKEY",
            "ExportTaskIdentifier": "wednesday-afternoon-test",
            "IamRoleArn": "arn:aws:iam::123456789012:role/export-to-s3",
            "TotalExtractedDataInGB": 0,
            "TaskStartTime": "2019-10-30T22:43:40.034Z",
            "SourceArn":
 "arn:aws:rds:AWS_Region:123456789012:snapshot:rds:example-1-2019-10-30-06-45"
        }
    ]
}
```

Per visualizzare informazioni su un'esportazione di snapshot specifica, includere l'opzione `--export-task-identifier` nel comando `describe-export-tasks`. Per filtrare l'output, includere l'opzione `--Filters`. Per ulteriori opzioni, consultare il comando describe-export-tasks.

API RDS

Per visualizzare informazioni sulle esportazioni di snapshot DB utilizzando l'API Amazon RDS, utilizzare l'operazione DescribeExportTasks.

Per tenere traccia del completamento del flusso di lavoro di esportazione o per attivare un altro flusso di lavoro, è possibile sottoscrivere gli argomenti del Servizio di notifica semplice Amazon. Per ulteriori informazioni su Amazon SNS, consulta Utilizzo della notifica degli eventi di Amazon RDS.

Annullamento di un'attività di esportazione di snapshot

È possibile annullare un'attività di esportazione di snapshot DB utilizzando la Console di gestione AWS, l'AWS CLI o l'API RDS.

> **Note**
>
> L'annullamento di un'attività di esportazione di snapshot non rimuove i dati esportati in Simple Storage Service (Amazon S3). Per informazioni su come eliminare i dati utilizzando la

console, consultare Come eliminare oggetti da un bucket S3? Per eliminare i dati utilizzando l'interfaccia della riga di comando (CLI), utilizzare il comando delete-object.

Console

Per annullare un'attività di esportazione di uno snapshot

1. Accedi alla Console di gestione AWS e apri la console Amazon RDS all'indirizzo https://console.aws.amazon.com/rds/.
2. Nel riquadro di navigazione scegli Exports in Amazon S3 (Esporta in Amazon S3).

   Le esportazioni di snapshot di database sono indicate nella colonna Source type (Tipo di origine). Lo stato dell'esportazione è visualizzato nella colonna Status (Stato).
3. Scegliere l'attività di esportazione di snapshot che si desidera annullare.
4. Seleziona Cancel (Annulla).
5. Scegli Cancel export task (Annulla attività di esportazione) nella pagina di conferma.

AWS CLI

Per annullare un'attività di esportazione di snapshot utilizzando l'AWS CLI, utilizzare il comando cancel-export-task. Il comando richiede l'opzione `--export-task-identifier`.

Example

```
aws rds cancel-export-task --export-task-identifier my_export
{
    "Status": "CANCELING",
    "S3Prefix": "",
    "ExportTime": "2019-08-12T01:23:53.109Z",
    "S3Bucket": "amzn-s3-demo-bucket",
    "PercentProgress": 0,
    "KmsKeyId": "arn:aws:kms:AWS_Region:123456789012:key/K7MDENG/bPxRfiCYEXAMPLEKEY",
    "ExportTaskIdentifier": "my_export",
    "IamRoleArn": "arn:aws:iam::123456789012:role/export-to-s3",
    "TotalExtractedDataInGB": 0,
    "TaskStartTime": "2019-11-13T19:46:00.173Z",
    "SourceArn": "arn:aws:rds:AWS_Region:123456789012:snapshot:export-example-1"
```

```
}
```

API RDS

Per annullare un'attività di esportazione di snapshot utilizzando l'API Amazon RDS, utilizzare l'operazione CancelExportTask con il parametro `ExportTaskIdentifier`.

Prestazioni di esportazione in Aurora MySQL

Gli snapshot del cluster di database Aurora MySQL versione 2 e versione 3 utilizzano un meccanismo di esportazione avanzato per migliorare le prestazioni e ridurre i tempi di esportazione. Il meccanismo include ottimizzazioni come i thread di esportazione multipli e la query parallela Aurora MySQL per sfruttare l'architettura di archiviazione condivisa di Aurora. Le ottimizzazioni vengono applicate in modo adattivo, a seconda delle dimensioni e della struttura del set di dati.

Non è necessario attivare la query parallela per utilizzare il processo di esportazione più rapido, ma il processo presenta le stesse limitazioni della query parallela. Inoltre, alcuni valori di dati non sono supportati, ad esempio le date in cui è il giorno del mese è `0` o l'anno è `0000`. Per ulteriori informazioni, consulta Query parallela per Amazon Aurora MySQL.

Quando si applicano le ottimizzazioni delle prestazioni, è possibile anche che i file Parquet diventino molto più grandi (~200 GB) per le esportazioni di Aurora MySQL versione 2 e 3.

Se non è possibile utilizzare il processo di esportazione più rapido, ad esempio a causa di tipi di dati o valori incompatibili, Aurora passa automaticamente a una modalità di esportazione a thread singolo senza query parallela. Le prestazioni di esportazione possono variare a seconda del processo utilizzato e della quantità di dati da esportare.

Risoluzione dei problemi relativi agli snapshot

Utilizza le seguenti sezioni per la risoluzione dei problemi relativi ai messaggi di errore e agli errori di autorizzazione PostgreSQL per le attività di esportazione di cluster di database in Amazon S3.

Messaggi di errore per le attività di esportazione di Amazon S3

Nella tabella seguente vengono descritti i messaggi restituiti quando le attività di esportazione di Amazon S3 non riescono.

Messaggio di errore	Descrizione
Si è verificato un errore interno sconosciuto.	L'elaborazione della richiesta non è riuscita a causa di un errore, un'eccezione o un guasto interno sconosciuto.
Si è verificato un errore interno sconosciuto durante la scrittura dei metadati dell'attività di esportazione nel bucket S3 [nome bucket].	L'elaborazione della richiesta non è riuscita a causa di un errore, un'eccezione o un guasto interno sconosciuto.
L'esportazione RDS non è riuscita a scrivere i metadati dell'attività di esportazione perché non può assumere il ruolo IAM [ruolo ARN].	L'attività di esportazione assume il ruolo IAM per verificare se è consentito scrivere metadati nel bucket S3. Se l'attività non può assumere il ruolo IAM, non riesce.
L'esportazione RDS non è riuscita a scrivere i metadati dell'attività di esportazione nel bucket S3 [nome bucket] utilizzando il ruolo IAM [ruolo ARN] con la chiave KMS [ID chiave]. Codice di errore: [codice di errore]	Mancano una o più autorizzazioni, quindi l'attività di esportazione non può accedere al bucket S3. Questo messaggio di errore viene generato quando si riceve uno dei seguenti codici di errore: • `AWSSecurityTokenServiceException` con il codice di errore `AccessDenied` • `AmazonS3Exception` con il codice di errore `NoSuchBucket`, `AccessDenied`, `KMS.KMSInvalidStateException`, `403 Forbidden`, oppure `KMS.DisabledException` Questi codici di errore indicano che le impostazioni non sono configurate correttamente per il ruolo IAM, il bucket S3 o la chiave KMS.
Il ruolo IAM [ruolo ARN] non è autorizzato a chiamare [azione S3] sul bucket S3 [nome bucket]. Controlla le tue autorizzazioni e riprova l'esportazione.	La policy IAM è configurata in modo errato. L'autorizzazione per l'azione S3 specifica sul bucket S3 è mancante e questa condizione causa l'esito negativo dell'attività di esportazione.

Messaggio di errore	Descrizione
Controllo chiave KMS non riuscito. Controlla le credenziali sulla tua chiave KMS e riprova.	Controllo delle credenziali della chiave KMS non riuscito.
Controllo delle credenziali S3 non riuscito. Controlla le autorizzazioni per il bucket S3 e la policy IAM.	Il controllo delle credenziali S3 non è riuscito.
Il bucket S3 [nome bucket] non è valido. Non si trova nella corrente Regione AWS oppure non esiste. Esamina il nome del bucket S3 e riprova l'esporta zione.	Bucket S3 non è valido.
Il bucket S3 [nome bucket] non si trova nella corrente Regione AWS. Esamina il nome del bucket S3 e riprova l'esporta zione.	Il bucket S3 è nella Regione AWS sbagliata.

Risoluzione degli errori di autorizzazione PostgreSQL

Quando si esportano i database PostgreSQL in Simple Storage Service (Amazon S3), è possibile che venga visualizzato un errore `PERMISSIONS_DO_NOT_EXIST` che indica che alcune tabelle sono state ignorate. Questo errore si verifica in genere quando l'utente con privilegi avanzati che hai specificato durante la creazione dell'istanza database, non dispone delle autorizzazioni per accedere alle tabelle.

Per risolvere questo errore, eseguire il comando seguente:

```
GRANT ALL PRIVILEGES ON ALL TABLES IN SCHEMA schema_name TO superuser_name
```

Per ulteriori informazioni sui privilegi utente con privilegi avanzati, vedere Privilegi dell'account utente master.

Ripristino di un cluster di database a un determinato momento

Per ripristinare un cluster di database a un determinato momento, crea un nuovo cluster di database.

Quando ripristini un cluster di database a un determinato momento, puoi scegliere il gruppo di sicurezza VPC (Virtual Private Cloud) predefinito. In alternativa, puoi applicare un gruppo di sicurezza VPC personalizzato al tuo cluster di database.

Il cluster di database ripristinati vengono associati automaticamente al cluster e ai gruppi di parametri predefiniti. Tuttavia, puoi applicare un gruppo di parametri personalizzati specificandoli durante un ripristino.

Amazon Aurora carica i log per i cluster di database in Simple Storage Service (Amazon S3) in modo continuativo. Per visualizzare l'ora di ripristino più recente per un cluster DB, usa il AWS CLI describe-db-clusterscomando e guarda il valore restituito nel `LatestRestorableTime` campo per il cluster DB.

Puoi eseguire il ripristino point-in-time durante il tempo di conservazione del backup. Per visualizzare il primo orario di ripristino per un cluster di DB, usa il AWS CLI describe-db-clusterscomando e guarda il valore restituito nel `EarliestRestorableTime` campo per il cluster di DB.

Il periodo di conservazione dei backup del cluster di database ripristinato è uguale a quello del database di origine.

> **Note**
>
> Le informazioni contenute in questo argomento si applicano ad Amazon Aurora. Per informazioni sul ripristino di un'istanza database Amazon RDS, consulta Ripristino di un'istanza database a un determinato momento.
> Per ulteriori informazioni sul backup e sul ripristino di un cluster di database Aurora, consulta Panoramica di backup e ripristino di un cluster di database Aurora.
> Per Aurora MySQL, puoi ripristinare un cluster di database assegnato a un cluster di database Aurora Serverless. Per ulteriori informazioni, consulta Ripristino di un cluster database Aurora Serverless v1.
> Puoi anche utilizzarlo AWS Backup per gestire i backup dei cluster Amazon Aurora DB. Se il cluster DB è associato a un piano di backup in AWS Backup, tale piano di backup viene utilizzato per il ripristino. point-in-time Per informazioni, consulta Ripristino di un cluster database a un'ora specificata mediante AWS Backup.

Per informazioni sul ripristino di un cluster di database Aurora o di un cluster globale con una versione Supporto esteso di RDS, consulta Ripristino di un cluster di database Aurora o un cluster globale con il supporto esteso di Amazon RDS.

Ripristina un cluster DB a un'ora specificata da un backup automatico, un backup automatizzato mantenuto o utilizzando AWS Backup.

Argomenti

- Ripristino point-in-time di un cluster di database
- Ripristino di un cluster database a un orario specificato da un backup automatico mantenuto
- Ripristino di un cluster database a un'ora specificata mediante AWS Backup

Ripristino point-in-time di un cluster di database

È possibile ripristinare un cluster DB in un determinato momento utilizzando l' Console di gestione AWS API AWS CLI, the o RDS.

Console

Per ripristinare un cluster di database a un determinato momento

1. Accedi a Console di gestione AWS e apri la console Amazon RDS all'indirizzo https://console.aws.amazon.com/rds/.
2. Nel riquadro di navigazione, selezionare Automated backups (Backup automatici).

   I backup automatici vengono visualizzati nella scheda Current Region (Regione corrente).

3. Scegli il cluster di database che desideri ripristinare.
4. In Actions (Operazioni), scegli Restore to point in time (Ripristina a un istante temporale).

   Viene visualizzata la finestra Restore to point in time (Ripristina a un istante temporale).
5. Scegliere Latest restorable time (Ultimo orario di ripristino) per eseguire il ripristino in base al momento più recente oppure scegliere Custom (Personalizzato) per scegliere una data e un'ora.

Se scegli Custom (Personalizzato), inserisci la data e l'ora alla quale desideri ripristinare il cluster.

> **Note**
>
> Gli orari vengono visualizzati nel fuso orario locale, indicato come un offset dell'ora UTC (Coordinated Universal Time). Ad esempio, UTC-5 è l'ora legale standard Time/Central orientale.

6. Per Identificatore cluster di database, inserisci il cluster database di destinazione ripristinato. Il nome deve essere univoco.
7. Scegliere le altre opzioni in base alle esigenze, ad esempio la classe dell'istanza database e la configurazione dell'archiviazione del cluster database.

   Per informazioni su ciascuna impostazione, consulta Impostazioni per cluster di database Aurora.
8. Scegli Restore to point in time (Ripristina per punto nel tempo).

AWS CLI

Per ripristinare un cluster DB a un'ora specificata, utilizzare il AWS CLI comando restore-db-cluster-to- point-in-time per creare un nuovo cluster DB.

È possibile specificare altre impostazioni. Per informazioni su ciascuna impostazione, consulta Impostazioni per cluster di database Aurora.

Se i tag vengono forniti nella richiesta, i tag forniti vengono applicati al cluster DB ripristinato. Se i tag non sono forniti nella richiesta e se il cluster DB di origine è attivo nella regione e dispone di tag, Aurora aggiunge i tag più recenti dal cluster DB di origine al cluster DB ripristinato.

Example

Per Linux, macOS o Unix:

```
aws rds restore-db-cluster-to-point-in-time \
       --source-db-cluster-identifier mysourcedbcluster \
       --db-cluster-identifier mytargetdbcluster \
       --restore-to-time 2017-10-14T23:45:00.000Z
```

Per Windows:

```
aws rds restore-db-cluster-to-point-in-time ^
       --source-db-cluster-identifier mysourcedbcluster ^
       --db-cluster-identifier mytargetdbcluster ^
       --restore-to-time 2017-10-14T23:45:00.000Z
```

Important

Se utilizzi la console per ripristinare un cluster di database a un determinato momento, Amazon RDS crea automaticamente l'istanza principale (writer) per il cluster di database. Se si utilizza il AWS CLI per ripristinare un cluster DB a un'ora specificata, è necessario creare in modo esplicito l'istanza principale per il cluster DB. L'istanza primaria è la prima istanza creata in un cluster di database.
Per creare l'istanza principale per il tuo cluster DB, chiama il create-db-instance AWS CLI comando. Includi il nome del cluster di database come valore dell'opzione `--db-cluster-identifier`.

API RDS

Per ripristinare un cluster di database a un punto temporale specifico, utilizza l'operazione API Amazon RDS `RestoreDBClusterToPointInTime` con i seguenti parametri:

- `SourceDBClusterIdentifier`
- `DBClusterIdentifier`
- `RestoreToTime`

Important

Se utilizzi la console per ripristinare un cluster di database a un determinato momento, Amazon RDS crea automaticamente l'istanza principale (writer) per il cluster di database.
Se utilizzi l'API RDS per ripristinare un cluster di database a un determinato momento, devi creare in modo esplicito l'istanza principale per il cluster di database. L'istanza primaria è la prima istanza creata in un cluster di database.

Per creare l'istanza principale per il tuo cluster DB, chiama l'operazione DBInstanceCreate dell'API RDS. Includi il nome del cluster di database come valore del parametro `DBClusterIdentifier`.

Ripristino di un cluster database a un orario specificato da un backup automatico mantenuto

È possibile ripristinare un cluster database da un backup automatico mantenuto dopo aver eliminato il cluster database di origine, se il backup rientra nel periodo di conservazione del cluster di origine. Il processo è simile al ripristino di un cluster database da un backup automatico.

> **Note**
>
> Non è possibile ripristinare un cluster di database Aurora Serverless v1 utilizzando questa procedura, poiché i backup automatici per i cluster Aurora Serverless v1 non vengono conservati.

Console

Per ripristinare un cluster di database a un determinato momento

1. Accedi alla Console di gestione AWS e apri la console Amazon RDS all'indirizzo https://console.aws.amazon.com/rds/.
2. Nel riquadro di navigazione, selezionare Automated backups (Backup automatici).
3. Scegli la scheda Conservato.

4. Scegli il cluster di database che desideri ripristinare.
5. In Actions (Operazioni), scegli Restore to point in time (Ripristina a un istante temporale).

   Viene visualizzata la finestra Restore to point in time (Ripristina a un istante temporale).

6. Scegliere Latest restorable time (Ultimo orario di ripristino) per eseguire il ripristino in base al momento più recente oppure scegliere Custom (Personalizzato) per scegliere una data e un'ora.

   Se scegli Custom (Personalizzato), inserisci la data e l'ora alla quale desideri ripristinare il cluster.

   > **Note**
   >
   > Gli orari vengono visualizzati nel fuso orario locale, indicato come un offset dell'ora UTC (Coordinated Universal Time). Ad esempio, UTC-5 è l'orario standard degli Stati Uniti orientali/ora legale degli Stati Uniti centrali.

7. Per Identificatore cluster di database, inserisci il cluster database di destinazione ripristinato. Il nome deve essere univoco.
8. Scegli altre opzioni in base alle esigenze, ad esempio la classe di istanza database.

   Per informazioni su ciascuna impostazione, consulta Impostazioni per cluster di database Aurora.
9. Scegli Restore to point in time (Ripristina per punto nel tempo).

AWS CLI

Per ripristinare un cluster database a un punto temporale specifico, utilizza il comando AWS CLI restore-db-cluster-to-point-in-time per creare un nuovo cluster database.

È possibile specificare altre impostazioni. Per informazioni su ciascuna impostazione, consulta Impostazioni per cluster di database Aurora.

Il tagging di risorse è supportato per questa operazione. Quando usi l'opzione `--tags`, i tag del cluster di database di origine vengono ignorati e vengono utilizzati quelli forniti. In caso contrario, vengono utilizzati i tag più recenti del cluster di origine.

Example

Per Linux, macOS o Unix:

```
aws rds restore-db-cluster-to-point-in-time \
    --source-db-cluster-resource-id cluster-123ABCEXAMPLE \
    --db-cluster-identifier mytargetdbcluster \
```

```
    --restore-to-time 2017-10-14T23:45:00.000Z
```

Per Windows:

```
aws rds restore-db-cluster-to-point-in-time ^
    --source-db-cluster-resource-id cluster-123ABCEXAMPLE ^
    --db-cluster-identifier mytargetdbcluster ^
    --restore-to-time 2017-10-14T23:45:00.000Z
```

Important

Se utilizzi la console per ripristinare un cluster di database a un determinato momento, Amazon RDS crea automaticamente l'istanza principale (writer) per il cluster di database. Se utilizzi AWS CLI per ripristinare un cluster di database, è necessario creare in modo esplicito l'istanza principale per il cluster. L'istanza primaria è la prima istanza creata in un cluster di database.

Invoca il comando AWS CLI create-db-instance per creare l'istanza principale per il cluster di database. Includi il nome del cluster di database come valore dell'opzione `--db-cluster-identifier`.

API RDS

Per ripristinare un cluster di database a un punto temporale specifico, utilizza l'operazione API Amazon RDS `RestoreDBClusterToPointInTime` con i seguenti parametri:

- `SourceDbClusterResourceId`
- `DBClusterIdentifier`
- `RestoreToTime`

Important

Se utilizzi la console per ripristinare un cluster di database a un determinato momento, Amazon RDS crea automaticamente l'istanza principale (writer) per il cluster di database. Se utilizzi l'API RDS per ripristinare un cluster di database a un determinato momento, devi creare in modo esplicito l'istanza principale per il cluster di database. L'istanza primaria è la prima istanza creata in un cluster di database.

Invoca l'operazione API RDS CreateDBInstance per creare l'istanza principale per il cluster di database. Includi il nome del cluster di database come valore del parametro `DBClusterIdentifier`.

Ripristino di un cluster database a un'ora specificata mediante AWS Backup

Puoi utilizzare AWS Backup per gestire i backup automatici e quindi ripristinarli a un'ora specificata. A tale scopo, devi creare un piano di backup in AWS Backup e assegnare il tuo cluster database come risorsa. Quindi abiliti i backup continui per il PITR nella regola di backup. Per ulteriori informazioni sui piani di backup e sulle regole di backup, consulta laGuida per gli sviluppatori di AWS Backup.

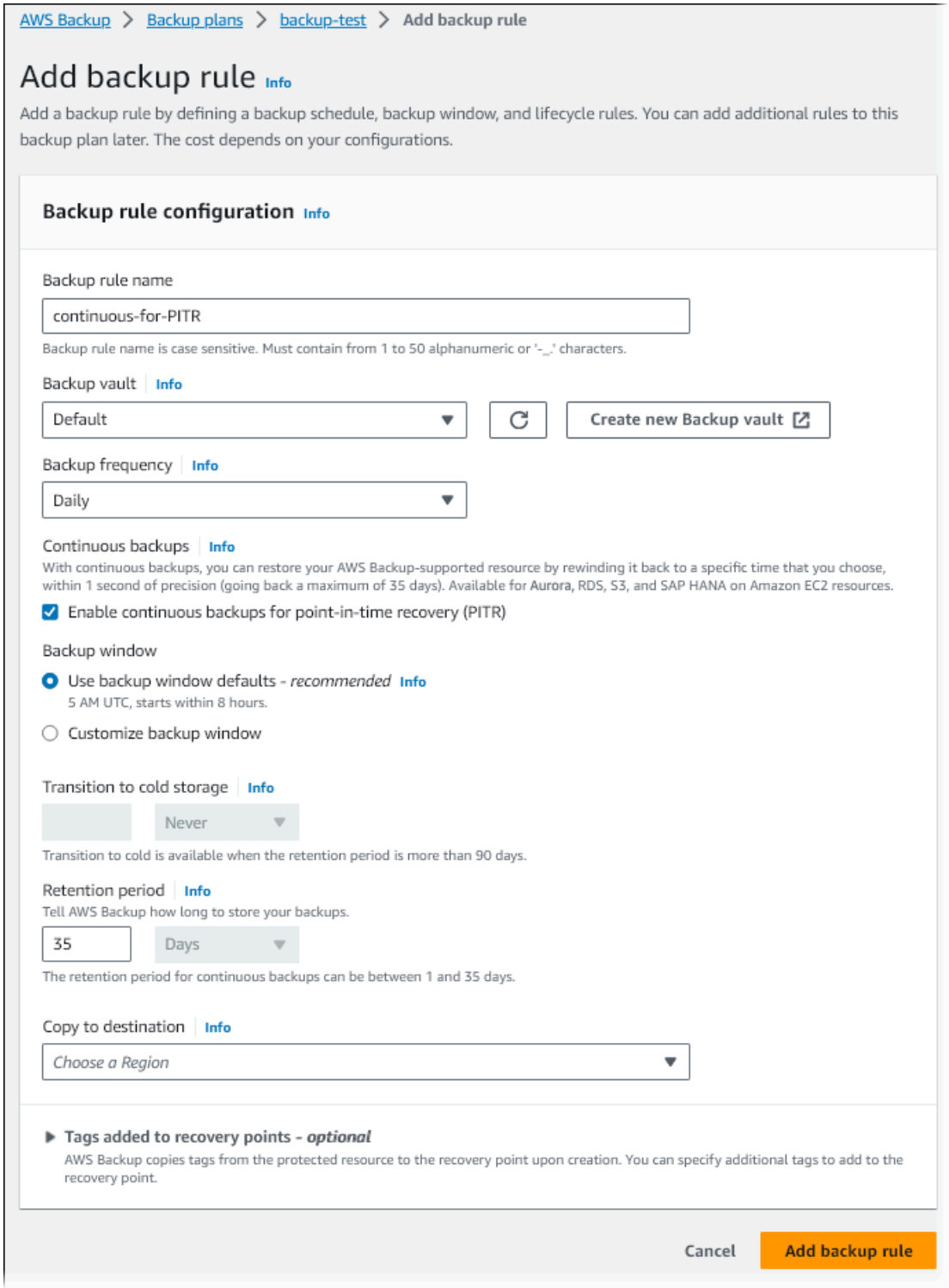
Abilitazione di backup continui in AWS Backup

È possibile abilitare i backup continui nelle regole di backup.

Abilitazione dei backup continui per il PITR

1. Accedi alla Console di gestione AWS e apri la console AWS Backup all'indirizzo https://console.aws.amazon.com/backup.
2. Nel riquadro di navigazione scegliere Backup plans (Piani di backup).
3. In Nome del piano di backup, seleziona il piano di backup che utilizzi per eseguire il backup del cluster database.
4. Nella sezione Regole di backup, scegli Aggiungi regola di backup.

   Viene visualizzata la pagina Aggiungi regola di backup.
5. Seleziona la casella di controllo Abilitazione dei backup continui per Ripristino point-in-time (PITR).



6. Scegli le altre impostazioni, se necessario, quindi scegli Aggiungi regola di backup.

Ripristino da un backup continuo in AWS Backup

Puoi eseguire il ripristino a un'ora specificata da un vault di backup.

Console

Per ripristinare un cluster database a un'ora specificata, puoi utilizzare Console di gestione AWS.

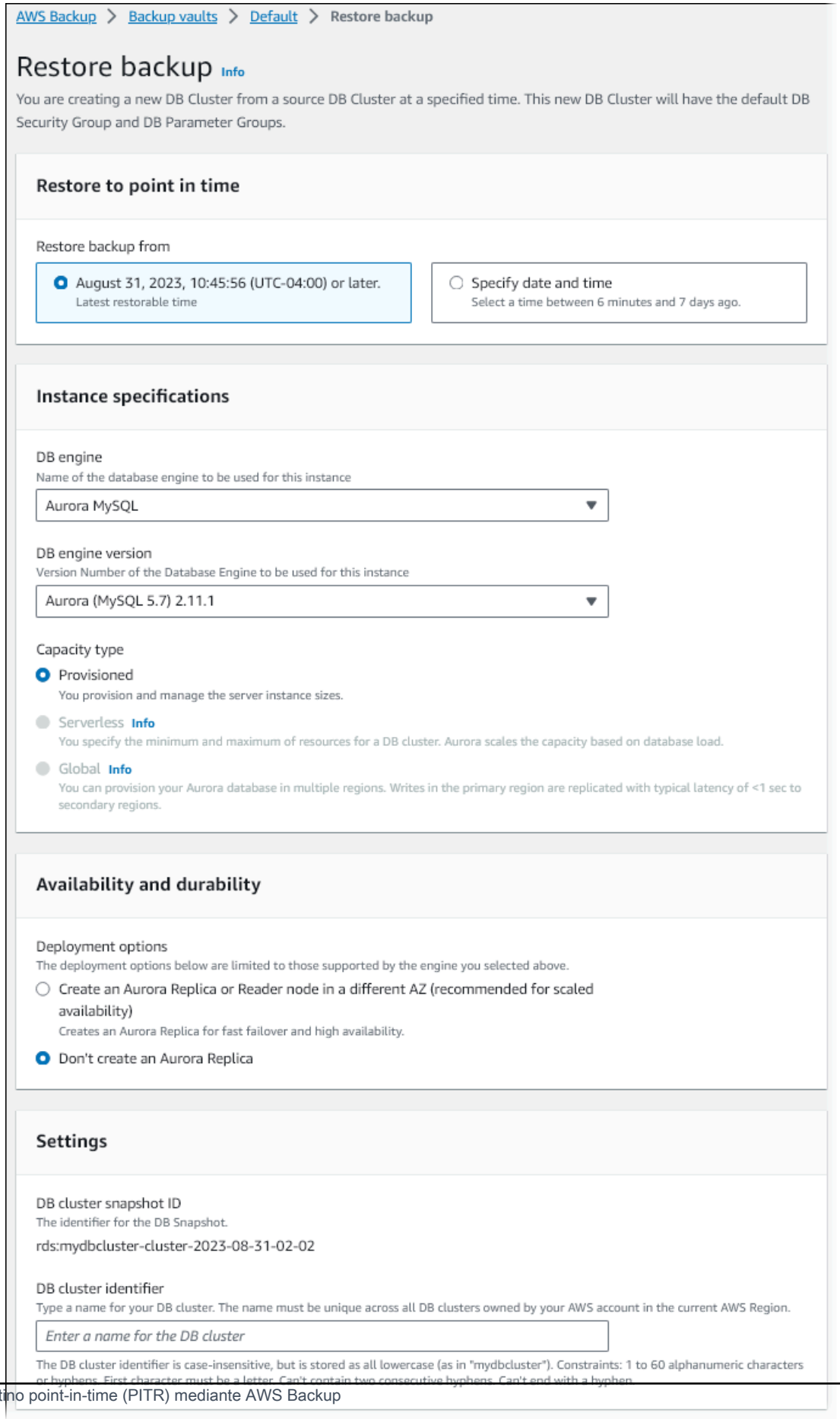
Ripristino da un backup continuo in AWS Backup

1. Accedi alla Console di gestione AWS e apri la console AWS Backup all'indirizzo https://console.aws.amazon.com/backup.
2. Nel riquadro di navigazione scegliere Backup vaults (Vault di backup).
3. Scegli il vault di backup contenente il backup continuo, ad esempio Predefinito.

   Viene visualizzata la pagina dei dettagli del vault di backup.
4. In Punti di ripristino, seleziona il punto di ripristino per il backup automatico.

   Il tipo di backup è Continuo e il nome e `continuous:cluster-`*`AWS-Backup-job-number`*.
5. In Operazioni, sceglie Riavvia.

   Viene visualizzata la pagina Ripristina backup.


AWS Backup > Backup vaults > Default > Restore backup
Restore backup Info
You are creating a new DB Cluster from a source DB Cluster at a specified time. This new DB Cluster will have the default DB Security Group and DB Parameter Groups.
Restore to point in time
Restore backup from
August 31, 2023, 10:45:56 (UTC-04:00) or later.
Latest restorable time
Specify date and time
Select a time between 6 minutes and 7 days ago.
Instance specifications
DB engine
Name of the database engine to be used for this instance
Aurora MySQL
DB engine version
Version Number of the Database Engine to be used for this instance
Aurora (MySQL 5.7) 2.11.1
Capacity type
Provisioned
You provision and manage the server instance sizes.
Serverless Info
You specify the minimum and maximum of resources for a DB cluster. Aurora scales the capacity based on database load.
Global Info
You can provision your Aurora database in multiple regions. Writes in the primary region are replicated with typical latency of <1 sec to secondary regions.
Availability and durability
Deployment options
The deployment options below are limited to those supported by the engine you selected above.
Create an Aurora Replica or Reader node in a different AZ (recommended for scaled availability)
Creates an Aurora Replica for fast failover and high availability.
Don't create an Aurora Replica
Settings
DB cluster snapshot ID
The identifier for the DB Snapshot.
rds:mydbcluster-cluster-2023-08-31-02-02
DB cluster identifier
Type a name for your DB cluster. The name must be unique across all DB clusters owned by your AWS account in the current AWS Region.
Enter a name for the DB cluster
The DB cluster identifier is case-insensitive, but is stored as all lowercase (as in "mydbcluster"). Constraints: 1 to 60 alphanumeric characters or hyphens. First character must be a letter. Can't contain two consecutive hyphens. Can't end with a hyphen.

6. PerRipristina al point-in-time, seleziona Specifica data e ora per eseguire il ripristino a un'ora specifica.
7. Scegli le altre impostazioni necessarie per ripristinare il cluster database, quindi scegli Ripristina backup.

   Viene visualizzata la pagina Processi con il riquadro Processi di ripristino. Un messaggio nella parte superiore della pagina fornisce informazioni sul lavoro di ripristino.

Dopo il ripristino del cluster database, devi aggiungere l'istanza database (scrittura) primaria. Invoca il comando AWS CLI create-db-instance per creare l'istanza principale per il cluster di database. Includi il nome del cluster di database come valore del parametro `--db-cluster-identifier`.

CLI

Per ripristinare il cluster database a un'ora specificata, devi utilizzare il comando AWS CLI start-restore-job. I parametri seguenti sono obbligatori:

- `--recovery-point-arn`: il nome della risorsa Amazon (ARN) per il punto di ripristino da cui eseguire il ripristino.
- `--resource-type`: utilizza `Aurora`.
- `--iam-role-arn`: il nome della risorsa Amazon (ARN) per il ruolo IAM utilizzato per le operazioni AWS Backup.
- `--metadata`: i metadati utilizzati per ripristinare il cluster database. I parametri seguenti sono obbligatori:
  - `DBClusterIdentifier`
  - `Engine`
  - `RestoreToTime` o `UseLatestRestorableTime`

L'esempio seguente mostra come ripristinare un cluster database a un'ora specificata.

```
aws backup start-restore-job \
--recovery-point-arn arn:aws:backup:eu-central-1:123456789012:recovery-
point:continuous:cluster-itsreallyjustanexample1234567890-487278c2 \
--resource-type Aurora \
--iam-role-arn arn:aws:iam::123456789012:role/service-role/AWSBackupDefaultServiceRole
 \
```

```
--metadata '{"DBClusterIdentifier":"backup-pitr-test","Engine":"aurora-
mysql","RestoreToTime":"2023-09-01T17:00:00.000Z"}'
```

L'esempio seguente mostra come ripristinare un cluster database all'ultima ora ripristinabile.

```
aws backup start-restore-job \
--recovery-point-arn arn:aws:backup:eu-central-1:123456789012:recovery-
point:continuous:cluster-itsreallyjustanexample1234567890-487278c2 \
--resource-type Aurora \
--iam-role-arn arn:aws:iam::123456789012:role/service-role/AWSBackupDefaultServiceRole
 \
--metadata '{"DBClusterIdentifier":"backup-pitr-latest","Engine":"aurora-
mysql","UseLatestRestorableTime":"true"}'
```

Dopo il ripristino del cluster database, devi aggiungere l'istanza database (scrittura) primaria. Invoca il comando AWS CLI create-db-instance per creare l'istanza principale per il cluster di database. Includi il nome del cluster di database come valore del parametro `--db-cluster-identifier`.

Eliminazione di una snapshot del cluster di database

Puoi eliminare snapshot del cluster di database gestite da Amazon RDS quando non ti servono più.

> **Note**
>
> Per eliminare backup gestiti da AWS Backup, utilizza la console AWS Backup. Per ulteriori informazioni su AWS Backup, consulta la Guida per sviluppatori di AWS Backup.

Eliminazione di una snapshot del cluster di database

Puoi eliminare una snapshot del cluster di database con la console, AWS CLI, o l'API RDS.

Per eliminare uno snapshot condiviso o pubblico, devi accedere all'account AWS proprietario dello snapshot.

Console

Per eliminare una snapshot del cluster di database

1. Accedi alla Console di gestione AWS e apri la console Amazon RDS all'indirizzo https://console.aws.amazon.com/rds/.
2. Nel riquadro di navigazione, selezionare Snapshots (Snapshot).
3. Scegliere la snapshot del cluster di database da eliminare.
4. Per Actions (Operazioni), scegliere Delete Snapshot (Elimina snapshot).
5. Nella pagina di conferma, scegliere Delete (Elimina).

AWS CLI

Puoi eliminare una snapshot del cluster di database usando il comando AWS CLI copy-db-snapshot.

Le seguenti opzioni vengono utilizzate per eliminare una snapshot del cluster di database.

- `--db-cluster-snapshot-identifier` – L'identificatore per la snapshot del cluster di database.

Example

Il seguente codice elimina la snapshot del cluster di database `mydbclustersnapshot`.

Per Linux, macOS o Unix:

```
aws rds delete-db-cluster-snapshot \
    --db-cluster-snapshot-identifier mydbclustersnapshot
```

Per Windows:

```
aws rds delete-db-cluster-snapshot ^
    --db-cluster-snapshot-identifier mydbclustersnapshot
```

API RDS

Puoi eliminare una snapshot del cluster di database usando l'operazione API di Amazon RDS DeleteDBClusterSnapshot.

I seguenti parametri vengono utilizzati per eliminare uno snapshot del cluster di database.

- `DBClusterSnapshotIdentifier` – L'identificatore per la snapshot del cluster di database.

Tutorial: Ripristino di un cluster di database Amazon Aurora da uno snapshot del cluster di database

Quando si utilizza Amazon Aurora è normale avere un'istanza database utilizzata occasionalmente ma che non serve a tempo pieno. Ad esempio, potrebbe essere necessario utilizzare un cluster di database per conservare i dati di un report che viene eseguito solo trimestralmente. Per risparmiare denaro su tale scenario, è possibile acquisire uno snapshot del cluster di database del cluster di database al termine del report. Eliminare quindi il cluster database e ripristinarlo quando è necessario caricare nuovi dati ed eseguire il report durante il trimestre successivo.

Quando si ripristina un cluster database, si fornisce il nome dello snapshot cluster DB da cui eseguire il ripristino. Quindi si fornisce un nome per il nuovo cluster database creato dall'operazione di ripristino. Per ulteriori informazioni sul ripristino di cluster database da snapshot, consulta Ripristino da uno snapshot cluster database.

In questo tutorial, il cluster database ripristinato viene anche aggiornato da Aurora MySQL versione 2 (compatibile con MySQL 5.7) ad Aurora MySQL versione 3 (compatibile con MySQL 8.0).

Ripristino di un cluster di database a un punto temporale specifico da uno snapshot del cluster di database mediante la console di Amazon RDS o la AWS CLI.

Per ulteriori informazioni sulla gestione della chiave AWS KMS per Amazon RDS, consulta Gestione di AWS KMS key.

Argomenti

- Tutorial: ripristinare un cluster di database da uno snapshot del cluster di database mediante la console di Amazon RDS
- Tutorial: Ripristino di un cluster di database da uno snapshot del cluster di database mediante la AWS CLI

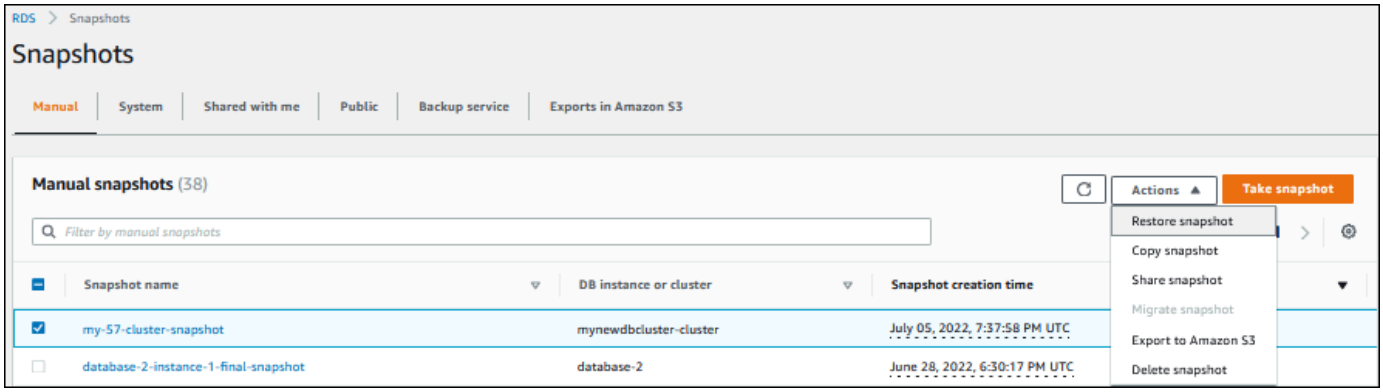
Tutorial: ripristinare un cluster di database da uno snapshot del cluster di database mediante la console di Amazon RDS

Con questo tutorial, è possibile ripristinare un cluster di database da uno snapshot del cluster di database mediante la console di Amazon RDS. Quando si ripristina un cluster di database da uno snapshot utilizzando la Console di gestione AWS, viene anche creata l'istanza database (di scrittura) primaria.

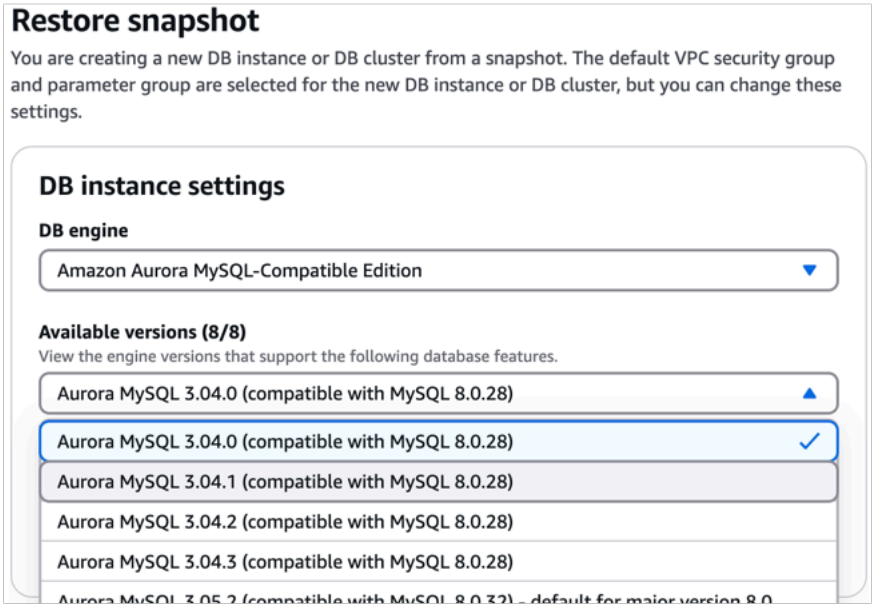
> **Note**
>
> Mentre l'istanza database primaria è in corso di creazione, viene visualizzata come istanza di lettura, ma al termine è un'istanza di scrittura.

Per ripristinare un cluster database da una snapshot cluster DB

1. Accedi alla Console di gestione AWS e apri la console Amazon RDS all'indirizzo https://console.aws.amazon.com/rds/.
2. Nel pannello di navigazione, scegli Snapshot.
3. Scegliere lo snapshot cluster database da cui eseguire il ripristino.
4. Per Actions (Operazioni), selezionare Restore Snapshot (Ripristina snapshot).



   Viene visualizzata la pagina Restore snapshot (Ripristina snapshot).
5. In DB instance settings (Impostazioni dell'istanza database), effettua le seguenti operazioni:
   a. Usa l'impostazione predefinita per DB engine (Motore database).
   b. Per Available versions (Versioni disponibili), scegli una versione compatibile con MySQL-8.0, ad esempio Aurora MySQL 3.04.0 (compatibile con MySQL 8.0.28).



6. Sotto Settings (Impostazioni), per DB instance identifier (Identificatore istanze DB) inserisci il nome univoco da usare per l'istanza database ripristinata, ad esempio **`my-80`**.

   > **Note**
   >
   > Per creare l'identificatore del cluster di database, Amazon RDS aggiunge `-cluster` all'identificatore dell'istanza database specificato.

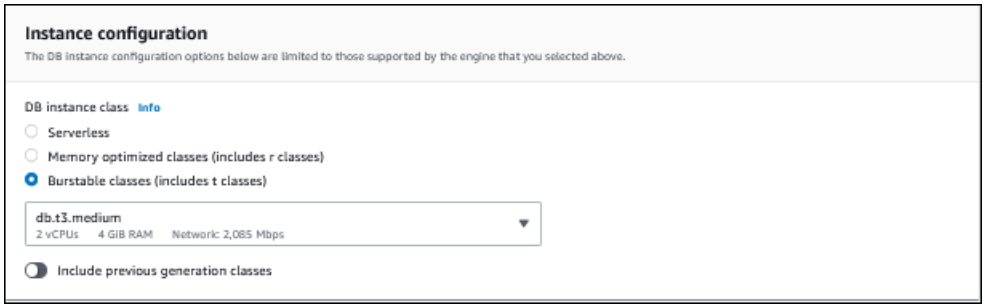
7. Sotto Connectivity (Connettività), utilizza le impostazioni di default per quanto segue:

   - Virtual Private Cloud (VPC) (Cloud privato virtuale (VPC)
   - DB subnet group (Gruppo di sottoreti DB)
   - Accesso pubblico
   - VPC security group (firewall) (Gruppo di sicurezza VPC (firewall)

8. Scegli la DB instance class (Classe di istanza database).

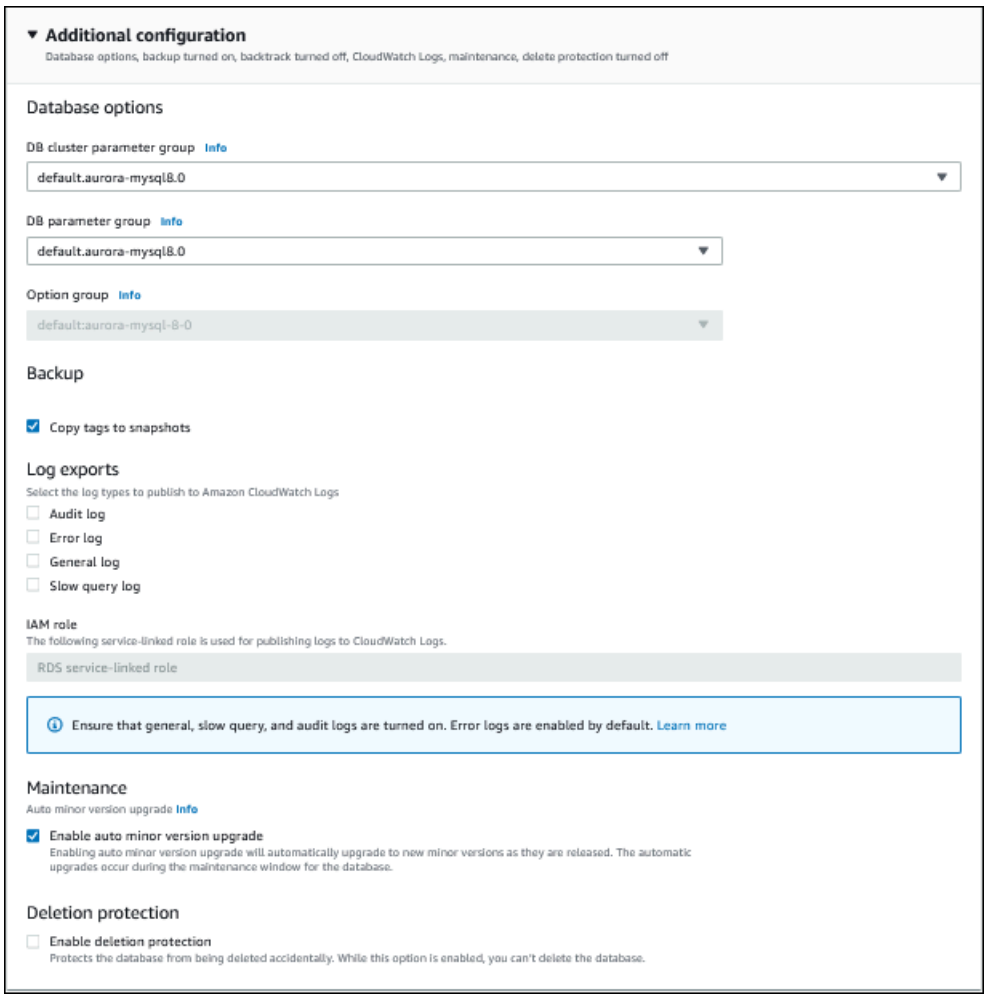
   Per questo tutorial, scegli Burstable classes (includes t classes) (Classi espandibili (include le classi t) e quindi seleziona db.t3.medium.

> **Note**
>
> Consigliamo di utilizzare le classi di istanza database T solo per i server di sviluppo e test o altri server non di produzione. Per ulteriori informazioni sulle classi di istanza T, consulta Tipi di classi di istanza database.



9. Per Database authentication (Autenticazione database), utilizza l'impostazione predefinita.
10. Per Encryption (Crittografia), utilizza le impostazioni di default.

    Se il cluster database di origine per lo snapshot è stato crittografato, anche il cluster database ripristinato viene crittografato. Non è possibile renderla non crittografato.
11. Espandi Additional configuration (Configurazione aggiuntiva) nella parte inferiore della pagina.



12. Effettua le scelte riportate di seguito:

    a. Per questo tutorial, utilizza il valore predefinito per DB cluster parameter group (Gruppo di parametri del cluster DB).

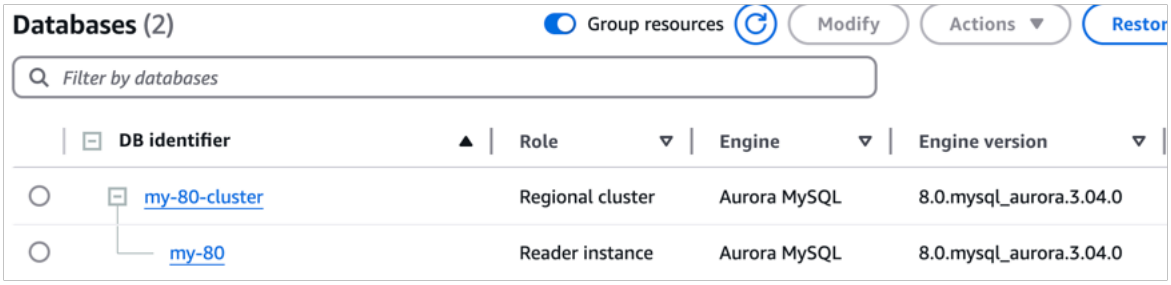
    b. Per questo tutorial, utilizza il valore predefinito per DB parameter group (Gruppo di parametri DB).

    c. Per Log exports (Esportazioni log), seleziona tutte le caselle di controllo.

    d. Per Deletion protection (Protezione da eliminazione), seleziona la casella di controllo Enable deletion protection (Abilita protezione da eliminazione).

13. Selezionare Ripristina istanza database.

La pagina Databases (Database) visualizza il cluster database ripristinato, con uno stato `Creating`.



Mentre l'istanza database primaria è in corso di creazione, viene visualizzata come istanza di lettura, ma al termine è un'istanza di scrittura.

Tutorial: Ripristino di un cluster di database da uno snapshot del cluster di database mediante la AWS CLI

Con questo tutorial, è possibile ripristinare un cluster di database da una snapshot del cluster di database mediante la AWS CLI. Il ripristino di un cluster di database da uno snapshot mediante la AWS CLI comprende due passaggi:

1. Ripristino del cluster database usando il comando restore-db-cluster-from-snapshot
2. Creazione dell'istanza database (scrittura) primaria usando il comando create-db-instance

Ripristino del cluster database

Si utilizza il comando `restore-db-cluster-from-snapshot`. Sono richieste le seguenti opzioni:

- `--db-cluster-identifier`: il nome del cluster database ripristinato.
- `--snapshot-identifier`: il nome dello snapshot DB da cui eseguire il ripristino.
- `--engine`: il motore di database del cluster database ripristinato. Deve essere compatibile con il motore di database del cluster database di origine.

  Le scelte sono le seguenti:

  - `aurora-mysql`: compatibile con Aurora MySQL 5.7 e 8.0.
  - `aurora-postgresql`: compatibile con Aurora PostgreSQL.

  In questo esempio viene utilizzato `aurora-mysql`.
- `--engine-version`: la versione del cluster database ripristinato. In questo esempio, si utilizza una versione compatibile con MySQL-8.0.

Nell'esempio seguente viene ripristinato un cluster database compatibile con Aurora MySQL 8.0 denominato `my-new-80-cluster` da uno snapshot cluster DB denominato `my-57-cluster-snapshot`.

Per ripristinare il cluster database

- Utilizzare uno dei seguenti comandi.

  Per Linux, macOS o Unix:

```
aws rds restore-db-cluster-from-snapshot \
    --db-cluster-identifier my-new-80-cluster \
    --snapshot-identifier my-57-cluster-snapshot \
    --engine aurora-mysql \
    --engine-version 8.0.mysql_aurora.3.02.0
```

  Per Windows:

```
aws rds restore-db-cluster-from-snapshot ^
    --db-cluster-identifier my-new-80-cluster ^
    --snapshot-identifier my-57-cluster-snapshot ^
    --engine aurora-mysql ^
    --engine-version 8.0.mysql_aurora.3.02.0
```

L'output è simile a quello riportato di seguito.

```
{
    "DBCluster": {
        "AllocatedStorage": 1,
        "AvailabilityZones": [
            "eu-central-1b",
            "eu-central-1c",
            "eu-central-1a"
        ],
        "BackupRetentionPeriod": 14,
        "DatabaseName": "",
        "DBClusterIdentifier": "my-new-80-cluster",
        "DBClusterParameterGroup": "default.aurora-mysql8.0",
        "DBSubnetGroup": "default",
        "Status": "creating",
```

```
        "Endpoint": "my-new-80-cluster.cluster-############.eu-
central-1.rds.amazonaws.com",
        "ReaderEndpoint": "my-new-80-cluster.cluster-ro-############.eu-
central-1.rds.amazonaws.com",
        "MultiAZ": false,
        "Engine": "aurora-mysql",
        "EngineVersion": "8.0.mysql_aurora.3.02.0",
        "Port": 3306,
        "MasterUsername": "admin",
        "PreferredBackupWindow": "01:55-02:25",
        "PreferredMaintenanceWindow": "thu:21:14-thu:21:44",
        "ReadReplicaIdentifiers": [],
        "DBClusterMembers": [],
        "VpcSecurityGroups": [
            {
                "VpcSecurityGroupId": "sg-########",
                "Status": "active"
            }
        ],
        "HostedZoneId": "Z1RLNU0EXAMPLE",
        "StorageEncrypted": true,
        "KmsKeyId": "arn:aws:kms:eu-central-1:123456789012:key/#######-5ccc-49cc-8aaa-
############",
        "DbClusterResourceId": "cluster-ZZ12345678ITSJUSTANEXAMPLE",
        "DBClusterArn": "arn:aws:rds:eu-central-1:123456789012:cluster:my-new-80-
cluster",
        "AssociatedRoles": [],
        "IAMDatabaseAuthenticationEnabled": false,
        "ClusterCreateTime": "2022-07-05T20:45:42.171000+00:00",
        "EngineMode": "provisioned",
        "DeletionProtection": false,
        "HttpEndpointEnabled": false,
        "CopyTagsToSnapshot": false,
        "CrossAccountClone": false,
        "DomainMemberships": [],
        "TagList": []
    }
}
```

Creazione dell'istanza database (scrittura) primaria

Per creare l'istanza database (scrittura) primaria, si utilizza il comando `create-db-instance`. Sono richieste le seguenti opzioni:

- `--db-cluster-identifier`: il nome del cluster database ripristinato.
- `--db-instance-identifier`: il nome dell'istanza database primaria.
- `--db-instance-class`: la classe istanza dell'istanza database primaria. In questo esempio viene utilizzato `db.t3.medium`.

> **Note**
>
> Consigliamo di utilizzare le classi di istanza database T solo per i server di sviluppo e test o altri server non di produzione. Per ulteriori informazioni sulle classi di istanza T, consulta Tipi di classi di istanza database.

- `--engine`- il motore di database dell'istanza database primaria. Deve essere identico a quello utilizzato dal cluster database ripristinato.

  Le scelte sono le seguenti:

  - `aurora-mysql`: compatibile con Aurora MySQL 5.7 e 8.0.
  - `aurora-postgresql`: compatibile con Aurora PostgreSQL.

  In questo esempio viene utilizzato `aurora-mysql`.

Nell'esempio seguente viene creata un'istanza database (scrittura) primaria denominata `my-new-80-cluster-instance` nel cluster database compatibile con Aurora MySQL 8.0 denominato `my-new-80-cluster`.

Per creare l'istanza database primaria

- Utilizzare uno dei seguenti comandi.

  Per Linux, macOS o Unix:

```
aws rds create-db-instance \
    --db-cluster-identifier my-new-80-cluster \
    --db-instance-identifier my-new-80-cluster-instance \
    --db-instance-class db.t3.medium \
    --engine aurora-mysql
```

  Per Windows:

```
aws rds create-db-instance ^
    --db-cluster-identifier my-new-80-cluster ^
    --db-instance-identifier my-new-80-cluster-instance ^
    --db-instance-class db.t3.medium ^
    --engine aurora-mysql
```

L'output è simile a quello riportato di seguito.


```
{
    "DBInstance": {
        "DBInstanceIdentifier": "my-new-80-cluster-instance",
        "DBInstanceClass": "db.t3.medium",
        "Engine": "aurora-mysql",
        "DBInstanceStatus": "creating",
        "MasterUsername": "admin",
        "AllocatedStorage": 1,
        "PreferredBackupWindow": "01:55-02:25",
        "BackupRetentionPeriod": 14,
        "DBSecurityGroups": [],
        "VpcSecurityGroups": [
            {
                "VpcSecurityGroupId": "sg-########",
                "Status": "active"
            }
        ],
        "DBParameterGroups": [
            {
                "DBParameterGroupName": "default.aurora-mysql8.0",
                "ParameterApplyStatus": "in-sync"
            }
        ],
        "DBSubnetGroup": {
            "DBSubnetGroupName": "default",
            "DBSubnetGroupDescription": "default",
            "VpcId": "vpc-2305ca49",
            "SubnetGroupStatus": "Complete",
            "Subnets": [
                {
                    "SubnetIdentifier": "subnet-########",
                    "SubnetAvailabilityZone": {
                        "Name": "eu-central-1a"
```

```
                    },
                    "SubnetOutpost": {},
                    "SubnetStatus": "Active"
                },
                {
                    "SubnetIdentifier": "subnet-########",
                    "SubnetAvailabilityZone": {
                        "Name": "eu-central-1b"
                    },
                    "SubnetOutpost": {},
                    "SubnetStatus": "Active"
                },
                {
                    "SubnetIdentifier": "subnet-########",
                    "SubnetAvailabilityZone": {
                        "Name": "eu-central-1c"
                    },
                    "SubnetOutpost": {},
                    "SubnetStatus": "Active"
                }
            ]
        },
        "PreferredMaintenanceWindow": "sat:02:41-sat:03:11",
        "PendingModifiedValues": {},
        "MultiAZ": false,
        "EngineVersion": "8.0.mysql_aurora.3.02.0",
        "AutoMinorVersionUpgrade": true,
        "ReadReplicaDBInstanceIdentifiers": [],
        "LicenseModel": "general-public-license",
        "OptionGroupMemberships": [
            {
                "OptionGroupName": "default:aurora-mysql-8-0",
                "Status": "in-sync"
            }
        ],
        "PubliclyAccessible": false,
        "StorageType": "aurora",
        "DbInstancePort": 0,
        "DBClusterIdentifier": "my-new-80-cluster",
        "StorageEncrypted": true,
        "KmsKeyId": "arn:aws:kms:eu-central-1:534026745191:key/#######-5ccc-49cc-8aaa-
############",
        "DbiResourceId": "db-5C6UT5PU0YETANOTHEREXAMPLE",
        "CACertificateIdentifier": "rds-ca-2019",
```

```
        "DomainMemberships": [],
        "CopyTagsToSnapshot": false,
        "MonitoringInterval": 0,
        "PromotionTier": 1,
        "DBInstanceArn": "arn:aws:rds:eu-central-1:123456789012:db:my-new-80-cluster-
instance",
        "IAMDatabaseAuthenticationEnabled": false,
        "PerformanceInsightsEnabled": false,
        "DeletionProtection": false,
        "AssociatedRoles": [],
        "TagList": []
    }
}
```

Monitoraggio dei parametri in un cluster di database Amazon Aurora

Amazon Aurora utilizza un cluster di server di database replicati. Il monitoraggio di un cluster Aurora in genere richiede il controllo dell'integrità di più istanze database. Le istanze potrebbero avere ruoli specializzati, gestendo principalmente operazioni di scrittura, solo operazioni di lettura o una combinazione di entrambe. È inoltre possibile monitorare lo stato generale del cluster misurando il ritardo di replica. Questa è la quantità di tempo per le modifiche apportate da un'istanza database per essere disponibili per le altre istanze.

Argomenti

- Piano di monitoraggio
- Baseline delle prestazioni
- Linee guida per le prestazioni
- Monitoraggio degli strumenti di Amazon Aurora
- Visualizzazione dello stato del cluster
- Suggerimenti di Amazon Aurora
- Visualizzazione delle metriche nella console Amazon RDS
- Visualizzazione delle metriche combinate con la dashboard di Approfondimenti sulle prestazioni
- Monitoraggio dei parametri di Amazon Aurora con Amazon CloudWatch
- Monitoraggio dei database Amazon Aurora con CloudWatch Database Insights
- Monitoraggio del carico DB con Performance Insights su Amazon Aurora
- Analisi delle anomalie prestazionali di Aurora con DevOps Amazon Guru per Amazon RDS
- Monitoraggio dei parametri del sistema operativo con il monitoraggio avanzato
- Riferimento per i parametri per Amazon Aurora

Piano di monitoraggio

Prima di iniziare il monitoraggio di , crea un piano di monitoraggio. Questo piano deve rispondere alle domande seguenti:

- Quali sono gli obiettivi del monitoraggio?

- Quali risorse verranno monitorate?
- Con quale frequenza eseguirai il monitoraggio di queste risorse?
- Quali strumenti di monitoraggio verranno usati?
- Chi eseguirà le attività di monitoraggio?
- Chi deve ricevere la notifica quando si verifica un problema?

Baseline delle prestazioni

Per raggiungere gli obiettivi di monitoraggio è necessario stabilire una baseline. Pertanto devi misurare le prestazioni in condizioni di carico diverse e in diversi momenti nell'ambiente Amazon Aurora. Puoi monitorare parametri come i seguenti:

- Throughput di rete
- Connessioni client
- I/O per operazioni di lettura, scrittura o metadati
- Saldi credito burst per le istanze database

Ti consigliamo di archiviare i dati cronologici delle prestazioni per Amazon Aurora. Utilizzando i dati archiviati puoi confrontare le prestazioni correnti con le tendenze passate. Puoi distinguere i normali modelli di prestazioni dalle anomalie e definire i metodi per risolvere i problemi.

Linee guida per le prestazioni

In generale, i valori accettabili per i parametri delle prestazioni dipendono dalle attività dell'applicazione in relazione alla tua baseline. Indagare le variazioni della baseline coerenti o che rappresentano dei trend. I seguenti parametri sono spesso fonte di problemi di prestazioni:

- Consumo elevato di CPU o RAM – Valori elevati per il consumo di CPU o RAM potrebbero essere appropriati, purché tengano conto degli obiettivi dell'applicazione (come throughput o concorrenza) e siano previsti.
- Consumo dello spazio su disco: esamina il consumo dello spazio su disco se lo spazio usato supera costantemente l'85% dello spazio su disco totale. Verifica se è possibile eliminare dati dall'istanza o archiviare dati su un sistema diverso per liberare spazio.

- Traffico di rete – Per il traffico di rete, rivolgiti al tuo amministratore di sistema per identificare il throughput previsto per la rete del dominio e la connessione Internet. Indaga il traffico di rete se il throughput è costantemente al di sotto del valore previsto.
- Connessioni al database – Se noti un numero elevato di connessioni utente insieme a un peggioramento delle prestazioni e del tempo di risposta dell'istanza, valuta se limitare le connessioni al database . Il numero ideale di connessioni utente per l'istanza database dipende dalla classe di istanza e dalla complessità delle operazioni eseguite. Per determinare il numero di connessioni di database, associa l'istanza database a un gruppo di parametri dove il parametro `User Connections` è impostato su un valore diverso da 0 (illimitato). Puoi utilizzare un gruppo di parametri esistente o crearne uno nuovo. Per ulteriori informazioni, consulta Gruppi di parametri per Amazon Aurora.
- Parametri di IOPS: poiché i valori previsti per i parametri di IOPS dipendono dalle specifiche del disco e dalla configurazione del server, usa i valori di riferimento per identificare i comportamenti tipici. Verifica se i valori sono costantemente diversi dalla baseline. Per prestazioni IOPS ottimali, verifica che il working set tipico possa essere caricato nella memoria per ridurre al minimo le operazioni di lettura e scrittura.

Quando le prestazioni non rientrano nella baseline stabilita, potrebbe essere necessario apportare modifiche per ottimizzare la disponibilità del database per il carico di lavoro. Ad esempio, potrebbe essere necessario modificare la classe di istanza dell'istanza database. In alternativa, potrebbe essere necessario modificare il numero di istanze database e leggere le repliche disponibili per i client.

Monitoraggio degli strumenti di Amazon Aurora

Il monitoraggio è importante per mantenere l'affidabilità, la disponibilità e le prestazioni di Amazon Aurora e delle altre soluzioni AWS. AWS fornisce strumenti di monitoraggio per controllare Amazon Aurora, segnalare eventuali problemi ed eseguire operazioni automatiche quando appropriato.

Argomenti

- Strumenti di monitoraggio automatici
- Strumenti di monitoraggio manuali

Strumenti di monitoraggio automatici

Si consiglia di automatizzare il più possibile i processi di monitoraggio.

Argomenti

- Stato di cluster Amazon Aurora e suggerimenti
- Parametri di Amazon CloudWatch per Amazon Aurora
- Amazon RDS Performance Insights e monitoraggio del sistema operativo
- Servizi integrati

Stato di cluster Amazon Aurora e suggerimenti

Per controllare Amazon Aurora e segnalare l'eventuale presenza di problemi, puoi usare gli strumenti automatici seguenti:

- Stato del cluster di Amazon Aurora: visualizzare i dettagli sullo stato corrente del cluster utilizzando la console Amazon RDS, la AWS CLI o l'API RDS.
- Suggerimenti di Amazon Aurora: consente di rispondere ai suggerimenti automatici per le risorse di database, come istanze database, cluster di database, e gruppi di parametri del cluster di database. Per ulteriori informazioni, consulta Suggerimenti di Amazon Aurora.

Parametri di Amazon CloudWatch per Amazon Aurora

Amazon Aurora si integra con Amazon CloudWatch per offrire funzionalità di monitoraggio aggiuntive.

- Amazon CloudWatch: questo servizio monitora in tempo reale le risorse AWS e le applicazioni in esecuzione su AWS. Puoi utilizzare le seguenti caratteristiche Amazon CloudWatch con Amazon Aurora:
  - I parametri Amazon CloudWatch – Amazon Aurora invia automaticamente parametri a CloudWatch ogni minuto per ogni database attivo. Non ricevi addebiti aggiuntivi per parametri Amazon RDS in CloudWatch. Per ulteriori informazioni, consulta CloudWatch Parametri Amazon per Amazon Aurora
  - Allarmi Amazon CloudWatch – È possibile visualizzare un singolo parametro Amazon Aurora in un determinato periodo di tempo. È quindi possibile eseguire una o più operazioni in base al valore del parametro rispetto a una soglia impostata.

Amazon RDS Performance Insights e monitoraggio del sistema operativo

Per monitorare le prestazioni di Amazon Aurora, puoi usare i seguenti strumenti automatici:

- Performance Insights Amazon RDS – Aiuta a valutare in modo rapido il carico del database e a determinare quando e dove intervenire. Per ulteriori informazioni, consulta Monitoraggio del carico DB con Performance Insights su Amazon Aurora.
- Monitoraggio avanzato di Amazon RDS – Osserva i parametri in tempo reale per il sistema operativo. Per ulteriori informazioni, consulta Monitoraggio dei parametri del sistema operativo con il monitoraggio avanzato.

Servizi integrati

I seguenti servizi AWS sono integrati con Amazon Aurora:

- Amazon EventBridge è un servizio bus di eventi serverless che semplifica la connessione delle applicazioni ai dati provenienti da un'ampia gamma di origini. Per ulteriori informazioni, consulta Monitoraggio di eventi Amazon Aurora.
- Amazon CloudWatch Logs consente di monitorare, archiviare e accedere ai file di log dalle istanze Amazon Aurora, CloudTrail o da altre origini. Per ulteriori informazioni, consulta Monitoraggio dei file di log di Amazon Aurora.
- AWS CloudTrail acquisisce le chiamate API e gli eventi correlati effettuati da o per conto del tuo Account AWSe fornisce i file di log a un bucket Simple Storage Service (Amazon S3) specificato. Per ulteriori informazioni, consulta Monitoraggio delle chiamate API di Amazon Aurora in AWS CloudTrail.

- I flussi di attività di database sono una caratteristica di Amazon Aurora che fornisce un flusso quasi in tempo reale dell'attività cluster di database . Per ulteriori informazioni, consulta Monitoraggio di Amazon Aurora tramite i flussi di attività del database.
- DevOps Guru per RDS è una caratteristica di Amazon DevOps Guru che applica il machine learning ai parametri di Performance Insights per i database Amazon Aurora. Per ulteriori informazioni, consulta Analisi delle anomalie prestazionali di Aurora con DevOps Amazon Guru per Amazon RDS.

Strumenti di monitoraggio manuali

È necessario monitorare manualmente gli elementi non coperti dagli allarmi CloudWatch. Amazon RDS, CloudWatch, AWS Trusted Advisor e altri pannelli di controllo della console AWS forniscono una panoramica dello stato dell'ambiente AWS. Consigliamo anche di controllare i file di log nell'istanza database.

- Dalla console Amazon RDS, puoi monitorare i seguenti elementi per le risorse:
  - Il numero di connessioni a un'istanza database
  - Il numero di operazioni di lettura e scrittura a un'istanza database
  - Quantità di storage utilizzato al momento dall'istanza database
  - La quantità di memoria e CPU utilizzati per un'istanza database
  - La quantità di traffico di rete verso e da un'istanza database
- Dal pannello di controllo Trusted Advisor, puoi rivedere i seguenti controlli di ottimizzazione dei costi, sicurezza, tolleranza ai guasti e miglioramento delle prestazioni:
  - Istanze database Amazon RDS inattive
  - Rischio accesso gruppo di sicurezza Amazon RDS
  - Backup Amazon RDS
  - Multi-AZ Amazon RDS
  - Accessibilità dell’istanza database Aurora

  Per ulteriori informazioni su questi controlli, consulta best practice Trusted Advisor (Controlli).
- La home page di CloudWatch mostra le seguenti informazioni:
  - Stato e allarmi attuali
  - Grafici degli allarmi e delle risorse
  - Stato di integrità dei servizi

Inoltre, è possibile utilizzare CloudWatch per effettuare le seguenti operazioni:

- Crea pannelli di controllo personalizzati per monitorare i servizi rilevanti.
- Creare grafici dei dati dei parametri per la risoluzione di problemi e il rilevamento di tendenze.
- Ricercare e analizzare tutti i parametri delle risorse AWS.
- Creare e modificare gli allarmi per ricevere le notifiche dei problemi.

Visualizzazione dello stato del cluster

Utilizzando la console Amazon RDS, è possibile accedere rapidamente allo stato del cluster database.

Argomenti

- Visualizzazione di un cluster di database Amazon Aurora
- Visualizzazione dello stato del cluster del DB
- Visualizzazione dello stato dell'istanza database di in un cluster Aurora

Visualizzazione di un cluster di database Amazon Aurora

Hai diverse opzioni per visualizzare le informazioni sui tuoi cluster di database Amazon Aurora e sulle istanze database nei tuoi cluster di database.

- Puoi visualizzare i cluster di database e le istanze database nella console Amazon RDS scegliendo Databases (Database) dal riquadro di navigazione.
- È possibile ottenere informazioni su cluster e istanze DB utilizzando (). AWS Command Line Interface AWS CLI
- Puoi ottenere informazioni sul cluster di database e le istanze database utilizzando l'API Amazon RDS.

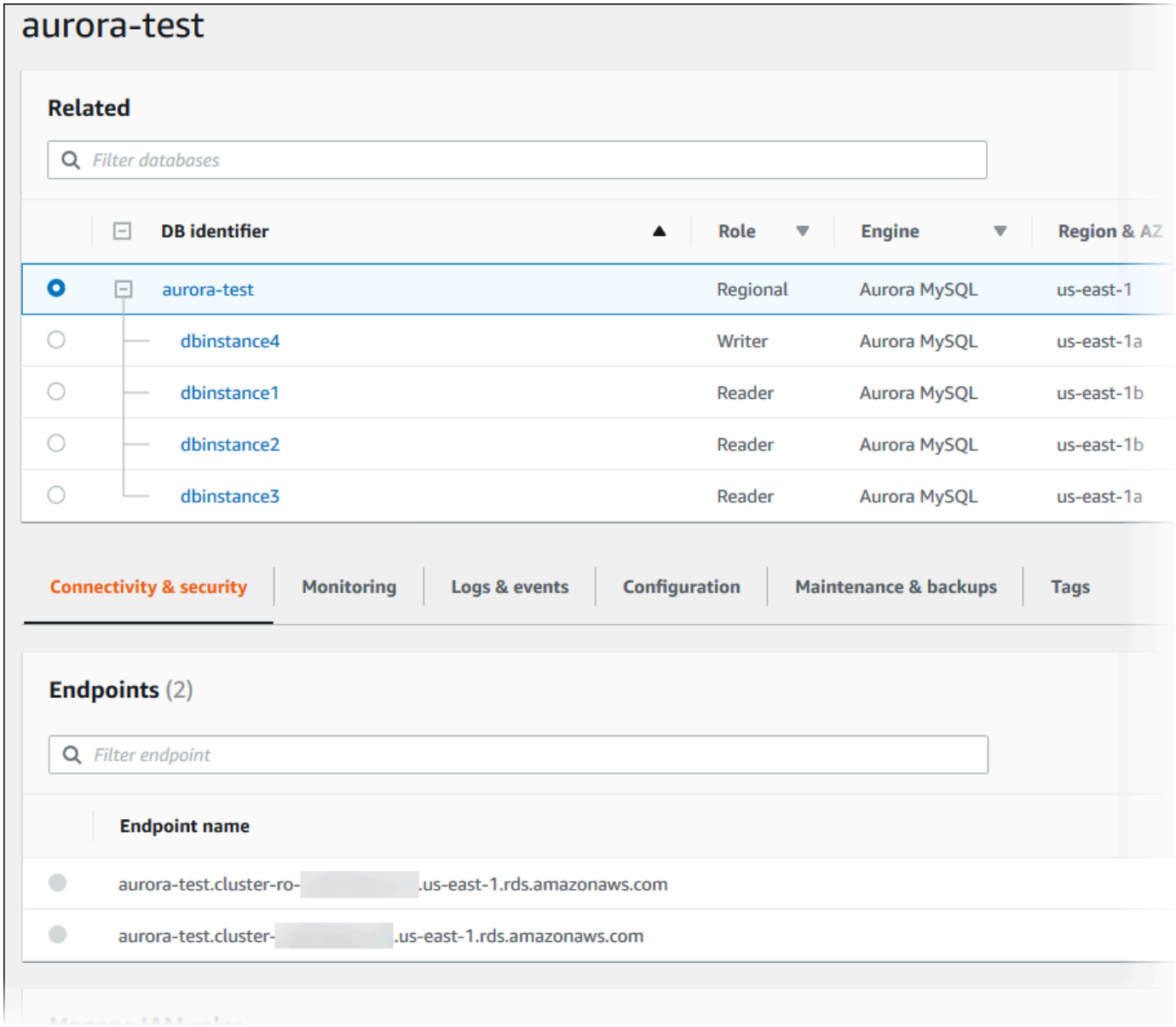
Console

Nella Amazon RDS console, puoi vedere i dettagli di un cluster di dB scegliendo Database dal riquadro di navigazione della console. Puoi anche vedere i dettagli delle istanze database che fanno parte di un cluster di database Amazon Aurora.

Per visualizzare o modificare i cluster DB nella console di Amazon RDS

1. Accedi a Console di gestione AWS e apri la console Amazon RDS all'indirizzo https://console.aws.amazon.com/rds/.
2. Nel pannello di navigazione, seleziona Database.
3. Scegli il nome del cluster di database Aurora che desideri visualizzare dall'elenco.

   Ad esempio, di seguito è riportata l'immagine della pagina dei dettagli per il cluster di database denominato `aurora-test`. Il cluster di database ha quattro istanze DB mostrate nell'elenco DB indentifier (Identificatore database). L'istanza database di scrittura, `dbinstance4`, è l'istanza database primaria per il cluster di database.



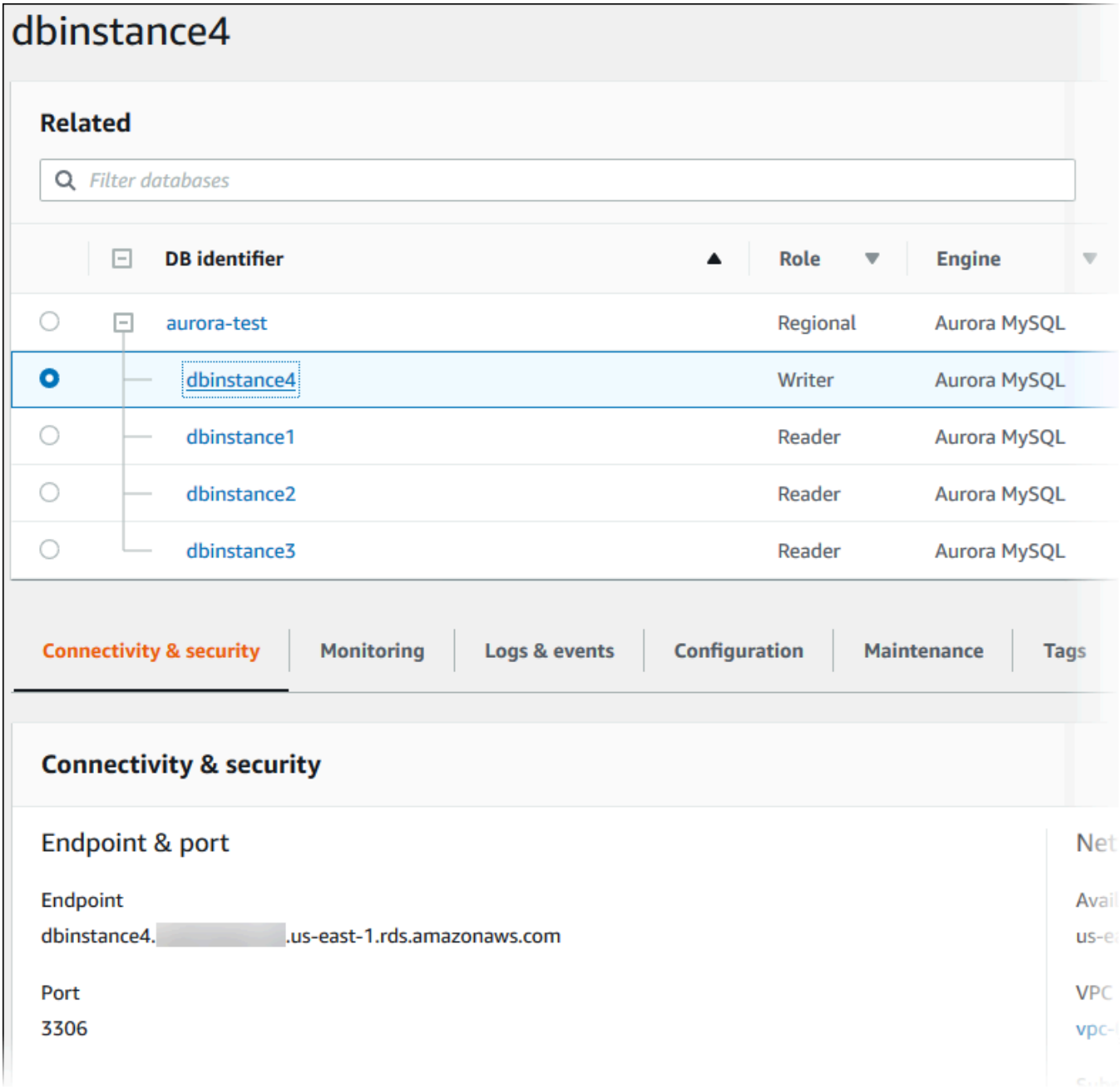
4. Per modificare un cluster di database, selezionalo nell'elenco e scegli Modify (Modifica).

Per visualizzare o modificare istanze database di un cluster di database nella console di Amazon RDS

1. Accedi a Console di gestione AWS e apri la console Amazon RDS all'indirizzo https://console.aws.amazon.com/rds/.
2. Nel pannello di navigazione, seleziona Database.
3. Esegui una di queste operazioni:

- Per visualizzare un'istanza database, scegline una dall'elenco che sia un membro del cluster di database Aurora.

  Ad esempio, se scegli l'identificatore istanze database `dbinstance4`, la console mostra la pagina dei dettagli per l'istanza database `dbinstance4`, come mostrato nell'immagine seguente.



- Per modificare un'istanza database, sceglila dall'elenco e seleziona Modify (Modifica). Per ulteriori informazioni sulla modifica di un cluster di database, consultare Modifica di un cluster database Amazon Aurora.

AWS CLI

Per visualizzare le informazioni sul cluster DB utilizzando il AWS CLI, usa il describe-db-clusterscomando. Ad esempio, il AWS CLI comando seguente elenca le informazioni sul cluster DB per tutti i cluster DB nell'`us-east-1`area di modifica per l' AWS account configurato.

```
aws rds describe-db-clusters --region us-east-1
```

Il comando restituisce il seguente output se si AWS CLI è configurati per l'output JSON.

```
{
    "DBClusters": [
        {
            "Status": "available",
            "Engine": "aurora-mysql",
            "Endpoint": "sample-cluster1.cluster-123456789012.us-
east-1.rds.amazonaws.com"
            "AllocatedStorage": 1,
            "DBClusterIdentifier": "sample-cluster1",
            "MasterUsername": "mymasteruser",
            "EarliestRestorableTime": "2023-03-30T03:35:42.563Z",
            "DBClusterMembers": [
                {
                    "IsClusterWriter": false,
                    "DBClusterParameterGroupStatus": "in-sync",
                    "DBInstanceIdentifier": "sample-replica"
                },
                {
                    "IsClusterWriter": true,
                    "DBClusterParameterGroupStatus": "in-sync",
                    "DBInstanceIdentifier": "sample-primary"
                }
            ],
            "Port": 3306,
            "PreferredBackupWindow": "03:34-04:04",
            "VpcSecurityGroups": [
                {
```

```
                    "Status": "active",
                    "VpcSecurityGroupId": "sg-ddb65fec"
                }
            ],
            "DBSubnetGroup": "default",
            "StorageEncrypted": false,
            "DatabaseName": "sample",
            "EngineVersion": "5.7.mysql_aurora.2.11.0",
            "DBClusterParameterGroup": "default.aurora-mysql5.7",
            "BackupRetentionPeriod": 1,
            "AvailabilityZones": [
                "us-east-1b",
                "us-east-1c",
                "us-east-1d"
            ],
            "LatestRestorableTime": "2023-03-31T20:06:08.903Z",
            "PreferredMaintenanceWindow": "wed:08:15-wed:08:45"
        },
        {
            "Status": "available",
            "Engine": "aurora-mysql",
            "Endpoint": "aurora-sample.cluster-123456789012.us-
east-1.rds.amazonaws.com",
            "AllocatedStorage": 1,
            "DBClusterIdentifier": "aurora-sample-cluster",
            "MasterUsername": "mymasteruser",
            "EarliestRestorableTime": "2023-03-30T10:21:34.826Z",
            "DBClusterMembers": [
                {
                    "IsClusterWriter": false,
                    "DBClusterParameterGroupStatus": "in-sync",
                    "DBInstanceIdentifier": "aurora-replica-sample"
                },
                {
                    "IsClusterWriter": true,
                    "DBClusterParameterGroupStatus": "in-sync",
                    "DBInstanceIdentifier": "aurora-sample"
                }
            ],
            "Port": 3306,
            "PreferredBackupWindow": "10:20-10:50",
            "VpcSecurityGroups": [
                {
                    "Status": "active",
```

```
                    "VpcSecurityGroupId": "sg-55da224b"
                }
            ],
            "DBSubnetGroup": "default",
            "StorageEncrypted": false,
            "DatabaseName": "sample",
            "EngineVersion": "5.7.mysql_aurora.2.11.0",
            "DBClusterParameterGroup": "default.aurora-mysql5.7",
            "BackupRetentionPeriod": 1,
            "AvailabilityZones": [
                "us-east-1b",
                "us-east-1c",
                "us-east-1d"
            ],
            "LatestRestorableTime": "2023-03-31T20:00:11.491Z",
            "PreferredMaintenanceWindow": "sun:03:53-sun:04:23"
        }
    ]
}
```

API RDS

Per visualizzare le informazioni sui cluster DB utilizzando l'API Amazon RDS, utilizza l'DBClustersoperazione Descrivi.

Visualizzazione dello stato del cluster del DB

Lo stato di un cluster DB indica il suo attuale stato operativo. Puoi visualizzare lo stato di un cluster DB e delle istanze del cluster utilizzando la console Amazon RDS AWS CLI, o l'API.

> **Note**
>
> Aurora usa anche un altro stato denominato stato di manutenzione, mostrato nella colonna Maintenance (Manutenzione) della console Amazon RDS. Questo valore indica lo stato delle patch di manutenzione da applicare a un cluster DB. Lo stato della manutenzione è indipendente dallo stato del cluster DB. Per ulteriori informazioni sullo stato della manutenzione, consulta Applicazione di aggiornamenti a un cluster di database.

Puoi trovare i valori di stato possibili per i cluster DB nella tabella seguente.

Stato cluster DB	Fatturata	Descrizione
Disponibilità	Fatturato	Il cluster DB è disponibile per le modifiche . Quando un cluster Aurora Serverless è disponibile e sospeso, viene addebitato solo lo spazio di archiviazione.
Backup	Fatturata	Il cluster DB è attualmente sottoposto a backup.
Backtracking	Fatturata	Il cluster DB è attualmente sottoposto a backtrack. Questo stato si applica solo ad Aurora MySQL.
Cloning-failed (Clonazione non riuscita)	Non fatturata	Errore nella clonazione di un cluster DB.
Creazione in corso	Non fatturata	Il cluster DB è in fase di creazione. Non è possibile accedere al cluster DB mentre è in fase di creazione.
Eliminazione in corso	Non fatturata	Il cluster DB è in fase di eliminazione.

Stato cluster DB	Fatturata	Descrizione
Failing-over (Failover)	Fatturata	È in corso di esecuzione un failover da parte dell'istanza primaria in una replica di Aurora.
Inaccessible-encryption-credentials	Non fatturato	Non è possibile accedere o recuperare il cluster DB AWS KMS key utilizzato per crittografare o decrittografare.
Inaccessible-encryption-credentials-recoverable	Fatturato per storage	Non è possibile accedere alla chiave KMS utilizzata per crittografare o decrittografare il cluster di database. Tuttavia, se la chiave KMS è attiva, il riavvio del cluster di database può ripristinarla. Per ulteriori informazioni, consulta Creazione di un cluster di database Amazon Aurora.
Maintenance (Manutenzione)	Fatturata	È in corso l'applicazione da parte di Amazon RDS di un aggiornamento di manutenzione al cluster di database. Questo stato viene utilizzato per la manutenzione a livello di cluster DB pianificata in anticipo da RDS.
Migrating (Migrazione in corso)	Fatturata	Uno snapshot cluster DB è stato ripristinato in un cluster DB.
Migration-failed (Migrazione non riuscita)	Non fatturata	Una migrazione non è riuscita.
Modifying (Modifica in corso)	Fatturata	È in corso la modifica del cluster DB in seguito alla richiesta da parte di un cliente.
Promoting (Promozione in corso)	Fatturata	Una replica di lettura è in corso di promozione su un cluster di database standalone.
Preparing-data-migration	Fatturato	Amazon RDS si sta preparando a migrare i dati su Aurora.

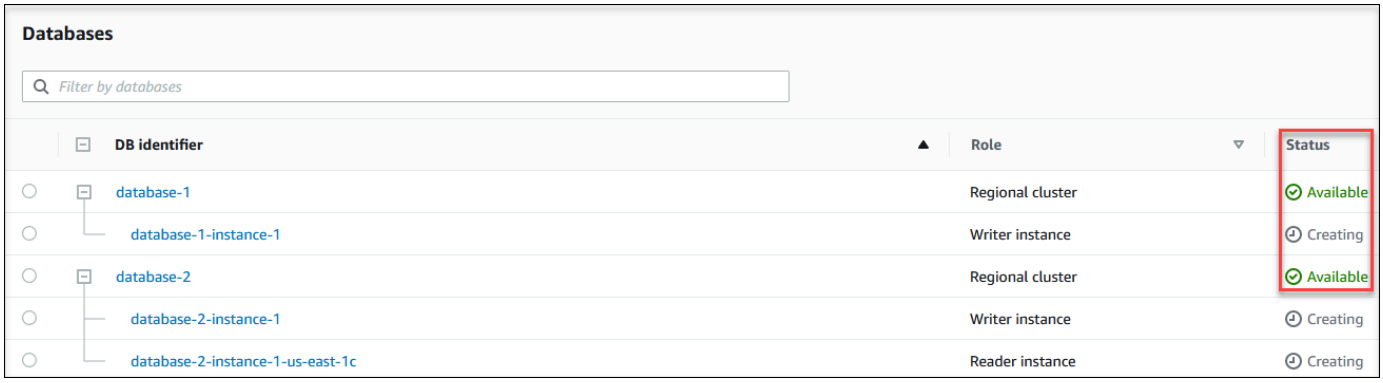
Stato cluster DB	Fatturata	Descrizione
Ridenominazione	Fatturata	È in corso la ridenominazione del cluster DB in seguito alla richiesta da parte di un cliente.
Resetting-master-credentials	Fatturato	È in corso il ripristino delle credenziali master del cluster DB in seguito alla richiesta da parte di un cliente.
Avvio di	Fatturato per storage	Il cluster di database è in fase di avvio.
Arrestate	Fatturato per storage	Il cluster di database è stato arrestato.
Stopping (In arresto)	Fatturato per storage	Il cluster di database è in fase di arresto.
Storage-optimization (Ottimizzazione archiviazione)	Fatturata	È in corso la modifica delle dimensioni o del tipo di storage dell'istanza database. L'istanza database è completamente operativa. Tuttavia, quando per l'istanza database è attivo lo stato storage-optimization, non è possibile richiedere modifiche dell'archiviazione dell'istanza database. Il processo di ottimizzazione dello storage è solitamente breve, ma a volte può richiedere oltre 24 ore.
Update-iam-db-auth	Fatturato	Autorizzazione IAM per il cluster DB che è attualmente sottoposto ad aggiornamento.
Aggiornamento	Fatturato	La versione del motore del cluster di database o del sistema operativo è in fase di aggiornamento.
upgrade_failed	Non fatturato	Il cluster DB non è riuscito a eseguire l'aggiornamento a una versione supportata. Aurora crea un'istantanea finale con il prefisso. `rds-final`

Console

Per visualizzare o stato di un cluster di database

1. Accedi a Console di gestione AWS e apri la console Amazon RDS all'indirizzo https://console.aws.amazon.com/rds/.
2. Nel pannello di navigazione, seleziona Database.

   La pagina Databases (Database) viene visualizzata con l'elenco dei cluster di database. Per ogni cluster di database, viene visualizzato un valore dello stato.



CLI

Per visualizzare solo lo stato dei cluster DB, usa la seguente query in. AWS CLI

```
aws rds describe-db-clusters --query 'DBClusters[*].[DBClusterIdentifier,Status]' --
output table
```

Visualizzazione dello stato dell’istanza database di in un cluster Aurora

Lo stato di un'istanza DB in un cluster Aurora indica il suo stato operativo corrente. Puoi utilizzare le seguenti procedure per visualizzare lo stato dell'istanza DB di un cluster nella console Amazon RDS, nel AWS CLI comando o nell'operazione API.

> **Note**
>
> Amazon RDS usa anche un altro stato denominato stato di manutenzione, mostrato nella colonna Maintenance (Manutenzione) della console Amazon RDS. Questo valore indica lo stato delle patch di manutenzione da applicare a un’istanza database. Lo stato della manutenzione è indipendente dallo stato dell’istanza database. Per ulteriori informazioni sullo stato della manutenzione, consulta Applicazione di aggiornamenti a un cluster di database.

I valori di stato possibili per le Istanze DB nella tabella seguente. Nella tabella viene inoltre indicato se è prevista la fatturazione per l’istanza database e lo storage, o solo per lo storage oppure se la fatturazione non è prevista. Per tutti gli stati delle istanze database, l’utilizzo del backup viene inserito in fattura.

Stato istanza database	Fatturata	Descrizione
available (disponibile)	Fatturato	L'istanza DB è disponibile per le modifiche.
backing-up (backup in corso)	Fatturata	L’istanza database è attualmente sottoposta a backup.
backtracking (backtrack in corso)	Fatturata	L’istanza database è attualmente sottoposta a backtrack. Questo stato si applica solo ad Aurora MySQL.
configuring-enhanced-monitoring	Fatturato	Il monitoraggio avanzato per questa istanza database è in fase di abilitazione/disabilitazione.
configuring-iam-database-auth	Fatturato	AWS Identity and Access Management L'autenticazione del database (IAM) è abilitata o disabilitata per questa istanza DB.

Stato istanza database	Fatturata	Descrizione
configuring-log-exports	Fatturato	La pubblicazione dei file di log su Amazon CloudWatch Logs è abilitata o disabilitata per questa istanza DB.
converting-to-vpc	Fatturato	È in corso la conversione dell'istanza database da un'istanza database che non si trova in un Amazon Virtual Private Cloud (Amazon VPC) a un'istanza database che si trova in un Amazon VPC.
creating (creazione in corso)	Non fatturata (non PITR) Fatturata (solo PITR)	È in corso la creazione dell'istanza database. Non è possibile accedere all'istanza database mentre è in fase di creazione. Se ripristini un database durante il point-in-time ripristino (PITR), ti verrà addebitato il costo mentre il database è in fase di creazione. Questo è l'unico scenario in cui lo stato di creazione incorre in addebiti.
delete-precheck (elimina precontrollo)	Non fatturato	Amazon RDS sta verificando che le repliche di lettura siano sicure da eliminare.
deleting (eliminazione in corso)	Non fatturata	È in corso l'eliminazione dell'istanza database.
Non riuscito	Non fatturata	L'istanza database ha restituito un errore e Amazon RDS non è stato in grado di recuperarla. Esegui un point-in-time ripristino all'ora di ripristino più recente dell'istanza DB per ripristinare i dati.
inaccessible-encryption-credentials	Non fatturato	Non è possibile accedere o recuperare l'istanza DB AWS KMS key utilizzata per crittografare o decrittografare l'istanza DB.

Stato istanza database	Fatturata	Descrizione
inaccessible-encryption-credentials-recoverable	Fatturato per storage	Non è possibile accedere alla chiave KMS utilizzata per crittografare o decrittografare l'istanza database. Tuttavia, se la chiave KMS è attiva, il riavvio dell'istanza database può ripristinarla. Per ulteriori informazioni, consulta Creazione di un cluster di database Amazon Aurora.
incompatible-create (creazione incompatibile)	Non fatturato	Il tentativo di Amazon RDS di creare un'istanza database ha esito negativo perché le risorse sono incompatibili con l'istanza database. Questo stato si può verificare ad esempio se il profilo dell'istanza database non dispone delle autorizzazioni corrette.
incompatible-network	Non fatturata	Il tentativo di Amazon RDS di eseguire un'operazione di ripristino su un'istanza database ha esito negativo perché il VPC si trova in uno stato che impedisce il completamento dell'operazione. Questo stato si verifica ad esempio se tutti gli indirizzi IP di una sottorete sono in uso e Amazon RDS non è in grado di ottenere un indirizzo IP per l'istanza database.
incompatible-option-group	Fatturato	Amazon RDS ha tentato di applicare una modifica a un gruppo di opzioni, ma non può farlo e non può eseguire il rollback allo stato del gruppo di opzioni precedente. Per ulteriori informazioni, consulta l'elenco Recent Events (Eventi recenti) per l'istanza database. Questo stato si verifica ad esempio se il gruppo di opzioni contiene un'opzione come TDE e l'istanza database non contiene informazioni crittografate.

Stato istanza database	Fatturata	Descrizione
parametri incompatibili	Fatturata	Amazon RDS non è in grado di avviare l'istanza database perché i parametri specificati nel gruppo di parametri database non sono compatibili con l'istanza database. È necessario annullare le modifiche apportate ai parametri o rendere tali parametri compatibili con l'istanza database per ottenere di nuovo l'accesso all'istanza database. Per ulteriori informazioni sui parametri incompatibili, consulta l'elenco Recent Events (Eventi recenti) per l'istanza database.
ripristino incompatibile	Non fatturato	Amazon RDS non è in grado di eseguire un point-in-time ripristino. Tra le cause più comuni di questo stato è incluso l'uso di tabelle temporanee o , i tabelle MyISAM con MySQL .
capacità insufficiente	Non fatturato	Amazon RDS non può creare l'istanza perché al momento non è disponibile una capacità sufficiente. Per creare l'istanza database nella stessa AZ con lo stesso tipo di istanza, elimina l'istanza database, attendi qualche ora e prova a crearla di nuovo. In alternativa, crea una nuova istanza utilizzando una classe di istanza o una AZ diversa.
manutenzione	Fatturata	È in corso l'applicazione da parte di Amazon RDS di un aggiornamento di manutenzione all'istan za database. Questo stato viene utilizzato per la manutenzione a livello di istanza pianificata in anticipo da RDS.
modifying (modifica in corso)	Fatturata	È in corso la modifica dell'istanza database in seguito alla richiesta da parte di un cliente.
moving-to-vpc	Fatturato	È in corso lo spostamento dell'istanza database in un nuovo Amazon Virtual Private Cloud (Amazon VPC).

Stato istanza database	Fatturata	Descrizione
riavvio in corso	Fatturata	È in corso il riavvio dell’istanza database a causa della richiesta di un cliente o perché è necessario per un processo Amazon RDS.
resetting-master-credentials	Fatturato	È in corso il ripristino delle credenziali master dell’istanza database in seguito alla richiesta da parte di un cliente.
renaming (ridenominazione)	Fatturata	È in corso la ridenominazione dell’istanza database in seguito alla richiesta da parte di un cliente.
restore-error (errore-ripristino)	Fatturato	L'istanza DB ha riscontrato un errore nel tentativo di ripristinare uno snapshot point-in-time o da uno snapshot.
avvio in corso	Fatturato per storage	L’istanza database è in fase di avvio.
stopped (arrestato)	Fatturato per storage	L’istanza database è stata arrestata.
stopping (in arresto)	Fatturato per storage	L’istanza database è in fase di arresto.
storage-config-upgrade	Fatturato	La configurazione del file system di archiviazione dell’istanza database è in fase di aggiornamento. Questo stato si applica solo ai database verdi all’interno di un’implementazione blu/verde o alle repliche di lettura delle istanze database.

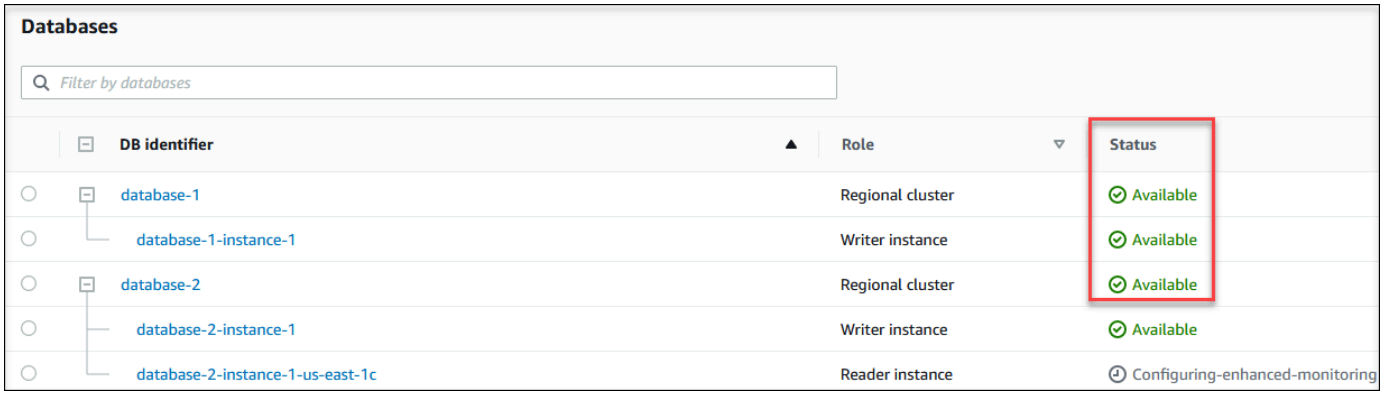
Stato istanza database	Fatturata	Descrizione
storage pieno	Fatturata	L’istanza database ha raggiunto la capacità di storage allocata. Si tratta di uno stato critico che richiede l’immediata risoluzione del problema. A tale scopo, è necessario aumentare la capacità di storage modificando l’istanza database. Per evitare questa situazione, imposta gli CloudWatch allarmi di Amazon per avvisarti quando lo spazio di archiviaz ione si sta esaurendo.
storage-initialization (inizializzazione del supporto di archiviaz ione)	Fatturato	L’istanza database sta caricando blocchi di dati da Amazon S3 per ottimizzare le prestazioni del volume dopo il ripristino da uno snapshot. Rimane disponibi le per le operazioni, ma le prestazioni potrebbero non essere ottimali fino al completamento dell’inizializzazi one.
storage-optimizati on (ottimizzazione-st orage)	Fatturato	Amazon RDS sta ottimizzando lo storage dell’istanza database. Il processo di ottimizzazione dello storage è solitamente breve, ma a volte può richiedere oltre 24 ore. Durante l’ottimizzazione dell’archiviazione, l’istanza database rimane disponibile. L’ottimizzazione dell’archiviazione è un processo in background che non influisce sulla disponibilità dell’istanza.
aggiornamento in corso	Fatturato	La versione del motore di database o del sistema operativo è in fase di aggiornamento.
upgrade_failed	Non fatturato	L'aggiornamento dell'istanza DB a una versione supportata non è riuscito. Aurora crea un'istantanea finale con il prefisso. `rds-final`

Console

Per visualizzare lo stato di un'istanza database

1. Accedi a Console di gestione AWS e apri la console Amazon RDS all'indirizzo https://console.aws.amazon.com/rds/.
2. Nel pannello di navigazione, seleziona Database.

   La pagina Databases (Database) viene visualizzata con l'elenco delle istanze database. Per ogni istanza database in un cluster, viene visualizzato il valore dello stato.



CLI

Per visualizzare l'istanza DB e le relative informazioni sullo stato utilizzando il AWS CLI, usa il describe-db-instancescomando. Ad esempio, il AWS CLI comando seguente elenca tutte le informazioni sulle istanze DB.

```
aws rds describe-db-instances
```

Per visualizzare un'istanza DB specifica e il relativo stato, chiamate il describe-db-instancescomando con l'opzione seguente:

- `DBInstanceIdentifier`: il nome dell'istanza database.

```
aws rds describe-db-instances --db-instance-identifier mydbinstance
```

Per visualizzare solo lo stato di tutte le istanze DB, usa la seguente query in AWS CLI.

```
aws rds describe-db-instances --query 'DBInstances[*].
[DBInstanceIdentifier,DBInstanceStatus]' --output table
```

"Hello, World!"

Per visualizzare lo stato dell'istanza DB utilizzando l'API Amazon RDS, chiama l'DBInstancesoperazione Descrivi.

Suggerimenti di Amazon Aurora

Amazon Aurora fornisce suggerimenti automatici per le risorse di del database, come le istanze database, i cluster di database, e i gruppi di parametri di database. Questi suggerimenti forniscono indicazioni sulle best practice analizzando la configurazione del cluster di database e la configurazione, l'utilizzo e i dati sulle prestazioni dell'istanza database.

Approfondimenti sulle prestazioni di Amazon RDS monitora metriche specifiche e crea automaticamente soglie analizzando quali livelli sono considerati potenzialmente problematici per una specifica risorsa. Quando i nuovi valori delle metriche superano una soglia predefinita in un determinato periodo di tempo, Approfondimenti sulle prestazioni genera un suggerimento proattivo. Questo suggerimento aiuta a evitare un impatto sulle prestazioni del database in futuro. Ad esempio, la raccomandazione "inattività nella transazione" viene generata per le istanze Aurora PostgreSQL quando le sessioni connesse al database non svolgono attività attive, ma possono mantenere bloccate le risorse del database. Per ricevere suggerimenti proattivi, è necessario attivare Approfondimenti sulle prestazioni con un periodo di conservazione del piano a pagamento. Per ulteriori informazioni sull'attivazione di Approfondimenti sulle prestazioni, consulta Attivazione e disattivazione di Approfondimenti sulle prestazioni per Aurora. Per informazioni sui prezzi e sulla conservazione dei dati per Approfondimenti sulle prestazioni, consulta Prezzi e conservazione dei dati per Performance Insights.

DevOps Guru per RDS monitora determinate metriche per rilevare quando il comportamento della metrica diventa molto insolito o anomalo. Queste anomalie vengono segnalate come approfondimenti reattivi con suggerimenti. Ad esempio, DevOps Guru per RDS potrebbe consigliare di aumentare la capacità della CPU o di indagare sugli eventi di attesa che contribuiscono al carico del database. DevOps Guru per RDS fornisce anche suggerimenti proattivi basati su soglia. Per questi suggerimenti, è necessario attivare DevOps Guru per RDS. Per informazioni sull'attivazione di DevOps Guru per RDS, consulta Attivare DevOps Guru e specificare la copertura delle risorse.

I suggerimenti avranno uno dei seguenti stati: attivi, ignorati, in sospeso o risolti. I suggerimenti risolti sono disponibili per 365 giorni.

È possibile visualizzare o ignorare i suggerimenti. È possibile applicare immediatamente un suggerimento attivo basato sulla configurazione, programmarlo nella successiva finestra di manutenzione o ignorarlo. Per i suggerimenti proattivi basati su soglia e quelli reattivi basati sul machine learning, è necessario esaminare la causa suggerita del problema e quindi eseguire le azioni suggerite per risolverlo.

I suggerimenti sono supportati nelle seguenti Regioni AWS:

- Stati Uniti orientali (Ohio)
- US East (N. Virginia)
- Stati Uniti occidentali (California settentrionale)
- Stati Uniti occidentali (Oregon)
- Asia Pacific (Mumbai)
- Asia Pacifico (Seoul)
- Asia Pacifico (Singapore)
- Asia Pacifico (Sydney)
- Asia Pacifico (Tokyo)
- Canada (Centrale)
- Europa (Francoforte)
- Europa (Irlanda)
- Europe (London)
- Europe (Paris)
- Europa (Stoccolma)
- Sud America (São Paulo)

Informazioni su come visualizzare, applicare, ignorare e modificare le raccomandazioni di Amazon Aurora nelle seguenti sezioni.

Argomenti

- Visualizzazione dei suggerimenti Amazon Aurora
- Applicazione delle raccomandazioni Amazon Aurora
- Ignorare le raccomandazioni Amazon Aurora
- Modifica delle raccomandazioni Amazon Aurora ignorate in raccomandazioni attive
- Guida di riferimento sui suggerimenti da Amazon Aurora

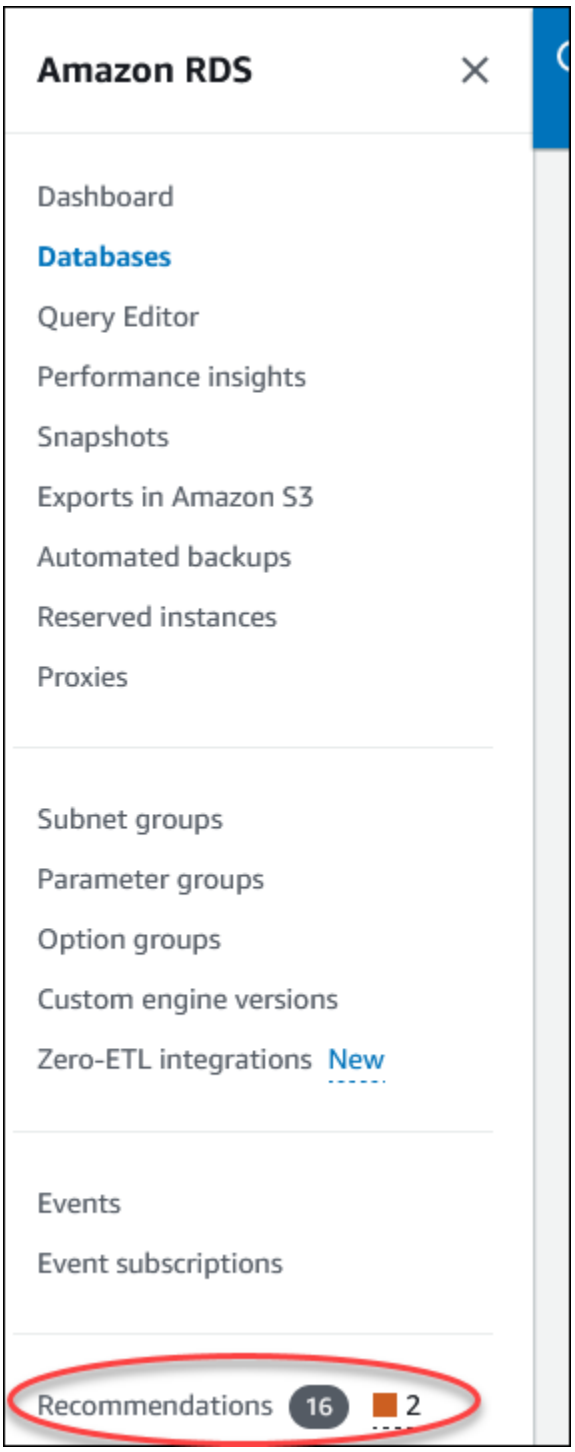
Visualizzazione dei suggerimenti Amazon Aurora

Utilizzando la console Amazon RDS, puoi visualizzare le raccomandazioni di Amazon Aurora per le tue risorse di database. Per un cluster di database, vengono visualizzati le raccomandazioni per il cluster di database e le relative istanze.

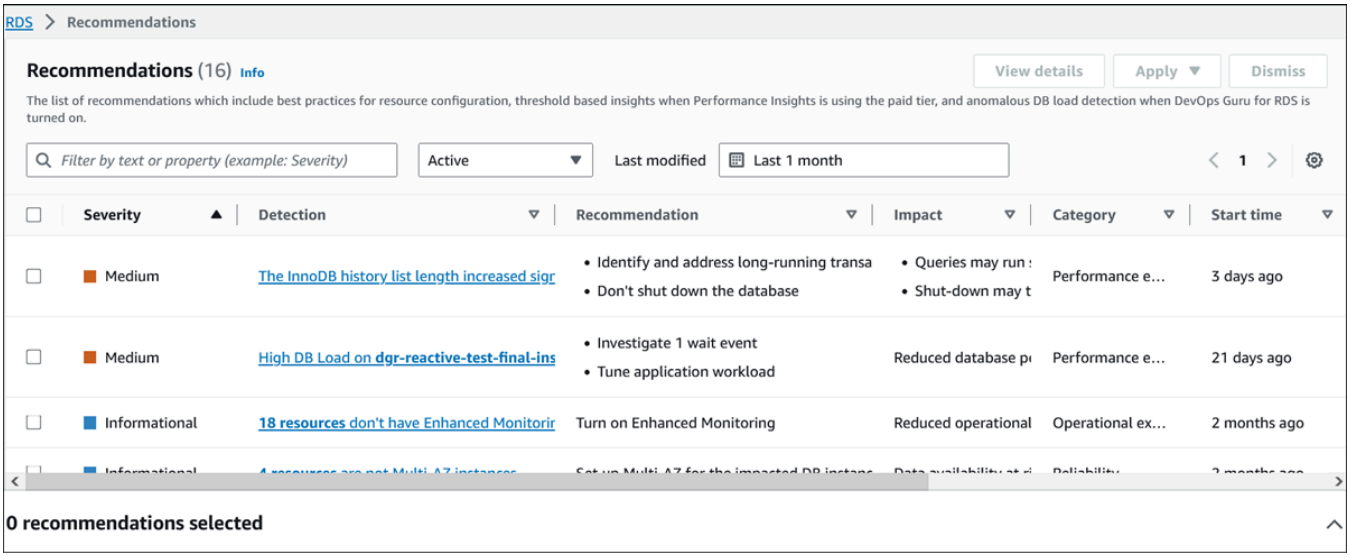
Console

Per visualizzare le raccomandazioni Amazon Aurora

1. Accedi alla Console di gestione AWS e apri la console Amazon RDS all'indirizzo https://console.aws.amazon.com/rds/.
2. Nel riquadro di navigazione, effettua una delle seguenti operazioni:
   - Scegli Raccomandazioni. Il numero di raccomandazioni attive per le tue risorse e il numero di raccomandazioni con la massima severità generati nell'ultimo mese sono disponibili accanto a Raccomandazioni. Per trovare il numero di raccomandazioni attive per ogni gravità, scegli il numero che mostra la gravità più alta.

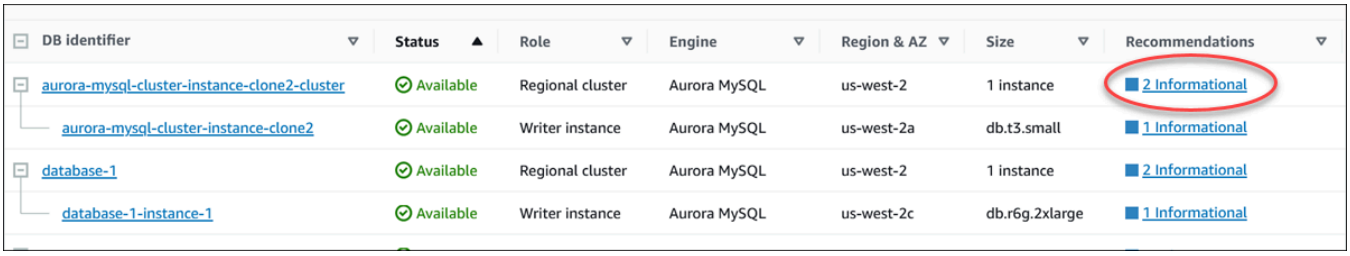


Per impostazione predefinita, la pagina Raccomandazioni mostra un elenco di nuove raccomandazioni nell'ultimo mese. Amazon Aurora fornisce raccomandazioni per tutte le risorse del tuo account e le ordina in base alla loro gravità.

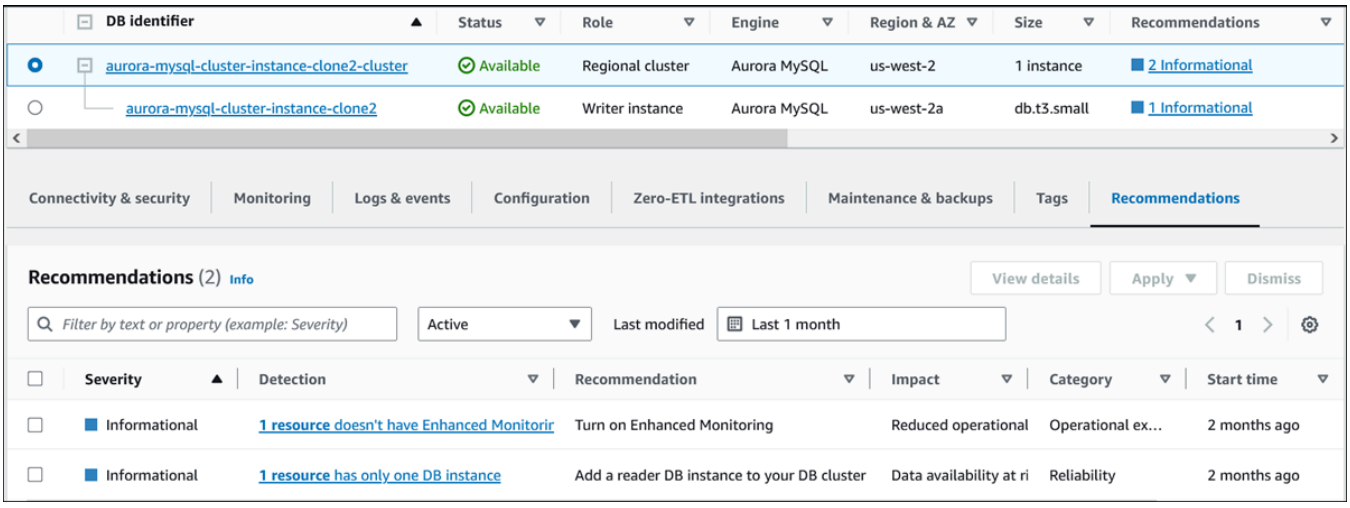


Puoi scegliere una raccomandazione per visualizzare una sezione nella parte inferiore della pagina che contiene le risorse interessate e i dettagli su come verrà applicata la raccomandazione.

- Nella pagina Database, scegli Raccomandazioni per una risorsa.



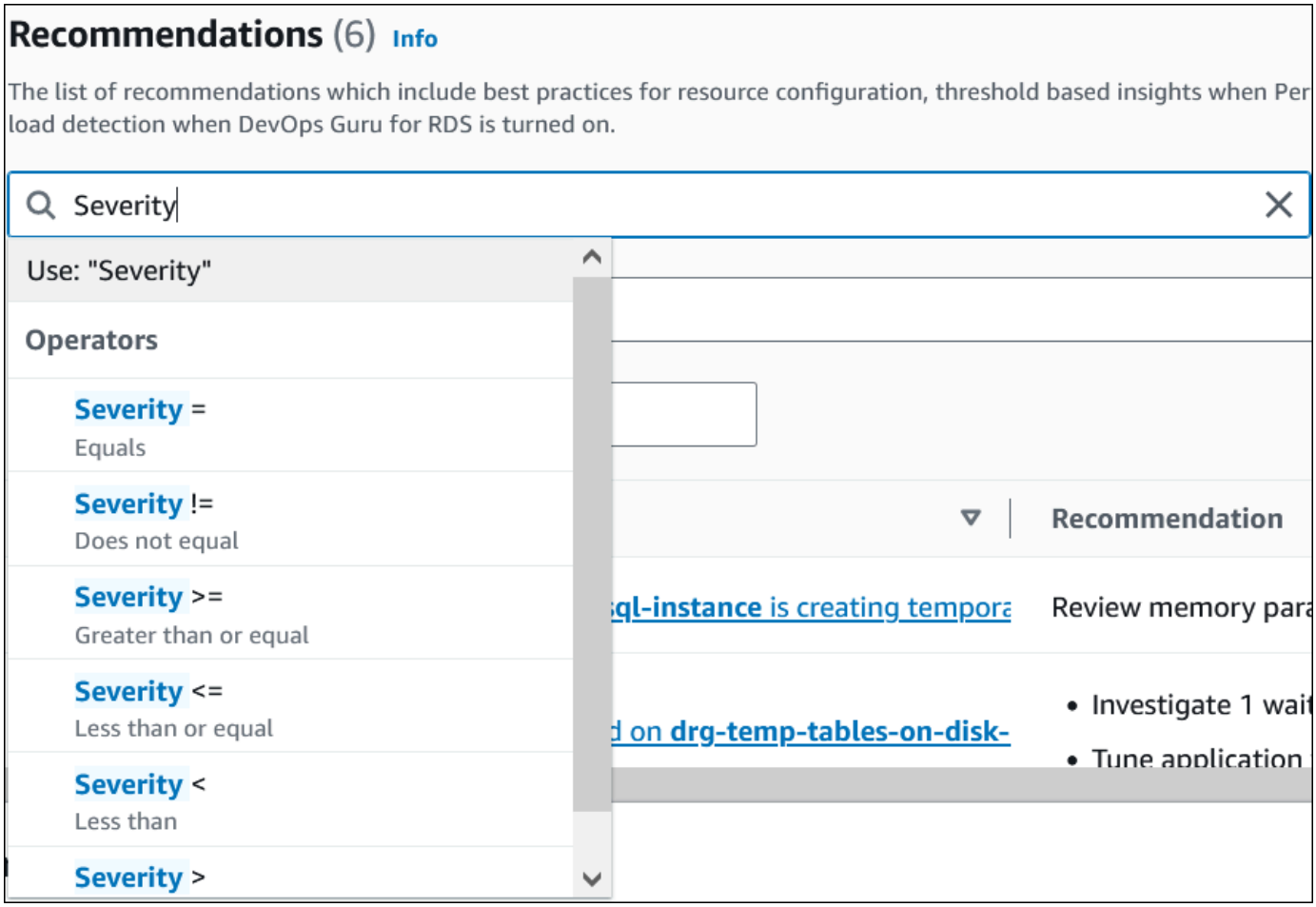
La scheda Raccomandazioni mostra le raccomandazioni e i relativi dettagli per la risorsa selezionata.



I seguenti dettagli sono disponibili per le raccomandazioni:

- Gravità: il livello di implicazione del problema. I livelli di gravità sono Alto, Medio, Basso e Informativo.
- Rilevamento: il numero di risorse interessate e una breve descrizione del problema. Scegli questo link per visualizzare la raccomandazione e i dettagli dell'analisi.
- Raccomandazione: una breve descrizione dell'azione consigliata da applicare.
- Impatto: una breve descrizione del possibile impatto quando la raccomandazione non viene applicata.
- Categoria: il tipo di raccomandazione. Le categorie sono Efficienza delle prestazioni, Sicurezza, Affidabilità, Ottimizzazione dei costi, Eccellenza operativa e Sostenibilità.
- Stato: lo stato attuale della raccomandazione. Gli stati possibili sono Tutti, Attivo, Ignorato, Risolto e In sospeso.
- Ora di inizio: ora in cui è iniziato il problema. Ad esempio, 18 ore fa.
- Ultima modifica: l'ora in cui la raccomandazione è stata aggiornata l'ultima volta dal sistema a causa di una modifica della gravità o l'ora in cui hai risposto alla raccomandazione. Ad esempio, 10 ore fa.
- Ora di fine: l'ora in cui è terminato il problema. L'ora non verrà visualizzata per eventuali problemi persistenti.
- Identificatore di risorsa: il nome di una o più risorse.

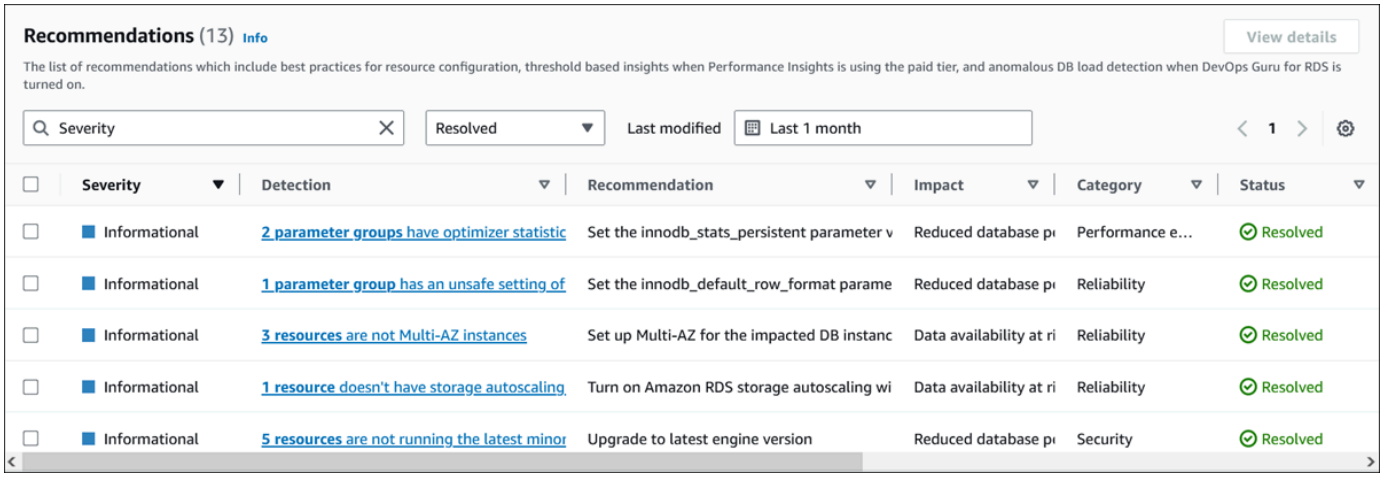
3. (Facoltativo) Scegli gli operatori di severità o categoria nel campo per filtrare l'elenco delle raccomandazioni.



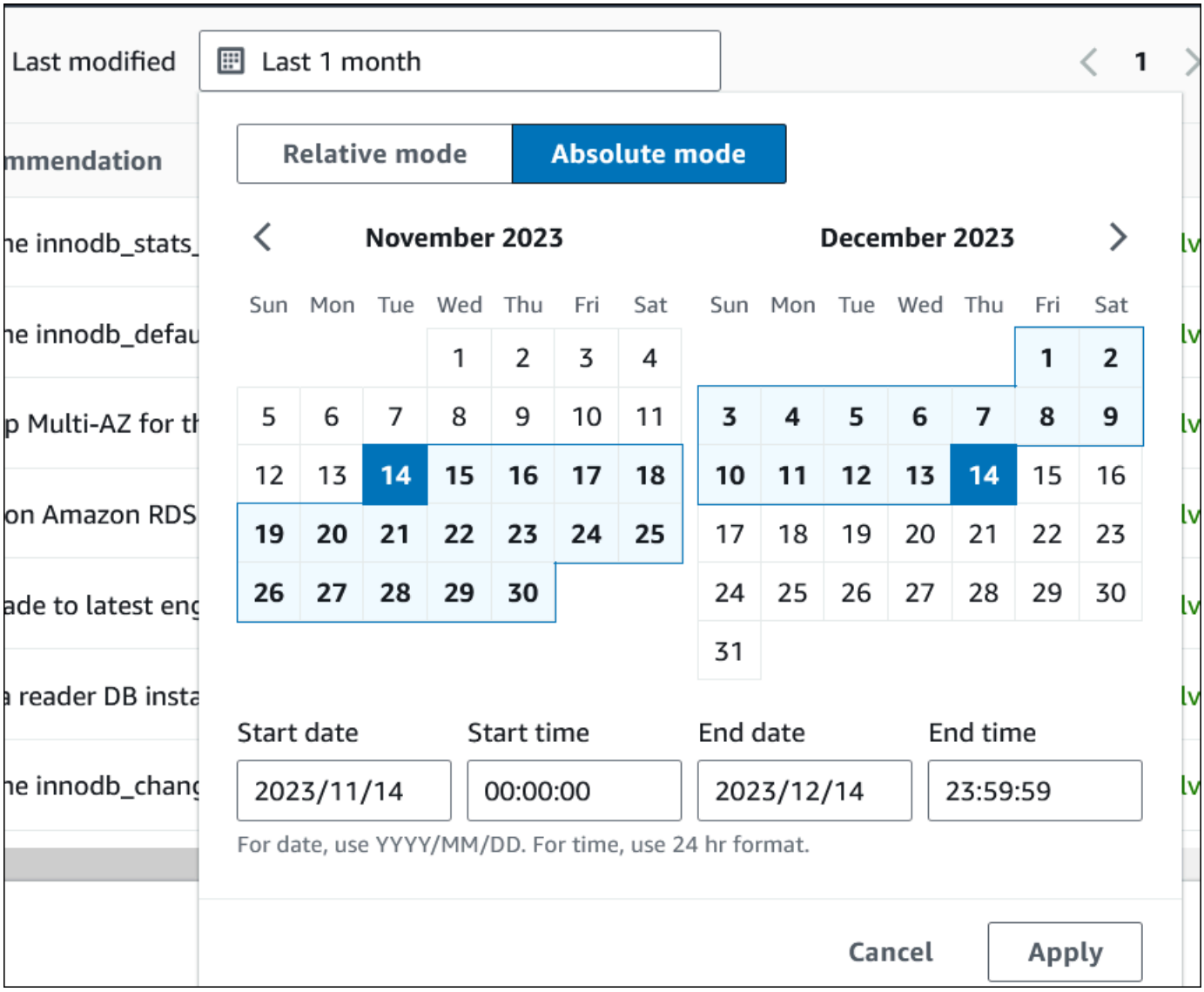
Vengono visualizzate le raccomandazioni per l'operazione selezionata.

4. (Facoltativo) Scegli uno dei seguenti stati di raccomandazione:

   - Attivo (predefinito): mostra le raccomandazioni correnti che è possibile applicare, annullare o pianificare.
   - Tutti: mostra tutte le raccomandazioni con lo stato corrente.
   - Ignorato: mostra le raccomandazioni ignorate.
   - Risolto: mostra le raccomandazioni che sono state risolte.
   - In sospeso: mostra le raccomandazioni le cui azioni raccomandate sono in corso o pianificate per la finestra di manutenzione successiva.



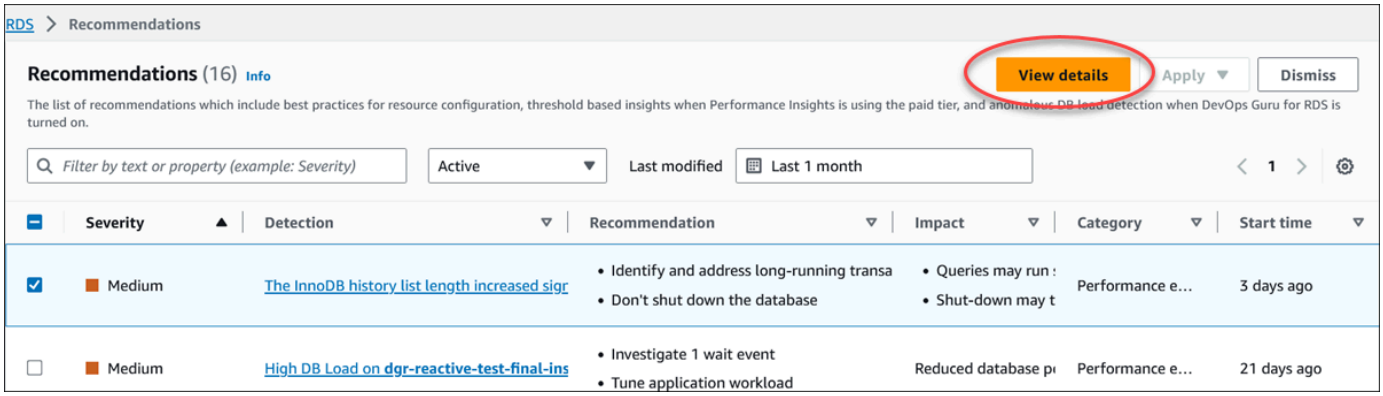
5. (Facoltativo) Scegli Modalità relativa o Modalità assoluta in Ultima modifica per modificare il periodo di tempo. La pagina Raccomandazioni mostra i consigli generati nel periodo di tempo. Il periodo di tempo predefinito è l'ultimo mese. In Modalità assoluta, puoi scegliere il periodo di tempo o inserire l'ora nei campi Data di inizio e Data di fine.



Vengono visualizzati le raccomandazioni per il periodo di tempo impostato.

Tieni presente che puoi visualizzare tutte le raccomandazioni relative alle risorse del tuo account impostando l'intervallo su Tutti.

6. (Facoltativo) Scegli Preferenze a destra per personalizzare i dettagli da visualizzare. Puoi scegliere una dimensione di pagina, disporre le righe del testo e consentire o nascondere le colonne.
7. (Facoltativo) Scegli una raccomandazione, quindi scegli Visualizza dettagli.



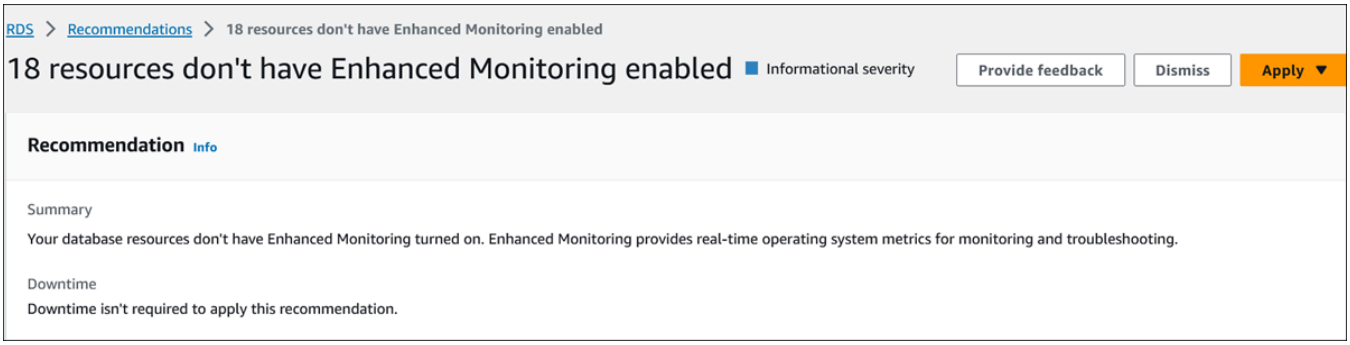
Viene visualizzata la pagina Raccomandazioni. Il titolo fornisce il conteggio totale delle risorse con il problema rilevato e la gravità.

Per informazioni sui componenti nella pagina dei dettagli per una raccomandazione reattiva basata sulle anomalie, consulta Viewing reactive anomalies nella Guida per l'utente di Amazon DevOps Guru.

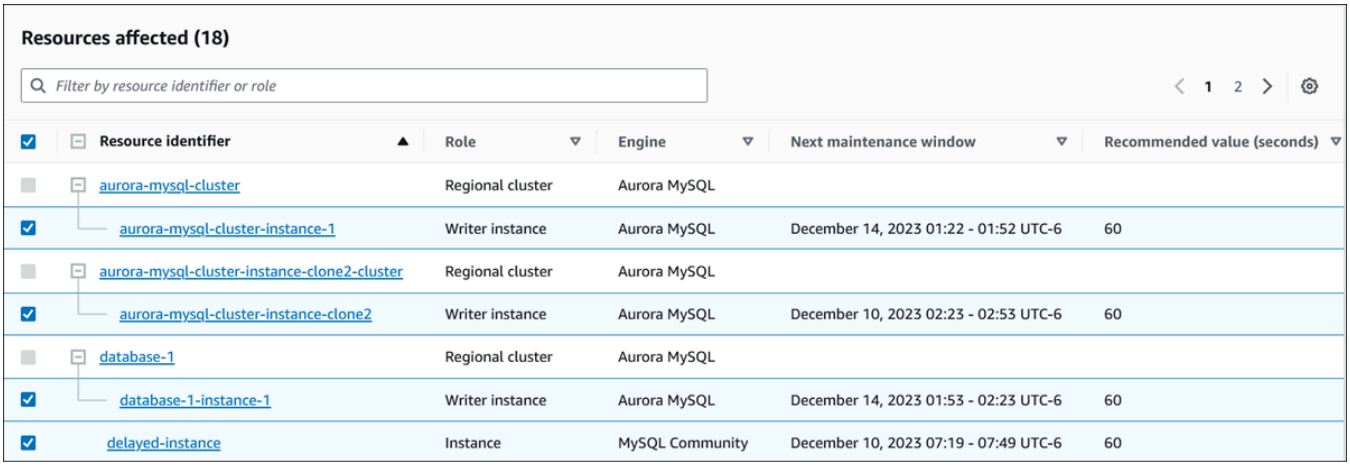
Per informazioni sui componenti nella pagina dei dettagli per una raccomandazione proattiva basata su una soglia, consulta Visualizzazione dei consigli proattivi di Approfondimenti sulle prestazioni.

Le altre raccomandazioni automatiche mostrano i seguenti componenti nella pagina dei dettagli delle raccomandazioni:

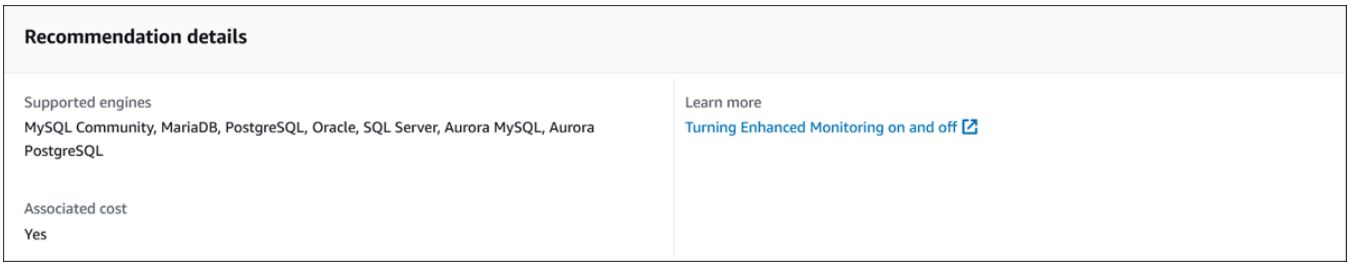
- Raccomandazione: un riepilogo della raccomandazione e indica se è necessario tempo di inattività per applicarla.



- Risorse interessate: dettagli delle risorse interessate.



- Dettagli della raccomandazione: informazioni sul motore supportato, eventuali costi associati necessari per applicare la raccomandazione e link alla documentazione per saperne di più.



CLI

Per visualizzare le raccomandazioni di Amazon RDS relative alle istanze o ai cluster di database, usa il seguente comando in AWS CLI.

```
aws rds describe-db-recommendations
```

API RDS

Per visualizzare le raccomandazioni Amazon RDS tramite l'API Amazon RDS, utilizzare l'operazione DescribeDBRecommendations.

Applicazione delle raccomandazioni Amazon Aurora

Per applicare le raccomandazioni di Amazon Aurora utilizzando la console Amazon RDS, seleziona un suggerimento basato sulla configurazione o una risorsa interessata nella pagina dei dettagli. Quindi, scegliere di applicare la raccomandazione subito o programmarla per la prossima finestra di manutenzione. Potrebbe essere necessario riavviare la risorsa per rendere effettiva la modifica. Per

alcune raccomandazioni sui gruppi di parametri di database, potrebbe essere necessario riavviare le risorse.

La raccomandazioni proattive basate sulla soglia o reattive basate sulle anomalie non avranno l'opzione Applica e potrebbero richiedere un'ulteriore revisione.

Console

Per applicare una raccomandazione basata sulla configurazione

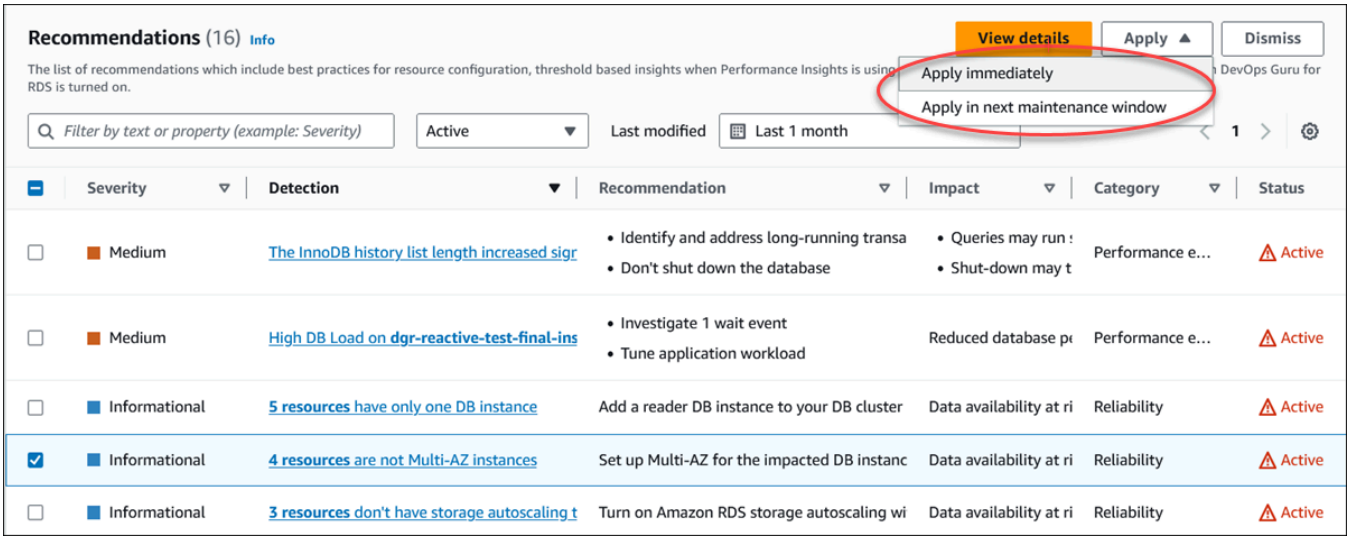
1. Accedi alla Console di gestione AWS e apri la console Amazon RDS all'indirizzo https://console.aws.amazon.com/rds/.
2. Nel riquadro di navigazione, effettua una delle seguenti operazioni:
   - Scegli Raccomandazioni.

     Viene visualizzata la pagina Raccomandazioni con l'elenco di tutte le raccomandazioni.
   - Scegli Database, quindi scegli Raccomandazioni per una risorsa nella pagina dei database.

     La scheda Raccomandazioni mostra le raccomandazioni per la risorsa selezionata.
   - Scegli Rilevamento per una raccomandazione attiva nella pagina Raccomandazioni o nella scheda Raccomandazioni nella pagina Database.

     Viene visualizzata la pagina Raccomandazioni.
3. Scegli una raccomandazione o una o più risorse interessate nella pagina dei dettagli della raccomandazione ed esegui una delle seguenti operazioni:
   - Scegli Applica, quindi scegli Applica immediatamente per applicare immediatamente la raccomandazione.
   - Scegli Applica e quindi scegli Applica nella finestra di manutenzione successiva per pianificare nella finestra di manutenzione successiva.

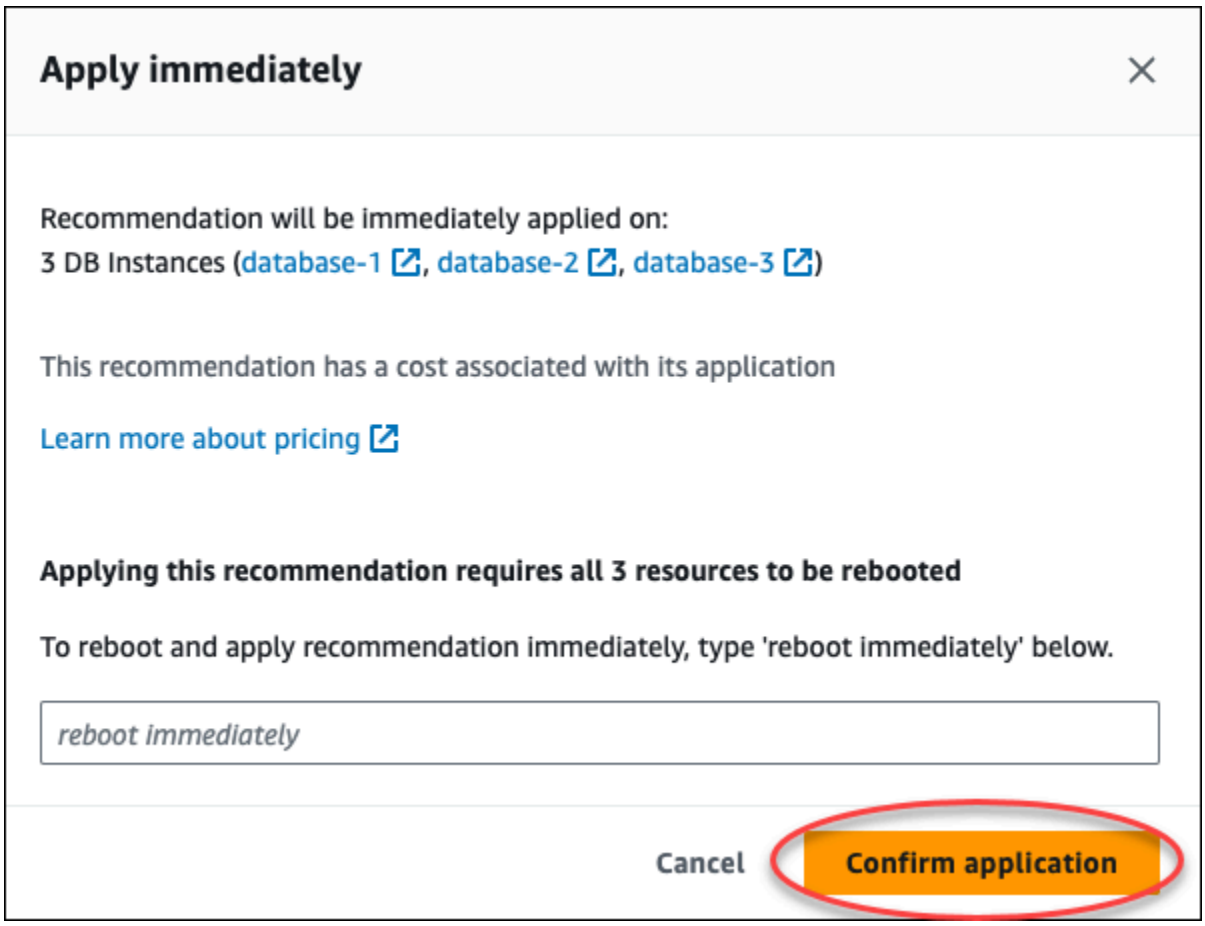
     Lo stato della raccomandazione selezionata viene aggiornato in sospeso fino alla prossima finestra di manutenzione.



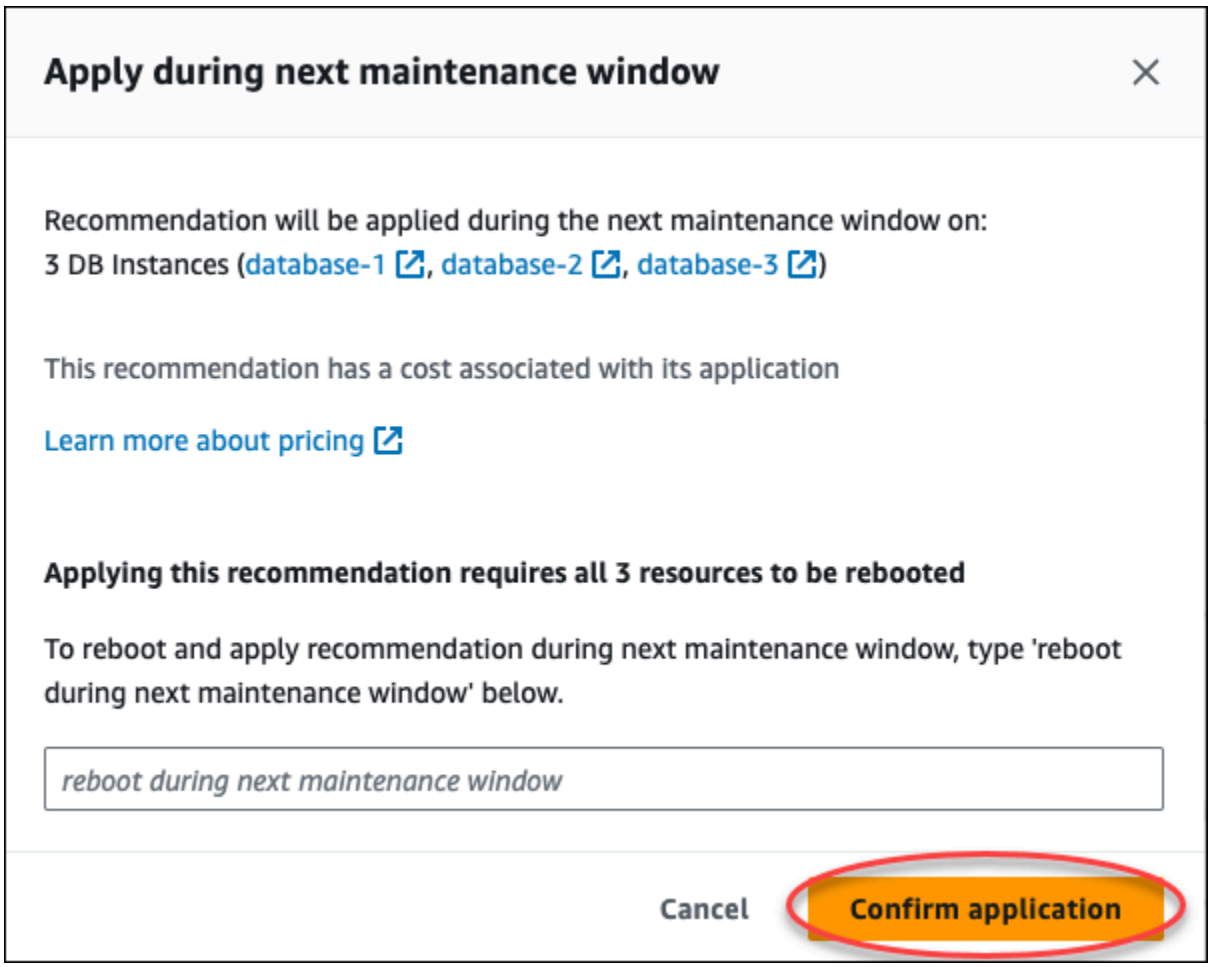
Viene visualizzata una finestra di conferma.

4. Scegli Conferma applicazione per applicare la raccomandazione. Questa finestra conferma se le risorse necessitano di un riavvio automatico o manuale per rendere effettive le modifiche.

   Nell'esempio seguente viene illustrata la finestra di conferma per applicare la raccomandazione subito.

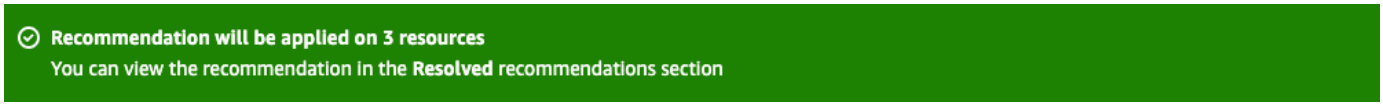


L’esempio seguente mostra la finestra di conferma per pianificare l’applicazione della raccomandazione nella finestra di manutenzione successiva.



Un banner mostra un messaggio a prescindere dall'esito della raccomandazione applicata.

L'esempio seguente mostra il banner con il messaggio che l'operazione è riuscita.



L'esempio seguente mostra il banner con il messaggio che l'operazione non è riuscita.

API RDS

Applicare una raccomandazione Aurora basata sulla configurazione utilizzando l'API Amazon RDS

1. Utilizza l'operazione DescribeDBRecommendations. La `RecommendedActions` nell'output può avere una o più azioni consigliate.
2. Utilizza l'oggetto RecommendedAction per ogni azione raccomandata del passaggio 1. L'output contiene. `Operation` e `Parameters`.

   Nell'esempio seguente viene illustrato l'output con un'azione raccomandata.

```
   "RecommendedActions": [
       {
           "ActionId": "0b19ed15-840f-463c-a200-b10af1b552e3",
           "Title": "Turn on auto backup", // localized
           "Description": "Turn on auto backup for my-mysql-instance-1", //
localized
           "Operation": "ModifyDbInstance",
           "Parameters": [
               {
                   "Key": "DbInstanceIdentifier",
                   "Value": "my-mysql-instance-1"
               },
               {
                   "Key": "BackupRetentionPeriod",
                   "Value": "7"
               }
           ],
           "ApplyModes": ["immediately", "next-maintenance-window"],
           "Status": "applied"
       },
       ... // several others
   ],
```

3. Utilizza il comando `operation` per ogni azione raccomandata dall'output del passaggio 2 e immetti i valori `Parameters`.
4. Una volta completata l'operazione nel passaggio 2, utilizza l'operazione ModifyDBRecommendation per modificare lo stato della raccomandazione.

Ignorare le raccomandazioni Amazon Aurora

Puoi ignorare uno o più raccomandazioni di Amazon Aurora utilizzando la console Amazon RDS, AWS CLI o l'API Amazon RDS.

Console

Per ignorare uno o più raccomandazioni

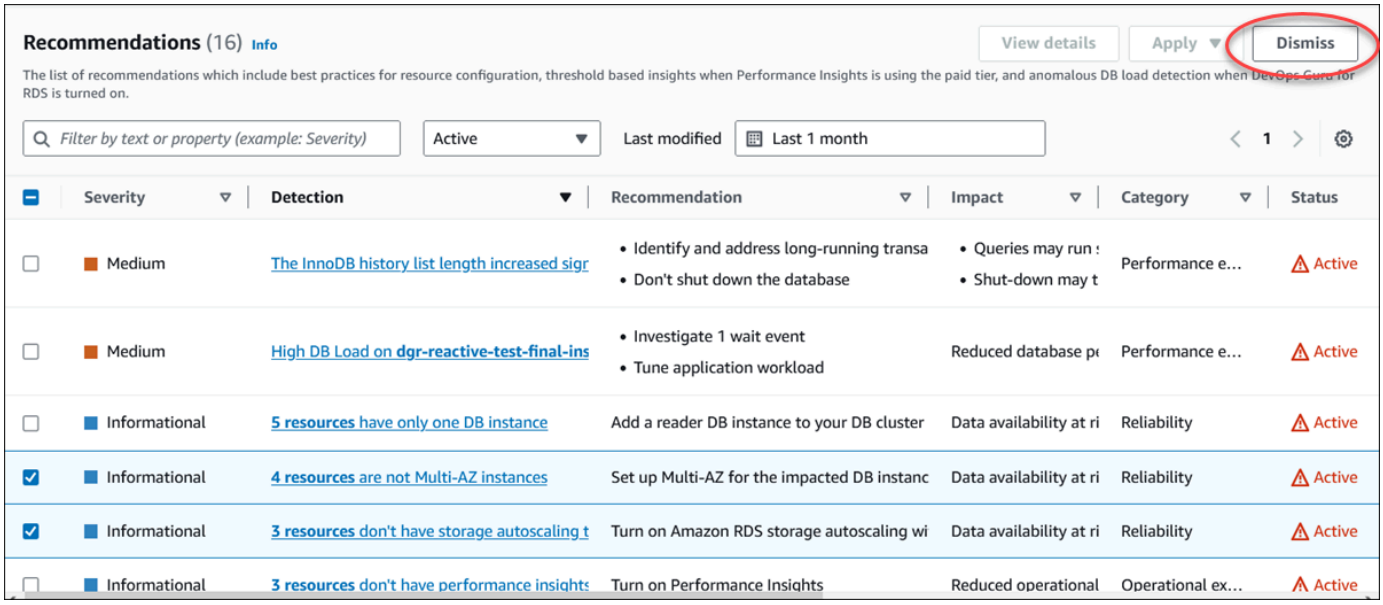
1. Accedi alla Console di gestione AWS e apri la console Amazon RDS all'indirizzo https://console.aws.amazon.com/rds/.
2. Nel riquadro di navigazione, effettua una delle seguenti operazioni:
   - Scegli Raccomandazioni.

     Viene visualizzata la pagina Raccomandazioni con l'elenco di tutte le raccomandazioni.
   - Scegli Database, quindi scegli Raccomandazioni per una risorsa nella pagina dei database.

     La scheda Raccomandazioni mostra le raccomandazioni per la risorsa selezionata.
   - Scegli Rilevamento per una raccomandazione attiva nella pagina Raccomandazioni o nella scheda Raccomandazioni nella pagina Database.

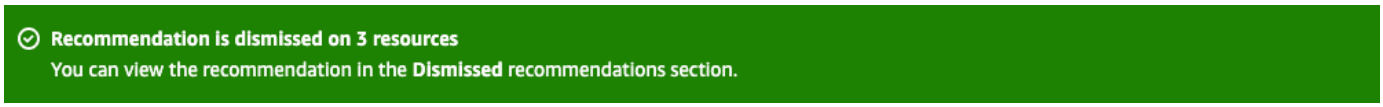
     La pagina dei dettagli delle raccomandazioni mostra l'elenco delle risorse interessate.
3. Scegli una o più raccomandazioni o una o più risorse interessate nella pagina dei dettagli della raccomandazione, quindi scegli Ignora.

   L'esempio seguente mostra la pagina Raccomandazioni con più raccomandazioni attive selezionate da ignorare.



Un banner mostra un messaggio quando una o più raccomandazioni selezionate vengono ignorate.

L'esempio seguente mostra il banner con il messaggio che l'operazione è riuscita.



L'esempio seguente mostra il banner con il messaggio che l'operazione non è riuscita.

CLI

Ignorare una raccomandazione Aurora utilizzando AWS CLI

1. Esegui il comando `aws rds describe-db-recommendations --filters "Name=status,Values=active"`.

   L'output fornisce un elenco di raccomandazioni in stato `active`.
2. Trova lo `recommendationId` per la raccomandazione che desideri eliminare dal passaggio 1.
3. Esegui il comando `>aws rds modify-db-recommendation --status dismissed --recommendationId <ID>` con lo `recommendationId` dal passaggio 2 per ignorare la raccomandazione.

API RDS

Per ignorare un suggerimento Aurora utilizzando l'API Amazon RDS, utilizza l'operazione ModifyDBRecommendation.

Modifica delle raccomandazioni Amazon Aurora ignorate in raccomandazioni attive

Puoi spostare una o più raccomandazioni Amazon Aurora ignorate in raccomandazioni attive utilizzando la console Amazon RDS, l'interfaccia AWS CLI o l'API Amazon RDS.

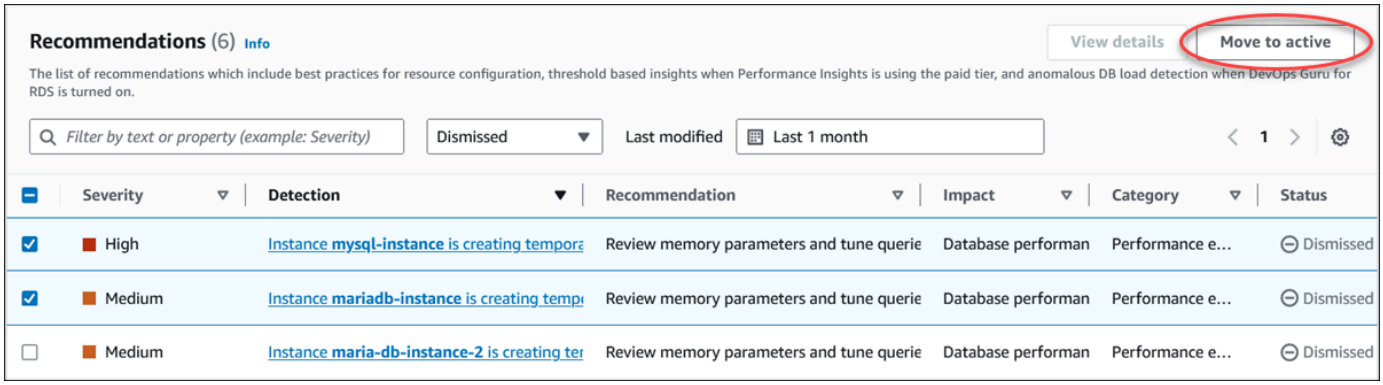
Console

Per spostare una o più raccomandazioni ignorate in attive

1. Accedi alla Console di gestione AWS e apri la console Amazon RDS all'indirizzo https://console.aws.amazon.com/rds/.
2. Nel riquadro di navigazione, effettua una delle seguenti operazioni:
   - Scegli Raccomandazioni.

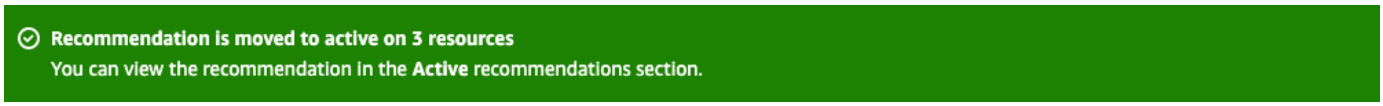
     La pagina Raccomandazioni mostra un elenco di raccomandazioni ordinate in base alla gravità per tutte le risorse dell'account.
   - Scegli Database, quindi scegli Raccomandazioni per una risorsa nella pagina dei database.

     La scheda Raccomandazioni mostra le raccomandazioni e i relativi dettagli per la risorsa selezionata.
3. Scegli una o più raccomandazioni ignorate dall'elenco, quindi scegli Sposta su attiva.



Un banner mostra un messaggio di successo o di insuccesso quando si spostano le raccomandazioni selezionate dallo stato ignorato a quello attivo.

L'esempio seguente mostra il banner con il messaggio che l'operazione è riuscita.



L'esempio seguente mostra il banner con il messaggio che l'operazione non è riuscita.

CLI

Per modificare una raccomandazione Aurora ignorata in una raccomandazione attiva utilizzando l'interfaccia AWS CLI

1. Esegui il comando `aws rds describe-db-recommendations --filters "Name=status,Values=dismissed"`.

   L'output fornisce un elenco di raccomandazioni in stato `dismissed`.
2. Trova lo `recommendationId` per la raccomandazione di cui desideri modificare lo stato dal passaggio 1.
3. Esegui il comando `>aws rds modify-db-recommendation --status active --recommendationId <ID>` con lo `recommendationId` dal passaggio 2 per passare alla raccomandazione attiva.

API RDS

Per modificare un suggerimento Aurora ignorato in un suggerimento attivo utilizzando l'API Amazon RDS, utilizza l'operazione ModifyDBRecommendation.

Guida di riferimento sui suggerimenti da Amazon Aurora

Amazon Aurora genera raccomandazioni per una risorsa quando questa viene creata o modificata. Puoi trovare esempi di suggerimenti da Amazon Aurora nella tabella seguente.

Tipo	Descrizione	Raccomandazione	Tempo di inattività richiest	Informazioni aggiuntive
I backup di Resource Automated sono disattivati	I backup automatici non sono attivati per le istanze database. I backup automatici sono consigliati perché consentono point-in-time il ripristino delle istanze DB.	Attivare i backup automatizzati con un periodo di conservazione fino a 14 giorni.	Sì	Panoramica di backup e ripristino di un cluster di database Aurora Spiegazione del funzionamento dei costi dell'archiviazione di backup di Amazon RDS sul Blog sui database di AWS
È necessario l'aggiornamento della versione secondaria del motore	Le risorse del database non eseguono la versione secondaria più recente del motore di database. L'ultima versione secondaria include le correzioni più recenti per la sicurezza e altri miglioramenti.	Esegui l'aggiornamento alla versione più recente del motore.	Sì	Manutenzione di un cluster di database Amazon Aurora
Il monitoraggio avanzato è disattivato	Il monitoraggio avanzato non è attivato per le risorse del database. Il monitoraggio avanzato offre le metriche del sistema	Attiva il Monitoraggio avanzato.	No	Monitoraggio dei parametri del sistema operativo con il monitoraggio avanzato

Tipo	Descrizione	Raccomandazione	Tempo di inattività richiest	Informazioni aggiuntive
	operativo in tempo reale per il monitoraggio e la risoluzione dei problemi.			

Tipo	Descrizione	Raccomandazione	Tempo di inattività richiest	Informazioni aggiuntive
La crittografia dell'archiviazione è disattivata	Amazon RDS supporta la crittografia a riposo per tutti i motori di database utilizzando le chiavi gestite in AWS Key Management Service (AWS KMS). In un'istanza database attiva con crittografia Amazon RDS, i dati a riposo memorizzati vengono crittografati, in modo simile ai backup automatici, alle repliche di lettura e agli snapshot. Se la crittografia non viene attivata durante la creazione di un cluster di database Aurora, è necessario ripristinare uno snapshot decrittografato in un cluster di database crittografato.	Attivare la crittografia dei dati a riposo per il cluster di database.	Sì	Sicurezza in Amazon Aurora

Tipo	Descrizione	Raccomandazione	Tempo di inattività richiest	Informazioni aggiuntive
Cluster di database con tutte le istanze nella stessa zona di disponibilità	I cluster di database si trovano attualmente in un'unica zona di disponibilità. Utilizza più zone di disponibilità per migliorare la disponibilità.	Aggiungi le istanze database a più zone di disponibilità nel cluster di database.	No	Elevata disponibilità di Amazon Aurora
Istanze database nei cluster con dimensioni di istanza eterogenee	È consigliabile utilizzare la stessa classe e le stesse dimensioni di istanza database per tutte le istanze database nel cluster di database.	Utilizza la stessa classe e le stesse dimensioni di istanza database per tutte le istanze database nel cluster di database.	Sì	Replica con Amazon Aurora
Istanze database nei cluster con classi di istanza eterogenee	È consigliabile utilizzare la stessa classe e le stesse dimensioni di istanza database per tutte le istanze database nel cluster di database.	Utilizza la stessa classe e le stesse dimensioni di istanza database per tutte le istanze database nel cluster di database.	Sì	Replica con Amazon Aurora
Istanze database nei cluster con gruppi di parametri eterogenei	È consigliabile che tutte le istanze database nel cluster di database utilizzino lo stesso gruppo di parametri database.	Associa l'istanza database al gruppo di parametri database associato all'istanza di scrittura nel cluster di database.	No	Gruppi di parametri per Amazon Aurora

Tipo	Descrizione	Raccomandazione	Tempo di inattività richiest	Informazioni aggiuntive
I cluster di database Amazon RDS dispongono di un'istanza database	Aggiungere almeno un'altra istanza database al cluster di database per migliorare la disponibilità e le prestazioni.	Aggiungere un'istanza database di lettura al cluster di database.	No	Elevata disponibilità di Amazon Aurora
La funzionalità Approfondimenti sulle prestazioni è disattivata	Approfondimenti sulle prestazioni monitora il carico delle istanze database per consentire di analizzare e risolvere i problemi di prestazioni del database. È consigliabile attivare Approfondimenti sulle prestazioni.	Attivare Performance Insights.	No	Monitoraggio del carico DB con Performance Insights su Amazon Aurora
È necessario eseguire l'aggiornamento delle versioni principali delle risorse RDS	I database con la versione principale corrente per il motore di database non saranno supportati. Si consiglia di eseguire l'aggiornamento alla versione principale più recente, che include nuove funzionalità e miglioramenti.	Esegui l'aggiornamento alla versione principale più recente per il motore di database.	Sì	Aggiornamenti di Amazon Aurora Creazione di una blue/green distribuzione in

Tipo	Descrizione	Raccomandazione	Tempo di inattività richiest	Informazioni aggiuntive
Dimensione massima del volume del cluster DB	Le versioni più recenti del motore supportano volumi di archiviazione più grandi per il cluster DB.	Ti consigliamo di aggiornare la versione del motore del tuo cluster DB alla versione più recente per beneficiare di una maggiore capacità di archiviazione.	Sì	Limiti di dimensione Amazon Aurora

Tipo	Descrizione	Raccomandazione	Tempo di inattività richiest	Informazioni aggiuntive
Cluster di database con tutte le istanze di lettura nella stessa zona di disponibilità	Le zone di disponibilità (AZs) sono località distinte l'una dall'altra per garantire l'isolamento in caso di interruzioni all'interno di ciascuna AWS regione. Ti consigliamo di distribuire l'istanza principale e le istanze di lettura nel cluster di database su più istanze AZs per migliorare la disponibilità del cluster di database. Puoi creare un cluster Multi-AZ utilizzando la console di AWS gestione, la AWS CLI o l'API Amazon RDS quando crei il cluster. È inoltre possibile modificare il cluster Aurora esistente in un cluster Multi-AZ aggiungendo una nuova istanza di lettura e specificando una zona di disponibilità diversa.	Tutte le istanze di lettura del cluster di database si trovano nella stessa zona di disponibilità. Consigliamo di distribuire le istanze di lettura in più zone di disponibilità. La distribuzione aumenta la disponibilità del database e migliora i tempi di risposta riducendo la latenza di rete tra client e database.	No	Elevata disponibilità di Amazon Aurora

Tipo	Descrizione	Raccomandazione	Tempo di inattività richiest	Informazioni aggiuntive
I parametri di memoria del database sono diversi da quelli predefiniti	I parametri di memoria delle istanze database sono significativamente diversi dai valori predefiniti. Queste impostazioni possono influire sulle prestazioni e causare errori. Ti consigliamo di reimpostare i parametri di memoria personalizzati per l'istanza database sui valori predefiniti nel gruppo di parametri del database.	Reimpostare i parametri di memoria sui valori predefiniti.	No	Gruppi di parametri per Amazon Aurora

Tipo	Descrizione	Raccomandazione	Tempo di inattività richiest	Informazioni aggiuntive
Il parametro della cache delle query è attivato	Quando le modifiche richiedono l'eliminazione della cache delle query, l'istanza database sembrerà bloccarsi. La maggior parte dei carichi di lavoro non beneficia della cache delle query. La cache delle query è stata rimossa da MySQL versione 8.0 e successive. È consigliabile impostare il parametro query_cache_type su 0.	Impostare il parametro `query_cache_type` su `0` nel gruppo di parametri del database.	Sì	Gruppi di parametri per Amazon Aurora

Tipo	Descrizione	Raccomandazione	Tempo di inattività richiest	Informazioni aggiuntive
Il parametro `log_output` è impostato su TABLE	Quando `log_output` è impostato su `TABLE`, viene utilizzato più spazio di archiviazione rispetto a quando `log_output` è impostato su `FILE`. Per evitare di raggiungere il limite di spazio di archiviazione, è consigliabile impostare il parametro su `FILE`. Impostato su `FILE` per impostazione predefinita in MySQL versione 8.4 e successive.	Impostare il parametro `log_output` su `FILE` nel gruppo di parametri del database.	No	File di log del database Aurora MySQL

Tipo	Descrizione	Raccomandazione	Tempo di inattività richiest	Informazioni aggiuntive
Il parametro `autovacuum` è disattivato	Il parametro autovacuum è disattivato per i delle istanze database. La disattivazione del parametro autovacuum aumenta le dimensioni della tabella e dell'indice e influisce sulle prestazioni. È consigliabile attivare il parametro autovacuum nei gruppi di parametri del database.	Attivare il parametro autovacuum nei gruppi di parametri del cluster di database.	No	Informazioni sull'autovacuum negli ambienti Amazon RDS for PostgreSQL nel Database Blog AWS

Tipo	Descrizione	Raccomandazione	Tempo di inattività richiest	Informazioni aggiuntive
Il parametro `synchronous_commit` è disattivato	Quando il parametro `synchronous_commit` è disattivato, i dati possono andare persi in caso di arresto anomalo del database. La durabilità del database è a rischio. Consigliamo di attivare il parametro `synchronous_commit`.	Attivare il parametro `synchronous_commit` nei gruppi di parametri del database.	Sì	Parametri PostgreSQL di Amazon Aurora: replica, sicurezza e registrazione nel blog del database AWS
Il parametro `track_counts` è disattivato	Quando il parametro `track_counts` è disattivato, il database non raccoglie le statistiche sulle attività del database. La funzione di autovacuum richiede che queste statistiche funzionino correttamente. Consigliamo di impostare il parametro `track_counts` su 1.	Impostare il parametro `track_counts` su 1.	No	Statistiche di esecuzione di PostgreSQL

Tipo	Descrizione	Raccomandazione	Tempo di inattività richiest	Informazioni aggiuntive
Il parametro `enable_indexonlyscan` è disattivato	Il pianificatore o l'ottimizzatore di query non può utilizzare il tipo di piano di scansione del solo indice se è disabilitato. È consigliabile impostare il parametro `enable_indexonlyscan` su 1.	Impostare il valore del parametro `enable_indexonlyscan` su 1.	No	Configurazione del metodo Planner per PostgreSQL
Il parametro `enable_indexscan` è disattivato	Il pianificatore o l'ottimizzatore di query non può utilizzare il tipo di piano di scansione del solo indice quando il parametro è disattivato. È consigliabile impostare `enable_indexscan` su 1.	Impostare il valore del parametro `enable_indexscan` su 1.	No	Configurazione del metodo Planner per PostgreSQL

Tipo	Descrizione	Raccomandazione	Tempo di inattività richiest	Informazioni aggiuntive
Il parametro `innodb_flush_log_at_trx` è disattivato	Il valore del parametro `innodb_flush_log_at_trx` dell'istanza database non è un valore sicuro. Questo parametro controlla la persistenza delle operazioni di commit su disco. Consigliamo di impostare il parametro `innodb_flush_log_at_trx` su 1.	Impostare il valore del parametro `innodb_flush_log_at_trx` su 1.	No	Configurazione della frequenza di svuotamento del buffer dei registri

Tipo	Descrizione	Raccomandazione	Tempo di inattività richiest	Informazioni aggiuntive
Il parametro `innodb_stats_persistent` è disattivato	L'istanza database non è configurata per memorizzare le statistiche InnoDB sul disco. Quando le statistiche non vengono archiviate, vengono ricalcolate ogni volta che l'istanza viene riavviata e accede alla tabella. Questo comporta variazioni nel piano di esecuzione delle query. Puoi modificare il valore di questo parametro globale a livello di tabella. È consigliabile impostare il parametro `innodb_stats_persistent` su `ON`.	Impostare il valore del parametro `innodb_stats_persistent` su `ON`.	No	Gruppi di parametri per Amazon Aurora

Tipo	Descrizione	Raccomandazione	Tempo di inattività richiest	Informazioni aggiuntive
Il valore del parametro `innodb_open_files` è basso	Il parametro `innodb_open_files` controlla il numero di file che InnoDB può aprire contemporaneamente. InnoDB apre tutti i file di log e di tablespace di sistema quando mysqld è in esecuzione. Il valore del numero massimo di file dell'istanza database che InnoDB può aprire contemporaneamente non è sufficiente. Consigliamo di impostare il parametro `innodb_open_files` almeno sul valore 65.	Impostare il parametro `innodb_open_files` su un valore minimo di 65.	Sì	InnoDB open files for MySQL

Tipo	Descrizione	Raccomandazione	Tempo di inattività richiest	Informazioni aggiuntive
Il valore del parametro `max_user_connections` è basso	Il valore del numero massimo di connessioni simultanee per ogni account di database dell'istanza database non è sufficiente. È consigliabile impostare il parametro `max_user_connections` su un numero maggiore di 5.	Aumentare il valore del parametro `max_user_connections` impostandolo su un numero maggiore di 5.	Sì	Impostazione dei limiti di risorse dell'account per MySQL
Le repliche di lettura sono aperte in modalità scrivibile	L'istanza database ha la replica di lettura in modalità scrivibile e consente gli aggiornamenti dei client. È consigliabile impostare il parametro `read_only` su `TrueIfReplica` affinché le repliche di lettura non siano in modalità scrivibile.	Impostare il valore del parametro `read_only` su `TrueIfReplica` .	No	Gruppi di parametri per Amazon Aurora

Tipo	Descrizione	Raccomandazione	Tempo di inattività richiest	Informazioni aggiuntive
L'impostazione del parametro `innodb_default_row_format` non è sicura	L'istanza database presenta un problema noto: una tabella creata in una versione di MySQL precedente alla versione 8.0.26 con `row_format` impostato su `COMPACT` o `REDUNDANT` diventa inaccessibile e irrecuperabile quando l'indice supera 767 byte. È consigliabile impostare il parametro `innodb_default_row_format` su `DYNAMIC`.	Impostare il valore del parametro `innodb_default_row_format` su `DYNAMIC`.	No	Modifiche in MySQL 8.0.26

Tipo	Descrizione	Raccomandazione	Tempo di inattività richiest	Informazioni aggiuntive
Il parametro `general_logging` è attivato	La registrazione di log generali dell'istanza database è attivata. Questa impostazione è utile per la risoluzione dei problemi del database. Tuttavia, l'attivazione della registrazione generale aumenta la quantità di I/O operazioni e lo spazio di storage allocato, il che potrebbe causare conflitti e un peggioramento delle prestazioni. Controllare i requisiti per l'utilizzo della registrazione di log generali. È consigliabile impostare il parametro `general_logging` su `0`.	Controllare i requisiti per l'utilizzo della registrazione di log generali. Se non è obbligatorio, è consigliabile impostare il valore del parametro `general_logging` su `0`.	No	Panoramica dei registri di database Aurora MySQL

Tipo	Descrizione	Raccomandazione	Tempo di inattività richiest	Informazioni aggiuntive
Capacità allocata del cluster di database insufficiente per il carico di lavoro di lettura	È consigliabile aggiungere un'istanza database di lettura al cluster di database con la stessa classe di istanza e le stesse dimensioni dell'istanza database di scrittura nel cluster. La configurazione corrente prevede un'istanza database con un carico di database costantemente elevato derivante principalmente da operazioni di lettura. Distribuire queste operazioni aggiungendo un'altra istanza database al cluster e indirizzando il carico di lavoro di lettura all'endpoint di sola lettura del cluster di database.	Aggiungere un'istanza database di lettura al cluster.	No	Aggiunta di repliche di Aurora a un cluster di database Gestione delle prestazioni e del dimensionamento dei cluster DB Aurora Prezzi di Amazon RDS

Tipo	Descrizione	Raccomandazione	Tempo di inattività richiest	Informazioni aggiuntive
La capacità allocata dell'istanza RDS è insufficiente rispetto alla capacità di memoria del sistema	È consigliabile ottimizzare le query in modo da utilizzare meno memoria o utilizzare un tipo di istanza database con una maggiore quantità di memoria allocata. Quando la memoria dell'istanza è prossima all'esaurimento, le prestazioni del database subiscono un peggioramento.	Utilizzare un'istanza database con una maggiore capacità di memoria	Sì	Scalabilità verticale e orizzontale dell'istanza Amazon RDS sul database Blog AWS Tipi di istanza Amazon RDS Prezzi di Amazon RDS
La capacità allocata dell'istanza RDS è insufficiente rispetto alla capacità di CPU del sistema	Ti consigliamo di ottimizzare le query in modo da utilizzare meno CPU o di modificare l'istanza DB per utilizzare una classe di istanza DB con un'allocazione maggiore di v. CPUs Le prestazioni del database potrebbero diminuire quando un'istanza DB sta esaurendo la CPU.	Utilizzare un'istanza database con una maggiore capacità di CPU	Sì	Scalabilità verticale e orizzontale dell'istanza Amazon RDS sul database Blog AWS Tipi di istanza Amazon RDS Prezzi di Amazon RDS

Tipo	Descrizione	Raccomandazione	Tempo di inattività richiest	Informazioni aggiuntive
Le risorse RDS non utilizzano correttamente il pooling delle connessioni	È consigliabile abilitare Server proxy per Amazon RDS per raggruppare e condividere in modo efficiente le connessioni a database esistenti. Se è già in uso un proxy per il database, configurarlo correttamente per migliorare il pooling delle connessioni e il bilanciamento del carico tra più istanze database. Server proxy per Amazon RDS può contribuire a ridurre il rischio di esaurimento delle connessioni e i tempi di inattività, migliorando al contempo la disponibilità e la scalabilità.	Abilitare Server proxy per Amazon RDS o modificare la configurazione proxy esistente	No	Scalabilità verticale e orizzontale dell'istanza Amazon RDS sul database Blog AWS Server proxy per Amazon RDS per Aurora Prezzi del Server proxy per Amazon RDS

Visualizzazione delle metriche nella console Amazon RDS

Amazon RDS si integra con Amazon CloudWatch per visualizzare una varietà di metriche per il cluster di database Aurora nella console RDS. Alcune metriche vengono applicate a livello di cluster, mentre altri si applicano a livello di istanza. Per le descrizioni di delle metriche a livello di istanza e cluster, consulta Riferimento per i parametri per Amazon Aurora.

Per il cluster database Aurora, vengono monitorate le seguenti categorie di metriche:

- CloudWatch: mostra le metriche Amazon CloudWatch per Aurora a cui è possibile accedere nella console RDS. Puoi visualizzare tali parametri anche nella console CloudWatch. Ogni parametro include un grafico che mostra il parametro monitorato in un periodo di tempo specifico. Per un elenco completo dei parametri CloudWatch, consulta CloudWatch Parametri Amazon per Amazon Aurora.
- Enhanced monitoring (Monitoraggio avanzato): mostra un riepilogo dei parametri del sistema operativo quando il cluster DB Aurora ha attivato il monitoraggio avanzato. RDS fornisce le metriche del monitoraggio avanzato al tuo account Amazon CloudWatch Logs. Ogni parametro del sistema operativo include un grafico che visualizza il parametro monitorato in un periodo di tempo specifico. Per una panoramica, consulta Monitoraggio dei parametri del sistema operativo con il monitoraggio avanzato. Per un elenco di metriche di monitoraggio avanzato, consulta Metriche del sistema operativo nel monitoraggio avanzato.
- OS Process list (Elenco processi sistema operativo): mostra i dettagli di ogni processo in esecuzione nel cluster di database.
- Performance Insights: apre il pannello di controllo di Amazon RDS Performance Insights per un'istanza database nel cluster di database Aurora. Performance Insights non è supportato a livello di cluster.Per una panoramica su Performance Insights, consulta Monitoraggio del carico DB con Performance Insights su Amazon Aurora. Per un elenco di parametri di Performance Insights, consulta Metriche Amazon CloudWatch per Approfondimenti sulle prestazioni di Amazon RDS.

Amazon RDS ora fornisce una visualizzazione consolidata delle metriche di Performance Insights e CloudWatch nel pannello di controllo di Performance Insights. Performance Insights deve essere attivato affinché il cluster database possa utilizzare questa visualizzazione. È possibile scegliere la nuova visualizzazione di monitoraggio nella scheda Monitoraggio o Performance Insights nel pannello di navigazione. Per visualizzare le istruzioni per scegliere questa visualizzazione, consultare Visualizzazione delle metriche combinate con la dashboard di Approfondimenti sulle prestazioni.

Visualizzazione delle metriche combinate con la dashboard di Approfondimenti sulle prestazioni

Important

AWS ha annunciato la end-of-life data di Performance Insights: 30 giugno 2026. Dopo questa data, Amazon RDS non supporterà più l'esperienza della console Approfondimenti sulle prestazioni, i periodi di conservazione flessibili (1-24 mesi) e i relativi prezzi. L'API Approfondimenti sulle prestazioni continuerà a esistere senza modifiche ai prezzi. I costi per l'API Performance Insights verranno visualizzati nella AWS fattura insieme al costo di CloudWatch Database Insights.
Si consiglia di aggiornare tutte DB Clusters di database utilizzando il livello a pagamento di Performance Insights alla modalità Advanced di Database Insights prima del 30 giugno 2026. Per informazioni sull'aggiornamento alla modalità avanzata di Database Insights, consulta Attivazione della modalità avanzata di Database Insights per Amazon Aurora.
Se non si esegue alcuna azione, i cluster di database che utilizzano Approfondimenti sulle prestazioni, per impostazione predefinita useranno la modalità standard di Database Insights. Con la modalità standard di Database Insights, è possibile perdere l'accesso alla cronologia dei dati sulle prestazioni di oltre 7 giorni e non essere in grado di utilizzare i piani di esecuzione e le funzionalità di analisi on demand nella console Amazon RDS. Dopo il 30 giugno 2026 solo la modalità Advanced di Database Insights supporterà i piani di esecuzione e l'analisi su richiesta.
Con CloudWatch Database Insights, puoi monitorare il carico del database per la tua flotta di database e analizzare e risolvere i problemi delle prestazioni su larga scala. Per ulteriori informazioni su Database Insights, consulta Monitoraggio dei database Amazon Aurora con CloudWatch Database Insights. Per informazioni sui prezzi, consulta la pagina CloudWatch dei prezzi di Amazon.

Amazon RDS offre una visualizzazione consolidata di Performance Insights e dei CloudWatch parametri per l'istanza DB nella dashboard di Performance Insights. È possibile utilizzare il pannello di controllo preconfigurato o crearne uno personalizzato. Il pannello di controllo preconfigurato fornisce le metriche più comunemente utilizzate per diagnosticare i problemi relativi alle prestazioni di un motore di database. In alternativa, è possibile creare un pannello di controllo personalizzato contenente le metriche di un motore di database che soddisfi i requisiti di analisi definiti. Quindi,

utilizza questa dashboard per tutte le istanze DB di quel tipo di motore di database presenti nel tuo account. AWS

È possibile scegliere la visualizzazione di monitoraggio nella scheda Monitoraggio o Approfondimenti sulle prestazioni nel pannello di navigazione.

Performance Insights deve essere attivato affinché il cluster database possa visualizzare le metriche combinate nel pannello di controllo di Performance Insights. Per ulteriori informazioni sull'attivazione di Performance Insights, consultare Attivazione e disattivazione di Approfondimenti sulle prestazioni per Aurora.

Nelle sezioni seguenti, puoi imparare a visualizzare Performance Insights e CloudWatch metriche.

Argomenti

- Scelta della nuova visualizzazione di monitoraggio nella scheda Monitoraggio
- Scelta della nuova visualizzazione di monitoraggio dalla pagina Approfondimenti sulle prestazioni
- Creazione di una dashboard personalizzata con Approfondimenti sulle prestazioni
- Scelta della dashboard preconfigurata con Approfondimenti sulle prestazioni

Scelta della nuova visualizzazione di monitoraggio nella scheda Monitoraggio

> **Important**
>
> AWS ha annunciato la end-of-life data di Performance Insights: 30 giugno 2026. Dopo questa data, Amazon RDS non supporterà più l'esperienza della console Approfondimenti sulle prestazioni, i periodi di conservazione flessibili (1-24 mesi) e i relativi prezzi. L'API Approfondimenti sulle prestazioni continuerà a esistere senza modifiche ai prezzi. I costi per l'API Performance Insights verranno visualizzati nella AWS fattura insieme al costo di CloudWatch Database Insights.
> Si consiglia di aggiornare tutte DB Clusters di database utilizzando il livello a pagamento di Performance Insights alla modalità Advanced di Database Insights prima del 30 giugno 2026. Per informazioni sull'aggiornamento alla modalità avanzata di Database Insights, consulta Attivazione della modalità avanzata di Database Insights per Amazon Aurora.
> Se non si esegue alcuna azione, i cluster di database che utilizzano Approfondimenti sulle prestazioni, per impostazione predefinita useranno la modalità standard di Database

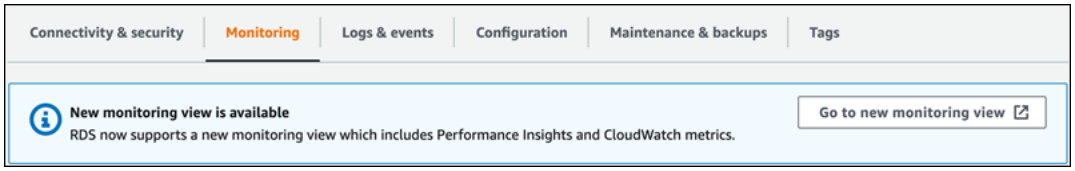
Insights. Con la modalità standard di Database Insights, è possibile perdere l'accesso alla cronologia dei dati sulle prestazioni di oltre 7 giorni e non essere in grado di utilizzare i piani di esecuzione e le funzionalità di analisi on demand nella console Amazon RDS. Dopo il 30 giugno 2026 solo la modalità Advanced di Database Insights supporterà i piani di esecuzione e l'analisi su richiesta.
Con CloudWatch Database Insights, puoi monitorare il carico del database per la tua flotta di database e analizzare e risolvere i problemi delle prestazioni su larga scala. Per ulteriori informazioni su Database Insights, consulta Monitoraggio dei database Amazon Aurora con CloudWatch Database Insights. Per informazioni sui prezzi, consulta la pagina CloudWatch dei prezzi di Amazon.

Dalla console Amazon RDS, puoi scegliere la nuova vista di monitoraggio per visualizzare Performance Insights e CloudWatch metriche per la tua istanza DB.

Per scegliere la nuova visualizzazione di monitoraggio nella scheda Monitoraggio

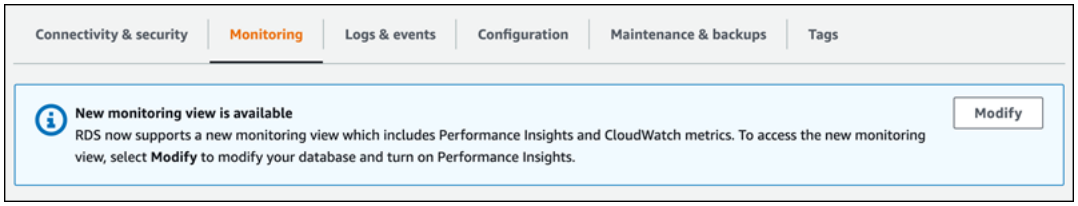
1. Accedi a Console di gestione AWS e apri la console Amazon RDS all'indirizzo https://console.aws.amazon.com/rds/.
2. Nel pannello di navigazione a sinistra, scegliere Database.
3. Scegliere il cluster database Aurora che si desidera monitorare.
4. Scorrere verso il basso e selezionare Monitoraggio.

   Viene visualizzato un banner con l'opzione che consente di scegliere la nuova visualizzazione di monitoraggio. Nell'esempio seguente è rappresentato il banner per scegliere la nuova visualizzazione di monitoraggio.



5. Scegli Vai alla nuova visualizzazione di monitoraggio per aprire la dashboard di Performance Insights con Performance Insights e CloudWatch metriche per l'.
6. (Facoltativo) Se Performance Insights è disattivato per l'istanza database, viene visualizzato un banner con l'opzione per modificare l'istanza database e attivare Performance Insights.

   L'esempio seguente mostra il banner per modificare l'istanza database nella scheda Monitoraggio.



Scegliere Modifica per modificare l'istanza database e attivare Performance Insights. Per ulteriori informazioni sull'attivazione di Performance Insights, consultare Attivazione e disattivazione di Approfondimenti sulle prestazioni per Aurora.

Scelta della nuova visualizzazione di monitoraggio dalla pagina Approfondimenti sulle prestazioni

Important

AWS ha annunciato la end-of-life data di Performance Insights: 30 giugno 2026. Dopo questa data, Amazon RDS non supporterà più l'esperienza della console Approfondimenti sulle prestazioni, i periodi di conservazione flessibili (1-24 mesi) e i relativi prezzi. L'API Approfondimenti sulle prestazioni continuerà a esistere senza modifiche ai prezzi. I costi per l'API Performance Insights verranno visualizzati nella AWS fattura insieme al costo di CloudWatch Database Insights.

Si consiglia di aggiornare tutte DB Clusters di database utilizzando il livello a pagamento di Performance Insights alla modalità Advanced di Database Insights prima del 30 giugno 2026. Per informazioni sull'aggiornamento alla modalità avanzata di Database Insights, consulta Attivazione della modalità avanzata di Database Insights per Amazon Aurora.

Se non si esegue alcuna azione, i cluster di database che utilizzano Approfondimenti sulle prestazioni, per impostazione predefinita useranno la modalità standard di Database Insights. Con la modalità standard di Database Insights, è possibile perdere l'accesso alla cronologia dei dati sulle prestazioni di oltre 7 giorni e non essere in grado di utilizzare i piani di esecuzione e le funzionalità di analisi on demand nella console Amazon RDS. Dopo il 30 giugno 2026 solo la modalità Advanced di Database Insights supporterà i piani di esecuzione e l'analisi su richiesta.

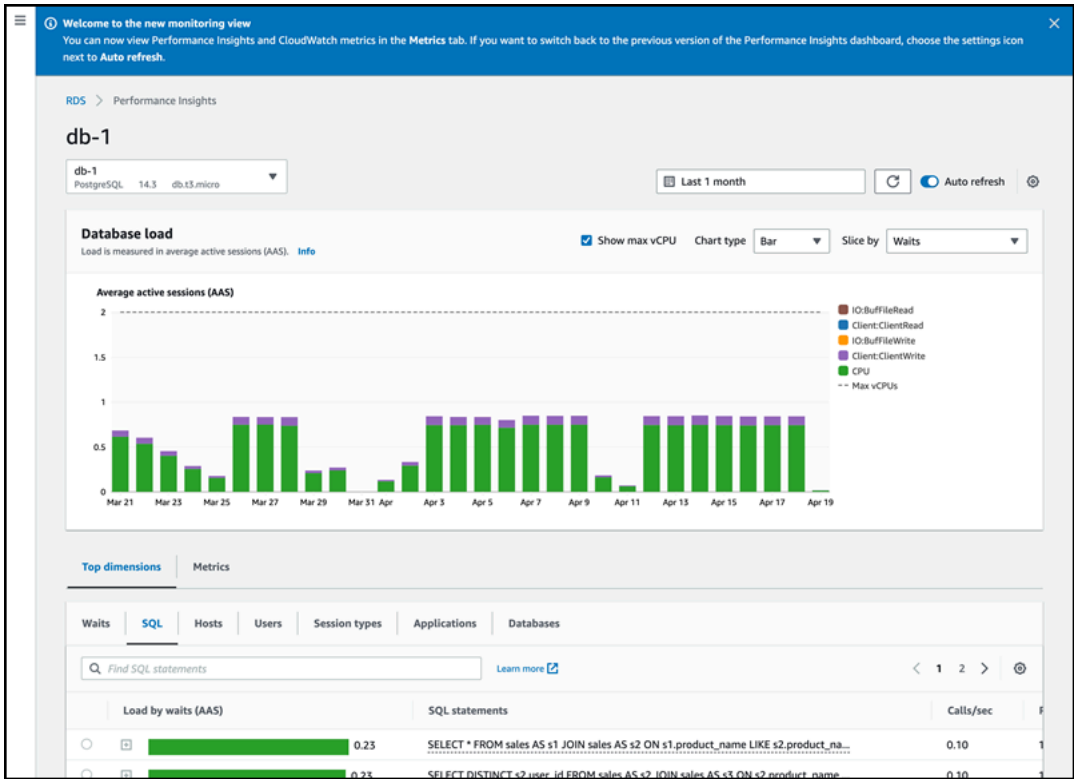
Con CloudWatch Database Insights, puoi monitorare il carico del database per la tua flotta di database e analizzare e risolvere i problemi delle prestazioni su larga scala. Per ulteriori informazioni su Database Insights, consulta Monitoraggio dei database Amazon Aurora con

CloudWatch Database Insights. Per informazioni sui prezzi, consulta la pagina CloudWatch dei prezzi di Amazon.

Dalla console Amazon RDS, puoi scegliere la nuova vista di monitoraggio per visualizzare Performance Insights e CloudWatch metriche per la tua istanza DB.

Per scegliere la nuova visualizzazione di monitoraggio con Approfondimenti sulle prestazioni nel riquadro di navigazione

1. Accedi a Console di gestione AWS e apri la console Amazon RDS all'indirizzo https://console.aws.amazon.com/rds/.
2. Nel pannello di navigazione scegli Performance Insights.
3. Scegli un'istanza DB per visualizzare la dashboard di Performance Insights che mostra sia Performance Insights che le CloudWatch metriche per la tua istanza DB.



Creazione di una dashboard personalizzata con Approfondimenti sulle prestazioni

Important

AWS ha annunciato la end-of-life data di Performance Insights: 30 giugno 2026. Dopo questa data, Amazon RDS non supporterà più l'esperienza della console Approfondimenti sulle prestazioni, i periodi di conservazione flessibili (1-24 mesi) e i relativi prezzi. L'API Approfondimenti sulle prestazioni continuerà a esistere senza modifiche ai prezzi. I costi per l'API Performance Insights verranno visualizzati nella AWS fattura insieme al costo di CloudWatch Database Insights.
Si consiglia di aggiornare tutte DB Clusters di database utilizzando il livello a pagamento di Performance Insights alla modalità Advanced di Database Insights prima del 30 giugno 2026. Per informazioni sull'aggiornamento alla modalità avanzata di Database Insights, consulta Attivazione della modalità avanzata di Database Insights per Amazon Aurora.
Se non si esegue alcuna azione, i cluster di database che utilizzano Approfondimenti sulle prestazioni, per impostazione predefinita useranno la modalità standard di Database Insights. Con la modalità standard di Database Insights, è possibile perdere l'accesso alla cronologia dei dati sulle prestazioni di oltre 7 giorni e non essere in grado di utilizzare i piani di esecuzione e le funzionalità di analisi on demand nella console Amazon RDS. Dopo il 30 giugno 2026 solo la modalità Advanced di Database Insights supporterà i piani di esecuzione e l'analisi su richiesta.
Con CloudWatch Database Insights, puoi monitorare il carico del database per la tua flotta di database e analizzare e risolvere i problemi delle prestazioni su larga scala. Per ulteriori informazioni su Database Insights, consulta Monitoraggio dei database Amazon Aurora con CloudWatch Database Insights. Per informazioni sui prezzi, consulta la pagina CloudWatch dei prezzi di Amazon.

Nella nuova visualizzazione di monitoraggio, è possibile creare un pannello di controllo personalizzato con le metriche necessarie per soddisfare gli specifici requisiti di analisi.

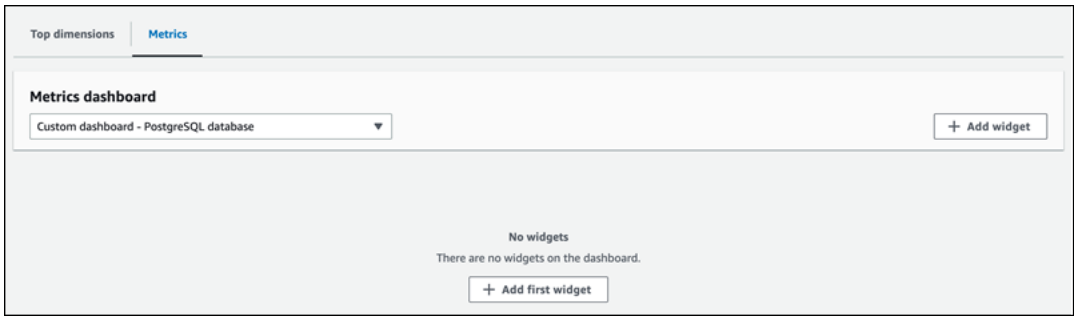
Puoi creare una dashboard personalizzata selezionando Performance Insights e CloudWatch metriche per la tua istanza DB. Puoi utilizzare questa dashboard personalizzata per altre istanze DB dello stesso tipo di motore di database nel tuo AWS account.

> **Note**
>
> Il pannello di controllo personalizzato supporta fino a 50 metriche.

Usare il menu delle impostazioni del widget per modificare o eliminare il pannello di controllo e spostare o ridimensionare la finestra del widget.

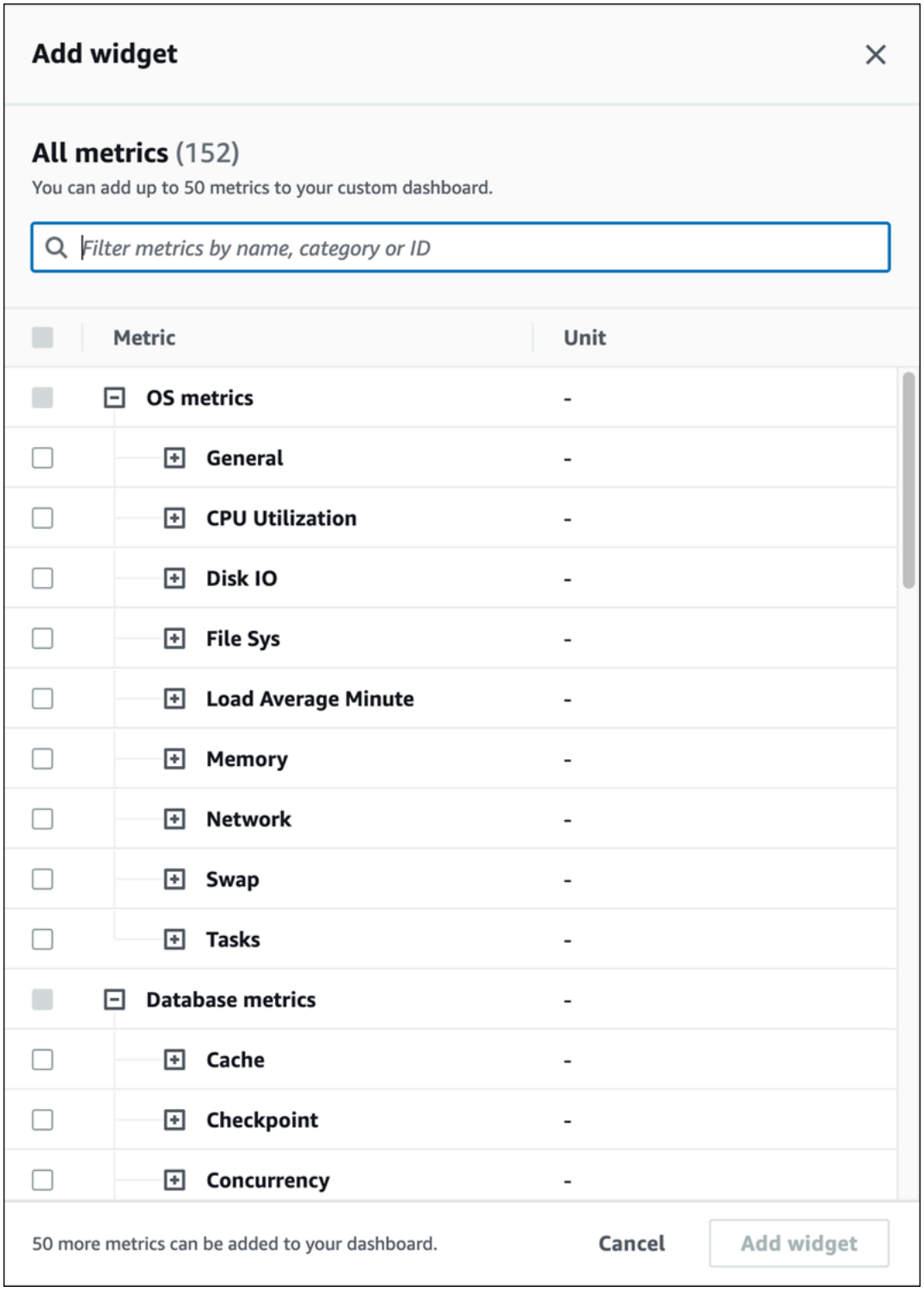
Per creare una dashboard personalizzata con Approfondimenti sulle prestazioni nel pannello di navigazione:

1. Accedi a Console di gestione AWS e apri la console Amazon RDS all'indirizzo https://console.aws.amazon.com/rds/.
2. Nel pannello di navigazione scegli Performance Insights.
3. Scegli un'istanza database.
4. Scorrere verso il basso fino alla scheda Metriche nella finestra.
5. Selezionare il pannello di controllo personalizzato nell'elenco a discesa. Nell'esempio seguente viene illustrata la creazione del pannello di controllo personalizzato.



6. Scegli Aggiungi widget per aprire la finestra Aggiungi widget. Puoi aprire e visualizzare i parametri del sistema operativo (OS), i parametri del database e i parametri disponibili nella CloudWatch finestra.

   L'esempio seguente mostra la finestra Aggiungi widget con le metriche.



7. Selezionare le metriche da visualizzare nel pannello di controllo e scegliere Aggiungi widget. È possibile utilizzare il campo di ricerca per cercare una metrica specifica.

   Le metriche selezionate vengono visualizzate nel pannello di controllo.

8. (Facoltativo) Per modificare o eliminare il pannello di controllo, scegliere l'icona delle impostazioni in alto a destra del widget, quindi selezionare una delle seguenti azioni nel menu.

   - Modifica: consente di modificare l'elenco delle metriche nella finestra. Scegliere Aggiorna widget dopo aver selezionato le metriche da visualizzare nel pannello di controllo.
   - Elimina: consente di eliminare il widget. Nella finestra di conferma scegliere Elimina.

Scelta della dashboard preconfigurata con Approfondimenti sulle prestazioni

Important

AWS ha annunciato la end-of-life data di Performance Insights: 30 giugno 2026. Dopo questa data, Amazon RDS non supporterà più l'esperienza della console Approfondimenti sulle prestazioni, i periodi di conservazione flessibili (1-24 mesi) e i relativi prezzi. L'API Approfondimenti sulle prestazioni continuerà a esistere senza modifiche ai prezzi. I costi per l'API Performance Insights verranno visualizzati nella AWS fattura insieme al costo di CloudWatch Database Insights.
Si consiglia di aggiornare tutte DB Clusters di database utilizzando il livello a pagamento di Performance Insights alla modalità Advanced di Database Insights prima del 30 giugno 2026. Per informazioni sull'aggiornamento alla modalità avanzata di Database Insights, consulta Attivazione della modalità avanzata di Database Insights per Amazon Aurora.
Se non si esegue alcuna azione, i cluster di database che utilizzano Approfondimenti sulle prestazioni, per impostazione predefinita useranno la modalità standard di Database Insights. Con la modalità standard di Database Insights, è possibile perdere l'accesso alla cronologia dei dati sulle prestazioni di oltre 7 giorni e non essere in grado di utilizzare i piani di esecuzione e le funzionalità di analisi on demand nella console Amazon RDS. Dopo il 30 giugno 2026 solo la modalità Advanced di Database Insights supporterà i piani di esecuzione e l'analisi su richiesta.
Con CloudWatch Database Insights, puoi monitorare il carico del database per la tua flotta di database e analizzare e risolvere i problemi delle prestazioni su larga scala. Per ulteriori informazioni su Database Insights, consulta Monitoraggio dei database Amazon Aurora con CloudWatch Database Insights. Per informazioni sui prezzi, consulta la pagina CloudWatch dei prezzi di Amazon.

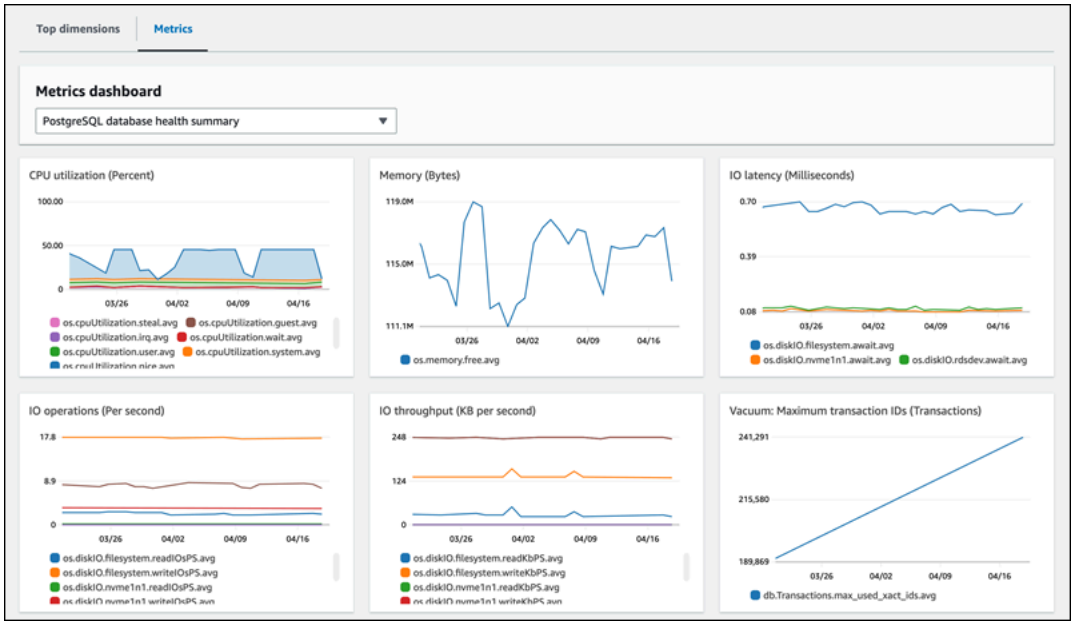
Nel pannello di controllo preconfigurato è possibile visualizzare le metriche di più frequente utilizzo. Questo pannello di controllo aiuta a diagnosticare i problemi di prestazioni di un motore di database e a ridurre il tempo medio di ripristino da ore a minuti.

> **Note**
>
> Questo pannello di controllo non può essere modificato.

Per scegliere la dashboard preconfigurata con Approfondimenti sulle prestazioni nel riquadro di navigazione

1. Accedi a Console di gestione AWS e apri la console Amazon RDS all'indirizzo https://console.aws.amazon.com/rds/.
2. Nel pannello di navigazione scegli Performance Insights.
3. Scegli un'istanza database.
4. Scorrere verso il basso fino alla scheda Metriche nella finestra.
5. Selezionare un pannello di controllo preconfigurato nell'elenco a discesa.

   Nel pannello di controllo è possibile visualizzare le metriche per l'istanza database. Nell'esempio seguente viene illustrato un pannello di controllo preconfigurato con le metriche.



Monitoraggio dei parametri di Amazon Aurora con Amazon CloudWatch

Amazon CloudWatch è un repository di parametri. Il repository raccoglie ed elabora i dati non elaborati da Amazon Aurora in parametri leggibili quasi in tempo reale. Per l'elenco completo dei parametri di Amazon Aurora inviati a CloudWatch, consulta Guida di riferimento per i parametri di Amazon Aurora.

Per analizzare e risolvere i problemi delle prestazioni dei database su larga scala, utilizza CloudWatch Database Insights.

Argomenti

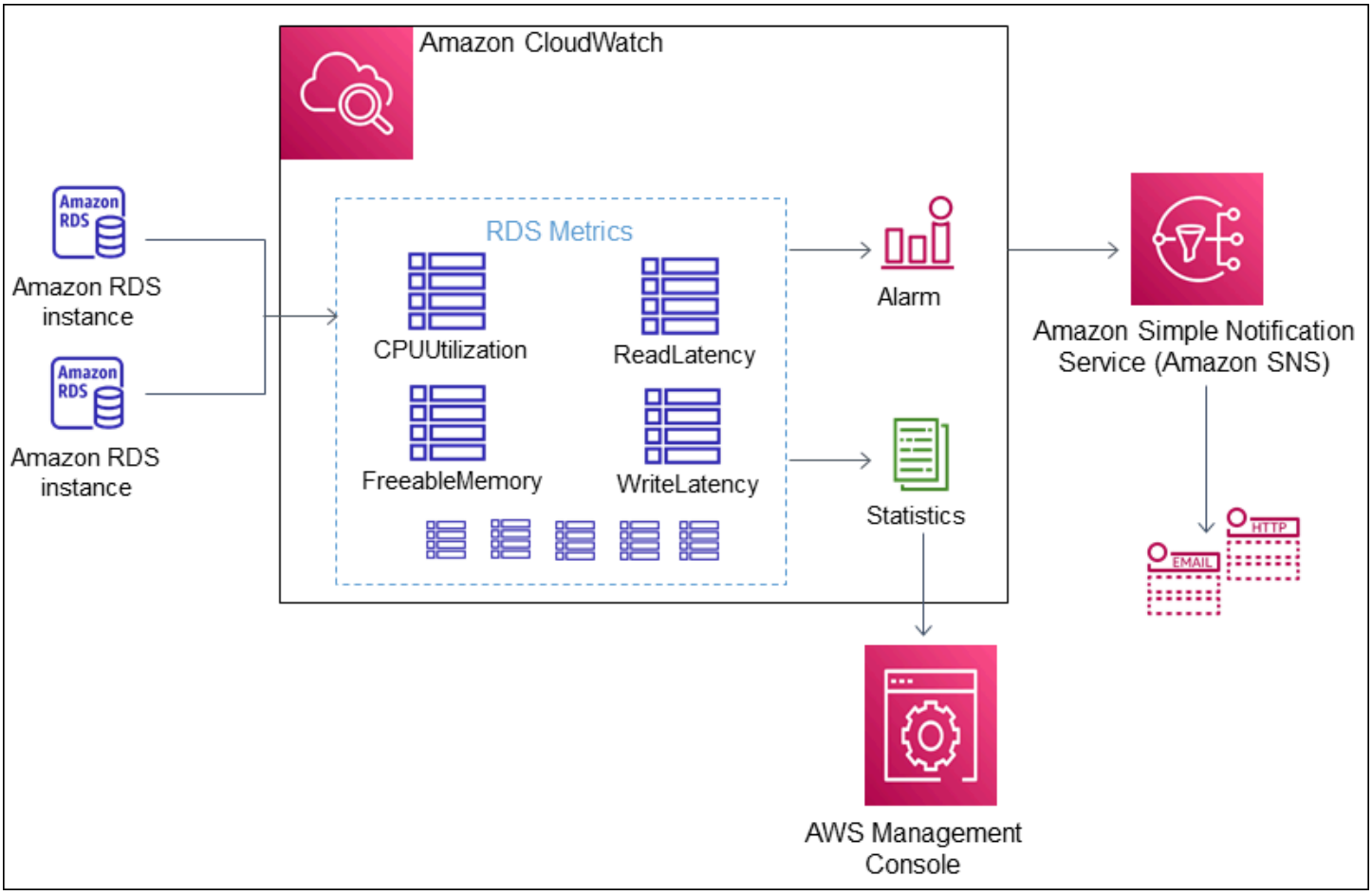
- Panoramica di Amazon Aurora e Amazon CloudWatch
- Visualizzazione dei parametri del cluster di database nella console CloudWatch e in AWS CLI
- Esportazione dei parametri di Approfondimenti sulle prestazioni in Amazon CloudWatch
- Creazione di allarmi CloudWatch per il monitoraggio di Amazon Aurora

Panoramica di Amazon Aurora e Amazon CloudWatch

Per impostazione predefinita, Amazon Aurora invia automaticamente i dati dei parametri a CloudWatch a intervalli di 1 minuto. Ad esempio, il parametro `CPUUtilization` registra la percentuale di utilizzo della CPU per un'istanza database nel tempo. I punti di dati con un periodo di 60 secondi (1 minuto) sono disponibili per 15 giorni. Ciò significa che è possibile accedere alle informazioni della cronologia e visualizzare le prestazioni del servizio o dell'applicazione Web.

Ora puoi esportare le dashboard dei parametri di Approfondimenti sulle prestazioni da Amazon RDS ad Amazon CloudWatch. Puoi esportare le dashboard dei parametri preconfigurate o personalizzate come nuove dashboard o aggiungerle a una dashboard CloudWatch esistente. La dashboard esportata è disponibile per la visualizzazione nella console CloudWatch. Per ulteriori informazioni su come esportare le dashboard dei parametri di Approfondimenti sulle prestazioni su CloudWatch, consulta Esportazione dei parametri di Approfondimenti sulle prestazioni in Amazon CloudWatch.

Come illustrato nel seguente diagramma, è possibile impostare gli allarmi per i parametri di CloudWatch. Ad esempio, puoi creare un allarme che segnali quando l'utilizzo della CPU per un'istanza è superiore al 70%. È possibile configurare Amazon Simple Notification Service in modo da ricevere un messaggio e-mail quando viene superata la soglia.



Amazon RDS pubblica i seguenti tipi di parametri in Amazon CloudWatch:

- Parametri Aurora a livello di cluster e di istanza

  Per una tabella di questi parametri, consulta CloudWatch Parametri Amazon per Amazon Aurora.
- Parametri Performance Insights

  Per una tabella di questi parametri, consulta Metriche Amazon CloudWatch per Approfondimenti sulle prestazioni di Amazon RDS e Metriche contatore di Performance Insights.
- Parametri di Monitoraggio avanzato (pubblicati in Amazon CloudWatch Logs)

  Per una tabella di questi parametri, consulta Metriche del sistema operativo nel monitoraggio avanzato.
- Parametri di utilizzo per le quote di servizio Amazon RDS nell'Account AWS

  Per una tabella di questi parametri, consulta Metriche CloudWatch di utilizzo di . Per ulteriori informazioni sulle quote di Amazon RDS, consulta Quote e vincoli per Amazon Aurora.

Per ulteriori informazioni su CloudWatch, consulta Che cos'è Amazon CloudWatch? nella Guida per l'utente di Amazon CloudWatch. Per ulteriori informazioni sulla conservazione dei parametri di CloudWatch, consulta Conservazione dei parametri.

Visualizzazione dei parametri del cluster di database nella console CloudWatch e in AWS CLI

Di seguito, è possibile trovare dettagli su come visualizzare i parametri per l'istanza database utilizzando CloudWatch. Per informazioni sul monitoraggio di parametri per il sistema operativo dell'istanza database in tempo reale tramite CloudWatch Logs, consulta Monitoraggio dei parametri del sistema operativo con il monitoraggio avanzato.

Quando utilizzi risorse Amazon Aurora, Amazon Aurora invia parametri e dimensioni ad Amazon CloudWatch ogni minuto.

Ora puoi esportare le dashboard dei parametri di Approfondimenti sulle prestazioni da Amazon RDS ad Amazon CloudWatch e visualizzare questi parametri nella console CloudWatch. Per ulteriori informazioni su come esportare le dashboard dei parametri di Approfondimenti sulle prestazioni su CloudWatch, consulta Esportazione dei parametri di Approfondimenti sulle prestazioni in Amazon CloudWatch.

Utilizza le procedure seguenti per visualizzare i parametri per Amazon Aurora nella console CloudWatch e nella CLI.

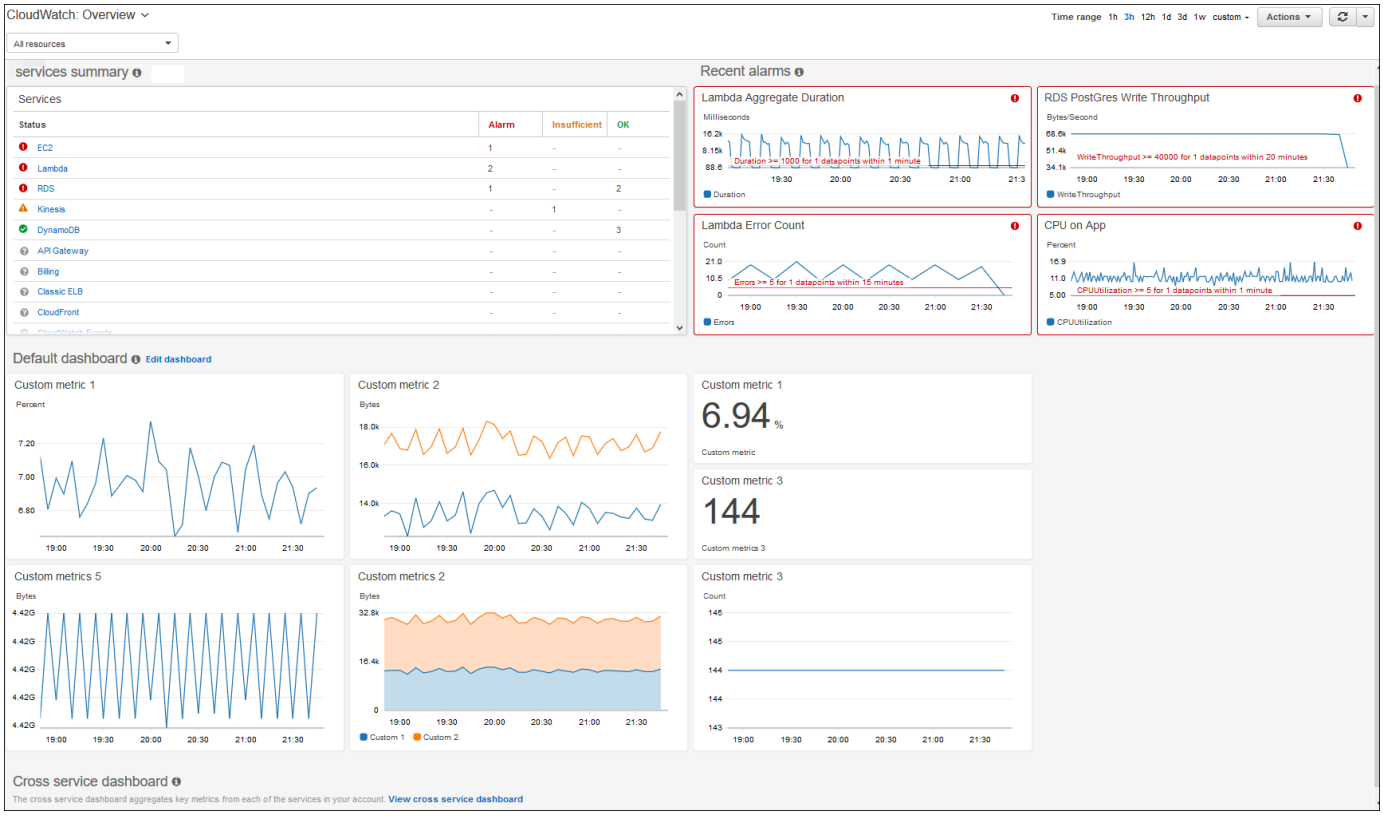
Console

Per visualizzare i parametri tramite la console Amazon CloudWatch

I parametri vengono raggruppati prima in base al namespace del servizio e successivamente in base alle diverse combinazioni di dimensioni all'interno di ogni namespace.

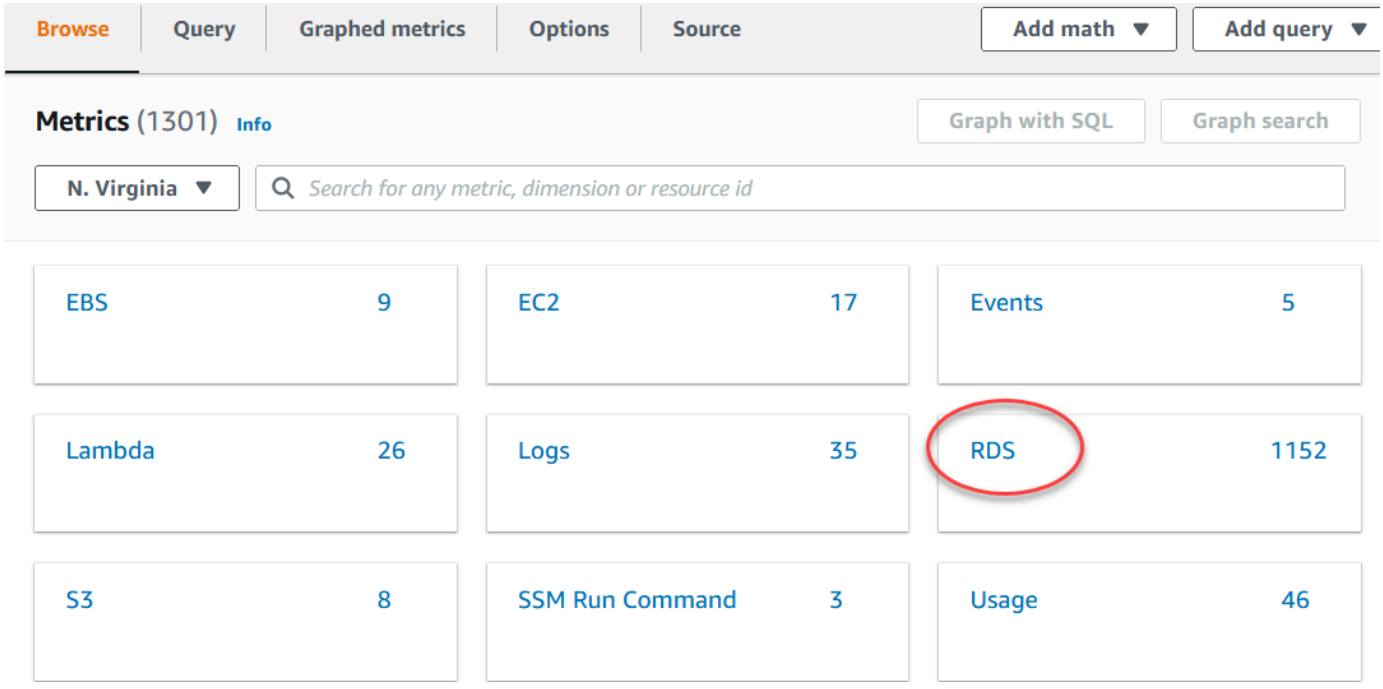
1. Apri la console CloudWatch all’indirizzo https://console.aws.amazon.com/cloudwatch/.

   Viene visualizzata la home page della panoramica di CloudWatch.



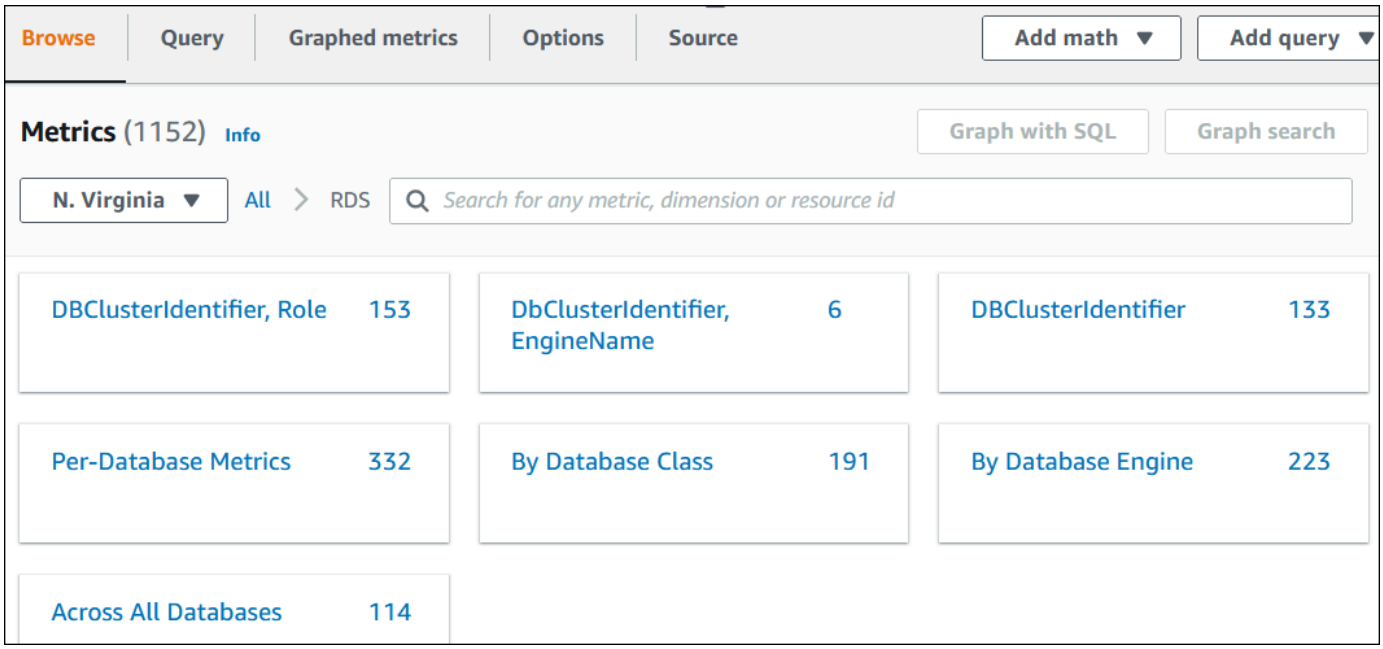
2. Se necessario, modifica Regione AWS. Dalla barra di navigazione, seleziona la Regione AWS in cui si trovano le risorse AWS. Per ulteriori informazioni, consulta Regioni ed endpoint.

3. Nel pannello di navigazione, scegli Metrics (Parametri), quindi scegli All metrics (Tutti i parametri).

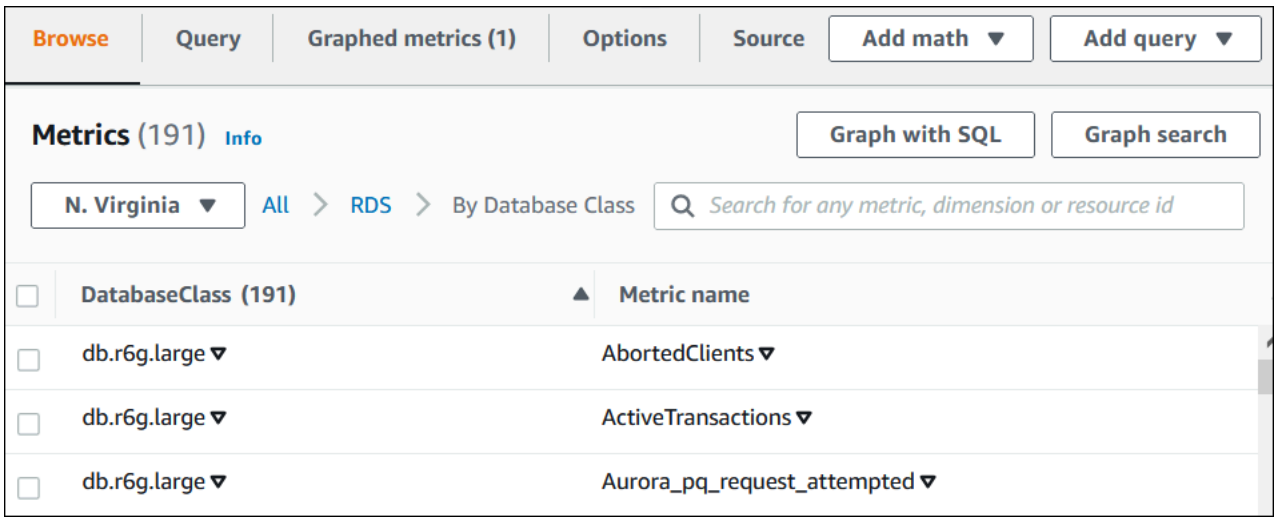


4. Scorri verso il basso e scegli il parametro namespace RDS.

   La pagina mostra le dimensioni Amazon Aurora. Per una descrizione di queste dimensioni, consulta CloudWatch Dimensioni Amazon per .



5. Scegli una dimensione di parametro, ad esempio By Database Class (Per classe di database).

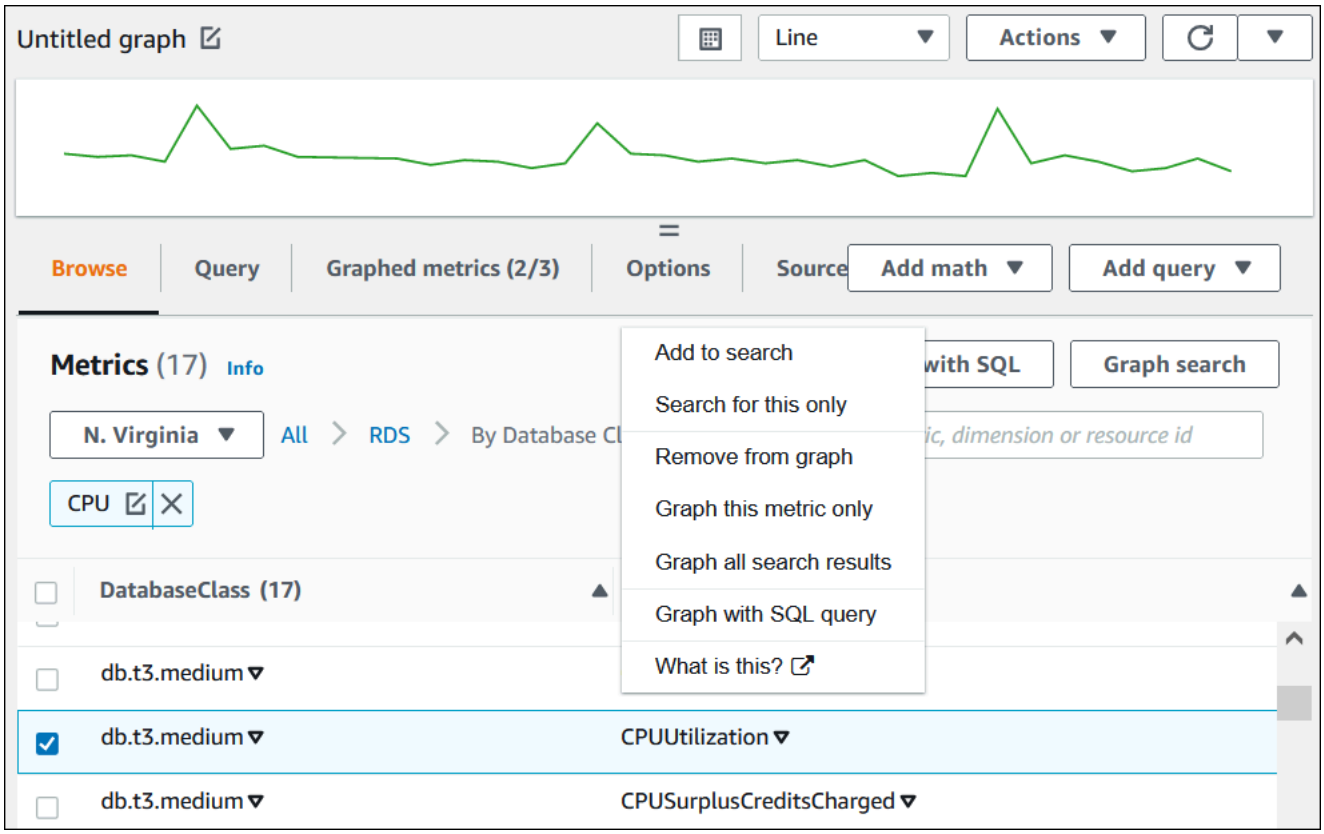


6. Effettua una delle seguenti operazioni:

   - Per ordinare i parametri, utilizza l'intestazione della colonna.
   - Per creare il grafico di un parametro, seleziona la casella di controllo accanto al parametro.
   - Per filtrare in base a una risorsa, scegli l'ID della risorsa e quindi Add to search (Aggiungi alla ricerca).

- Per filtrare in base a un parametro, scegli il nome del parametro e quindi Add to search (Aggiungi alla ricerca).

Il seguente esempio filtra sui grafici e sulla classe db.t3.medium il parametro CPUUtilization.



Sono disponibili i dettagli su come analizzare l'utilizzo delle risorse per Aurora PostgreSQL usando i parametri di CloudWatch. Per ulteriori informazioni, consulta Utilizzo dei CloudWatch parametri di Amazon per analizzare l'utilizzo delle risorse per Aurora PostgreSQL

AWS CLI

Per ottenere informazioni sui parametri utilizzando la AWS CLI, usa il comando `list-metrics` di CloudWatch. Nell'esempio seguente, vengono elencati tutti i parametri nello spazio dei nomi `AWS/RDS`.

```
aws cloudwatch list-metrics --namespace AWS/RDS
```

Per ottenere i dati delle metriche, utilizzare il comando `get-metric-data`.

Il seguente esempio ottiene statistiche `CPUUtilization` per l'istanza `my-instance` nel periodo specifico di 24 ore, con una granularità di 5 minuti.

Crea un file JSON `CPU_metric.json` con i seguenti contenuti.

```
{
   "StartTime" : "2023-12-25T00:00:00Z",
   "EndTime" : "2023-12-26T00:00:00Z",
   "MetricDataQueries" : [{
     "Id" : "cpu",
     "MetricStat" : {
    "Metric" : {
        "Namespace" : "AWS/RDS",
        "MetricName" : "CPUUtilization",
        "Dimensions" : [{ "Name" : "DBInstanceIdentifier" , "Value" : my-instance}]
    },
       "Period" : 360,
       "Stat" : "Minimum"
     }
   }]
}
```

Example

Per Linux, macOS o Unix:

```
aws cloudwatch get-metric-data \
    --cli-input-json file://CPU_metric.json
```

Per Windows:

```
aws cloudwatch get-metric-data ^
     --cli-input-json file://CPU_metric.json
```

L'output di esempio viene visualizzato come segue:

```
{
    "MetricDataResults": [
        {
            "Id": "cpu",
            "Label": "CPUUtilization",
```

```
            "Timestamps": [
                "2023-12-15T23:48:00+00:00",
                "2023-12-15T23:42:00+00:00",
                "2023-12-15T23:30:00+00:00",
                "2023-12-15T23:24:00+00:00",
                ...
            ],
            "Values": [
                13.299778337027714,
                13.677507543049558,
                14.24976250395827,
                13.02521708695145,
                ...
            ],
            "StatusCode": "Complete"
        }
    ],
    "Messages": []
}
```


Per ulteriori informazioni, consultare Ottenere statistiche per un parametro nella Guida per l'utente di Amazon CloudWatch.

Esportazione dei parametri di Approfondimenti sulle prestazioni in Amazon CloudWatch

Approfondimenti sulle prestazioni ti consente di esportare la dashboard dei parametri preconfigurata o personalizzata per la tua istanza database su Amazon CloudWatch. Puoi esportare la dashboard dei parametri come nuova dashboard o aggiungerla a una dashboard CloudWatch esistente. Quando scegli di aggiungere la dashboard a una dashboard CloudWatch esistente, puoi creare un'etichetta di intestazione in modo che i parametri vengano visualizzati in una sezione separata della dashboard CloudWatch.

Puoi visualizzare la dashboard dei parametri esportata anche nella console CloudWatch. Se aggiungi nuovi parametri a una dashboard dei parametri di Approfondimenti sulle prestazioni dopo averla esportata, è necessario esportare nuovamente la dashboard per visualizzare i nuovi parametri nella console CloudWatch.

Puoi anche selezionare un widget di parametro nella dashboard Approfondimenti sulle prestazioni e visualizzare i dati dei parametri nella console CloudWatch.

Per ulteriori informazioni sulla visualizzazione dei parametri nella console CloudWatch, consulta Visualizzazione dei parametri del cluster di database nella console CloudWatch e in AWS CLI.

Nelle sezioni seguenti, esportare le metriche di Approfondimenti sulle prestazioni in CloudWatch come nuova dashboard o in una dashboard esistente e visualizzare le metriche di Performance Insights in CloudWatch.

Argomenti

- Esportazione dei parametri di Approfondimenti sulle prestazioni come nuova dashboard in CloudWatch
- Aggiunta di parametri di Approfondimenti sulle prestazioni a una dashboard CloudWatch esistente
- Visualizzazione di un widget di parametri Approfondimenti sulle prestazioni in CloudWatch

Esportazione dei parametri di Approfondimenti sulle prestazioni come nuova dashboard in CloudWatch

Scegli una dashboard dei parametri preconfigurata o personalizzata dalla dashboard di Approfondimenti sulle prestazioni ed esportala come nuova dashboard in CloudWatch. Puoi visualizzare la dashboard esportata nella console CloudWatch.

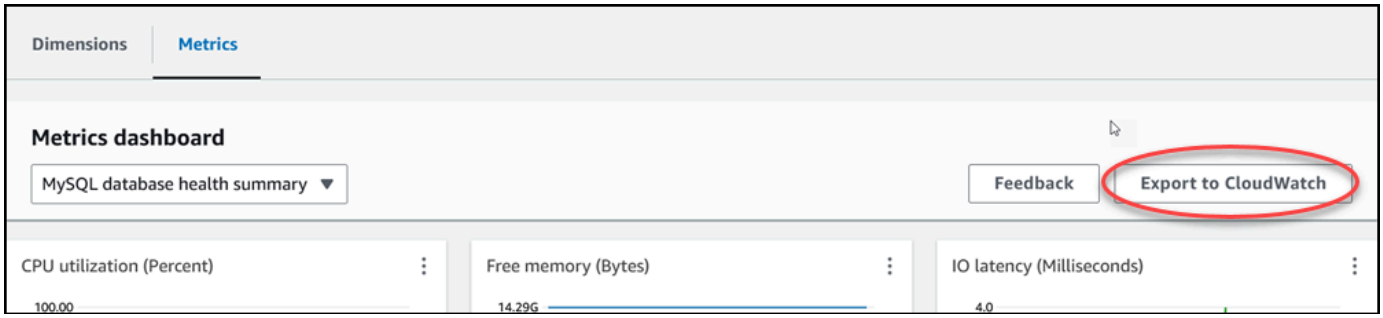
Per esportare una dashboard dei parametri di Approfondimenti sulle prestazioni come nuova dashboard in CloudWatch

1. Apri la console Amazon RDS all'indirizzo https://console.aws.amazon.com/rds/.
2. Nel pannello di navigazione scegli Performance Insights.
3. Scegli un'istanza database.

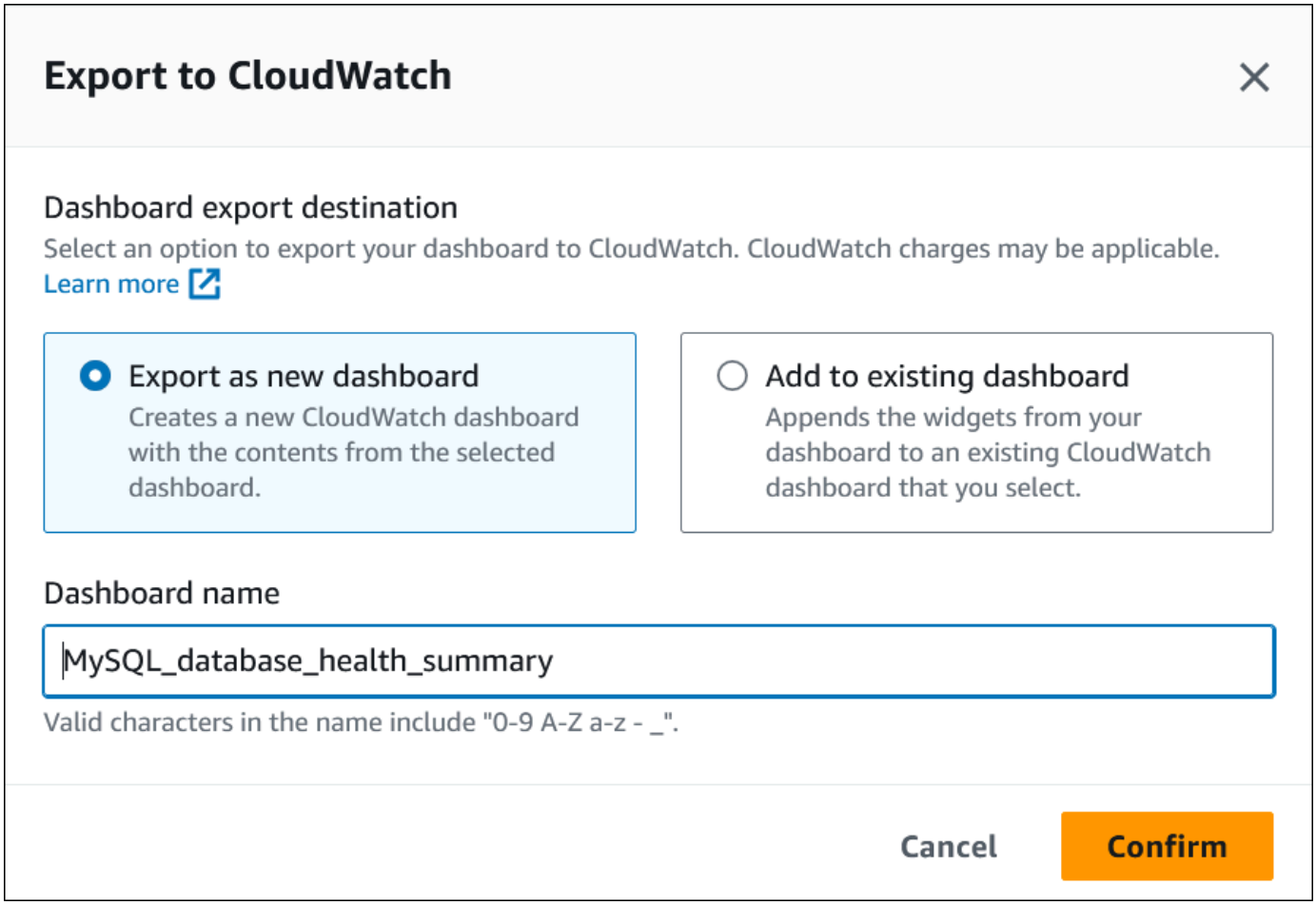
   Viene visualizzato il pannello di controllo di Approfondimenti sulle prestazioni per l'istanza database.
4. Scorri verso il basso e scegli Parametri.

   Per impostazione predefinita, viene visualizzata la dashboard preconfigurata con i parametri di Approfondimenti sulle prestazioni.
5. Scegli una dashboard preconfigurata o personalizzata, quindi seleziona Esporta in CloudWatch.

   Viene visualizzata la finestra Esporta in CloudWatch.

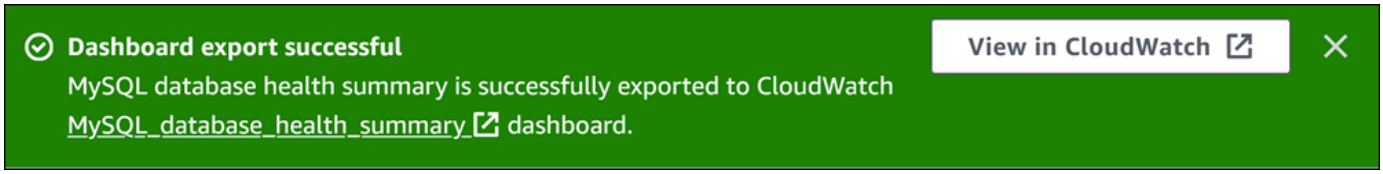


6. Scegli Esporta come nuova dashboard.



7. Immetti un nome per la nuova dashboard nel campo Nome della dashboard e scegli Conferma.

   Un banner mostra un messaggio al completamento dell'esportazione della dashboard.



8. Scegli il collegamento o Visualizza in CloudWatch nel banner per visualizzare la dashboard dei parametri nella console CloudWatch.

Aggiunta di parametri di Approfondimenti sulle prestazioni a una dashboard CloudWatch esistente

Aggiungi una dashboard dei parametri preconfigurata o personalizzata a una dashboard CloudWatch esistente. Puoi aggiungere un'etichetta alla dashboard dei parametri da visualizzare in una sezione separata della dashboard CloudWatch.

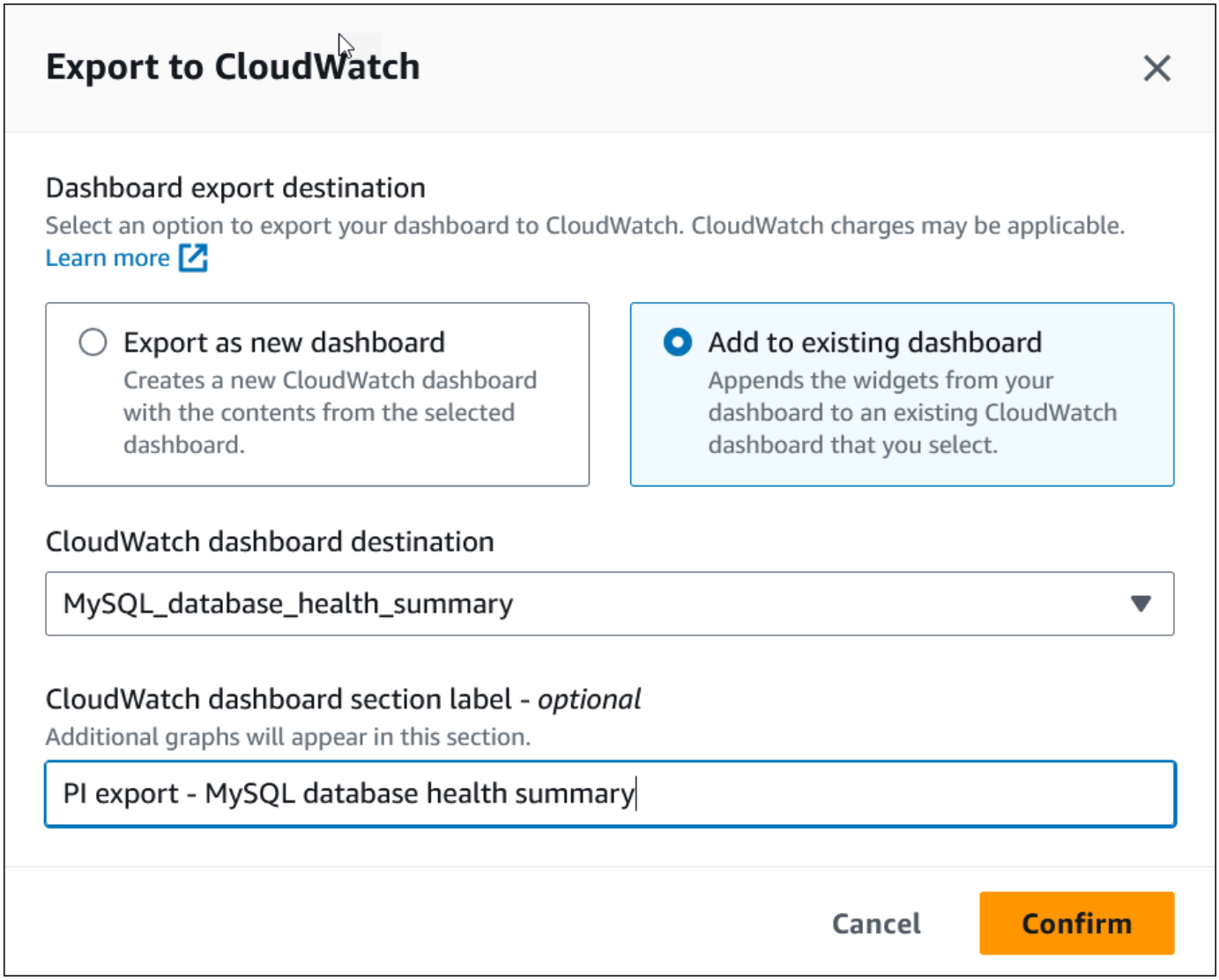
Per esportare i parametri in una dashboard CloudWatch esistente

1. Apri la console Amazon RDS all'indirizzo https://console.aws.amazon.com/rds/.
2. Nel pannello di navigazione scegli Performance Insights.
3. Scegli un'istanza database.

   Viene visualizzato il pannello di controllo di Approfondimenti sulle prestazioni per l'istanza database.
4. Scorri verso il basso e scegli Parametri.

   Per impostazione predefinita, viene visualizzata la dashboard preconfigurata con i parametri di Approfondimenti sulle prestazioni.
5. Scegli la dashboard preconfigurata o personalizzata, quindi seleziona Esporta in CloudWatch.

   Viene visualizzata la finestra Esporta in CloudWatch.
6. Scegli Aggiungi alla dashboard esistente.



7. Specifica la destinazione e l'etichetta della dashboard, quindi scegli Conferma.

   - Destinazione della dashboard CloudWatch: scegli una dashboard CloudWatch esistente.
   - Etichetta della sezione dashboard CloudWatch (facoltativa): immetti un nome per i parametri di Approfondimenti sulle prestazioni da visualizzare in questa sezione della dashboard CloudWatch.

   Un banner mostra un messaggio al completamento dell'esportazione della dashboard.

8. Scegli il collegamento o Visualizza in CloudWatch nel banner per visualizzare la dashboard dei parametri nella console CloudWatch.

Visualizzazione di un widget di parametri Approfondimenti sulle prestazioni in CloudWatch

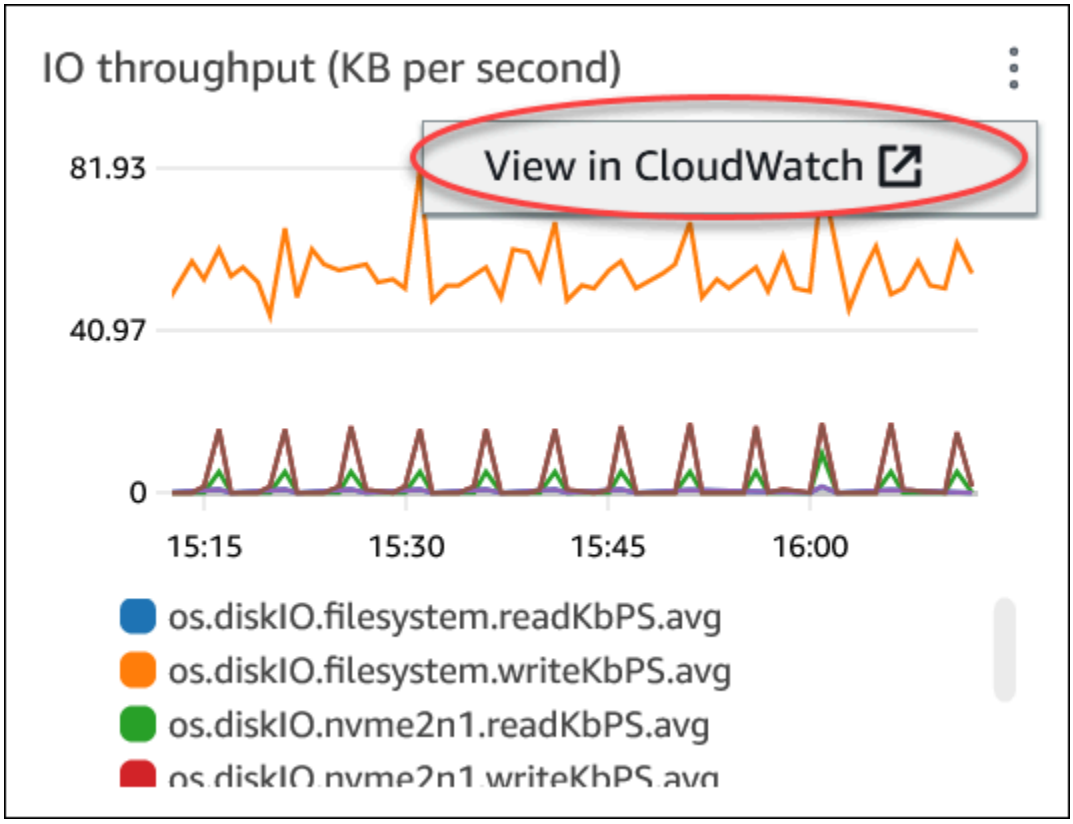
Seleziona un widget di parametri Approfondimenti sulle prestazioni nella dashboard di Approfondimenti sulle prestazioni di Amazon RDS e visualizza i dati dei parametri nella console CloudWatch.

Per esportare un widget di parametro e visualizzare i dati dei parametri nella console CloudWatch

1. Apri la console Amazon RDS all'indirizzo https://console.aws.amazon.com/rds/.
2. Nel pannello di navigazione scegli Performance Insights.
3. Scegli un'istanza database.

   Viene visualizzato il pannello di controllo di Approfondimenti sulle prestazioni per l'istanza database.
4. Scorri verso il basso fino a Parametri.

   Per impostazione predefinita, viene visualizzata la dashboard preconfigurata con i parametri di Approfondimenti sulle prestazioni.
5. Scegli un widget di parametro, quindi nel menu scegli Visualizza in CloudWatch.



I dati dei parametri vengono visualizzati nella console CloudWatch.

Creazione di allarmi CloudWatch per il monitoraggio di Amazon Aurora

Puoi creare un avviso CloudWatch che invia un messaggio Amazon SNS quando l'avviso cambia stato. Un allarme monitora un singolo parametro per un periodo di tempo specificato. L'allarme può anche eseguire una o più operazioni basate sul valore del parametro relativo a una soglia prestabilita per un certo numero di periodi. L'operazione corrisponde all'invio di una notifica a un argomento Amazon SNS o a una policy Amazon EC2 Auto Scaling.

Gli allarmi richiamano operazioni solo per le modifiche di stato prolungate. Gli allarmi CloudWatch non richiamano le operazioni semplicemente perché si trovano in uno stato particolare. È necessario che lo stato cambi e rimanga costante per un periodo specificato

> **Note**
>
> Per Aurora, utilizza `WRITER` o utilizza i parametri del ruolo `READER` per impostare gli avvisi anziché affidarti ai parametri per istanze DB specifiche. I ruoli di istanza database Aurora possono modificare i ruoli nel tempo. Questi parametri basati sui ruoli sono disponibili nella console CloudWatch.
> L'Auto Scaling di Aurora imposta automaticamente gli avvisi in base ai parametri del ruolo `READER`. Per ulteriori informazioni sull'Auto Scaling Aurora, consultare Dimensionamento automatico di Amazon Aurora con le repliche Aurora.

Puoi utilizzare la funzione matematica composta da parametri DB_PERF_INSIGHTS nella console CloudWatch per eseguire query su Amazon RDS per i parametri contatore di Performance Insights. La funzione DB_PERF_INSIGHTS include anche la metrica DBLoad a intervalli inferiori al minuto. Puoi impostare gli allarmi CloudWatch sui questi parametri.

Per maggiori dettagli su come creare un allarme, consulta Creazione di un allarme sui parametri contatore di Performance Insights da un database AWS.

Impostazione di un allarme mediante AWS CLI

- Chiamare `put-metric-alarm`. Per ulteriori informazioni, consulta il Riferimento ai comandi AWS CLI.

Per impostare un allarme mediante l'API di CloudWatch

- Chiamare `PutMetricAlarm`. Per maggiori informazioni, consulta la Documentazione di riferimento delle API di Amazon CloudWatch.

Per ulteriori informazioni sull'impostazione degli argomenti di Amazon SNS e sulla creazione degli allarmi, consulta Utilizzo degli allarmi di Amazon CloudWatch.

Monitoraggio dei database Amazon Aurora con CloudWatch Database Insights

Monitora il carico del database (Carico DB) per il parco di istanze database Amazon Aurora con Database Insights. L'opzione Carico DB misura il livello di attività della sessione nel database. Puoi utilizzare Database Insights per analizzare e risolvere i problemi di prestazioni dei database Amazon Aurora su larga scala.

Con Database Insights, è possibile visualizzare il carico del database sul parco di database e filtrare il carico in base alle attese, alle istruzioni SQL, agli host o agli utenti.

Per impostazione predefinita, RDS abilita la modalità standard di Database Insights per i database Amazon Aurora. Quando attivi la modalità avanzata di Database Insights per un cluster di database, RDS abilita Database Insights per ogni istanza database nel cluster.

Per informazioni sull'utilizzo di Database Insights nella console Amazon CloudWatch, consulta CloudWatch Database Insights nella Guida per l'utente di Amazon CloudWatch.

Prezzi

Per informazioni sui prezzi, consulta Prezzi di Amazon CloudWatch.

Argomenti

- Supporto di classe di istanza, Regione e motore di database Amazon Aurora per Database Insights
- Attivazione della modalità avanzata di Database Insights per Amazon Aurora
- Attivazione della modalità standard di Database Insights per Amazon Aurora
- Configurazione del database per monitorare le query SQL lente con Database Insights per Amazon Aurora
- Considerazioni su Database Insights per Amazon Aurora

Supporto di classe di istanza, Regione e motore di database Amazon Aurora per Database Insights

Nella tabella seguente vengono forniti i motori di database Amazon Aurora che supportano Database Insights.

Motore database Amazon Aurora	Versioni motore e regioni supportate	Restrizioni delle classi di istanza
Amazon Aurora edizione compatibile con MySQL	Per ulteriori informazioni sulla disponibilità di versioni e Regioni di Database Insights con Aurora MySQL, consulta Performance Insights con Aurora MySQL.	Database Insights ha le seguenti restrizioni della classe di motori: • db.t2: non supportato • db.t3: non supportato • db.t4g.micro e db.t4g.small: non supportate
Amazon Aurora PostgreSQL-Compatible Edition	Per ulteriori informazioni sulla disponibilità di versioni e Regioni di Database Insights con Aurora PostgreSQL, consulta Performance Insights con Aurora PostgreSQL.	Non applicabile
Aurora PostgreSQL Limitless Database	Per ulteriori informazioni sull'utilizzo di Database Insights con Aurora PostgreSQL Limitless Database, consulta Monitoraggio di Aurora PostgreSQL Limitless Database con CloudWatch Database Insights.	Non applicabile

Database Insights supporta Amazon Aurora Serverless v2.

Supporto di classe di istanza, Regione e motore di database Amazon Aurora per funzionalità Database Insights

Nella tabella seguente vengono forniti i motori di database Amazon Aurora che supportano funzionalità Database Insights.

Funzionalità	Livello di prezzi	Regioni supportate	Motori di database supportati	Classi di istanza supportate
Statistiche SQL per Performance Insights	Tutti	Tutti	Tutti	Tutti
Analisi delle prestazioni del database per un periodo di tempo	Solo livello a pagamento	Tutti	Tutti	Tutti tranne db.serverless (Aurora Serverless v2)
Visualizzazione dei consigli proattivi di Approfondimenti sulle prestazioni	Solo livello a pagamento	• Stati Uniti orientali (Ohio) • Stati Uniti orientali (Virginia settentrionale) • Stati Uniti occidentali (California settentrionale) • Stati Uniti occidentali (Oregon) • Asia Pacifico (Mumbai) • Asia Pacifico (Seoul) • Asia Pacifico (Singapore) • Asia Pacifico (Sydney)	Tutti	Tutti tranne db.serverless (Aurora Serverless v2)

Funzionalità	Livello di prezzi	Regioni supportate	Motori di database supportati	Classi di istanza supportate
		• Asia Pacifico (Tokyo) • Canada (Centrale) • Europa (Francoforte) • Europa (Irlanda) • Europe (London) • Europe (Paris) • Europa (Stoccolma) • Sud America (San Paolo)		

Supporto di Regione Amazon Aurora per Database Insights

Aurora supporta Database Insights nelle seguenti Regioni AWS.

- Stati Uniti orientali (Virginia settentrionale)
- Stati Uniti orientali (Ohio)
- Stati Uniti occidentali (California settentrionale)
- Stati Uniti occidentali (Oregon)
- Africa (Cape Town)
- Asia Pacifico (Hong Kong)
- Asia Pacific (Hyderabad)
- Asia Pacifico (Giacarta)
- Asia Pacifico (Malesia)

- Asia Pacifico (Melbourne)
- Asia Pacifico (Mumbai)
- Asia Pacifico (Osaka-Locale)
- Asia Pacifico (Seoul)
- Asia Pacifico (Singapore)
- Asia Pacifico (Sydney)
- Asia Pacifico (Tokyo)
- Canada (Centrale)
- Canada occidentale (Calgary)
- Europa (Francoforte)
- Europa (Irlanda)
- Europe (London)
- Europa (Milano)
- Europe (Paris)
- Europa (Spagna)
- Europa (Stoccolma)
- Europa (Zurigo)
- Israele (Tel Aviv)
- Medio Oriente (Bahrein)
- Medio Oriente (Emirati Arabi Uniti)
- Sud America (San Paolo)
- AWS GovCloud (Stati Uniti orientali)
- AWS GovCloud (Stati Uniti occidentali)

Attivazione della modalità avanzata di Database Insights per Amazon Aurora

Utilizza le seguenti procedure per attivare la modalità avanzata di Database Insights per Amazon Aurora.

Attivazione della modalità avanzata di Database Insights durante la creazione di un cluster di database

Attiva la modalità avanzata di Database Insights durante la creazione di un database per Amazon Aurora.

Console

Nella console è possibile attivare la modalità avanzata di Database Insights quando si crea un cluster di database. Le impostazioni per Database Insights si applicano a tutte le istanze database nel cluster di database.

Per attivare la modalità avanzata di Database Insights quando si crea utilizzando la console

1. Accedi alla Console di gestione AWS e apri la console Amazon RDS all'indirizzo https://console.aws.amazon.com/rds/.
2. Scegli Database.
3. Scegliere Crea database.
4. Nella sezione Database Insights, seleziona Modalità avanzata. Quindi, scegli le seguenti opzioni:
   - Conservazione – Quantità di tempo per cui conservare i dati di Performance Insights. Il periodo di conservazione deve essere di 15 mesi per la modalità avanzata di Database Insights.
   - AWS KMS key: specificare la chiave KMS. Performance Insights crittografa tutti i dati potenzialmente sensibili con la chiave KMS. I dati vengono crittografati mentre sono in transito o inattivi. Per ulteriori informazioni, consulta Crittografia delle risorse Amazon Aurora.
5. Scegliere Crea database.

AWS CLI

Per attivare la modalità avanzata di Database Insights durante la creazione di un cluster di database, chiama il comando AWS CLI create-db-cluster e fornisci i seguenti valori:

- `--database-insights-mode advanced` per attivare la modalità avanzata di Database Insights.
- `--engine`: il motore del database per il cluster di database.

- `--db-cluster-identifier`: l'identificatore per il cluster di database .
- `--enable-performance-insights` per attivare Approfondimenti sulle prestazioni per Database Insights.
- `--performance-insights-retention-period`: il periodo di conservazione dei dati per . Per attivare Database Insights, il periodo di conservazione deve essere di almeno 465 giorni.

L'esempio seguente abilita la modalità avanzata di Database Insights durante la creazione di un cluster di database.

Per Linux, macOS o Unix:

```
aws rds create-db-cluster \
    --database-insights-mode advanced \
    --engine aurora-postgresql \
    --db-cluster-identifier sample-db-identifier \
    --enable-performance-insights \
    --performance-insights-retention-period 465
```

Per Windows:

```
aws rds create-db-cluster ^
    --database-insights-mode advanced ^
    --engine aurora-postgresql ^
    --db-cluster-identifier sample-db-identifier ^
    --enable-performance-insights ^
    --performance-insights-retention-period 465
```

RDS API

Per attivare la modalità avanzata di Database Insights quando crei , specifica i seguenti parametri per l'operazione API CreateDBCluster di Amazon RDS.

- Da `DatabaseInsightsMode` a `advanced`
- Da `EnablePerformanceInsights` a `True`
- `PerformanceInsightsRetentionPeriod` ad almeno 465 giorni

Attivazione della modalità avanzata di Database Insights durante la modifica di un cluster di database

Attiva Database Insights durante la modifica di un database per Amazon Aurora. La modifica di un cluster di database per abilitare la modalità avanzata di Database Insights non causa tempi di inattività.

> **Note**
>
> Per abilitare Database Insights, ogni istanza database in un cluster di database deve avere le stesse impostazioni di Approfondimenti sulle prestazioni e Monitoraggio avanzato.

Console

Nella console è possibile attivare la modalità avanzata di Database Insights quando si modifica un cluster di database. Le impostazioni per Database Insights si applicano a tutte le istanze database nel cluster di database.

Per attivare la modalità avanzata di Database Insights quando si modifica utilizzando la console

1. Accedi alla Console di gestione AWS e apri la console Amazon RDS all'indirizzo https://console.aws.amazon.com/rds/.
2. Scegli Database.
3. Scegli e scegli Modifica.
4. Nella sezione Database Insights, seleziona Modalità avanzata. Quindi, scegli le seguenti opzioni:
   - Conservazione – Quantità di tempo per cui conservare i dati di Performance Insights. Il periodo di conservazione deve essere di 15 mesi per la modalità avanzata di Database Insights.
   - AWS KMS key: specificare la chiave KMS. Performance Insights crittografa tutti i dati potenzialmente sensibili con la chiave KMS. I dati vengono crittografati mentre sono in transito o inattivi. Per ulteriori informazioni, consulta Crittografia delle risorse Amazon Aurora.
5. Scegli Continua.

6. In Scheduling of Modifications (Pianificazione delle modifiche), scegli Apply immediately (Applica immediatamente). Se scegli Applica durante la prossima finestra di manutenzione pianificata, il database ignora questa impostazione e attiva immediatamente la modalità avanzata di Database Insights.
7. Scegli Modifica cluster.

AWS CLI

Per attivare la modalità avanzata di Database Insights durante la modifica di un cluster di database, chiama il comando AWS CLI modify-db-cluster e fornisci i seguenti valori:

- `--database-insights-mode advanced` per attivare la modalità avanzata di Database Insights.
- `--db-cluster-identifier`: l'identificatore per il cluster di database .
- `--enable-performance-insights` per attivare Approfondimenti sulle prestazioni per Database Insights.
- `--performance-insights-retention-period`: il periodo di conservazione dei dati per il cluster di database. Per attivare la modalità avanzata di Database Insights, il periodo di conservazione deve essere di almeno 465 giorni.

L'esempio seguente abilita la modalità avanzata di Database Insights quando si modifica un cluster di database.

Per Linux, macOS o Unix:

```
aws rds modify-db-cluster \
    --database-insights-mode advanced \
    --db-cluster-identifier sample-db-identifier \
    --enable-performance-insights \
    --performance-insights-retention-period 465
```

Per Windows:

```
aws rds modify-db-cluster ^
    --database-insights-mode advanced ^
    --db-cluster-identifier sample-db-identifier ^
    --enable-performance-insights ^
```

```
    --performance-insights-retention-period 465
```

RDS API

Per attivare la modalità avanzata di Database Insights quando modifichi un cluster di database, specifica i seguenti parametri per l'operazione API ModifyDBCluster di Amazon RDS.

- Da `DatabaseInsightsMode` a `advanced`
- Da `EnablePerformanceInsights` a `True`
- `PerformanceInsightsRetentionPeriod` ad almeno 465 giorni

Attivazione della modalità standard di Database Insights per Amazon Aurora

Utilizza le seguenti procedure per attivare la modalità standard di Database Insights per Amazon Aurora.

Attivazione della modalità standard di Database Insights durante la creazione di un cluster di database

Attiva la modalità standard di Database Insights durante la creazione di un database per Amazon Aurora.

Console

Nella console è possibile attivare la modalità standard di Database Insights quando si crea un cluster di database. Le impostazioni per Database Insights si applicano a tutte le istanze database nel cluster di database.

Per attivare la modalità standard di Database Insights quando si crea utilizzando la console

1. Accedi alla Console di gestione AWS e apri la console Amazon RDS all'indirizzo https://console.aws.amazon.com/rds/.
2. Scegli Database.
3. Scegliere Crea database.
4. Nella sezione Database Insights, seleziona Modalità standard, Quindi scegli tra le seguenti opzioni per attivare o disattivare Approfondimenti sulle prestazioni:

- Per disattivare Approfondimenti sulle prestazioni, deseleziona Abilita Performance Insights.
- Per attivare Approfondimenti sulle prestazioni, seleziona Abilita Performance Insights. Per configurare Approfondimenti sulle prestazioni, specifica le seguenti opzioni:
  - Conservazione – Quantità di tempo per cui conservare i dati di Performance Insights. Il periodo di conservazione deve essere di almeno 7 giorni.
  - AWS KMS key: specificare la chiave KMS. Performance Insights crittografa tutti i dati potenzialmente sensibili con la chiave KMS. I dati vengono crittografati mentre sono in transito o inattivi. Per ulteriori informazioni, consulta Crittografia delle risorse Amazon Aurora.

5. Scegliere Crea database.

AWS CLI

Per attivare la modalità standard di Database Insights durante la creazione di un cluster di database, chiama il comando AWS CLI create-db-cluster e fornisci i seguenti valori:

- `--database-insights-mode standard` per attivare la modalità standard di Database Insights.
- `--engine`: il motore del database per il cluster di database.
- `--db-cluster-identifier`: l'identificatore per il cluster di database .
- `--enable-performance-insights` o `--no-enable-performance-insights` per attivare o disattivare Approfondimenti sulle prestazioni. Se si specifica `--enable-performance-insights`, è necessario specificare anche `--performance-insights-retention-period`. Il periodo di conservazione dei dati per il cluster di database. Il periodo di conservazione deve essere di almeno 7 giorni.

L'esempio seguente abilita la modalità standard di Database Insights e Approfondimenti sulle prestazioni durante la creazione di un cluster di database.

Per Linux, macOS o Unix:

```
aws rds create-db-cluster \
    --database-insights-mode standard \
    --engine aurora-postgresql \
    --db-cluster-identifier sample-db-identifier \
```

```
    --enable-performance-insights \
    --performance-insights-retention-period 7
```

Per Windows:

```
aws rds create-db-cluster ^
    --database-insights-mode standard ^
    --engine aurora-postgresql ^
    --db-cluster-identifier sample-db-identifier ^
    --enable-performance-insights ^
    --performance-insights-retention-period 7
```

L'esempio seguente abilita la modalità standard di Database Insights e disabilita Approfondimenti sulle prestazioni durante la creazione di un cluster di database.

Per Linux, macOS o Unix:

```
aws rds create-db-cluster \
    --database-insights-mode standard \
    --engine aurora-postgresql \
    --db-cluster-identifier sample-db-identifier \
    --no-enable-performance-insights
```

Per Windows:

```
aws rds create-db-cluster ^
    --database-insights-mode standard ^
    --engine aurora-postgresql ^
    --db-cluster-identifier sample-db-identifier ^
    --no-enable-performance-insights
```

RDS API

Per attivare la modalità avanzata di Database Insights quando crei un cluster di database, specifica i seguenti parametri per l'operazione API CreateDBCluster di Amazon RDS.

- Da `DatabaseInsightsMode` a `standard`
- `EnablePerformanceInsights` su `True` o `False` Se imposti `EnablePerformanceInsights` su `True`, devi impostare `PerformanceInsightsRetentionPeriod` su almeno 7 giorni.

Attivazione della modalità standard di Database Insights durante la modifica di un cluster di database

Attiva la modalità standard di Database Insights durante la modifica di un database per Amazon Aurora. La modifica di un cluster di database per abilitare la modalità standard di Database Insights non causa tempi di inattività.

> **Note**
>
> Per abilitare Database Insights, ogni istanza database in un cluster di database deve avere le stesse impostazioni di Approfondimenti sulle prestazioni e Monitoraggio avanzato.

Console

Nella console è possibile attivare la modalità standard di Database Insights quando si modifica un cluster di database. Le impostazioni per Database Insights si applicano a tutte le istanze database nel cluster di database.

Per attivare la modalità standard di Database Insights quando si modifica utilizzando la console

1. Accedi alla Console di gestione AWS e apri la console Amazon RDS all'indirizzo https://console.aws.amazon.com/rds/.
2. Scegli Database.
3. Scegli e scegli Modifica.
4. Nella sezione Database Insights, seleziona Modalità standard, Seleziona quindi una delle opzioni seguenti:
   - Per disattivare Approfondimenti sulle prestazioni, deseleziona Abilita Performance Insights.
   - Per attivare Approfondimenti sulle prestazioni, seleziona Abilita Performance Insights. Per configurare Approfondimenti sulle prestazioni, specifica le seguenti opzioni:
     - Conservazione – Quantità di tempo per cui conservare i dati di Performance Insights. Il periodo di conservazione deve essere di almeno 7 giorni.
     - AWS KMS key: specificare la chiave KMS. Performance Insights crittografa tutti i dati potenzialmente sensibili con la chiave KMS. I dati vengono crittografati mentre sono in transito o inattivi. Per ulteriori informazioni, consulta Crittografia delle risorse Amazon Aurora.

5. Scegli Continua.
6. In Scheduling of Modifications (Pianificazione delle modifiche), scegli Apply immediately (Applica immediatamente). Se scegli Applica durante la prossima finestra di manutenzione pianificata, il database ignora questa impostazione e attiva immediatamente la modalità standard di Database Insights.
7. Scegli Modifica cluster.

AWS CLI

Per attivare la modalità standard di Database Insights durante la modifica di un cluster di database, chiama il comando AWS CLI modify-db-cluster e fornisci i seguenti valori:

- `--database-insights-mode standard` per attivare la modalità standard di Database Insights.
- `--db-cluster-identifier`: l'identificatore per il cluster di database .
- `--enable-performance-insights` o `--no-enable-performance-insights` per attivare o disattivare Approfondimenti sulle prestazioni. Se si specifica `--enable-performance-insights`, è necessario specificare anche `--performance-insights-retention-period`. Il periodo di conservazione dei dati per . Il periodo di conservazione deve essere di almeno 7 giorni.

L'esempio seguente abilita la modalità standard di Database Insights e abilita Approfondimenti sulle prestazioni durante la modifica di un cluster di database.

Per Linux, macOS o Unix:

```
aws rds modify-db-cluster \
    --database-insights-mode standard \
    --db-cluster-identifier sample-db-identifier \
    --enable-performance-insights \
    --performance-insights-retention-period 7
```

Per Windows:

```
aws rds modify-db-cluster ^
    --database-insights-mode standard ^
    --db-cluster-identifier sample-db-identifier ^
    --enable-performance-insights ^
```

```
    --performance-insights-retention-period 7
```

L'esempio seguente abilita la modalità standard di Database Insights e disabilita Approfondimenti sulle prestazioni durante la modifica di un cluster di database.

Per Linux, macOS o Unix:

```
aws rds modify-db-cluster \
    --database-insights-mode standard \
    --db-cluster-identifier sample-db-identifier \
    --no-enable-performance-insights
```

Per Windows:

```
aws rds modify-db-cluster ^
    --database-insights-mode standard ^
    --db-cluster-identifier sample-db-identifier ^
    --no-enable-performance-insights
```

RDS API

Per attivare la modalità standard di Database Insights quando modifichi un cluster di database, specifica i seguenti parametri per l'operazione API ModifyDBCluster di Amazon RDS.

- Da `DatabaseInsightsMode` a `standard`
- `EnablePerformanceInsights` su `True` o `False` Se imposti `EnablePerformanceInsights` su `True`, devi impostare `PerformanceInsightsRetentionPeriod` su almeno 7 giorni.

Configurazione del database per monitorare le query SQL lente con Database Insights per Amazon Aurora

Per monitorare le query SQL lente per il database, è possibile utilizzare la sezione Query SQL lente nella dashboard Database Insights. Prima di configurare il database per monitorare le query SQL lente, la sezione Query SQL lente è vuota.

Per ulteriori informazioni sul monitoraggio delle query SQL lente nella dashboard Database Insights, consulta Viewing the Database Instance Dashboard for CloudWatch Database Insights nella Guida per l'utente di Amazon CloudWatch.

Per configurare il database per monitorare le query SQL lente con Database Insights, completare i seguenti passaggi:

1. Abilita l'esportazione dei log in CloudWatch Logs.
2. Crea o modifica il gruppo di parametri del cluster di database per il cluster di database.

Per informazioni sulla configurazione delle esportazioni di log, consulta Pubblicazione di log di database su Amazon CloudWatch Logs nella Guida per l'utente di Amazon Aurora.

Per creare o modificare il gruppo di parametri del cluster di database, consulta i seguenti argomenti.

- Creazione di un gruppo di parametri del cluster di database in Amazon Aurora
- Modifica dei parametri in un gruppo di parametri del cluster di database in Amazon Aurora

Amazon Aurora MySQL

Per configurare il cluster di database MySQL Amazon Aurora per monitorare le query SQL lente, è possibile utilizzare la seguente combinazione di parametri come esempio:

- `slow_query_log` – imposta il valore su `1`
- `long_query_time` – imposta il valore su `1.0`
- `log_output` – imposta il valore su `FILE`

Questa è una possibile configurazione. Per una guida completa ai parametri di log delle query lente MySQL e alle opzioni di configurazione aggiuntive, consulta la documentazione MySQL per il log delle query lente.

Amazon Aurora PostgreSQL

Per configurare il cluster di database PostgreSQL Amazon Aurora per monitorare le query SQL lente, è possibile utilizzare la seguente combinazione di parametri come esempio. L'impostazione di questi parametri potrebbe ridurre le prestazioni del cluster di database.

- `log_min_duration_statement` – imposta il valore su `1000`
- `log_statement` – imposta il valore su `none`
- `log_destination` – imposta il valore su `stderr`

Questa è una possibile configurazione. Per una guida completa ai parametri di log di PostgreSQL e alle opzioni di configurazione aggiuntive, consulta la documentazione PostgreSQL per la configurazione di log.

> **Note**
>
> Per Aurora MySQL, è possibile configurare il parametro `long_query_time` con una granularità di 1 microsecondo. Ad esempio, è possibile impostare questo parametro su `0.000001`. A seconda della quantità di query sull'istanza database, il valore del parametro `long_query_time` può ridurre le prestazioni. Iniziare con il valore `1.0` e modificarlo in base al carico di lavoro. Quando si imposta questo parametro su `0`, Database Insights registra nel log tutte le query.

Per informazioni sui log Aurora MySQL e Aurora PostgreSQL, consulta quanto segue.

- File di log del database Aurora MySQL
- File di log del database Aurora PostgreSQL

Considerazioni su Database Insights per Amazon Aurora

Di seguito sono riportate alcune considerazioni su Database Insights per Amazon Aurora.

- Non è possibile gestire Database Insights per un'istanza database in un cluster di database.
- Per abilitare la modalità avanzata di Database Insights, è necessario abilitare Approfondimenti sulle prestazioni e impostare il periodo di conservazione di Approfondimenti sulle prestazioni su almeno 465 giorni (15 mesi). Oltre al costo di Database Insights, non sono previsti costi aggiuntivi per impostare il periodo di conservazione di Approfondimenti sulle prestazioni su 15 mesi. Per informazioni sui prezzi di Database Insights, consulta Prezzi di Amazon CloudWatch.
- Per abilitare Database Insights, ogni istanza database in un cluster di database deve avere le stesse impostazioni di Approfondimenti sulle prestazioni e Monitoraggio avanzato.
- La modifica di un cluster di database per abilitare la modalità di Database Insights non causa tempi di inattività.

Monitoraggio del carico DB con Performance Insights su Amazon Aurora

Important

AWS ha annunciato la end-of-life data di Performance Insights: 30 giugno 2026. Dopo questa data, Amazon RDS non supporterà più l'esperienza della console Approfondimenti sulle prestazioni, i periodi di conservazione flessibili (1-24 mesi) e i relativi prezzi. L'API Approfondimenti sulle prestazioni continuerà a esistere senza modifiche ai prezzi. I costi per l'API Performance Insights verranno visualizzati nella AWS fattura insieme al costo di CloudWatch Database Insights.

Si consiglia di aggiornare tutte DB Clusters di database utilizzando il livello a pagamento di Performance Insights alla modalità Advanced di Database Insights prima del 30 giugno 2026. Per informazioni sull'aggiornamento alla modalità avanzata di Database Insights, consulta Attivazione della modalità avanzata di Database Insights per Amazon Aurora.

Se non si esegue alcuna azione, i cluster di database che utilizzano Approfondimenti sulle prestazioni, per impostazione predefinita useranno la modalità standard di Database Insights. Con la modalità standard di Database Insights, è possibile perdere l'accesso alla cronologia dei dati sulle prestazioni di oltre 7 giorni e non essere in grado di utilizzare i piani di esecuzione e le funzionalità di analisi on demand nella console Amazon RDS. Dopo il 30 giugno 2026 solo la modalità Advanced di Database Insights supporterà i piani di esecuzione e l'analisi su richiesta.

Con CloudWatch Database Insights, puoi monitorare il carico del database per la tua flotta di database e analizzare e risolvere i problemi delle prestazioni su larga scala. Per ulteriori informazioni su Database Insights, consulta Monitoraggio dei database Amazon Aurora con CloudWatch Database Insights. Per informazioni sui prezzi, consulta la pagina CloudWatch dei prezzi di Amazon.

Performance Insights si espande sulle caratteristiche di monitoraggio esistenti di Amazon Aurora per illustrare e aiutare ad analizzare le prestazioni del cluster. Con il pannello di controllo di Performance Insights, puoi visualizzare il carico del database sul cluster Amazon Aurora e filtrare il carico in base alle attese, alle istruzioni SQL, agli host o agli utenti. Per informazioni sull'uso di Performance Insights con Amazon DocumentDB, consulta Guida per gli sviluppatori di Amazon DocumentDB.

Argomenti

- Panoramica di Performance Insights su Amazon Aurora
- Attivazione e disattivazione di Approfondimenti sulle prestazioni per Aurora
- Panoramica dello schema di prestazioni per Approfondimenti sulle prestazioni su Aurora MySQL
- Configurazione delle policy di accesso per Performance Insights
- Analisi delle metriche utilizzando il pannello di controllo Performance Insights
- Visualizzazione dei consigli proattivi di Approfondimenti sulle prestazioni
- Recupero delle metriche con l'API Approfondimenti sulle prestazioni per Aurora
- Registrazione di log delle chiamate di Performance Insights mediante AWS CloudTrail
- API Approfondimenti sulle prestazioni ed endpoint VPC di interfaccia (AWS PrivateLink)

Panoramica di Performance Insights su Amazon Aurora

Important

AWS ha annunciato la end-of-life data di Performance Insights: 30 giugno 2026. Dopo questa data, Amazon RDS non supporterà più l'esperienza della console Approfondimenti sulle prestazioni, i periodi di conservazione flessibili (1-24 mesi) e i relativi prezzi. L'API Approfondimenti sulle prestazioni continuerà a esistere senza modifiche ai prezzi. I costi per l'API Performance Insights verranno visualizzati nella AWS fattura insieme al costo di CloudWatch Database Insights.

Si consiglia di aggiornare tutte DB Clusters di database utilizzando il livello a pagamento di Performance Insights alla modalità Advanced di Database Insights prima del 30 giugno 2026. Per informazioni sull'aggiornamento alla modalità avanzata di Database Insights, consulta Attivazione della modalità avanzata di Database Insights per Amazon Aurora.

Se non si esegue alcuna azione, i cluster di database che utilizzano Approfondimenti sulle prestazioni, per impostazione predefinita useranno la modalità standard di Database Insights. Con la modalità standard di Database Insights, è possibile perdere l'accesso alla cronologia dei dati sulle prestazioni di oltre 7 giorni e non essere in grado di utilizzare i piani di esecuzione e le funzionalità di analisi on demand nella console Amazon RDS. Dopo il 30 giugno 2026 solo la modalità Advanced di Database Insights supporterà i piani di esecuzione e l'analisi su richiesta.

Con CloudWatch Database Insights, puoi monitorare il carico del database per la tua flotta di database e analizzare e risolvere i problemi delle prestazioni su larga scala. Per ulteriori informazioni su Database Insights, consulta Monitoraggio dei database Amazon Aurora con

CloudWatch Database Insights. Per informazioni sui prezzi, consulta la pagina CloudWatch dei prezzi di Amazon.

Per impostazione predefinita, RDS abilita Approfondimenti sulle prestazioni nella procedura guidata di creazione della console per tutti i motori Amazon RDS. Se si attiva Approfondimenti sulle prestazioni a livello di cluster di database, RDS abilita Approfondimenti sulle prestazioni per ogni istanza database nel cluster. Se in un'istanza database sono presenti più database, Performance Insights aggrega i dati sulle prestazioni.

Puoi trovare una panoramica di Performance Insights per Amazon Aurora nel seguente video.

Uso di Performance Insights per analizzare le prestazioni di Amazon Aurora PostgreSQL

Argomenti

- Carico del database
- CPU massima
- Supporto di classe di istanza, regione e motore di database Amazon Aurora per Performance Insights
- Prezzi e conservazione dei dati per Performance Insights

Carico del database

L'opzione Carico del database (Carico DB) misura il livello di attività della sessione nel database. `DBLoad` è la metrica chiave di Approfondimenti sulle prestazioni e Approfondimenti sulle prestazioni raccoglie il carico del database ogni secondo.

Argomenti

- Sessioni attive
- Media delle sessioni attive
- Media delle esecuzioni attive
- Dimensioni

Sessioni attive

Una sessione database rappresenta il dialogo di un'applicazione con un database relazionale. Una sessione attiva è una connessione che ha inviato lavoro a un motore del database ed è in attesa di una risposta dal motore del database.

Una sessione è attiva quando è in esecuzione sulla CPU o in attesa che una risorsa diventi disponibile in modo che possa proseguire. Ad esempio, una sessione attiva potrebbe attendere la lettura di una pagina (o blocco) in memoria e quindi consumare la CPU mentre legge i dati dalla pagina.

Media delle sessioni attive

La media delle sessioni attive (AAS) è l'unità per il parametro `DBLoad` in Performance Insights. Misura quante sessioni sono attive contemporaneamente nel database.

Ogni secondo, Performance Insights esegue il campionamento del numero di sessioni che eseguono contemporaneamente una query. Per ogni sessione attiva, Performance Insights raccoglie i seguenti dati:

- Istruzione SQL
- Stato della sessione (in esecuzione sulla CPU o in attesa)
- Host
- Utente che esegue SQL

Performance Insights calcola il valore delle sessioni attive medie (AAS) dividendo il numero totale di sessioni per il numero totale di campioni per un periodo di tempo specifico. Ad esempio, nella tabella seguente vengono riportati 5 campioni consecutivi di una query in esecuzione, dove ogni campione viene acquisito a intervalli di 1 secondo.

Project N.E.M.O.	Numero di sessioni che eseguono query	AAS	Calcolo
1	2	2	2 sessioni totali / 1 campione
2	0	1	2 sessioni totali / 2 campioni

Project N.E.M.O.	Numero di sessioni che eseguono query	AAS	Calcolo
3	4	2	6 sessioni totali / 3 campioni
4	0	1.5	6 sessioni totali / 4 campioni
5	4	2	10 sessioni totali / 5 campioni

Nell'esempio precedente, il carico DB per l'intervallo di tempo è 2 AAS. Questa misurazione significa che, in media, sono state attive 2 sessioni alla volta durante il periodo in cui sono stati acquisiti i 5 campioni.

Media delle esecuzioni attive

La media delle esecuzioni attive (AAE) al secondo è correlata all'AAS. Per calcolare l'AAE, Performance Insights divide il tempo totale di esecuzione di una query per l'intervallo di tempo. Nella tabella seguente viene illustrato il calcolo AAE per la stessa query nella tabella precedente.

Tempo trascorso (sec)	Tempo di esecuzione totale (sec)	AAE	Calcolo
60	120	2	120 secondi di esecuzione/60 secondi trascorsi
120	120	1	120 secondi di esecuzione/120 secondi trascorsi
180	380	2.11	380 secondi di esecuzione/180 secondi trascorsi

Tempo trascorso (sec)	Tempo di esecuzione totale (sec)	AAE	Calcolo
240	380	1.58	380 secondi di esecuzione/240 secondi trascorsi
300	600	2	600 secondi di esecuzione/300 secondi trascorsi

Nella maggior parte dei casi, l'AAS e AAE per una query sono quasi uguali. Tuttavia, poiché gli input per i calcoli sono origini dati diverse, i calcoli spesso variano leggermente.

Dimensioni

Il parametro `db.load` è diverso dagli altri parametri di serie temporali in quanto può essere suddiviso in sottocomponenti detti dimensioni. Le dimensioni possono essere considerate come categorie "slice by" (dividi per) per le diverse caratteristiche del parametro `DBLoad`.

Quando si diagnosticano problemi di prestazioni, le dimensioni seguenti sono spesso le più utili:

Argomenti

- Eventi di attesa
- Prime istruzioni SQL

Per un elenco completo delle dimensioni per i motori Aurora, consulta Carico del database suddiviso per dimensioni.

Eventi di attesa

Un evento di attesa fa sì che un'istruzione SQL attenda che si verifichi un evento specifico prima che possa continuare l'esecuzione. Gli eventi di attesa sono una dimensione o una categoria importante per il caricamento del database perché indicano dove il lavoro è impedito.

Ogni sessione attiva è in esecuzione sulla CPU o in attesa. Ad esempio, le sessioni consumano la CPU quando cercano in memoria un buffer, eseguono un calcolo o eseguono codice procedurale. Quando le sessioni non consumano la CPU, potrebbero essere in attesa che un buffer di memoria

diventi libero, un file di dati da leggere o un registro in cui scrivere. Maggiore è il tempo in cui una sessione attende le risorse, minore è il tempo in cui viene eseguita sulla CPU.

Quando si sintonizza un database, si tenta spesso di scoprire le risorse che le sessioni sono in attesa. Ad esempio, due o tre eventi di attesa potrebbero rappresentare il 90% del carico DB. Questa misura significa che, in media, le sessioni attive trascorrono la maggior parte del tempo in attesa di un numero limitato di risorse. Se riesci a scoprire la causa di queste attese, puoi provare a fornire una soluzione.

Gli eventi di attesa variano in base al motore database:

- Per un elenco degli eventi di attesa comuni per Aurora MySQL, consulta Eventi di attesa Aurora MySQL. Per informazioni su come sintonizzarsi utilizzando questi eventi di attesa, consulta Ottimizzazione di Aurora MySQL.
- Per informazioni su tutti gli eventi di attesa MySQL, consulta Wait Event Summary Tables nella documentazione di MySQL.
- Per un elenco degli eventi di attesa comuni per Aurora PostgreSQL, consulta Eventi di attesa Amazon Aurora PostgreSQL. Per informazioni su come sintonizzarsi utilizzando questi eventi di attesa, consulta Sintonizzazione degli eventi di attesa per Aurora PostgreSQL.
- Per informazioni su tutti gli eventi di attesa PostgreSQL, consulta la pagina relativa al processo di raccolta delle statistiche e alle tabelle degli eventi di attesa nella documentazione di PostgreSQL.

Prime istruzioni SQL

Mentre gli eventi di attesa mostrano i colli di bottiglia, il primo SQL mostra quali query stanno contribuendo maggiormente al caricamento del DB. Ad esempio, molte query potrebbero essere attualmente in esecuzione nel database, ma una singola query potrebbe consumare il 99 percento del carico DB. In questo caso, il carico elevato potrebbe indicare un problema con la query.

Per impostazione predefinita, la console Performance Insights visualizza le prime query SQL che contribuiscono al caricamento del database. La console mostra anche le statistiche pertinenti per ogni istruzione. Per diagnosticare problemi di prestazioni per un'istruzione specifica, è possibile esaminarne il piano di esecuzione.

CPU massima

Nel dashboard, il grafico di caricamento del database raccoglie, aggrega e visualizza le informazioni sulla sessione. Per verificare se le sessioni attive superano la CPU massima, esaminare la loro relazione con la linea vCPU massima. Approfondimenti sulle prestazioni determina il valore vCPU

max dal numero di core vCPU (CPU virtuale) per l'istanza database. Per Aurora Serverless v2, Max vCPU (Numero massimo di vCPU) rappresenta il numero stimato di vCPU.

Un processo può essere eseguito su una vCPU alla volta. Se il numero di processi supera il numero di vCPU, i processi vengono messi in coda. Quando la coda aumenta, le prestazioni del database diminuiscono. Se il carico è spesso sopra la linea vCPU massima e lo stato di attesa primario è CPU, la CPU è sovraccarica. In questo caso, si potrebbero limitare le connessioni all'istanza, ottimizzare le eventuali query SQL con un elevato carico CPU o valutare la possibilità di una classe istanza di maggiori dimensioni. Istanze elevate e costanti di qualsiasi stato di attesa indicano che possono verificarsi colli di bottiglia o problemi di conflitto delle risorse da risolvere. Questo può valere anche se il carico database non supera il valore della riga CPU massima.

Supporto di classe di istanza, regione e motore di database Amazon Aurora per Performance Insights

Important

AWS ha annunciato la end-of-life data di Performance Insights: 30 giugno 2026. Dopo questa data, Amazon RDS non supporterà più l'esperienza della console Approfondimenti sulle prestazioni, i periodi di conservazione flessibili (1-24 mesi) e i relativi prezzi. L'API Approfondimenti sulle prestazioni continuerà a esistere senza modifiche ai prezzi. I costi per l'API Performance Insights verranno visualizzati nella AWS fattura insieme al costo di CloudWatch Database Insights.
Si consiglia di aggiornare tutte DB Clusters di database utilizzando il livello a pagamento di Performance Insights alla modalità Advanced di Database Insights prima del 30 giugno 2026. Per informazioni sull'aggiornamento alla modalità avanzata di Database Insights, consulta Attivazione della modalità avanzata di Database Insights per Amazon Aurora.
Se non si esegue alcuna azione, i cluster di database che utilizzano Approfondimenti sulle prestazioni, per impostazione predefinita useranno la modalità standard di Database Insights. Con la modalità standard di Database Insights, è possibile perdere l'accesso alla cronologia dei dati sulle prestazioni di oltre 7 giorni e non essere in grado di utilizzare i piani di esecuzione e le funzionalità di analisi on demand nella console Amazon RDS. Dopo il 30 giugno 2026 solo la modalità Advanced di Database Insights supporterà i piani di esecuzione e l'analisi su richiesta.
Con CloudWatch Database Insights, puoi monitorare il carico del database per la tua flotta di database e analizzare e risolvere i problemi delle prestazioni su larga scala. Per ulteriori informazioni su Database Insights, consulta Monitoraggio dei database Amazon Aurora con

CloudWatch Database Insights. Per informazioni sui prezzi, consulta la pagina CloudWatch dei prezzi di Amazon.

Nella tabella seguente vengono forniti i motori di database Amazon Aurora che supportano Performance Insights.

Motore database Amazon Aurora	Versioni motore e regioni supportate	Restrizioni delle classi di istanza
Amazon Aurora edizione compatibile con MySQL	Per ulteriori informazioni sulla disponibilità di versioni e Regioni di Approfondimenti sulle prestazioni con Aurora MySQL, consulta Performance Insights con Aurora MySQL.	Performance Insights ha le seguenti restrizioni della classe di motori: • db.t2: non supportato • db.t3: non supportato • db.t4g.micro e db.t4g.small: non supportate
Amazon Aurora PostgreSQL-Compatible Edition	Per ulteriori informazioni sulla disponibilità di versioni e Regioni di Performance Insights con Aurora PostgreSQL, consulta Performance Insights con Aurora PostgreSQL.	N/D

Supporto di classe di istanza, regione e motore di database Amazon Aurora per funzionalità Performance Insights

Nella tabella seguente vengono forniti i motori di database Amazon Aurora che supportano funzionalità Performance Insights.

Funzionalità	Livello di prezzi	Regioni supportate	Motori di database supportati	Classi di istanza supportate
Statistiche SQL per Performance Insights	Tutti	Tutti	Tutti	Tutti
Analisi delle prestazioni del database per un periodo di tempo	Solo livello a pagamento	Tutti	Tutti	Tutti tranne db.serverless (Aurora Serverless v2)
Visualizzazione dei consigli proattivi di Approfondimenti sulle prestazioni	Solo livello a pagamento	• Stati Uniti orientali (Ohio) • Stati Uniti orientali (Virginia settentrionale) • Stati Uniti occidentali (California settentrionale) • Stati Uniti occidentali (Oregon) • Asia Pacifico (Mumbai) • Asia Pacifico (Seoul) • Asia Pacifico (Singapore) • Asia Pacifico (Sydney)	Tutti	Tutti tranne db.serverless (Aurora Serverless v2)

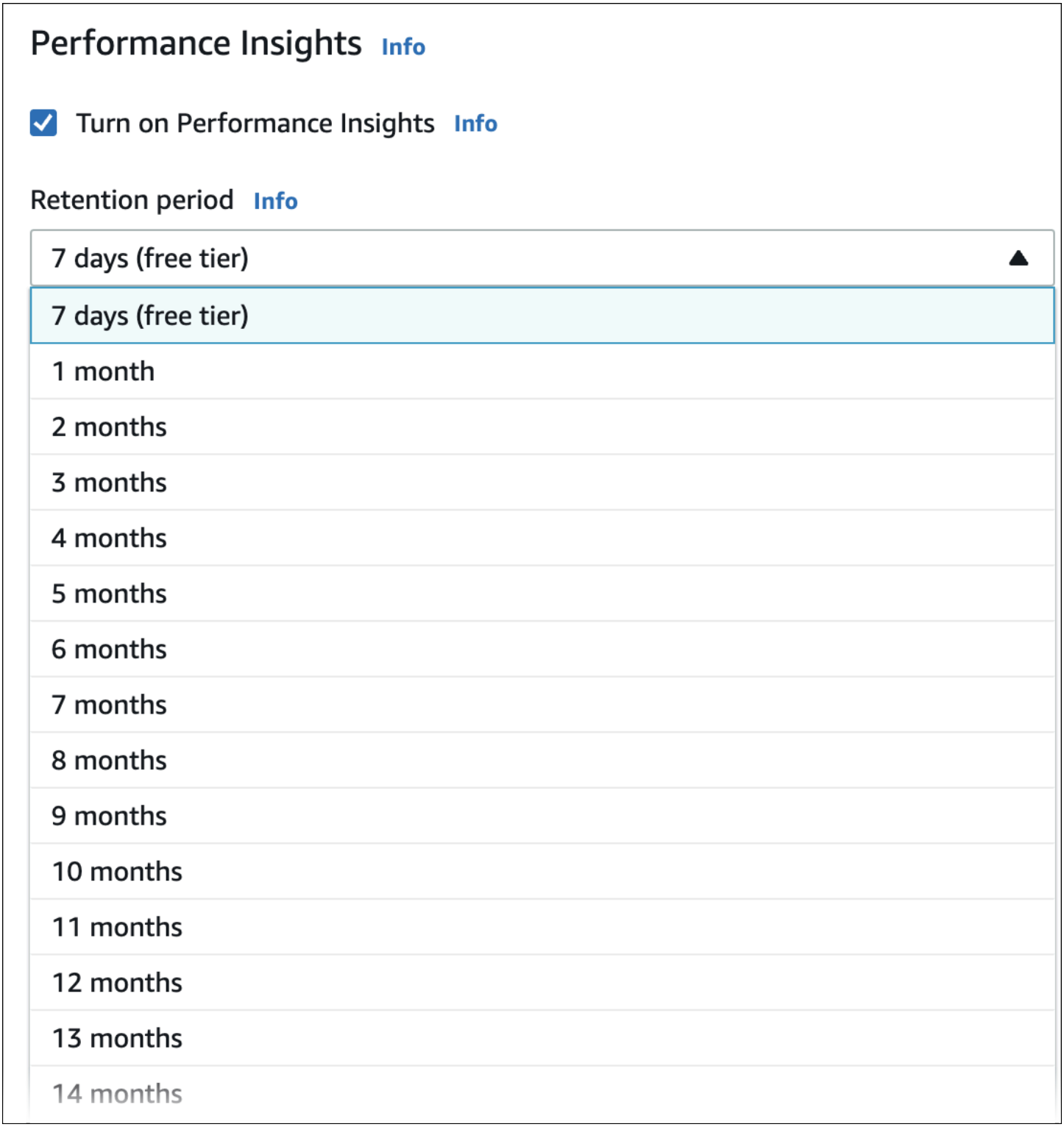
Funzionalità	Livello di prezzi	Regioni supportate	Motori di database supportati	Classi di istanza supportate
		• Asia Pacifico (Tokyo) • Canada (Centrale) • Europa (Francoforte) • Europa (Irlanda) • Europe (London) • Europe (Paris) • Europa (Stoccolma) • Sud America (San Paolo)		

Prezzi e conservazione dei dati per Performance Insights

Per impostazione predefinita, Approfondimenti sulle prestazioni include 7 giorni di cronologia dei dati sulle prestazioni e 1 milione di richieste API al mese. Puoi anche acquistare periodi di conservazione più lunghi. Per informazioni sui prezzi, consulta Prezzi di Performance Insights.

Nella console RDS, puoi scegliere uno dei seguenti periodi di conservazione per i dati di Performance Insights:

- Default (7 giorni)
- ***n*** mesi, dove ***n*** è un numero compreso tra 1 e 24



Per informazioni su come impostare un periodo di conservazione utilizzando la AWS CLI, consulta Attivazione e disattivazione di Approfondimenti sulle prestazioni per Aurora.

> **Note**
>
> L'arresto di un cluster di database con Approfondimenti sulle prestazioni abilitato non influisce sulla conservazione dei dati. Mentre un cluster di database viene arrestato, Approfondimenti sulle prestazioni non raccoglie dati.

Attivazione e disattivazione di Approfondimenti sulle prestazioni per Aurora

> **Important**
>
> AWS ha annunciato la end-of-life data di Performance Insights: 30 giugno 2026. Dopo questa data, Amazon RDS non supporterà più l'esperienza della console Approfondimenti sulle prestazioni, i periodi di conservazione flessibili (1-24 mesi) e i relativi prezzi. L'API Approfondimenti sulle prestazioni continuerà a esistere senza modifiche ai prezzi. I costi per l'API Performance Insights verranno visualizzati nella AWS fattura insieme al costo di CloudWatch Database Insights.
> Si consiglia di aggiornare tutte DB Clusters di database utilizzando il livello a pagamento di Performance Insights alla modalità Advanced di Database Insights prima del 30 giugno 2026. Per informazioni sull'aggiornamento alla modalità avanzata di Database Insights, consulta Attivazione della modalità avanzata di Database Insights per Amazon Aurora.
> Se non si esegue alcuna azione, i cluster di database che utilizzano Approfondimenti sulle prestazioni, per impostazione predefinita useranno la modalità standard di Database Insights. Con la modalità standard di Database Insights, è possibile perdere l'accesso alla cronologia dei dati sulle prestazioni di oltre 7 giorni e non essere in grado di utilizzare i piani di esecuzione e le funzionalità di analisi on demand nella console Amazon RDS. Dopo il 30 giugno 2026 solo la modalità Advanced di Database Insights supporterà i piani di esecuzione e l'analisi su richiesta.
> Con CloudWatch Database Insights, puoi monitorare il carico del database per la tua flotta di database e analizzare e risolvere i problemi delle prestazioni su larga scala. Per ulteriori informazioni su Database Insights, consulta Monitoraggio dei database Amazon Aurora con CloudWatch Database Insights. Per informazioni sui prezzi, consulta la pagina CloudWatch dei prezzi di Amazon.

È possibile attivare Approfondimenti sulle prestazioni per il cluster di database al momento della creazione. Se necessario, è possibile disattivarlo in un secondo momento modificando il cluster di

database dalla console. L'attivazione e la disattivazione di Performance Insights non determina tempi di inattività, riavvio o failover.

> **Note**
>
> Performance Schema è uno strumento di prestazioni opzionale utilizzato da Aurora MySQL. Se si attiva o disattiva Performance Schema, è necessario riavviare il sistema. Se si attiva o disattiva Performance Insights, tuttavia, non è necessario riavviare. Per ulteriori informazioni, consulta Panoramica dello schema di prestazioni per Approfondimenti sulle prestazioni su Aurora MySQL.

Se si utilizza Approfondimenti sulle prestazioni insieme ai database globali Aurora, è necessario attivare Approfondimenti sulle prestazioni individualmente per i database in ciascuna Regione AWS. Per dettagli, consultare Monitoraggio di un database globale Amazon Aurora con Amazon RDS Performance Insights.

L'agente Performance Insights consuma CPU e memoria limitate sull'host DB. Quando il carico del DB è elevato, l'agente limita l'impatto sulle prestazioni raccogliendo i dati meno frequentemente.

Console

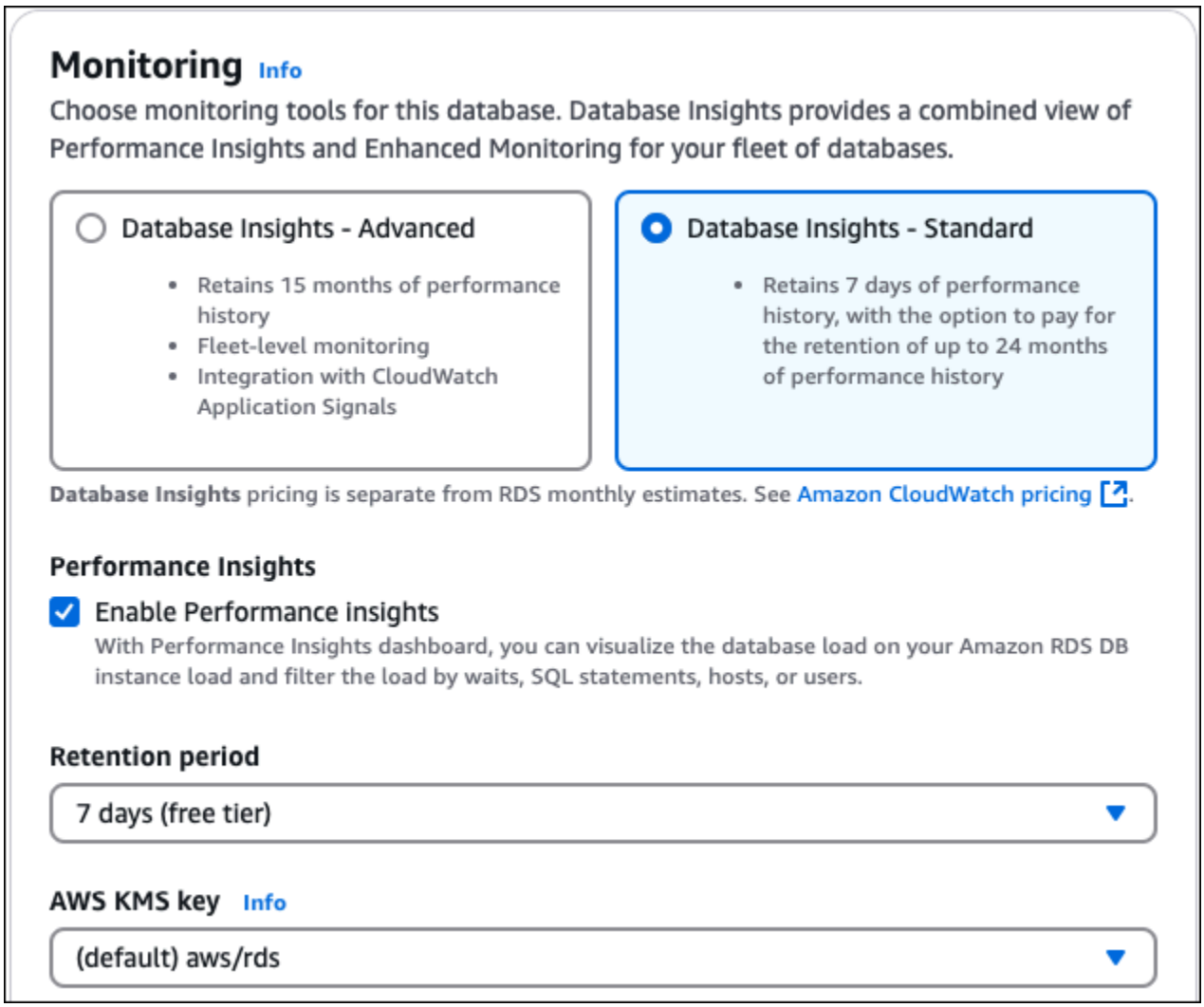
Nella console è possibile attivare o disattivare la funzionalità Performance Insights quando crei o modifichi un cluster database. L’attivazione di Approfondimenti sulle prestazioni consente di gestire le impostazioni e le opzioni di Approfondimenti sulle prestazioni per il cluster di database. Le impostazioni a livello di cluster si applicano a tutte le istanze database nel cluster.

Attivazione o disattivazione di Approfondimenti sulle prestazioni durante la creazione di un cluster di database

Dopo aver creato un nuovo cluster di database, Amazon RDS attiva Approfondimenti sulle prestazioni per impostazione predefinita. Per disattivare Approfondimenti sulle prestazioni per il cluster di database, scegliere l’opzione Informazioni sul database - Standard e deselezionare l’opzione Abilita Performance Insights.

Per creare un cluster di database, seguire le istruzioni per il proprio motore di database in Creazione di un cluster database Amazon Aurora.

L'immagine seguente mostra la sezione Performance Insights.



Quando selezioni Abilita Performance Insights, sono disponibili le opzioni seguenti:

- Conservazione (solo per la modalità Standard di Database Insights): il tempo di conservazione dei dati di Approfondimenti sulle prestazioni. L'impostazione del periodo di conservazione è Impostazione predefinita (7 giorni). Per mantenere i dati sulle prestazioni più a lungo, specifica da 1 a 24 mesi. Per altre informazioni sui periodi di conservazione, consulta Prezzi e conservazione dei dati per Performance Insights.
- AWS KMS key— Specificate il vostro AWS KMS key. Performance Insights crittografa tutti i dati potenzialmente sensibili con la chiave KMS. I dati vengono crittografati mentre sono in transito o inattivi. Per ulteriori informazioni, consulta Modifica di una policy AWS KMS per Approfondimenti sulle prestazioni.

Attivazione o disattivazione di Approfondimenti sulle prestazioni durante la modifica di un cluster di database

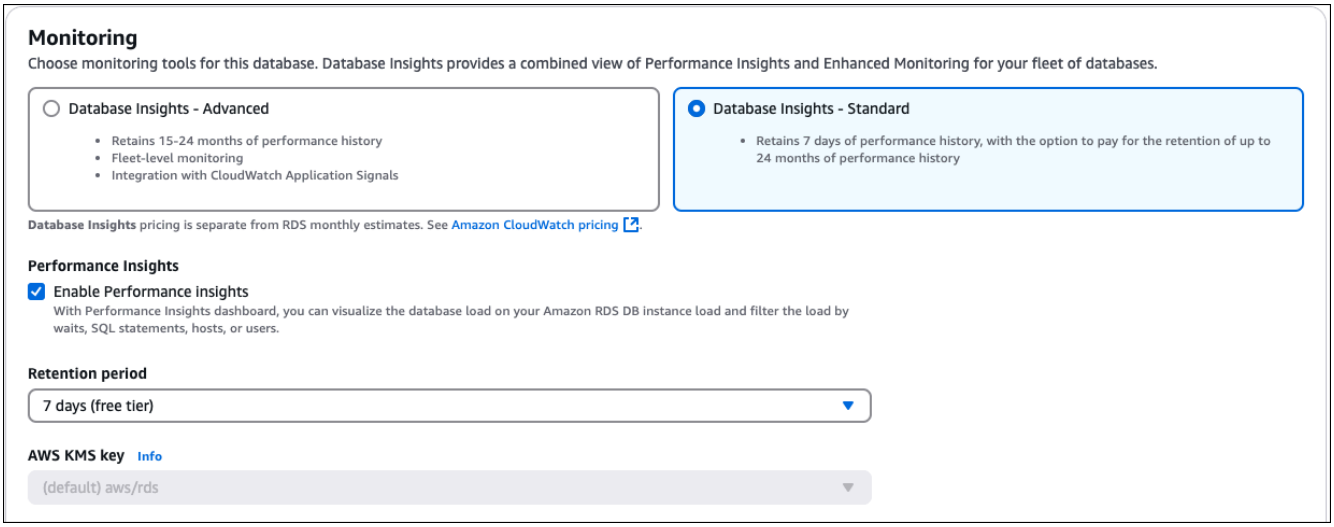
Nella console è possibile modificare un cluster di database per gestire Approfondimenti sulle prestazioni.

Per gestire Approfondimenti sulle prestazioni per un cluster di database utilizzando la console

1. Accedi a Console di gestione AWS e apri la console Amazon RDS all'indirizzo https://console.aws.amazon.com/rds/.
2. Scegli Database.
3. Scegli un cluster di database e scegli Modifica.
4. Per attivare Approfondimenti sulle prestazioni, seleziona Abilita Performance Insights. Per disattivare Approfondimenti sulle prestazioni per il cluster di database, scegliere l'opzione Informazioni sul database - Standard e deselezionare l'opzione Abilita Performance Insights.

   Quando selezioni Abilita Performance Insights, sono disponibili le opzioni seguenti:

   - Conservazione (solo per la modalità Standard di Database Insights): il tempo di conservazione dei dati di Approfondimenti sulle prestazioni. L'impostazione del periodo di conservazione è Impostazione predefinita (7 giorni). Per mantenere i dati sulle prestazioni più a lungo, specifica da 1 a 24 mesi. Per altre informazioni sui periodi di conservazione, consulta Prezzi e conservazione dei dati per Performance Insights.
   - AWS KMS key: specificare la chiave KMS. Performance Insights crittografa tutti i dati potenzialmente sensibili con la chiave KMS. I dati vengono crittografati mentre sono in transito o inattivi. Per ulteriori informazioni, consulta Crittografia delle risorse Amazon Aurora.



5. Scegli Continue (Continua).
6. In Scheduling of Modifications (Pianificazione delle modifiche), scegli Apply immediately (Applica immediatamente). Se scegli Apply (Applica) durante la prossima finestra di manutenzione pianificata, l'istanza ignora questa impostazione e attiva immediatamente Performance Insights.
7. Scegli Modify instance (Modifica istanza).

AWS CLI

Quando usi il create-db-instance AWS CLI comando, attiva Performance Insights specificando `--enable-performance-insights` e impostando `--database-insights-mode` su `advanced` o `standard`. Per disattivare Approfondimenti sulle prestazioni, specifica `--no-enable-performance-insights` e imposta `database-insights-mode` su `standard`.

È inoltre possibile specificare questi valori utilizzando i seguenti AWS CLI comandi:

- create-db-cluster
- modify-db-cluster
- create-db-instance-read-replica
- modify-db-instance
- restore-db-instance-from-s3

Per gestire Performance Insights per un cluster DB utilizzando AWS CLI

- Chiamate il modify-db-cluster AWS CLI comando e fornite i seguenti valori:
  - `--db-cluster-identifier` - Il nome dell'istanza database nel cluster di database.
  - `--enable-performance-insights` per attivare o `--no-enable-performance-insights` per disattivare
  - `database-insights-mode` - La modalità di Database Insights per il cluster di database. Per disattivare Approfondimenti sulle prestazioni, imposta questo valore su `standard`.

  Il seguente esempio attiva Performance Insights per `sample-db-cluster`.

  Per Linux, macOS o Unix:

```
aws rds modify-db-cluster \
 --database-insights-mode standard \
    --db-cluster-identifier sample-db-instance \
    --enable-performance-insights
```

  Per Windows:

```
aws rds modify-db-cluster ^
 --database-insights-mode standard ^
    --db-cluster-identifier sample-db-instance ^
    --enable-performance-insights
```

Quando attivi Performance Insights nella CLI, puoi specificare, in via facoltativa, il periodo di tempo, in giorni, per cui mantenere i dati di Performance Insights con l'opzione `--performance-insights-retention-period`. È possibile specificare7, *`month`* * 31 (dove *`month`* è un numero compreso tra 1 e 23) o. 731 Ad esempio, se desideri mantenere i dati sulle prestazioni per 3 mesi, specifica 93, che è 3 * 31. L'impostazione di default è 7 giorni. Per altre informazioni sui periodi di conservazione, consulta Prezzi e conservazione dei dati per Performance Insights.

Il seguente esempio attiva Performance Insights per `sample-db-cluster` e specifica che i dati Performance Insights sono mantenuti per 93 giorni (3 mesi).

Per Linux, macOS o Unix:

```
aws rds modify-db-cluster \
 --database-insights-mode standard \
    --db-cluster-identifier sample-db-instance \
    --enable-performance-insights \
    --performance-insights-retention-period 93
```

Per Windows:

```
aws rds modify-db-cluster ^
 --database-insights-mode standard ^
    --db-cluster-identifier sample-db-instance ^
    --enable-performance-insights ^
    --performance-insights-retention-period 93
```

Se il periodo di conservazione specificato, ad esempio 94 giorni, non è un valore valido, RDS genera un errore.

```
An error occurred (InvalidParameterValue) when calling the CreateDBInstance
 operation:
Invalid Performance Insights retention period. Valid values are: [7, 31, 62, 93,
 124, 155, 186, 217,
248, 279, 310, 341, 372, 403, 434, 465, 496, 527, 558, 589, 620, 651, 682, 713, 731]
```

> **Note**
>
> È possibile attivare/disattivare Approfondimenti sulle prestazioni solo per un'istanza in un cluster di database in cui Approfondimenti sulle prestazioni non è gestito a livello di cluster.

RDS API

Quando crei una nuova istanza DB nel tuo cluster DB utilizzando l'DBInstanceoperazione Create operation Amazon RDS API, attiva Performance Insights `EnablePerformanceInsights` impostando su. `True` Per disattivare Approfondimenti sulle prestazioni, imposta `EnablePerformanceInsights` su `False` e `DatabaseInsightsMode` su `standard`.

Puoi anche specificare il valore `EnablePerformanceInsights` utilizzando le seguenti operazioni API:

- CreaDBCluster
- ModificaDBCluster
- ModificaDBInstance
- CreaDBInstanceReadReplica
- Ripristina DBInstance da S3

Quando si attiva Performance Insights, è possibile specificare, in via facoltativa, il periodo di tempo, in giorni, per cui conservare i dati degli approfondimenti delle prestazioni con il parametro `PerformanceInsightsRetentionPeriod`. È possibile specificare7, *`month`* * 31 (dove *`month`* è un numero compreso tra 1 e 23) o. `731` Ad esempio, se desideri mantenere i dati sulle prestazioni per 3 mesi, specifica `93`, che è 3 * 31. L'impostazione di default è 7 giorni. Per altre informazioni sui periodi di conservazione, consulta Prezzi e conservazione dei dati per Performance Insights.

Panoramica dello schema di prestazioni per Approfondimenti sulle prestazioni su Aurora MySQL

Performance Schema è una funzionalità facoltativa per il monitoraggio delle prestazioni di runtime di Aurora MySQL a un dettaglio di basso livello. Performance Schema è progettato per avere un impatto minimo sulle prestazioni del database. Performance Insights è una funzionalità separata che puoi utilizzare con o senza Performance Schema.

Argomenti

- Panoramica dello schema di prestazioni
- Performance Insights e lo schema di prestazioni
- Gestione automatica di Performance Schema da parte di Performance Insights
- Effetto di un riavvio su Performance Schema
- Determinazione della gestione dello schema delle prestazioni da parte di Performance Insights
- Attivazione dello Schema delle prestazioni per Aurora MySQL

Panoramica dello schema di prestazioni

Performance Schema monitora gli eventi nei database Aurora MySQL. Un evento è un'azione del server di database che consuma tempo ed è stata strumentata in modo che possano essere raccolte le informazioni di temporizzazione. Ecco alcuni esempi di eventi:

- Chiamate di funzione
- Attendi il sistema operativo
- Fasi dell'esecuzione SQL
- Gruppi di istruzioni SQL

Il motore di archiviazione `PERFORMANCE_SCHEMA` è un meccanismo per l'implementazione della funzionalità Performance Schema. Questo motore raccoglie i dati degli eventi utilizzando la strumentazione nel codice sorgente del database. Il motore memorizza gli eventi raccolti nelle tabelle in memoria nel database `performance_schema`. È possibile interrogare `performance_schema` proprio come puoi interrogare qualsiasi altra tabella. Per ulteriori informazioni, consulta Performance Schema di MySQL nel Manuale di riferimento di MySQL.

Performance Insights e lo schema di prestazioni

Performance Insights e Performance Schema sono funzionalità separate, ma sono connesse. Il comportamento di Performance Insights per Aurora MySQL varia a seconda che lo schema di prestazioni sia attivato e, in questo caso, se Performance Insights gestisce automaticamente lo schema di prestazioni. Il comportamento viene descritto nella tabella seguente.

Schema di prestazioni attivato	Modalità di gestione di Performance Insights	Comportamento di Performance Insights
Sì	Automatica	• Raccoglie informazioni dettagliate di monitoraggio a basso livello • Raccoglie le metriche di sessione attive ogni secondo •

Schema di prestazioni attivato	Modalità di gestione di Performance Insights	Comportamento di Performance Insights
		Visualizza il carico del database classificato in base a eventi di attesa dettagliati, che è possibile utilizzare per identificare i colli di bottiglia
Sì	Manuale	• Raccoglie gli eventi di attesa e le metriche per SQL • Raccoglie le metriche di sessione attive ogni secondo • Segnala gli stati utente, ad esempio l'inserimento e l'invio, che non consentono di identificare i colli di bottiglia
No	N/D	• Non raccoglie eventi di attesa, metriche per SQL o altre informazioni dettagliate di monitoraggio di basso livello • Raccoglie le metriche di sessione attive ogni cinque secondi anziché ogni secondo • Segnala gli stati utente, ad esempio l'inserimento e l'invio, che non consentono di identificare i colli di bottiglia

Gestione automatica di Performance Schema da parte di Performance Insights

Quando crei un'istanza database Aurora MySQL con Performance Insights abilitato, anche la funzionalità Performance Schema viene abilitata. In questo caso, Performance Insights gestisce automaticamente i parametri di Performance Schema. Questa è la configurazione consigliata.

Quando Approfondimenti sulle prestazioni gestisce automaticamente lo schema delle prestazioni l'origine di `performance_schema` è `System default`.

> **Note**
>
> La gestione automatica dello schema di prestazioni non è supportata per la classe di istanza t4g.medium.

Puoi anche gestire Performance Schema manualmente. Se scegli questa opzione, imposta i parametri in base ai valori nella tabella riportata di seguito.

Nome del parametro	Valore del parametro
`performance_schema`	1 (la colonna Source (Origine) è impostata al valore `Modified`)
`performance-schema-consumer-events-waits-current`	`ON`
`performance-schema-instrument`	`wait/%=ON`
`performance_schema_consumer_global_instrumentation`	`1`
`performance_schema_consumer_thread_instrumentation`	`1`

Se modifichi il manualmente il parametro `performance_schema` e in seguito desideri ripristinare la gestione automatica, consulta Attivazione dello Schema delle prestazioni per Aurora MySQL.

> **Important**
>
> Quando Performance Insights abilita Performance Schema, non modifica i valori del gruppo di parametri. Tuttavia, i valori vengono modificati sulle istanze database in esecuzione. L'unico modo per vedere i valori modificati è eseguire il comando `SHOW GLOBAL VARIABLES`.

Effetto di un riavvio su Performance Schema

Performance Insights e Performance Schema differiscono per i requisiti relativi al riavvio delle istanze DB:

Performance Schema

Per attivare o disattivare questa funzionalità, è necessario riavviare l'istanza database.

Performance Insights

Per attivare o disattivare questa funzionalità, non è necessario riavviare l'istanza database.

Se Performance Schema non è attualmente attivato e si attiva Performance Insights senza riavviare l'istanza database, Performance Schema non verrà attivato.

Determinazione della gestione dello schema delle prestazioni da parte di Performance Insights

Per determinare se Approfondimenti sulle prestazioni gestisce lo schema delle prestazioni per tutte le versioni principali del motore, consulta la tabella riportata di seguito.

Impostazione del parametro performance_schema	Impostazione della colonna Source	Performance Insights sta gestendo Performance Schema?
`0`	`System default`	Sì
`0` o `1`	`Modified`	No

La seguente procedura consente di determinare se Approfondimenti sulle prestazioni gestisce automaticamente lo schema delle prestazioni.

Per determinare se Performance Insights sta gestendo automaticamente Performance Schema

1. Accedi alla Console di gestione AWS e apri la console Amazon RDS all'indirizzo https://console.aws.amazon.com/rds/.
2. Scegliere Gruppi di parametri.
3. Selezionare il nome del gruppo di parametri per l'istanza database.

4. Inserire **`performance_schema`** nella barra di ricerca.
5. Controllare se Origine è il valore predefinito di sistema e Valore è impostato su 0. In tal caso, Performance Insights gestisce automaticamente Performance Schema.

   In questo esempio, Approfondimenti sulle prestazioni non gestisce automaticamente lo schema delle prestazioni.

Attivazione dello Schema delle prestazioni per Aurora MySQL

Supponiamo che Performance Insights sia attivato per l'istanza database ma al momento non stia gestendo Performance Schema. Se desideri consentire a Performance Insights di gestire automaticamente Performance Schema, completa la procedura seguente.

Configurazione di Performance Schema per la gestione automatica

1. Accedi alla Console di gestione AWS e apri la console Amazon RDS all'indirizzo https://console.aws.amazon.com/rds/.
2. Scegliere Gruppi di parametri.
3. Selezionare il nome del gruppo di parametri per l'istanza database.
4. Scegli Modifica.
5. Inserire **`performance_schema`** nella barra di ricerca.
6. Selezionare il parametro `performance_schema`.
7. Scegli Imposta sul valore predefinito.
8. Conferma scegliendo Imposta i valori sull'impostazione predefinita.
9. Seleziona Salva modifiche.
10. Riavvia l'istanza database.

> **Important**
>
> Ogni volta che si abilita o disabilita Performance Schema, è necessario riavviare l'istanza database.

Per ulteriori informazioni sulla modifica dei parametri di un'istanza, consulta Modifica dei parametri in un gruppo di parametri database in Amazon Aurora. Per ulteriori informazioni sulle pagine del pannello di controllo, consulta Analisi delle metriche utilizzando il pannello di controllo Performance Insights. Per ulteriori informazioni sullo schema delle prestazioni di MySQL, consulta Schema delle prestazioni di MySQL (per 8.0) e Schema delle prestazioni di MySQL (per 8.4) nella documentazione MySQL.

Configurazione delle policy di accesso per Performance Insights

Per accedere a Performance Insights, un principale deve disporre delle autorizzazioni appropriate di AWS Identity and Access Management (IAM).

> **Note**
>
> Per utilizzare Performance Insights con una chiave gestita dal cliente, concedi agli utenti le `kms:GenerateDataKey` autorizzazioni `kms:Decrypt` e per la tua chiave. AWS AWS KMS

Accedi a Performance Insights utilizzando questi metodi:

- Allega la policy `AmazonRDSPerformanceInsightsReadOnly` gestita per l'accesso in sola lettura
- Allega la policy `AmazonRDSPerformanceInsightsFullAccess` gestita per l'accesso a tutte le operazioni dell'API Performance Insights
- Crea una policy IAM personalizzata con autorizzazioni specifiche
- Configurare le AWS KMS autorizzazioni per i dati crittografati di Performance Insights
- Configura un accesso granulare utilizzando autorizzazioni a livello di risorsa
- Utilizza il controllo degli accessi basato su tag per gestire le autorizzazioni tramite tag di risorse

Collegamento della policy AmazonRDSPerformanceInsightsReadOnly a un principale IAM

`AmazonRDSPerformanceInsightsReadOnly`è una policy AWS gestita che garantisce l'accesso a tutte le operazioni di sola lettura dell'API Amazon RDS Performance Insights.

Se ti colleghi `AmazonRDSPerformanceInsightsReadOnly` a un set di autorizzazioni o a un ruolo, devi allegare anche le seguenti autorizzazioni: CloudWatch

- `GetMetricStatistics`
- `ListMetrics`
- `GetMetricData`

Con queste autorizzazioni, il destinatario può utilizzare Performance Insights con altre funzionalità della console.

Per ulteriori informazioni sulle CloudWatch autorizzazioni, consulta Amazon CloudWatch permissions reference.

Per ulteriori informazioni su `AmazonRDSPerformanceInsightsReadOnly`, consultare Policy gestita AWS: AmazonRDSPerformanceInsightsReadOnly.

Collegamento della policy AmazonRDSPerformanceInsightsFullAccess a un principale IAM

`AmazonRDSPerformanceInsightsFullAccess`è una policy AWS gestita che garantisce l'accesso a tutte le operazioni dell'API Amazon RDS Performance Insights.

Se ti colleghi `AmazonRDSPerformanceInsightsFullAccess` a un set di autorizzazioni o a un ruolo, devi allegare anche le seguenti CloudWatch autorizzazioni:

- `GetMetricStatistics`
- `ListMetrics`
- `GetMetricData`

Con queste autorizzazioni, il destinatario può utilizzare Performance Insights con altre funzionalità della console.

Per ulteriori informazioni sulle CloudWatch autorizzazioni, consulta Amazon CloudWatch permissions reference.

Per ulteriori informazioni su `AmazonRDSPerformanceInsightsFullAccess`, consultare Policy gestita da AWS: AmazonRDSPerformanceInsightsFullAccess.

Creazione di una policy IAM personalizzata per Approfondimenti sulle prestazioni

Per gli utenti che non dispongono della policy `AmazonRDSPerformanceInsightsReadOnly` o `AmazonRDSPerformanceInsightsFullAccess`, è possibile fornire l'accesso a Approfondimenti

sulle prestazioni creando o modificando una policy IAM gestita dall'utente. Quando si collega la policy a un set di autorizzazioni o un ruolo, il destinatario può utilizzare Performance Insights.

Per creare una policy personalizzata

1. Aprire la console IAM all'indirizzo https://console.aws.amazon.com/iam/.
2. Nel pannello di navigazione, selezionare Policies (Policy).
3. Scegli Create Policy (Crea policy).
4. Nella pagina Crea policy, seleziona l'opzione JSON.
5. Copiare e incollare il testo fornito nella sezione del documento di policy JSON nella Guida di riferimento sulle policy gestite da AWS per la policy AmazonRDSPerformanceInsightsReadOnly o AmazonRDSPerformanceInsightsFullAccess.
6. Scegliere Esamina policy.
7. Specifica un nome per la policy e, facoltativamente, una descrizione e quindi scegli Crea policy.

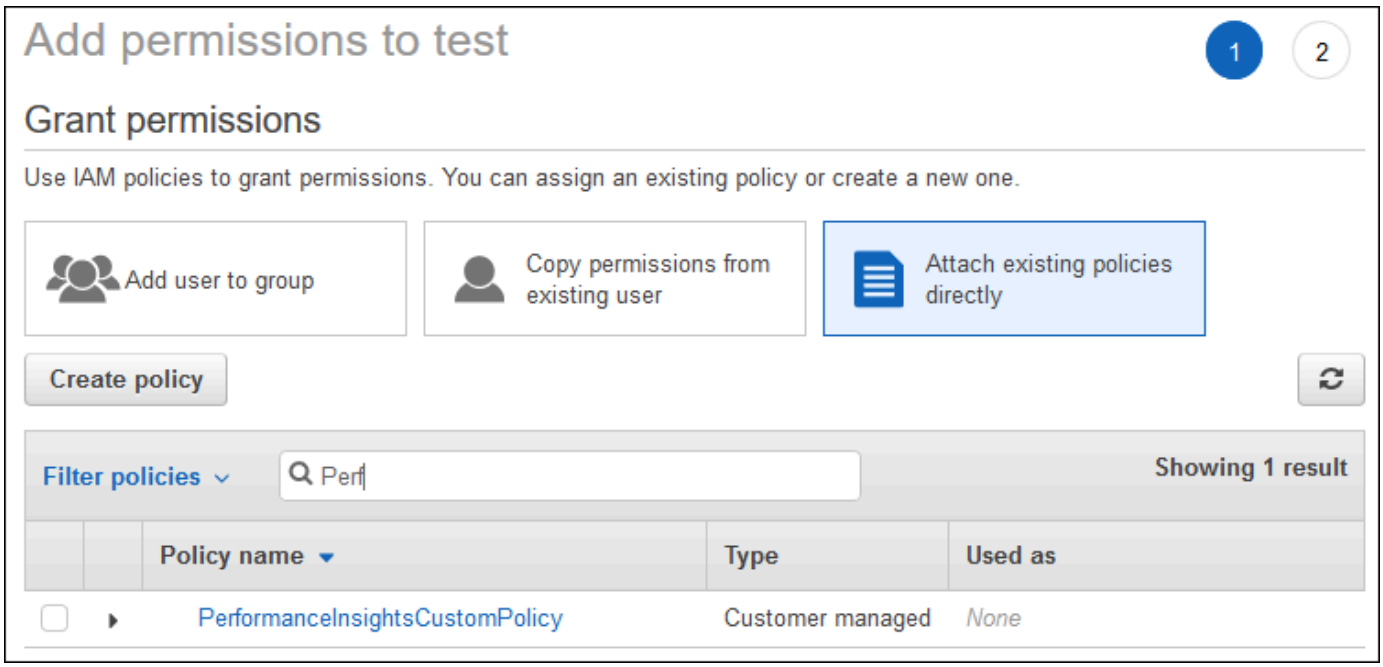
Ora è possibile collegare la policy a un set di autorizzazioni o un ruolo. La seguente procedura presuppone che si disponga già di un utente disponibile allo scopo.

Per collegare la policy a un utente

1. Aprire la console IAM all'indirizzo https://console.aws.amazon.com/iam/.
2. Nel pannello di navigazione, seleziona Utenti.
3. Seleziona un utente esistente dall'elenco.

> **Important**
>
> Per utilizzare Performance Insights, l'utente deve avere accesso a Amazon RDS nonché alla policy personalizzata. Ad esempio, la policy predefinita `AmazonRDSPerformanceInsightsReadOnly` concede l'accesso in sola lettura ad Amazon RDS. Per ulteriori informazioni, consulta Gestione dell'accesso tramite policy.

4. Nella pagina Summary (Riepilogo), scegli Add permissions (Aggiungi autorizzazioni).
5. Scegli Attach existing policies directly (Collega direttamente le policy esistenti). In Cerca digita i primi caratteri del nome della policy, come mostrato nella seguente immagine.



6. Scegli la policy e quindi seleziona Next: Review (Successivo: Rivedi).
7. Scegli Add Permissions (Aggiungi autorizzazioni).

Modifica di una policy AWS KMS per Approfondimenti sulle prestazioni

Performance Insights utilizza an AWS KMS key per crittografare i dati sensibili. Quando abiliti Performance Insights mediante l'API o la console, sono disponibili le seguenti opzioni:

- Scegli l'impostazione predefinitaChiave gestita da AWS.

  Amazon RDS lo utilizza Chiave gestita da AWS per la tua nuova istanza DB. Amazon RDS crea una Chiave gestita da AWS per il tuo Account AWS. Il tuo Account AWS ha un Amazon RDS diverso Chiave gestita da AWS per ognunoRegione AWS.

- Scegli una chiave gestita dal cliente.

  Se si specifica una chiave gestita dal cliente, gli utenti dell'account che chiamano l'API Performance Insights necessitano delle autorizzazioni `kms:Decrypt` e `kms:GenerateDataKey` per la chiave KMS. È possibile configurare queste autorizzazioni mediante le policy IAM. Tuttavia, è consigliabile gestire queste autorizzazioni mediante la policy della chiave KMS. Per ulteriori informazioni, consulta Policy delle chiavi in AWS KMS nella Guida per gli sviluppatori di AWS Key Management Service.

Example

Il seguente esempio mostra come aggiungere istruzioni alla policy della chiave KMS. Queste istruzioni consentono l'accesso a Performance Insights. A seconda della modalità di utilizzare la chiave KMS, potrebbe essere necessario modificare alcune restrizioni. Prima di aggiungere istruzioni alle policy, ai criteri, rimuovi tutti i commenti.

JSON

```
{
    "Version":"2012-10-17",
    "Id" : "your-policy",
    "Statement" : [
        {
            "Sid" : "AllowViewingRDSPerformanceInsights",
            "Effect": "Allow",
            "Principal": {
                "AWS": [
                    "arn:aws:iam::444455556666:role/Role1"
                ]
                },
             "Action": [
                "kms:Decrypt",
                "kms:GenerateDataKey"
                ],
            "Resource": "*",
            "Condition" : {
            "StringEquals" : {
                "kms:ViaService" : "rds.us-east-1.amazonaws.com"
                },
            "ForAnyValue:StringEquals": {
                "kms:EncryptionContext:aws:pi:service": "rds",
                "kms:EncryptionContext:service": "pi",
                "kms:EncryptionContext:aws:rds:db-id": "db-
AAAAABBBBBCCCCDDDDDEEEEE"
                }
            }
        }
    ]
}
```

In che modo Performance Insights utilizza la chiave gestita dal AWS KMS cliente

La funzionalità Performance Insights utilizza una chiave gestita dal cliente per crittografare i dati sensibili. Quando attivi la funzionalità Performance Insights, puoi specificare una chiave AWS KMS tramite l'API. Performance Insights crea AWS KMS autorizzazioni su questa chiave. Utilizza la chiave ed esegue le operazioni necessarie per elaborare i dati sensibili. I dati sensibili includono campi come utente, database, applicazione e testo di query SQL. La funzionalità Performance Insights garantisce che i dati rimangano crittografati mentre sono sia in transito che inattivi.

Come funziona Performance Insights con IAM AWS KMS

IAM concede autorizzazioni a persone specifiche APIs. Performance Insights ha il seguente pubblico APIs, che puoi limitare utilizzando le policy IAM:

- `DescribeDimensionKeys`
- `GetDimensionKeyDetails`
- `GetResourceMetadata`
- `GetResourceMetrics`
- `ListAvailableResourceDimensions`
- `ListAvailableResourceMetrics`

Puoi utilizzare le seguenti richieste API per recuperare i dati sensibili.

- `DescribeDimensionKeys`
- `GetDimensionKeyDetails`
- `GetResourceMetrics`

Quando utilizzi l'API per recuperare i dati sensibili, la funzionalità Approfondimenti sulle prestazioni usa le credenziali del chiamante. Questo controllo garantisce che l'accesso ai dati sensibili sia limitato a coloro che hanno accesso alla chiave KMS.

Quando le chiami APIs, hai bisogno delle autorizzazioni per chiamare l'API tramite la policy IAM e delle autorizzazioni per richiamare l'`kms:decrypt`azione tramite la AWS KMS policy chiave.

L'API `GetResourceMetrics` può restituire dati sensibili e non sensibili. I parametri della richiesta determinano se la risposta deve includere dati sensibili. L'API restituisce dati sensibili quando la richiesta include una dimensione sensibile nei parametri del filtro o nei parametri di raggruppamento.

Per ulteriori informazioni sulle dimensioni che puoi utilizzare con l'`GetResourceMetrics`API, consulta. DimensionGroup

Example Esempi

L'esempio seguente richiede i dati sensibili per il gruppo `db.user`:

```
POST / HTTP/1.1
Host: <Hostname>
Accept-Encoding: identity
X-Amz-Target: PerformanceInsightsv20180227.GetResourceMetrics
Content-Type: application/x-amz-json-1.1
User-Agent: <UserAgentString>
X-Amz-Date: <Date>
Authorization: AWS4-HMAC-SHA256 Credential=<Credential>, SignedHeaders=<Headers>,
 Signature=<Signature>
Content-Length: <PayloadSizeBytes>
{
  "ServiceType": "RDS",
  "Identifier": "db-ABC1DEFGHIJKL2MNOPQRSTUV3W",
  "MetricQueries": [
    {
      "Metric": "db.load.avg",
      "GroupBy": {
        "Group": "db.user",
        "Limit": 2
      }
    }
  ],
  "StartTime": 1693872000,
  "EndTime": 1694044800,
  "PeriodInSeconds": 86400
}
```

Example

L'esempio seguente richiede i dati non sensibili per la metrica `db.load.avg`:

```
POST / HTTP/1.1
Host: <Hostname>
```

```
Accept-Encoding: identity
X-Amz-Target: PerformanceInsightsv20180227.GetResourceMetrics
Content-Type: application/x-amz-json-1.1
User-Agent: <UserAgentString>
X-Amz-Date: <Date>
Authorization: AWS4-HMAC-SHA256 Credential=<Credential>, SignedHeaders=<Headers>,
 Signature=<Signature>
Content-Length: <PayloadSizeBytes>
{
    "ServiceType": "RDS",
    "Identifier": "db-ABC1DEFGHIJKL2MNOPQRSTUV3W",
    "MetricQueries": [
        {
            "Metric": "db.load.avg"
        }
    ],
    "StartTime": 1693872000,
    "EndTime": 1694044800,
    "PeriodInSeconds": 86400
}
```

Concessione dell'accesso granulare per Approfondimenti sulle prestazioni

Il controllo granulare dell'accesso offre ulteriori modi per controllare l'accesso a Approfondimenti sulle prestazioni. Questo controllo dell'accesso può consentire o negare l'accesso alle singole dimensioni e alle azioni `GetResourceMetrics`, `DescribeDimensionKeys` e `GetDimensionKeyDetails` di Approfondimenti sulle prestazioni. Per utilizzare l'accesso granulare, specificare le dimensioni nella policy IAM utilizzando le chiavi di condizione. La valutazione dell'accesso segue la logica di valutazione della policy IAM. Per ulteriori informazioni, consulta Logica di valutazione delle policy nella Guida per l'utente di IAM. Se l'istruzione della policy IAM non specifica alcuna dimensione, l'istruzione controlla l'accesso a tutte le dimensioni per l'azione specificata. Per l'elenco delle dimensioni disponibili, consulta DimensionGroup.

Per determinare a quali dimensioni le credenziali sono autorizzate ad accedere, utilizzare il parametro `AuthorizedActions` in `ListAvailableResourceDimensions` e specificare l'azione. I valori consentiti per `AuthorizedActions` sono:

- `GetResourceMetrics`
- `DescribeDimensionKeys`
- `GetDimensionKeyDetails`

Ad esempio, se si specifica `GetResourceMetrics` per il parametro `AuthorizedActions`, `ListAvailableResourceDimensions` restituisce l'elenco delle dimensioni a cui l'azione `GetResourceMetrics` è autorizzata ad accedere. Se si specificano più azioni nel parametro `AuthorizedActions`, `ListAvailableResourceDimensions` restituisce un'intersezione di dimensioni a cui tali azioni sono autorizzate ad accedere.

Example

L'esempio seguente fornisce l'accesso alle dimensioni specificate per le azioni `GetResourceMetrics` e `DescribeDimensionKeys`.

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Sid": "AllowToDiscoverDimensions",
            "Effect": "Allow",
            "Action": [
                "pi:ListAvailableResourceDimensions"
            ],
            "Resource": [
                "arn:aws:pi:us-east-1:123456789012:metrics/rds/db-
ABC1DEFGHIJKL2MNOPQRSTUV3W"
            ]
        },
        {
            "Sid": "SingleAllow",
            "Effect": "Allow",
            "Action": [
                "pi:GetResourceMetrics",
                "pi:DescribeDimensionKeys"
            ],
            "Resource": [
                "arn:aws:pi:us-east-1:123456789012:metrics/rds/db-
ABC1DEFGHIJKL2MNOPQRSTUV3W"
            ],
            "Condition": {
                "ForAllValues:StringEquals": {
                    "pi:Dimensions": [
                        "db.sql_tokenized.id",
```

```
                        "db.sql_tokenized.statement"
                    ]
                }
            }
        }


    ]
}
```

Di seguito è riportata la risposta per la dimensione richiesta:

```
 // ListAvailableResourceDimensions API
// Request
{
    "ServiceType": "RDS",
    "Identifier": "db-ABC1DEFGHIJKL2MNOPQRSTUV3W",
    "Metrics": [ "db.load" ],
    "AuthorizedActions": ["DescribeDimensionKeys"]
}

// Response
{
    "MetricDimensions": [ {
        "Metric": "db.load",
        "Groups": [
            {
                "Group": "db.sql_tokenized",
                "Dimensions": [
                    { "Identifier": "db.sql_tokenized.id" },
                  //  { "Identifier": "db.sql_tokenized.db_id" }, // not included
 because not allows in the IAM Policy
                    { "Identifier": "db.sql_tokenized.statement" }
                ]
            }

        ] }
    ]
}
```

L’esempio seguente specifica un accesso consentito e due accessi negati per le dimensioni.

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
          {
            "Sid": "AllowToDiscoverDimensions",
            "Effect": "Allow",
            "Action": [
                "pi:ListAvailableResourceDimensions"
            ],
            "Resource": [
                "arn:aws:pi:us-east-1:123456789012:metrics/rds/db-
ABC1DEFGHIJKL2MNOPQRSTUV3W"
            ]
          },

          {
            "Sid": "O01AllowAllWithoutSpecifyingDimensions",
            "Effect": "Allow",
            "Action": [
                "pi:GetResourceMetrics",
                "pi:DescribeDimensionKeys"
            ],
            "Resource": [
                "arn:aws:pi:us-east-1:123456789012:metrics/rds/db-
ABC1DEFGHIJKL2MNOPQRSTUV3W"
            ]
        },

        {
            "Sid": "O01DenyAppDimensionForAll",
            "Effect": "Deny",
            "Action": [
                "pi:GetResourceMetrics",
                "pi:DescribeDimensionKeys"
            ],
            "Resource": [
                "arn:aws:pi:us-east-1:123456789012:metrics/rds/db-
ABC1DEFGHIJKL2MNOPQRSTUV3W"
            ],
            "Condition": {
                "ForAnyValue:StringEquals": {
```

```
                    "pi:Dimensions": [
                        "db.application.name"
                    ]
                }
            }
        },

        {
            "Sid": "O01DenySQLForGetResourceMetrics",
            "Effect": "Deny",
            "Action": [
                "pi:GetResourceMetrics"
            ],
            "Resource": [
                "arn:aws:pi:us-east-1:123456789012:metrics/rds/db-
ABC1DEFGHIJKL2MNOPQRSTUV3W"
            ],
            "Condition": {
                "ForAnyValue:StringEquals": {
                    "pi:Dimensions": [
                        "db.sql_tokenized.statement"
                    ]
                }
            }
        }
    ]
}
```

Di seguito sono riportate le risposte per le dimensioni richieste:

```
   // ListAvailableResourceDimensions API
// Request
{
    "ServiceType": "RDS",
    "Identifier": "db-ABC1DEFGHIJKL2MNOPQRSTUV3W",
    "Metrics": [ "db.load" ],
    "AuthorizedActions": ["GetResourceMetrics"]
}

// Response
```

```
{
    "MetricDimensions": [ {
        "Metric": "db.load",
        "Groups": [
            {
                "Group": "db.application",
                "Dimensions": [

                  // removed from response because denied by the IAM Policy
                  //  { "Identifier": "db.application.name" }
                ]
            },
            {
                "Group": "db.sql_tokenized",
                "Dimensions": [
                    { "Identifier": "db.sql_tokenized.id" },
                    { "Identifier": "db.sql_tokenized.db_id" },

                  // removed from response because denied by the IAM Policy
                  //  { "Identifier": "db.sql_tokenized.statement" }
                ]
            },
            ...
        ] }
    ]
}
```

```
// ListAvailableResourceDimensions API
// Request
{
    "ServiceType": "RDS",
    "Identifier": "db-ABC1DEFGHIJKL2MNOPQRSTUV3W",
    "Metrics": [ "db.load" ],
    "AuthorizedActions": ["DescribeDimensionKeys"]
}

// Response
{
    "MetricDimensions": [ {
        "Metric": "db.load",
        "Groups": [
            {
```

```
                "Group": "db.application",
                "Dimensions": [
                  // removed from response because denied by the IAM Policy
                  //  { "Identifier": "db.application.name" }
                ]
            },
            {
                "Group": "db.sql_tokenized",
                "Dimensions": [
                    { "Identifier": "db.sql_tokenized.id" },
                    { "Identifier": "db.sql_tokenized.db_id" },

                  // allowed for DescribeDimensionKeys because our IAM Policy
                  // denies it only for GetResourceMetrics
                    { "Identifier": "db.sql_tokenized.statement" }
                ]
            },
            ...
        ] }
    ]
}
```

Utilizzo del controllo degli accessi basato su tag per Performance Insights

Puoi controllare l'accesso alle metriche di Performance Insights utilizzando tag ereditati dall'istanza DB principale. Per controllare l'accesso alle operazioni di Performance Insights, utilizza le policy IAM. Queste policy possono controllare i tag sull'istanza DB per determinare le autorizzazioni.

Come funzionano i tag con Performance Insights

Performance Insights applica automaticamente i tag dell'istanza DB per autorizzare le metriche di Performance Insights. Quando aggiungi tag alla tua istanza DB, puoi utilizzarli immediatamente per controllare l'accesso ai dati di Performance Insights.

- Per aggiungere o aggiornare i tag per le metriche di Performance Insights, modifica i tag sulla tua istanza DB.
- Per visualizzare i tag per le metriche di Performance Insights, richiama la `ListTagsForResource` risorsa metrica Performance Insights. Restituirà i tag dall'istanza DB associata alla metrica.

> **Note**
>
> Le `UntagResource` operazioni `TagResource` and restituiscono un errore se si tenta di utilizzarle direttamente nelle metriche di Performance Insights.

Creazione di politiche IAM basate su tag

Per controllare l'accesso alle operazioni di Performance Insights, utilizza la chiave `aws:ResourceTag` condition nelle tue policy IAM. Queste politiche controllano i tag sulla tua istanza DB.

Example

Questa politica impedisce l'accesso alle metriche di Performance Insights per i database di produzione. La politica nega l'`pi:GetResourceMetrics`operazione in Performance Insights per qualsiasi risorsa del database contrassegnata con`env:prod`.

```
 {
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Deny",
            "Action": "pi:GetResourceMetrics",
            "Resource": "*",
            "Condition": {
                "StringEquals": {
                    "aws:ResourceTag/env": "prod"
                }
            }
        }
    ]
}
```

Analisi delle metriche utilizzando il pannello di controllo Performance Insights

Important

AWS ha annunciato la end-of-life data di Performance Insights: 30 giugno 2026. Dopo questa data, Amazon RDS non supporterà più l'esperienza della console Approfondimenti sulle prestazioni, i periodi di conservazione flessibili (1-24 mesi) e i relativi prezzi. L'API Approfondimenti sulle prestazioni continuerà a esistere senza modifiche ai prezzi. I costi per l'API Performance Insights verranno visualizzati nella AWS fattura insieme al costo di CloudWatch Database Insights.
Si consiglia di aggiornare tutte DB Clusters di database utilizzando il livello a pagamento di Performance Insights alla modalità Advanced di Database Insights prima del 30 giugno 2026. Per informazioni sull'aggiornamento alla modalità avanzata di Database Insights, consulta Attivazione della modalità avanzata di Database Insights per Amazon Aurora.
Se non si esegue alcuna azione, i cluster di database che utilizzano Approfondimenti sulle prestazioni, per impostazione predefinita useranno la modalità standard di Database Insights. Con la modalità standard di Database Insights, è possibile perdere l'accesso alla cronologia dei dati sulle prestazioni di oltre 7 giorni e non essere in grado di utilizzare i piani di esecuzione e le funzionalità di analisi on demand nella console Amazon RDS. Dopo il 30 giugno 2026 solo la modalità Advanced di Database Insights supporterà i piani di esecuzione e l'analisi su richiesta.
Con CloudWatch Database Insights, puoi monitorare il carico del database per la tua flotta di database e analizzare e risolvere i problemi delle prestazioni su larga scala. Per ulteriori informazioni su Database Insights, consulta Monitoraggio dei database Amazon Aurora con CloudWatch Database Insights. Per informazioni sui prezzi, consulta la pagina CloudWatch dei prezzi di Amazon.

Il pannello di controllo di Performance Insights contiene informazioni sulle performance del database, per consentire di analizzare e risolvere i problemi di performance. Nella pagina principale del pannello di controllo è possibile visualizzare le informazioni relative al carico del database. Puoi "dividere" il carico del database per dimensioni come eventi di attesa o SQL.

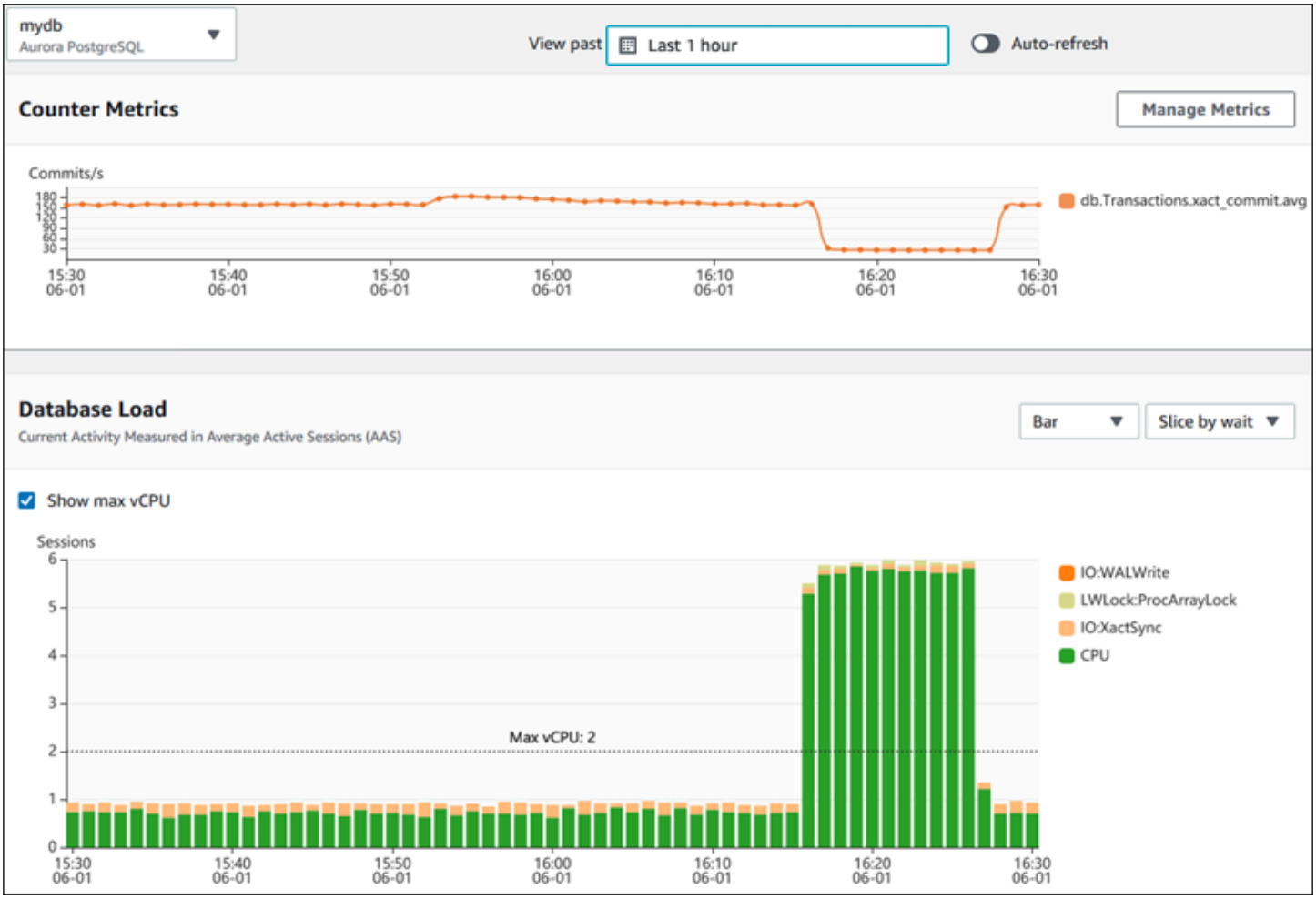
Pannello di controllo Performance Insights

- Panoramica del pannello di controllo di Performance Insights

- Accesso al pannello di controllo di Performance Insights
- Analisi del carico del database per eventi di attesa
- Analisi delle prestazioni del database per un periodo di tempo
- Analisi delle query con la scheda SQL principale in Approfondimenti sulle prestazioni

Panoramica del pannello di controllo di Performance Insights

Il pannello di controllo è il modo più semplice per interagire con Performance Insights. L'esempio seguente mostra la dashboard per un'istanza database PostgreSQL.



Argomenti

- Filtro intervallo temporale
- Grafico Parametri contatore
- Grafico di carico database

- Tabella dimensioni superiori

Filtro intervallo temporale

Di default, il pannello di controllo di Performance Insights mostra il carico del database relativo all'ultima ora. Puoi regolare questo intervallo di tempo da 5 minuti o fino a 2 anni. Puoi inoltre selezionare un intervallo relativo personalizzato.

Puoi selezionare un intervallo assoluto con data e ora di inizio e fine. L'esempio seguente mostra l'intervallo di tempo che inizia a mezzanotte del 25/9/24 e termina alle 23:59 del 28/9/24.

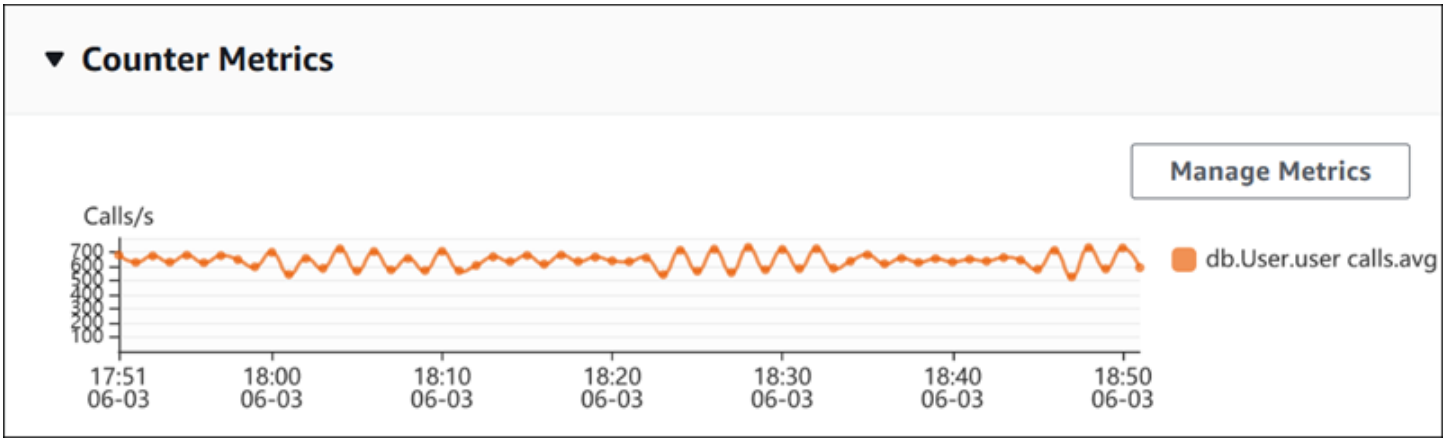
Per impostazione predefinita, il fuso orario per la dashboard di Approfondimenti sulle prestazioni è in formato Universal Time Coordinated (UTC). È anche possibile scegliere il fuso orario locale.

Grafico Parametri contatore

Con i parametri contatore, puoi personalizzare il pannello di controllo di Performance Insights per includere fino a 10 grafici aggiuntivi. Questi grafici mostrano una selezione di decine di parametri prestazionali di sistema operativo e database. Queste informazioni possono essere correlate ai carichi dei database per agevolare l'individuazione e l'analisi di problemi legati alle prestazioni.

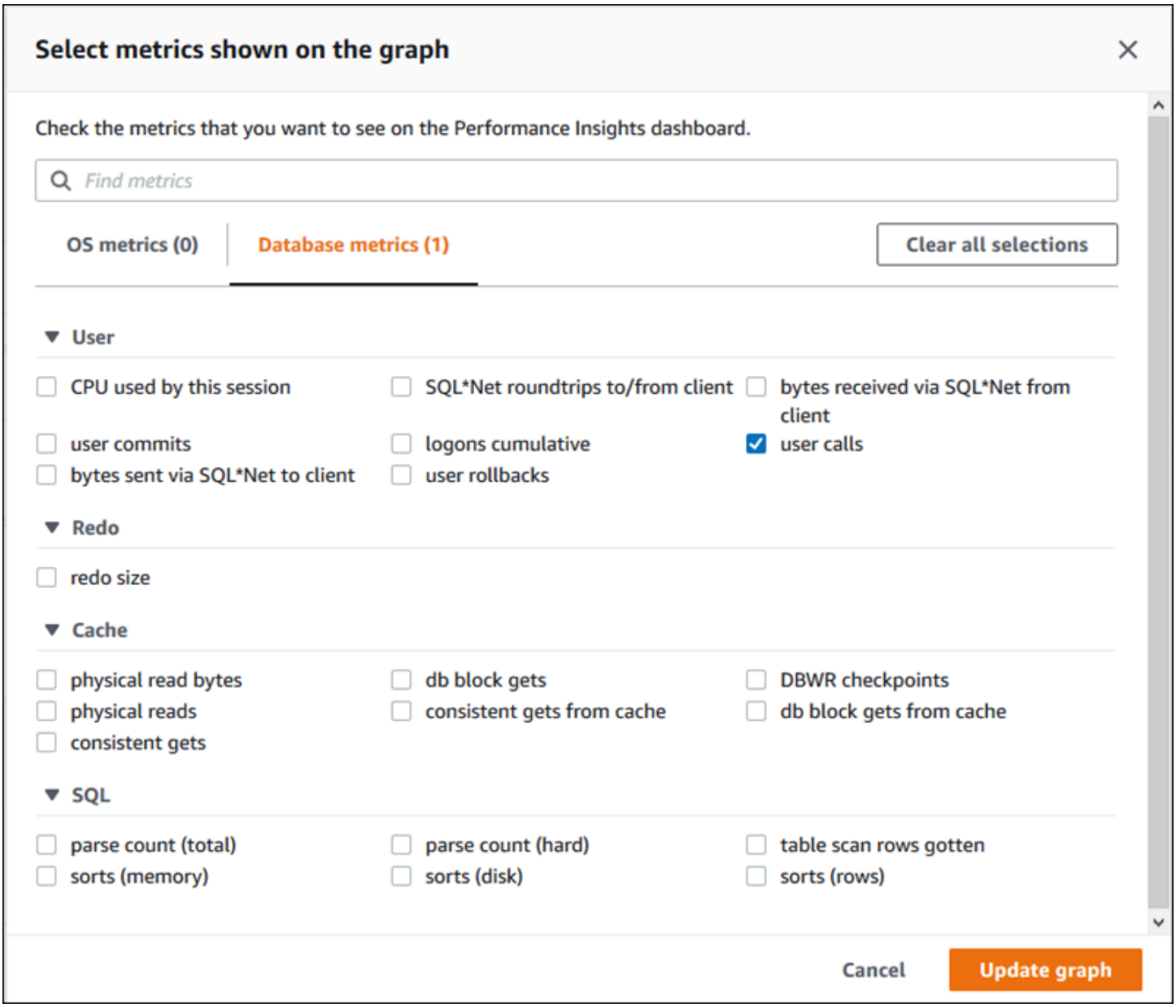
Il grafico Counter Metrics (Parametri contatore) visualizza i dati per i contatori delle prestazioni. I parametri predefiniti dipendono dal motore DB:

- Aurora MySQL– `db.SQL.Innodb_rows_read.avg`
- Aurora PostgreSQL – `db.Transactions.xact_commit.avg`



Per modificare i contatori delle prestazioni, scegli Manage Metrics (Gestisci parametri). È possibile selezionare più parametri del sistema operativo o metriche del database, come mostrato nello

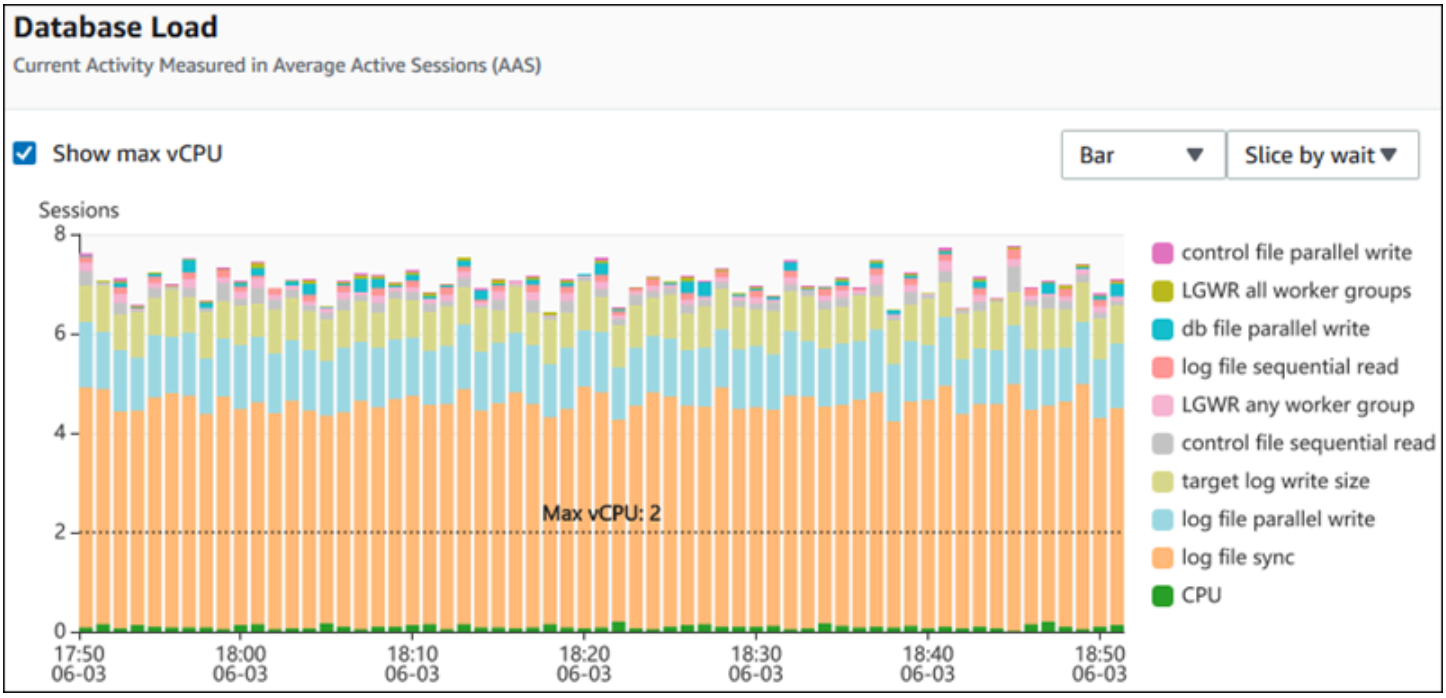
screenshot seguente. Per visualizzare i dettagli relativi a qualsiasi metrica, passare il mouse sul nome della metrica.



Per le descrizioni dei parametri contatore che è possibile aggiungere per ciascun motore database, consultare Metriche contatore di Performance Insights.

Grafico di carico database

Il grafico Database load (Carico database) mostra le differenze dell'attività del database in base alla capacità dell'istanza database, rappresentate dalla riga Max vCPU (vCPU massima). Per impostazione predefinita, il grafico a linee in pila rappresenta il carico DB come sessioni attive medie per unità di tempo. Il carico DB viene suddiviso (raggruppato) in base agli stati di attesa.

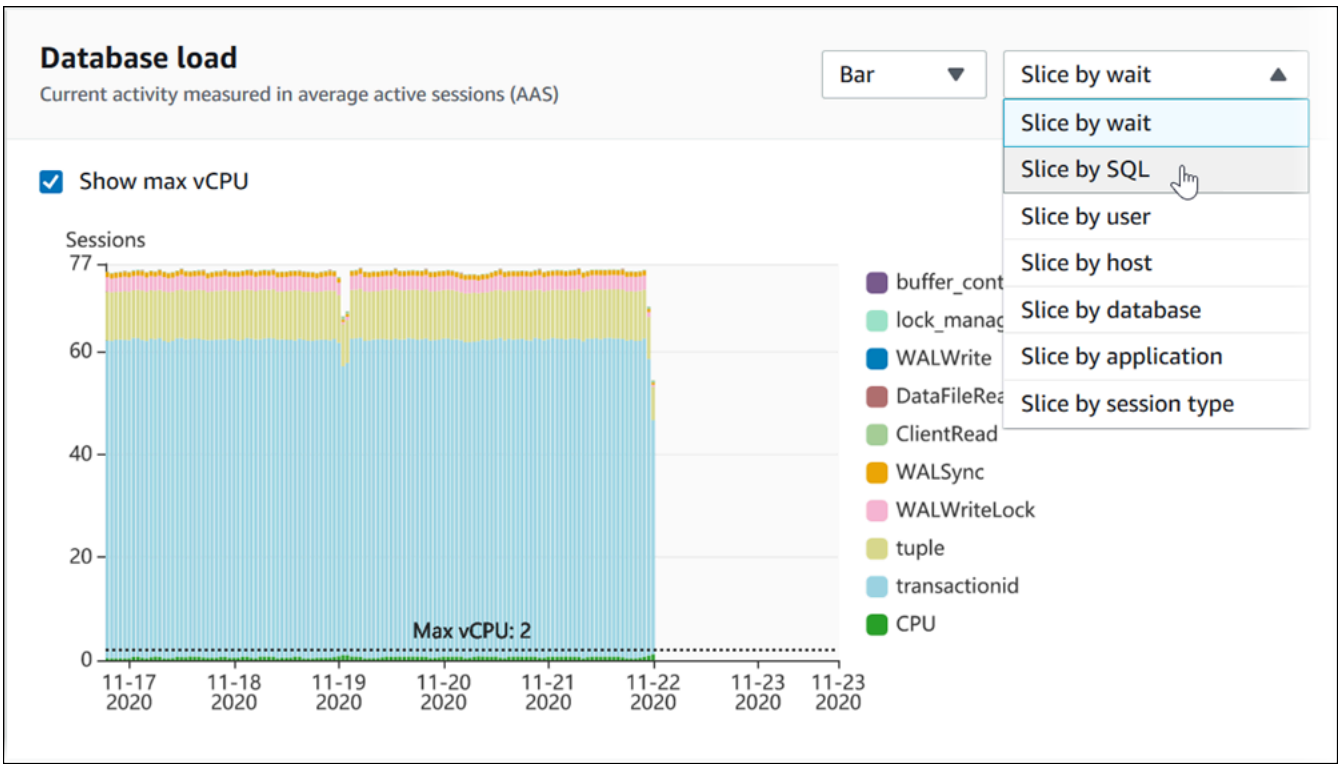


Carico del database suddiviso per dimensioni

È possibile scegliere di visualizzare il carico sotto forma di sessioni attive raggruppate in base alle dimensioni supportate. La tabella seguente mostra le dimensioni supportate per i diversi motori.

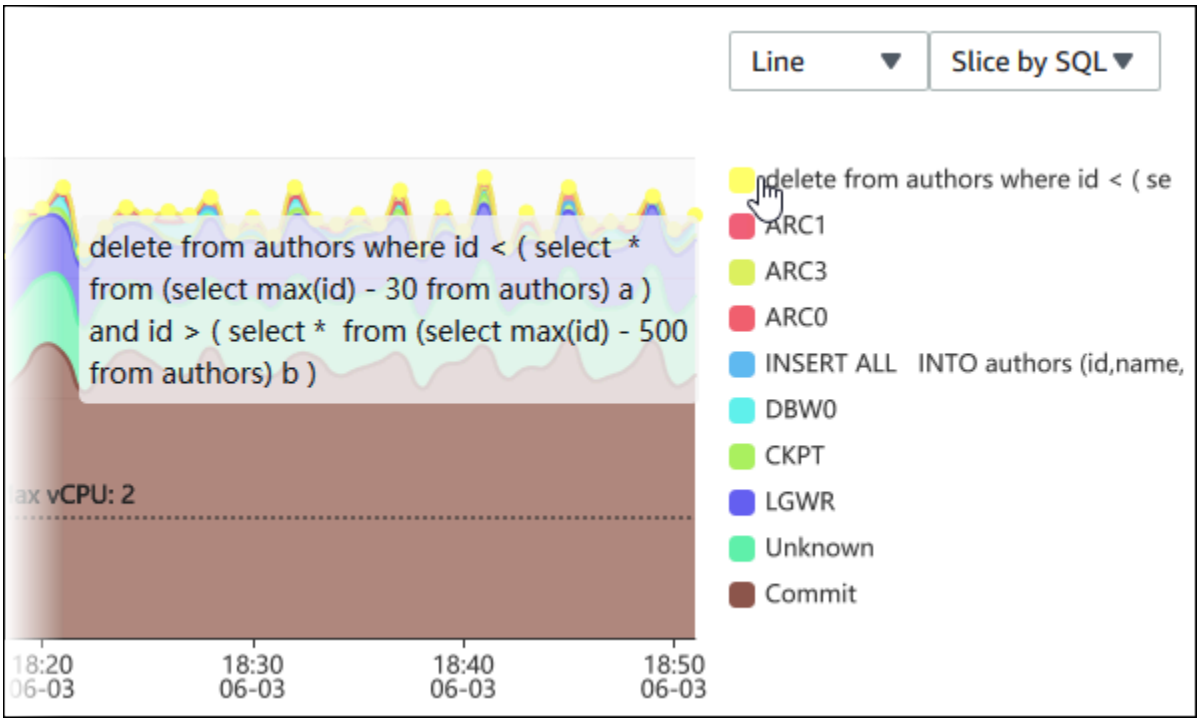
Dimensione	Aurora PostgreSQL	Aurora MySQL
Host	Sì	Sì
SQL	Sì	Sì
Utente	Sì	Sì
Stati di attesa	Sì	Sì
Applicazione	Sì	No
Database	Sì	Sì
Tipo di sessione	Sì	No

L'immagine seguente mostra le dimensioni di un'istanza database PostgreSQL.



Dettagli del carico DB per un elemento della dimensione

Per visualizzare i dettagli su un elemento del carico del database all'interno di una dimensione, passa il mouse sul nome dell'elemento. L'immagine seguente mostra i dettagli di un'istruzione SQL.



Per visualizzare i dettagli relativi a qualsiasi elemento per il periodo di tempo selezionato nella legenda, passa il mouse su tale elemento.

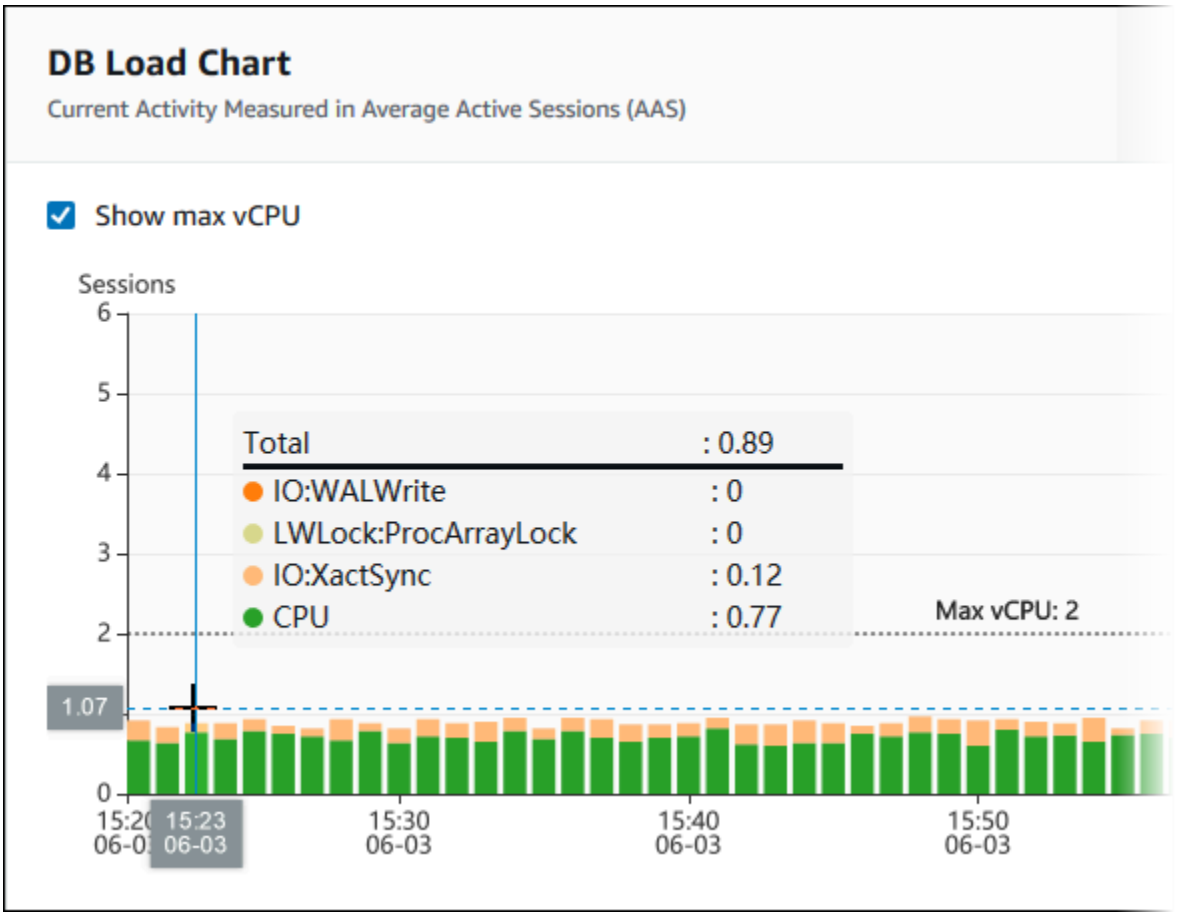
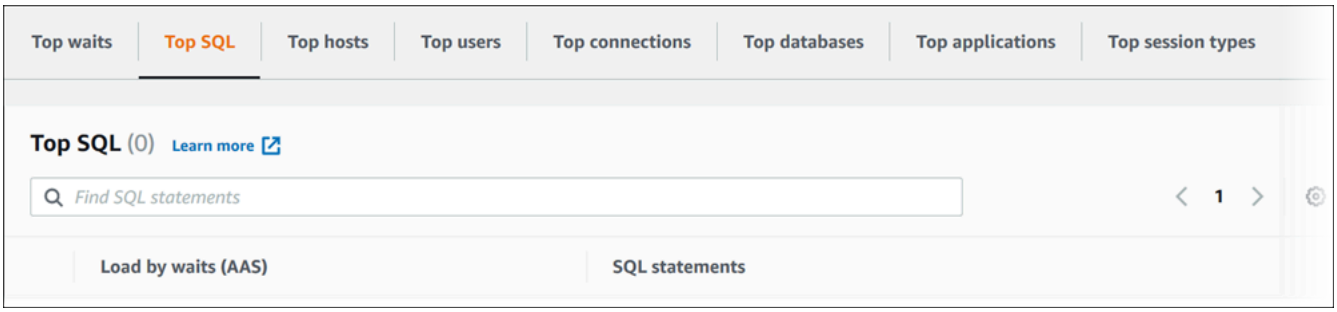


Tabella dimensioni superiori

La tabella delle dimensioni superiori seziona il carico DB in diverse dimensioni. Una dimensione è una categoria o una suddivisione per le diverse caratteristiche del carico del database. Se la dimensione è SQL, Top SQL (Prime istruzioni SQL) mostra le istruzioni SQL che contribuiscono maggiormente al carico DB.



Scegli una delle seguenti schede di dimensione.

Scheda	Descrizione	Motori supportati
Prime istruzioni SQL	Le istruzioni SQL correntem ente in esecuzione	Tutti
Principali stati d’attesa	L’evento per il quale il back-end del database è in attesa	Tutti
Host principali	Il nome host del client connesso	Tutti
Utenti principali	L’utente collegato al database	Tutti
Applicazioni principali	Il nome dell’applicazione connessa al database	Solo Aurora PostgreSQL
Tipi di sessione principali	Il tipo di sessione corrente	Solo Aurora PostgreSQL

Per informazioni sull’analisi delle query tramite la scheda Top SQL (Prime istruzioni SQL), consulta Panoramica della scheda Prime istruzioni SQL.

Accesso al pannello di controllo di Performance Insights

Amazon RDS pernisce una visualizzazione consolidata delle metriche di Performance Insights e CloudWatch nel pannello di controllo di Performance Insights.

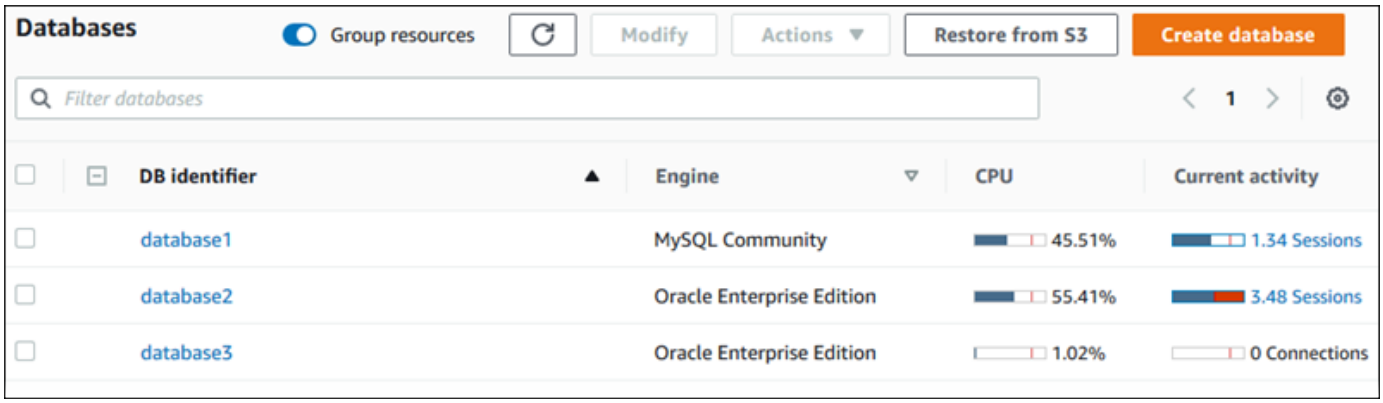
Per visualizzare il pannello di controllo di Performance Insights, procedi come indicato di seguito.

Per visualizzare il pannello di controllo di Performance Insights nella console di gestione AWS

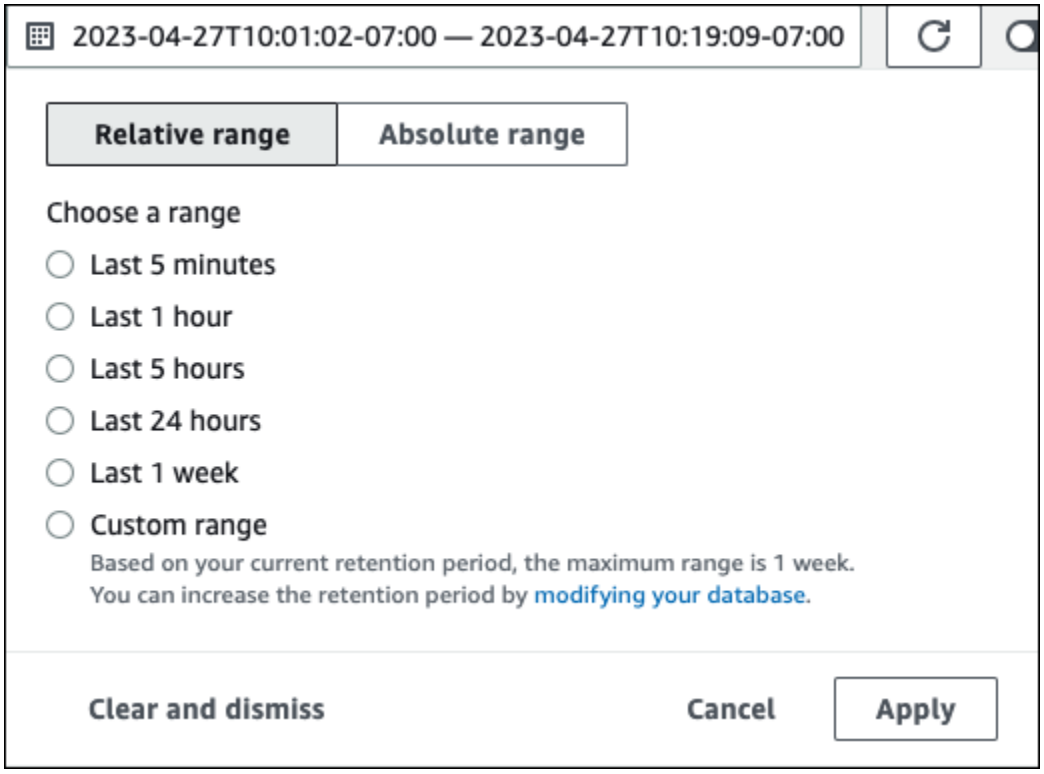
1. Apri la console Amazon RDS all’indirizzo https://console.aws.amazon.com/rds/.
2. Nel pannello di navigazione scegli Performance Insights.
3. Scegli un’istanza database.

   Per le istanze database con Approfondimenti sulle prestazioni attivato, è possibile accedere alla dashboard Approfondimenti sulle prestazioni anche scegliendo la voce Sessioni nell’elenco delle istanze database. In Attività corrente la voce Sessioni mostra il carico del database in sessioni attive medie negli ultimi cinque minuti. Il grafico mostra graficamente il carico: Quando la barra è vuota, l’istanza database è inattiva. Con l’aumentare del carico, la barra si riempie ed è di colore

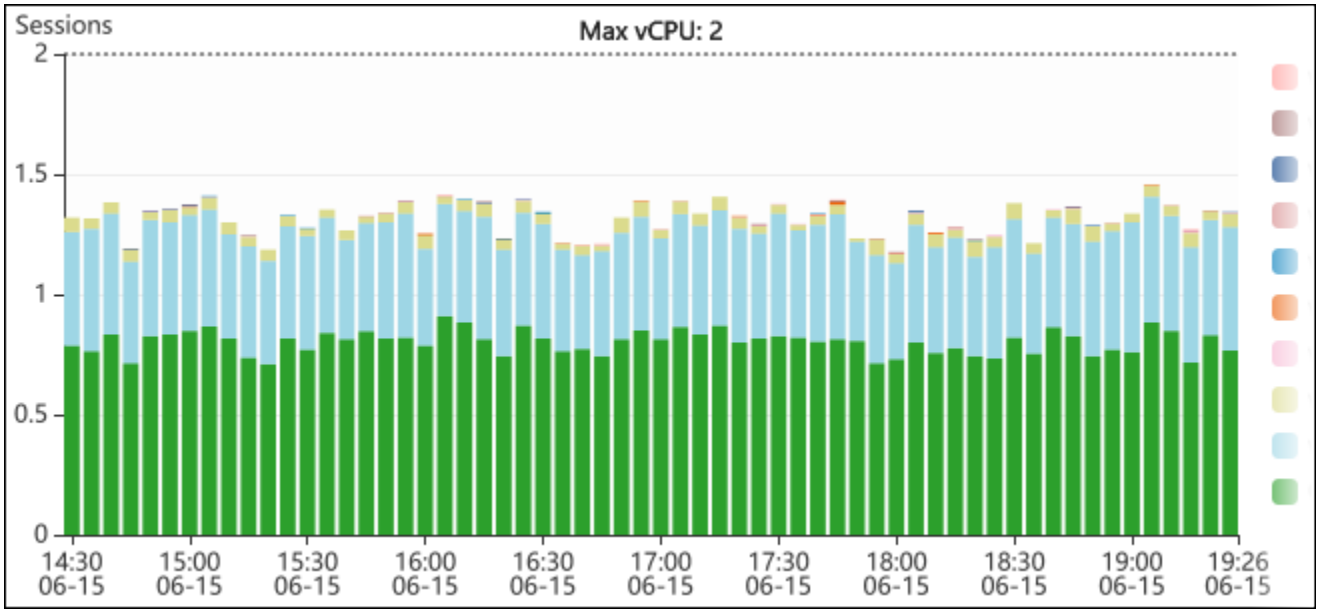
blu. Quando il carico supera il numero di CPU virtuali (vCPU) nella classe di istanza database, la barra diventa rossa, a indicare un possibile collo di bottiglia.



4. (Facoltativo) Scegliere la data o l'intervallo di ore in alto a destra e specificare un intervallo di tempo relativo o assoluto diverso. È ora possibile specificare un periodo di tempo e generare un report di analisi delle prestazioni del database. Il report fornisce le informazioni e i suggerimenti identificati. Per ulteriori informazioni, consulta Creazione di un report di analisi delle prestazioni in Approfondimenti sulle prestazioni.

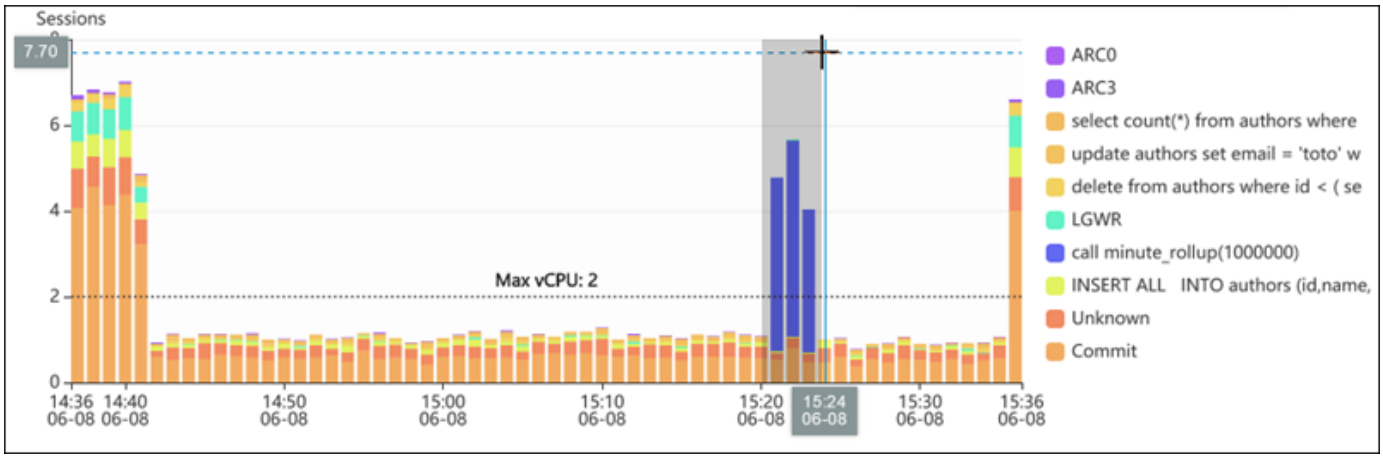


Nella schermata seguente, l'intervallo di caricamento DB è di 5 ore.

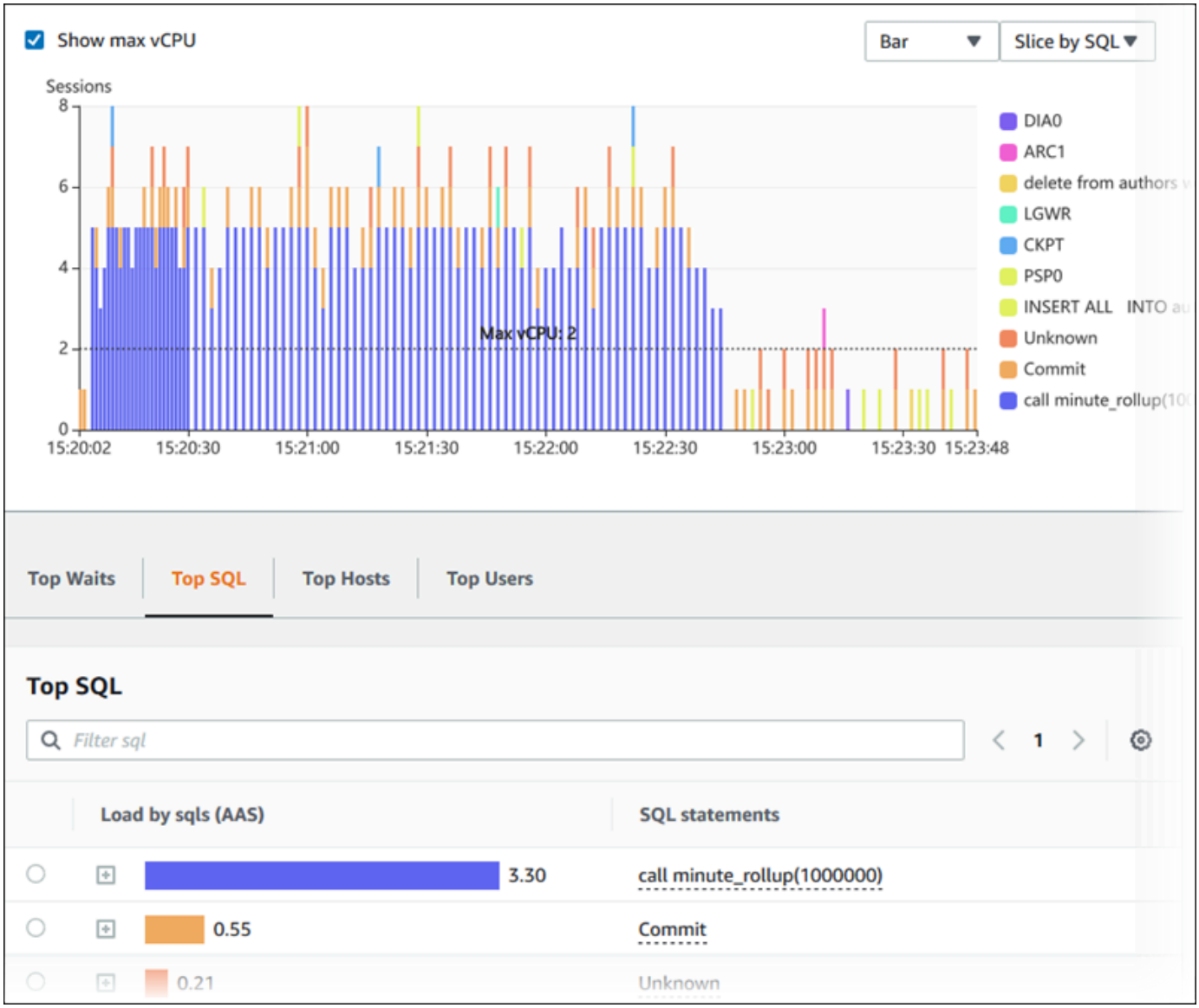


5. (Facoltativo) Per ingrandire una parte del grafico di carico del database, scegli l'ora di inizio e trascina fino alla fine del periodo di tempo che ti interessa.

   L'area selezionata viene evidenziata nel grafico del carico del database.



   Quando rilasci il mouse, la parte selezionata del grafico del carico del database si ingrandisce nella regione AWS selezionata e la tabella Top dimensions (Dimensioni principali) viene ricalcolata.



6. (Facoltativo) Per aggiornare automaticamente i dati, selezionare Aggiornamento automatico.

Il pannello di controllo di Performance Insights si aggiorna automaticamente in base ai nuovi dati. La frequenza di aggiornamento dipende dalla quantità di dati visualizzati:

- Se scegli 5 minuti, l'aggiornamento avviene ogni 10 secondi.
- Se scegli 1 ora, l'aggiornamento avviene ogni 5 minuti.
- Se scegli 5 ore, l'aggiornamento avviene ogni 5 minuti.
- Se scegli 24 ore, l'aggiornamento avviene ogni 30 minuti.
- Se scegli 1 settimana, l'aggiornamento avviene ogni giorno.

- Se scegli 1 mese, l’aggiornamento avviene ogni giorno.

Analisi del carico del database per eventi di attesa

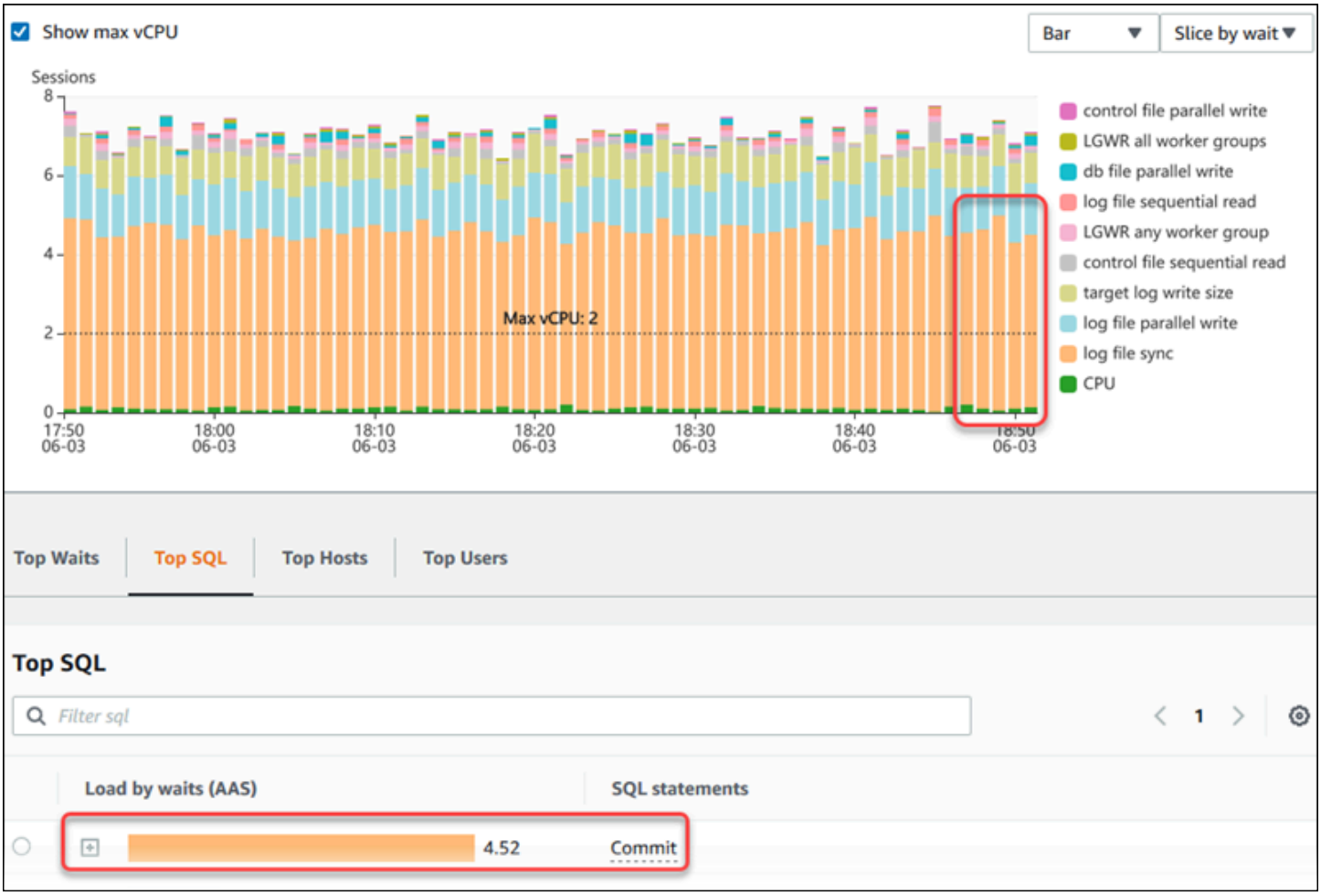
Se il grafico Database load (Carico database) mostra un collo di bottiglia, puoi identificare l’origine del carico. A questo scopo, osserva la tabella Top Load Items (Elementi con carico) sotto la tabella Database load (Carico database). Scegli uno specifico elemento, come una query SQL o un utente, ed effettua il drill-down di tale elemento per vedere i relativi dettagli.

Il carico del database raggruppato per attese e principali query SQL è la visualizzazione predefinita del pannello di controllo di Performance Insights. Questa combinazione offre di norma il maggior numero di informazioni sui problemi di prestazioni. Il carico del database raggruppato in base alle attese mostra la presenza di eventuali colli di bottiglia nel database relativamente alle risorse o alla simultaneità. In questo caso, la scheda SQL della tabella Top Load Items (Elementi con carico massimo) mostra quali query fanno aumentare il carico.

Il flusso di lavoro tipico per diagnosticare problemi di performance è il seguente:

1. Esaminare il grafico Database load (Carico database) per determinare se sono presenti eventi imprevisti di superamento della riga Max CPU (CPU max) da parte del carico del database.
2. Se sono presenti, osservare il grafico Database load (Carico database) e individuare lo stato o gli stati di attesa che sono i principali responsabili.
3. Identificare le query digest che provocano il carico individuando quali delle query della scheda SQL nella tabella Top Load Items (Elementi con carico massimo) contribuiscono maggiormente agli stati di attesa. È possibile identificarle attraverso la colonna DB Load by Waits (Carico del database in base alle attese).
4. Scegliere una delle query digest nella scheda SQL per espanderla e osservare le query figlio da cui è composta.

Ad esempio, nel dashboard seguente, la sincronizzazione file di registro attende account per la maggior parte del carico DB. Anche l’attesa di tutti i gruppi di lavoro LGWR è elevata. Il grafico Top SQL mostra ciò che causa le attese di sincronizzazione del file di registro: istruzioni `COMMIT` frequenti. In questo caso, il commit meno frequentemente ridurrà il carico del DB.



Analisi delle prestazioni del database per un periodo di tempo

Analizza le prestazioni del database con l'analisi on demand creando un report di analisi delle prestazioni per un periodo di tempo. I report di analisi delle prestazioni consentono di individuare problemi di prestazioni, come colli di bottiglia delle risorse o modifiche in una query nell'istanza database. Il pannello di controllo di Performance Insights consente di selezionare un periodo di tempo e creare un report di analisi delle prestazioni. Puoi anche aggiungere uno o più tag al report.

Per utilizzare questa funzionalità, devi utilizzare il periodo di conservazione del piano a pagamento. Per ulteriori informazioni, consulta Prezzi e conservazione dei dati per Performance Insights

Il report è disponibile nella scheda Report di analisi delle prestazioni - nuovi per la selezione e la visualizzazione. Il report contiene informazioni dettagliate, parametri correlati e suggerimenti per risolvere il problema relativo alle prestazioni. Il report è disponibile per la visualizzazione per tutta la durata del periodo di conservazione di Performance Insights.

Il report viene eliminato se l'ora di inizio del periodo di analisi del report è esterna al periodo di conservazione. È anche possibile eliminare il report prima della fine del periodo di conservazione.

Per individuare i problemi di prestazioni e generare il report di analisi per l'istanza database, è necessario attivare Performance Insights. Per ulteriori informazioni sull'attivazione di Performance Insights, consulta Attivazione e disattivazione di Approfondimenti sulle prestazioni per Aurora.

Per informazioni sul supporto di Regioni, motori di database e classi di istanza per questa funzionalità, consulta Supporto di classe di istanza, regione e motore di database Amazon Aurora per funzionalità Performance Insights

Nelle sezioni seguenti è possibile creare, visualizzare, aggiungere tag ed eliminare un report di analisi delle prestazioni.

Argomenti

- Creazione di un report di analisi delle prestazioni in Approfondimenti sulle prestazioni
- Visualizzazione di un report di analisi delle prestazioni in Approfondimenti sulle prestazioni
- Aggiunta di tag a un report di analisi delle prestazioni in Approfondimenti sulle prestazioni
- Eliminazione di un report di analisi delle prestazioni in Approfondimenti sulle prestazioni

Creazione di un report di analisi delle prestazioni in Approfondimenti sulle prestazioni

È possibile creare un report di analisi delle prestazioni per un periodo specifico nella pannello di controllo di Performance Insights. È possibile selezionare un periodo di tempo e aggiungere uno o più tag al report di analisi.

Il periodo di analisi può variare da 5 minuti a 6 giorni. Occorre disporre di almeno 24 ore di dati di prestazioni prima dell'ora di inizio dell'analisi.

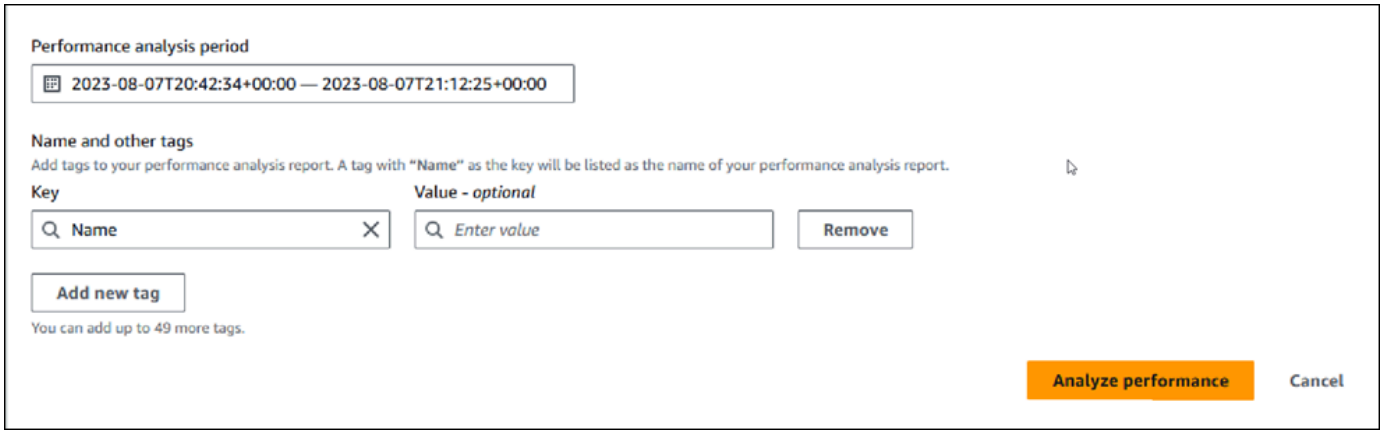
Per informazioni sul supporto di Regioni, motori di database e classi di istanza per questa funzionalità, consulta Supporto di classe di istanza, regione e motore di database Amazon Aurora per funzionalità Performance Insights

Per creare un report di analisi delle prestazioni per un periodo di tempo

1. Apri la console di Amazon RDS all'indirizzo https://console.aws.amazon.com/rds/.
2. Nel pannello di navigazione scegli Performance Insights.
3. Scegli un'istanza database.

4. Scegli Analizza le prestazioni nella sezione Carico del database nella dashboard di Approfondimenti sulle prestazioni.

   Vengono visualizzati i campi per impostare il periodo di tempo e aggiungere uno o più tag al report di analisi delle prestazioni.

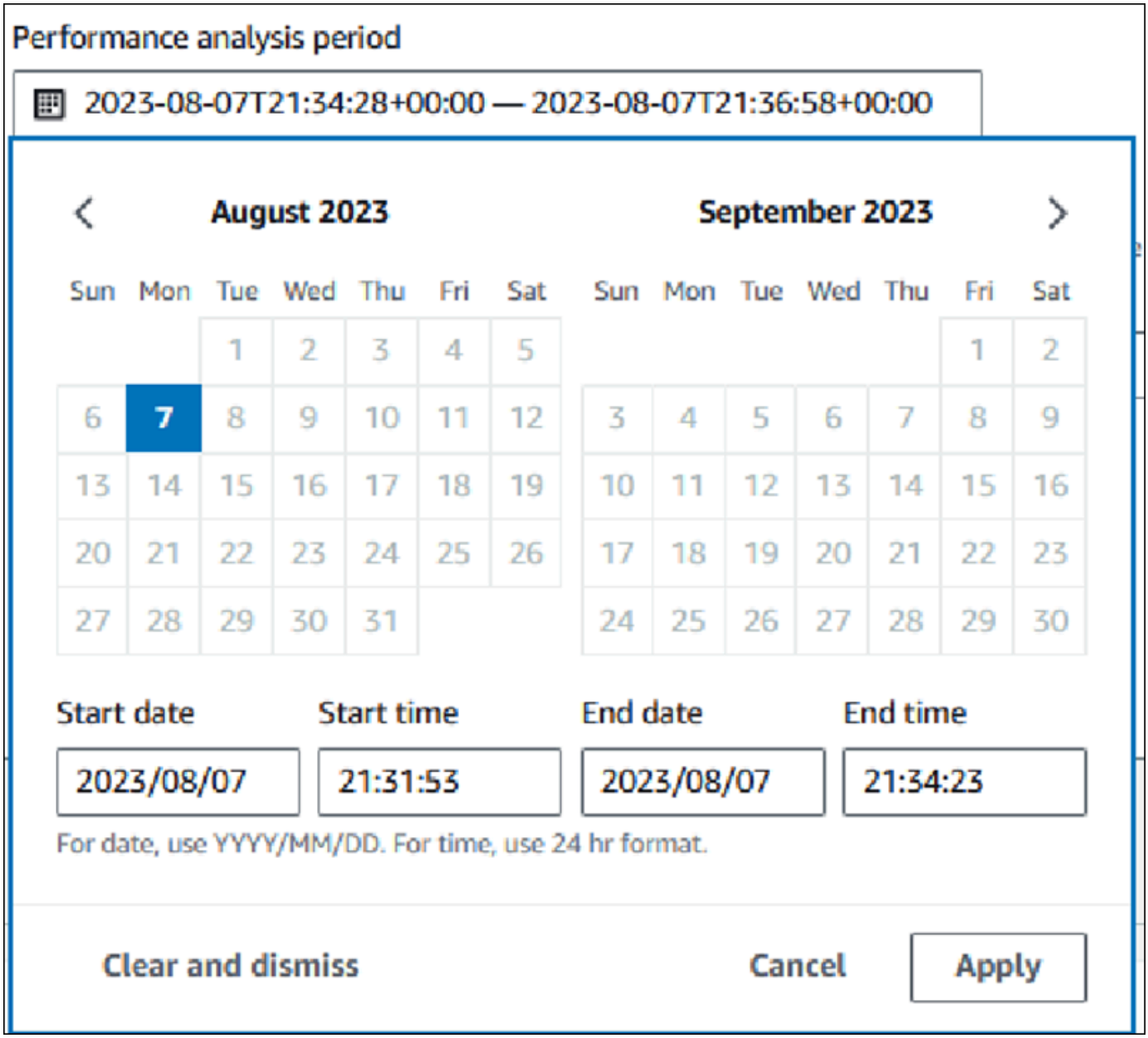


5. Scegli il periodo di tempo. Se imposti un periodo di tempo nel campo Intervallo relativo o Intervallo assoluto nell'angolo in alto a destra, puoi inserire o selezionare la data e l'ora del report di analisi solo entro questo periodo di tempo. Se selezioni il periodo di analisi al di fuori di questo periodo di tempo, viene visualizzato un messaggio di errore.

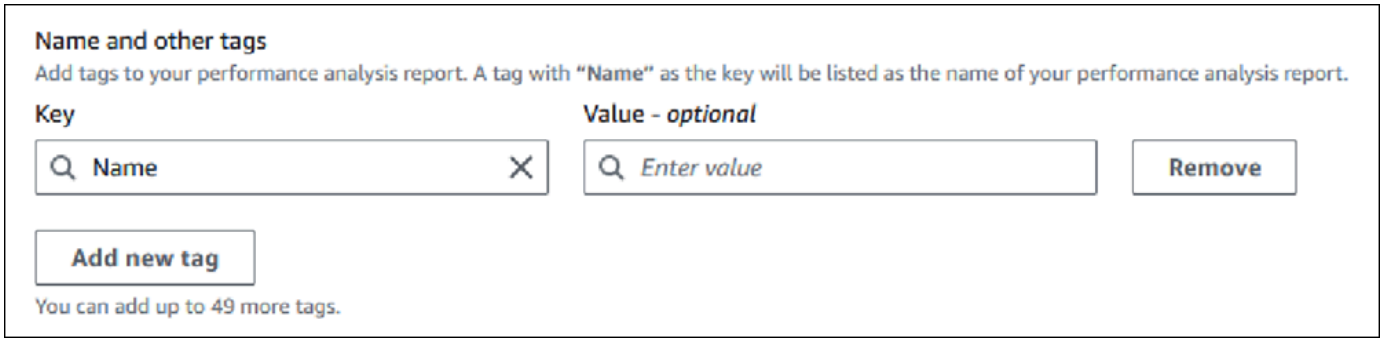
   Per impostare il periodo di tempo, puoi effettuare una delle seguenti operazioni:

   - Premi e trascina uno qualsiasi dei dispositivi di scorrimento sul grafico del carico del database.

     Nella casella Periodo di analisi delle prestazioni viene visualizzato il periodo di tempo selezionato e il grafico del carico del database evidenzia il periodo di tempo selezionato.
   - Scegli Data di inizio, Ora di inizio, Data di fine e Ora di fine nella casella Periodo di analisi delle prestazioni.



6. (Facoltativo) Inserisci Chiave e Valore-opzionale per aggiungere un tag per il report.



7. Scegli Analizza le prestazioni.

   Un banner mostra un messaggio a prescindere dall'esito della generazione del report. Il messaggio fornisce anche il collegamento per visualizzare il report.

   L'esempio seguente mostra il banner con il messaggio di creazione del report completata.

   Il report è disponibile per la visualizzazione nella scheda Report di analisi delle prestazioni - nuovi.

Puoi creare un report di analisi delle prestazioni utilizzando la AWS CLI. Per un esempio su come creare un report utilizzando la AWS CLI, consulta Creazione di un report di analisi delle prestazioni per un periodo di tempo.

Visualizzazione di un report di analisi delle prestazioni in Approfondimenti sulle prestazioni

Nella scheda Report di analisi delle prestazioni - nuovi vengono elencati tutti i report creati per l'istanza database. Per ogni report viene visualizzato quanto segue:

- ID: identificatore univoco del report.
- Nome: chiave di tag aggiunta al report.
- Ora di creazione del report: ora in cui il report è stato creato.
- Ora di inizio dell'analisi: ora di inizio dell'analisi nel report.
- Ora di fine dell'analisi: ora di fine dell'analisi nel report.

Per visualizzare un report di analisi delle prestazioni

1. Accedi alla Console di gestione AWS e apri la console Amazon RDS all'indirizzo https://console.aws.amazon.com/rds/.
2. Nel pannello di navigazione scegli Performance Insights.
3. Scegli un'istanza database per la quale desideri visualizzare il report di analisi.
4. Scorri verso il basso e scegli la scheda Report di analisi delle prestazioni - nuovo nella dashboard di Approfondimenti sulle prestazioni.

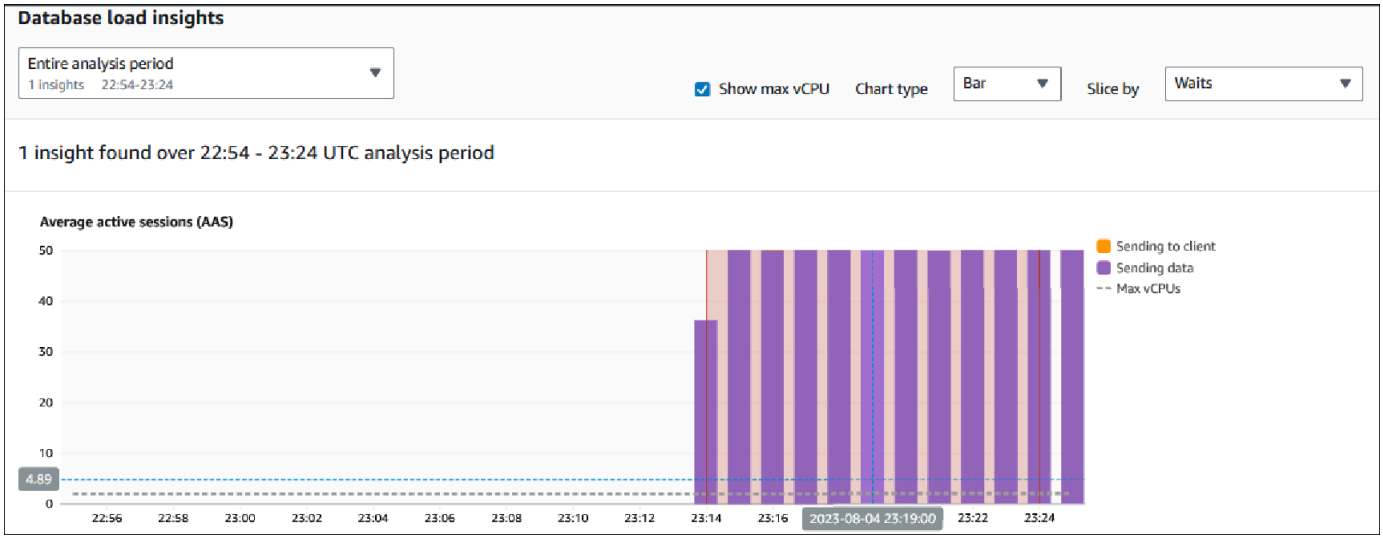
   Vengono visualizzati tutti i report di analisi per i diversi periodi di tempo.

5. Scegli ID del report che desideri visualizzare.

   Il grafico del carico del database mostra l'intero periodo di analisi per impostazione predefinita se vengono identificati più approfondimenti. Se il report ha identificato un approfondimento, il grafico del carico del database visualizza l'approfondimento per impostazione predefinita.

   Nel pannello di controllo vengono elencati anche i tag per il report nella sezione Tag.

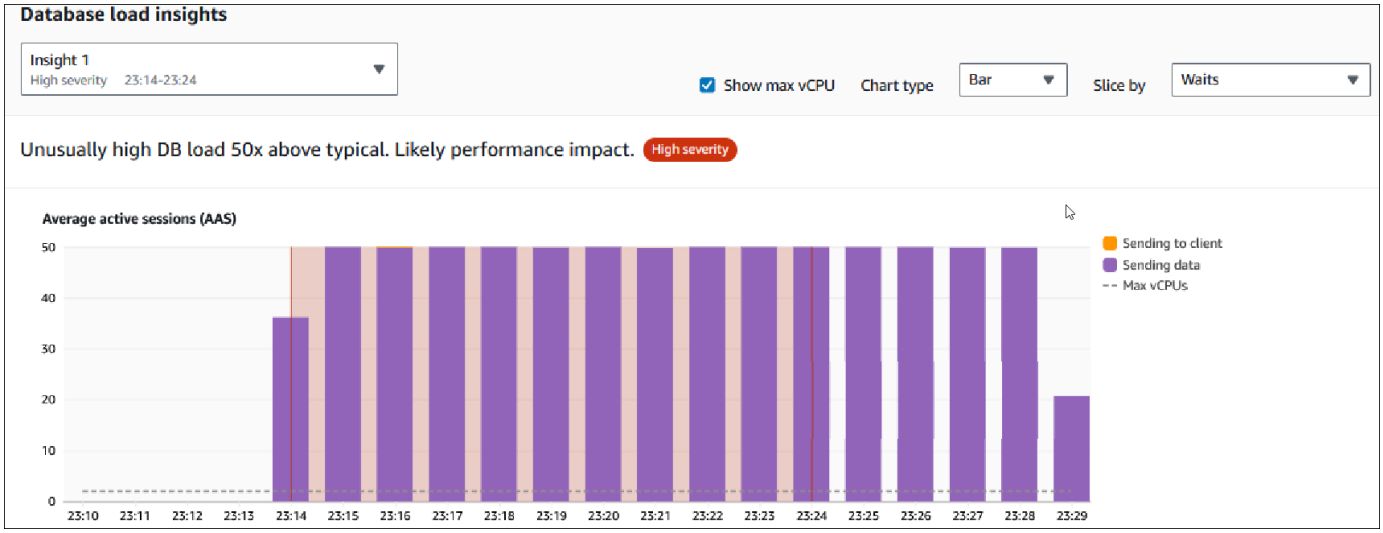
   L'esempio seguente mostra l'intero periodo di analisi per il report.

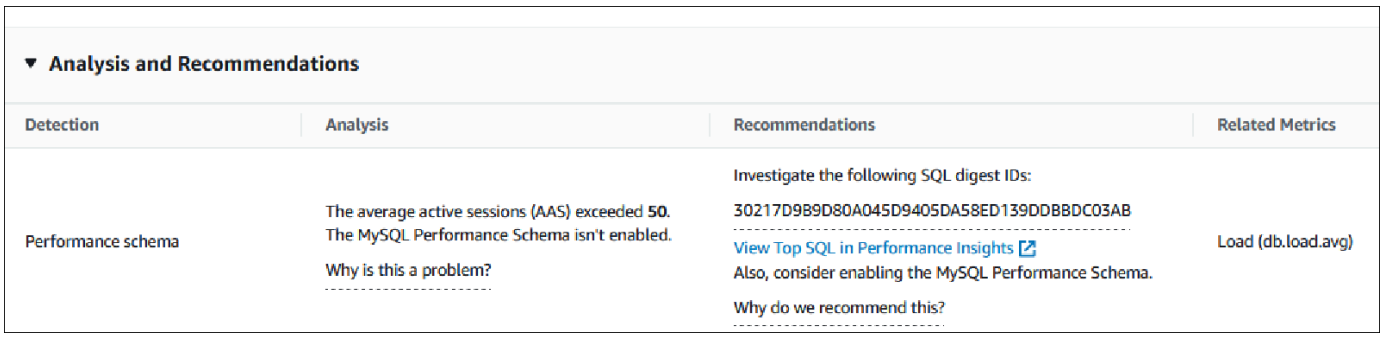


6. Scegli l'approfondimento nell'elenco Informazioni dettagliate sul carico del database che desideri visualizzare se nel report vengono identificati più approfondimenti.

   Il pannello di controllo mostra il messaggio di approfondimento, il grafico del carico del database evidenziando il periodo di tempo dell'approfondimento, l'analisi e i suggerimenti e l'elenco dei tag del report.

   Nell'esempio seguente viene mostrato l'approfondimento del carico del database nel report.





Aggiunta di tag a un report di analisi delle prestazioni in Approfondimenti sulle prestazioni

È possibile aggiungere un tag quando si crea o visualizza un report. È possibile aggiungere fino a 50 tag per un report.

Per aggiungere i tag sono richieste autorizzazioni. Per ulteriori informazioni sulle policy di accesso per Performance Insights, consulta Configurazione delle policy di accesso per Performance Insights

Per aggiungere uno o più tag durante la creazione di un report, consulta il passaggio 6 della procedura Creazione di un report di analisi delle prestazioni in Approfondimenti sulle prestazioni.

Per aggiungere uno o più tag durante la visualizzazione di un report

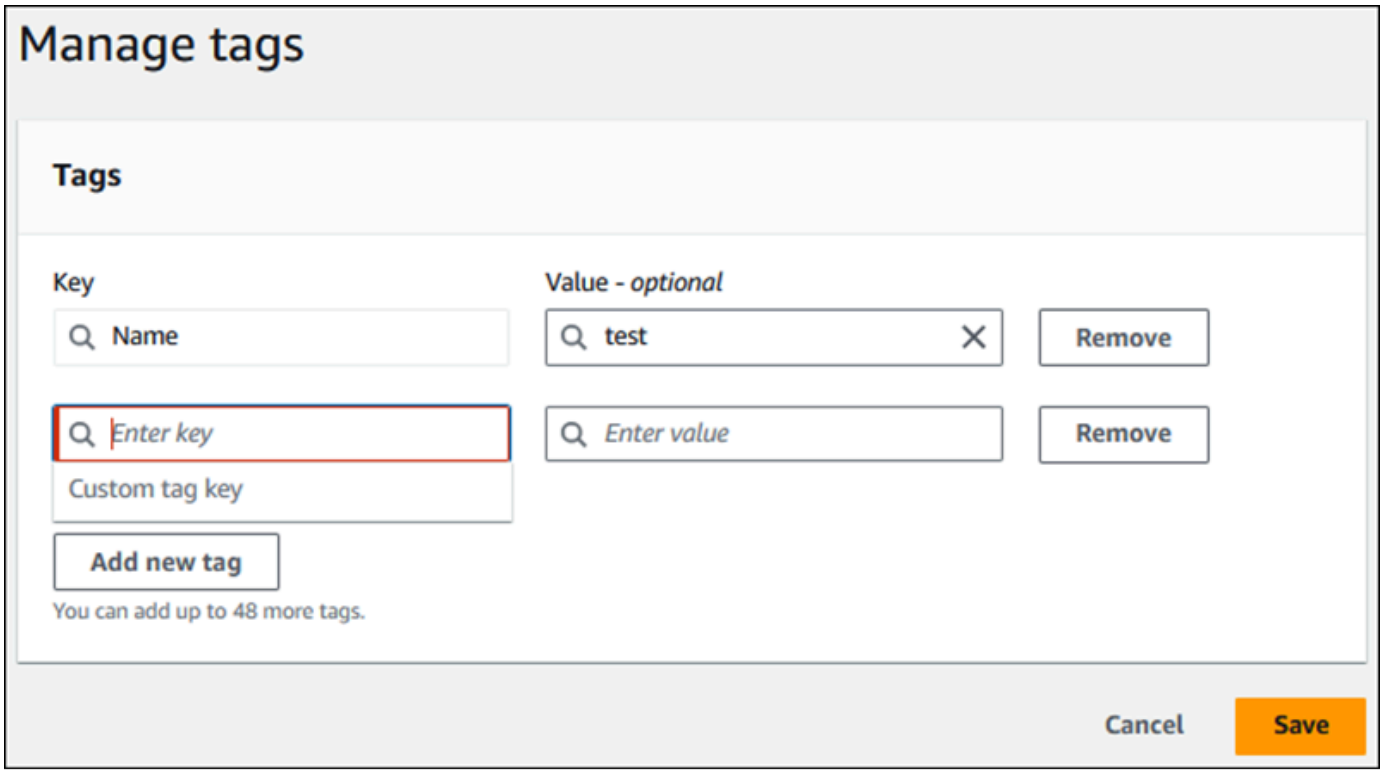
1. Apri la console di Amazon RDS all'indirizzo https://console.aws.amazon.com/rds/.
2. Nel pannello di navigazione scegli Performance Insights.
3. Scegli un'istanza database.

Viene visualizzato il pannello di controllo di Approfondimenti sulle prestazioni per l'istanza database.

4. Scorri verso il basso e scegli la scheda Report di analisi delle prestazioni - nuovi.
5. Scegli il report per il quale desideri aggiungere i tag.

   Il pannello di controllo visualizza il report.
6. Scorri verso il basso fino a Tag e scegli Gestisci i tag.
7. Scegli Aggiungi nuovo tag.
8. Immetti Chiave e Valore-facoltativo e scegli Aggiungi nuovo tag.

   Nell'esempio seguente viene fornita la possibilità di aggiungere un nuovo tag per il report selezionato.



   Viene creato un nuovo tag per il report.

   L'elenco dei tag per il report viene visualizzato nella sezione Tag sul pannello di controllo. Se desideri rimuovere un tag dal report, scegli Rimuovi accanto al tag.

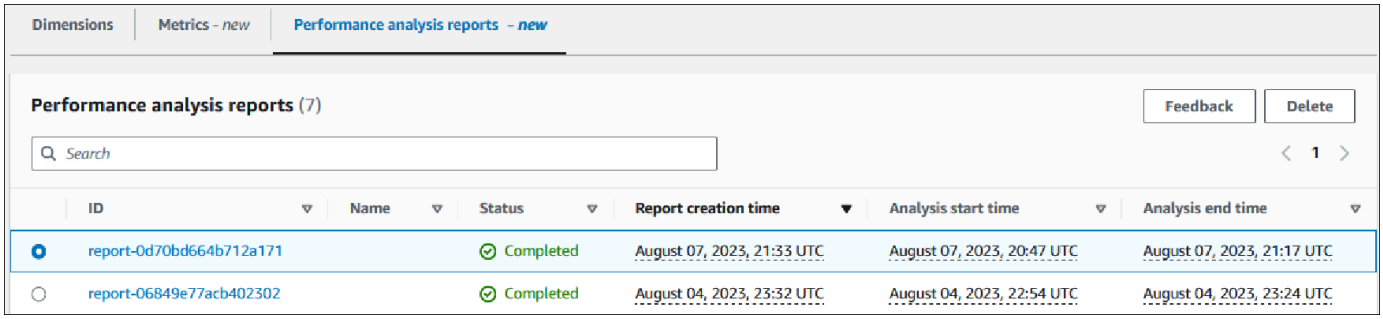
Eliminazione di un report di analisi delle prestazioni in Approfondimenti sulle prestazioni

È possibile eliminare un report dall'elenco dei report visualizzato nella scheda Report di analisi delle prestazioni o durante la visualizzazione di un report.

Per eliminare un report

1. Apri la console di Amazon RDS all'indirizzo https://console.aws.amazon.com/rds/.
2. Nel pannello di navigazione scegli Performance Insights.
3. Scegli un'istanza database.

   Viene visualizzato il pannello di controllo di Approfondimenti sulle prestazioni per l'istanza database.
4. Scorri verso il basso e scegli la scheda Report di analisi delle prestazioni - nuovi.
5. Seleziona il report che desideri eliminare e scegli Elimina nell'angolo in alto a destra.



   Viene visualizzata una finestra di conferma. Il report viene eliminato dopo aver scelto l'opzione di conferma.
6. (Facoltativo) Scegli ID del report che desideri eliminare.

   Nella pagina del report, scegli Elimina nell'angolo in alto a destra.

   Viene visualizzata una finestra di conferma. Il report viene eliminato dopo aver scelto l'opzione di conferma.

Analisi delle query con la scheda SQL principale in Approfondimenti sulle prestazioni

Nel pannello di controllo di Amazon RDS Performance Insights è possibile trovare informazioni relative alle query in esecuzione e recenti nella scheda Top SQL (Prime istruzioni SQL) nella tabella Top dimensions (Dimensioni principali). Queste informazioni possono essere utilizzate per ottimizzare le query.

Argomenti

- Panoramica della scheda Prime istruzioni SQL
- Accesso a una maggiore quantità di testo SQL nel pannello di controllo di Performance Insights
- Visualizzazione delle statistiche SQL nel pannello di controllo di Performance Insights

Panoramica della scheda Prime istruzioni SQL

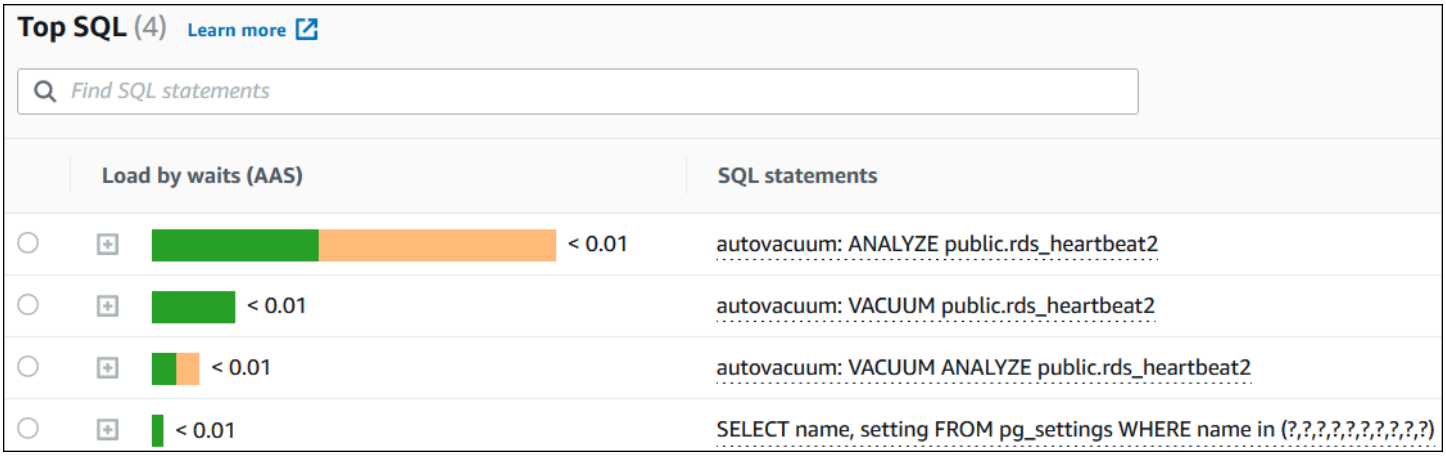
Per impostazione predefinita, la scheda Top SQL (Prime istruzioni SQL) mostra le 25 query che contribuiscono di più al carico del database. Per ottimizzare le query puoi analizzare le informazioni, ad esempio il testo della query e le statistiche SQL. È inoltre possibile scegliere le statistiche che desideri visualizzare nella scheda Top SQL (Prime istruzioni SQL).

Argomenti

- Testo SQL
- Statistiche SQL
- Caricamento per attesa (AAS)
- Visualizzazione delle informazioni SQL
- Scelta delle preferenze statistiche

Testo SQL

Per impostazione predefinita, ciascuna riga nella tabella Top SQL (Prime istruzioni SQL) mostra 500 byte di testo per ogni istruzione.

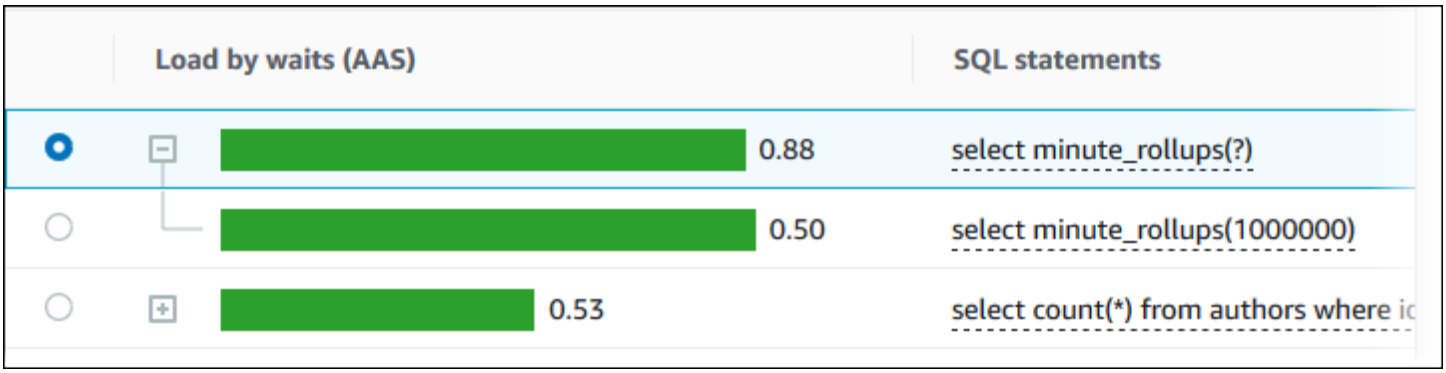


Per informazioni su come visualizzare più dei 500 byte di testo SQL di default, consulta Accesso a una maggiore quantità di testo SQL nel pannello di controllo di Performance Insights.

Un digest SQL è un composito di più query effettive strutturalmente simili ma potrebbero avere valori letterali diversi. Il digest sostituisce i valori codificati con un punto interrogativo. Ad esempio, un digest potrebbe essere `SELECT * FROM emp WHERE lname= ?`. Questo digest può includere le seguenti query figlio:

```
SELECT * FROM emp WHERE lname = 'Sanchez'
SELECT * FROM emp WHERE lname = 'Olagappan'
SELECT * FROM emp WHERE lname = 'Wu'
```

Per visualizzare le istruzioni SQL letterali in un digest, selezionare la query, quindi scegliere il simbolo più (+). Nell'esempio seguente, la query selezionata è un sunto.

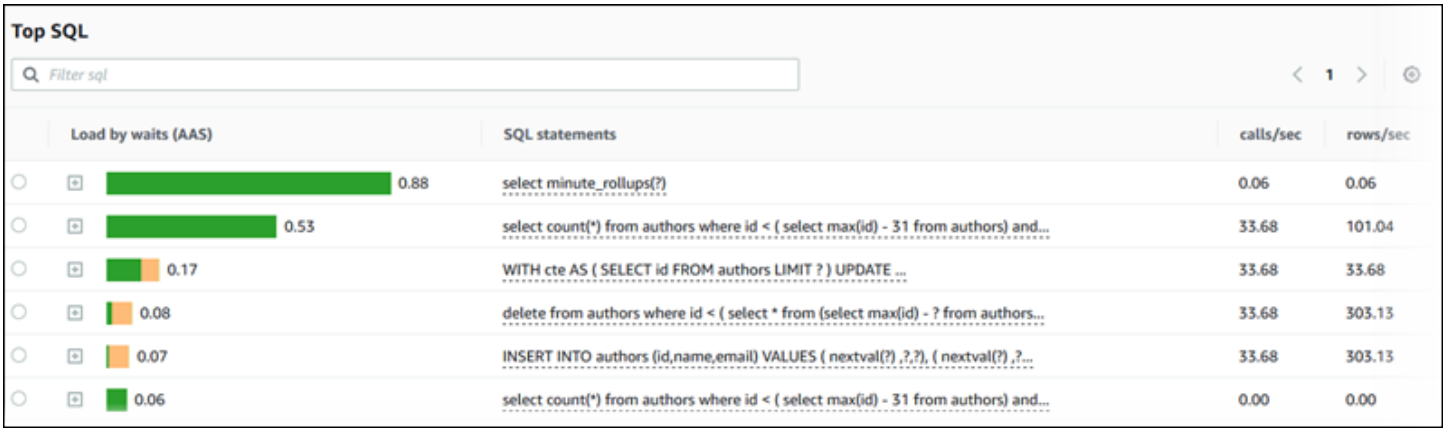


> **Note**
>
> Un sunto SQL raggruppa istruzioni SQL simili, ma non oscura le informazioni riservate.

Statistiche SQL

Statistiche SQL sono parametri relativi alle prestazioni relative alle query SQL. Ad esempio, Performance Insights potrebbe mostrare esecuzioni al secondo o righe elaborate al secondo. Performance Insights raccoglie statistiche solo per le query più comuni. In genere, queste query corrispondono alle prime query per carico mostrate nel dashboard di Performance Insights.

Tutte le righe della tabella Top SQL (Prime istruzioni SQL) mostra le statistiche rilevanti per l'istruzione SQL o il digest, come illustrato nell'esempio seguente.



Performance Insights può segnalare `0.00` e `-` (sconosciuto) per le statistiche SQL. Questa situazione si verifica nelle seguenti condizioni:

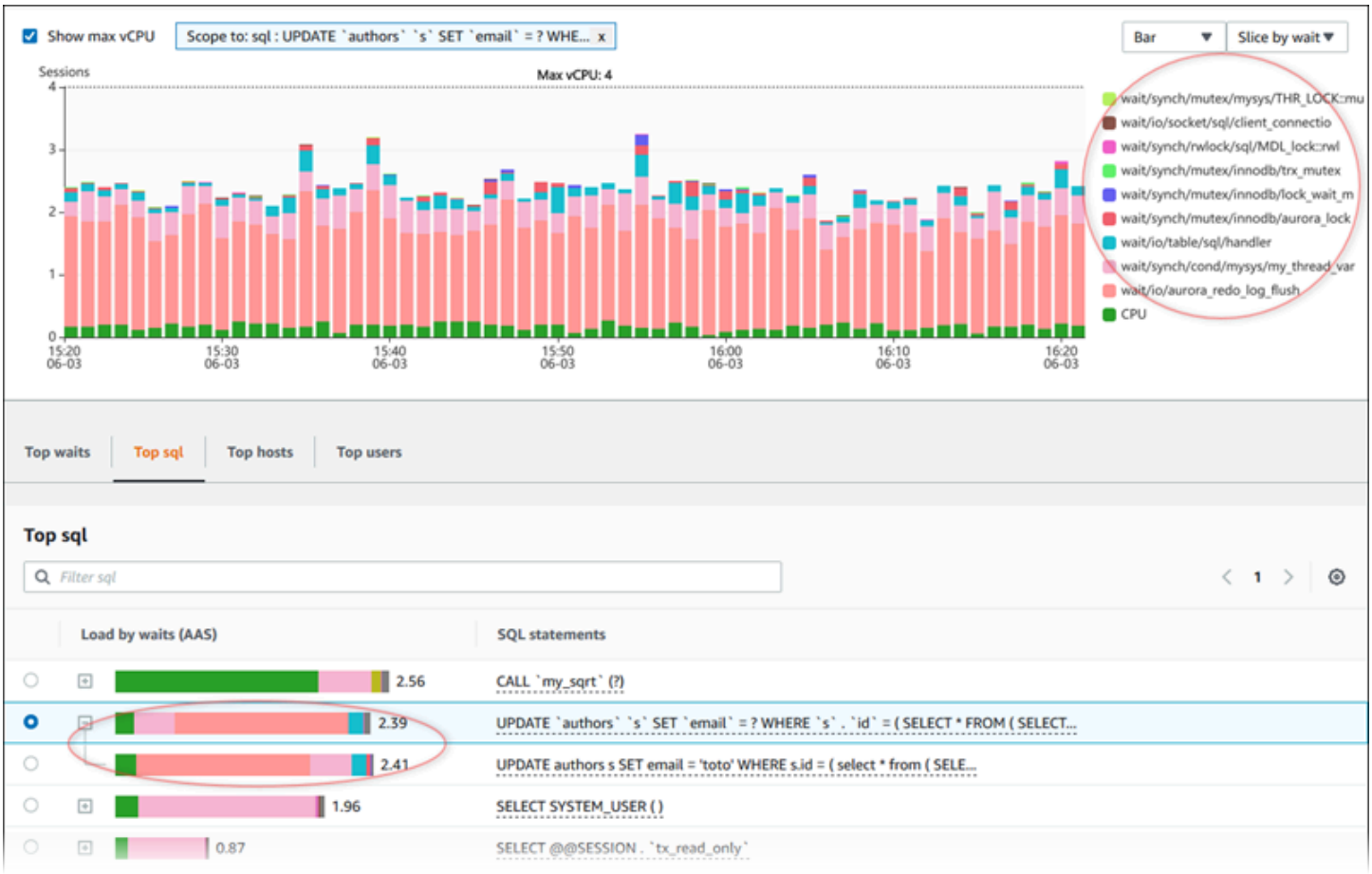
- Esiste un solo campione. Ad esempio, Performance Insights calcola i tassi di variazione per le query PostgreSQL di Aurora sulla basa di molteplici campioni della vista `pg_stat_statements`. Quando un carico di lavoro viene eseguito per un breve periodo, Performance Insights potrebbe raccogliere solo un campione, il che significa che non è in grado di calcolare un tasso di varaizione. Il valore sconosciuto è rappresentato da un trattino (`-`).
- Due campioni hanno gli stessi valori. Performance Insights non è in grado di calcolare un tasso di variazione perché non si è verificata alcuna variazione, quindi riporta il tasso come `0.00`.
- Un'istruzione SQL Aurora manca di un identificatore valido. PostgreSQL crea un identificatore per un'istruzione solo dopo la parsificazione e l'analisi. Pertanto, può esistere nelle strutture interne in memoria di PostgreSQL un'istruzione senza identificatore. Poiché Performance Insights esegue il campionamento delle strutture interne in memoria una volta al secondo, le query a bassa latenza potrebbero apparire solo in un singolo campione. Se l'identificatore della query non è disponibile per questo campione, Performance Insights non può associare questa istruzione alle relative statistiche. Il valore sconosciuto è rappresentato da un trattino (`-`).

Per una descrizione delle statistiche SQL per i motori Aurora, consulta Statistiche SQL per Performance Insights.

Caricamento per attesa (AAS)

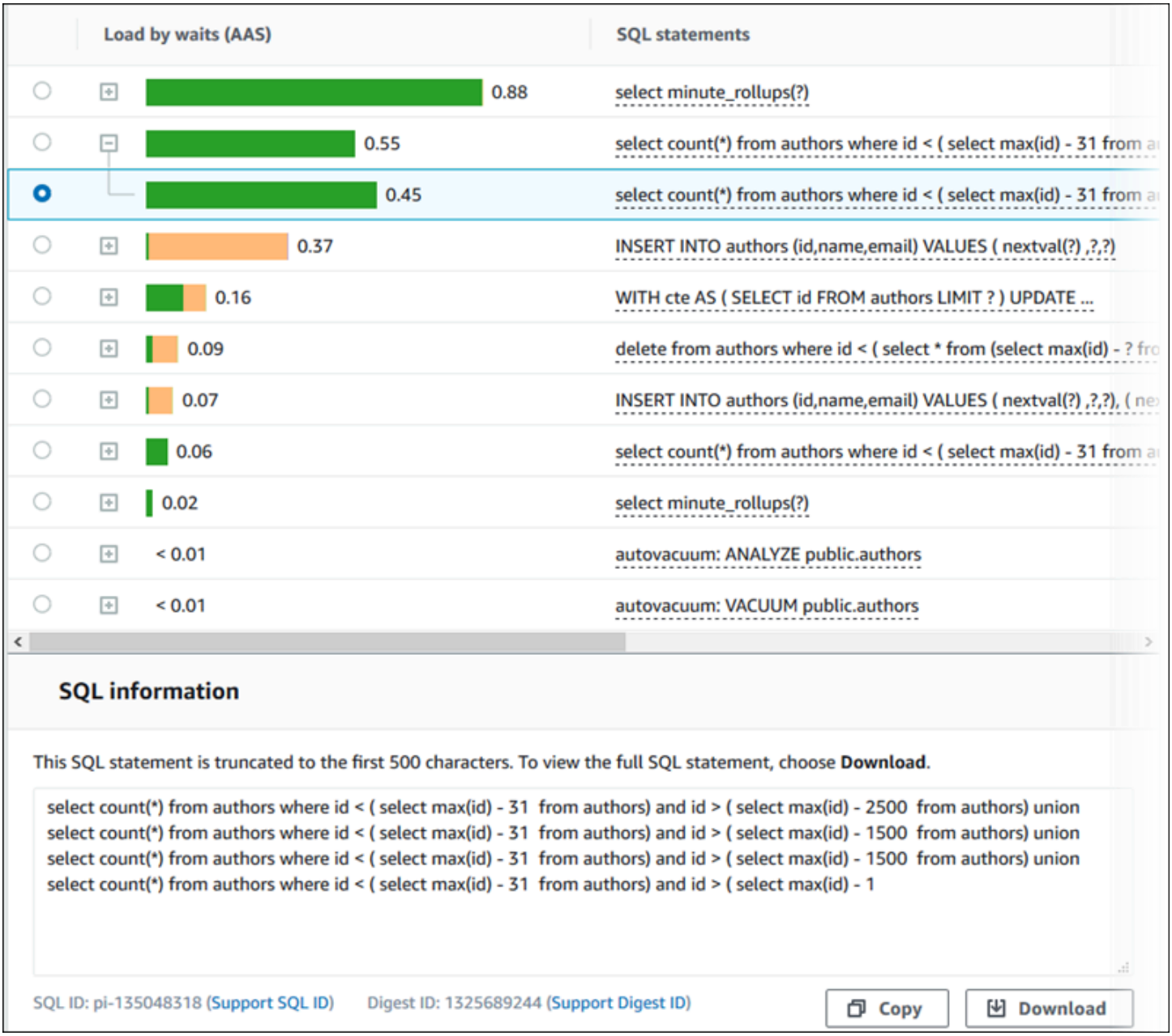
In Top SQL, la colonna Load by waits (AAS) illustra la percentuale del carico del database associato a ciascun elemento di caricamento superiore. Questa colonna indica il carico per questo elemento in base a qualunque raggruppamento attualmente selezionato nel grafico DB Load. Per ulteriori informazioni sulle sessioni attive medie (AAS), consulta Media delle sessioni attive.

Ad esempio, è possibile raggruppare il Carico DB in base agli stati di attesa. Esaminare le query SQL nella tabella degli elementi di caricamento superiore. In questo caso, la barra DB Load by Waits (Carico del database in base alle attese) è dimensionata, segmentata e rappresentata da un colore per mostrare qual è il contributo della query a un dato stato di attesa. Mostra anche quali stati di attesa stanno influenzando la query selezionata.



Visualizzazione delle informazioni SQL

Nella tabella Top SQL (Prime istruzioni SQL) è possibile aprire un'istruzione per visualizzarne le informazioni. Le informazioni vengono visualizzate nel riquadro inferiore.

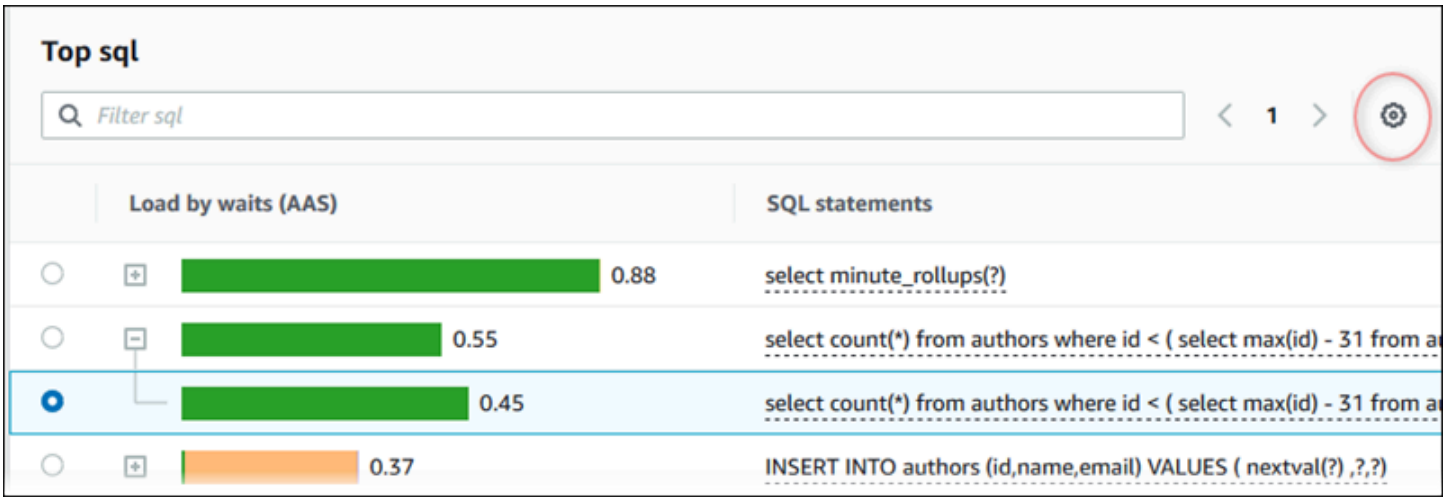


Puoi visualizzare i seguenti tipi di identificatori (ID) associati alle istruzioni SQL:

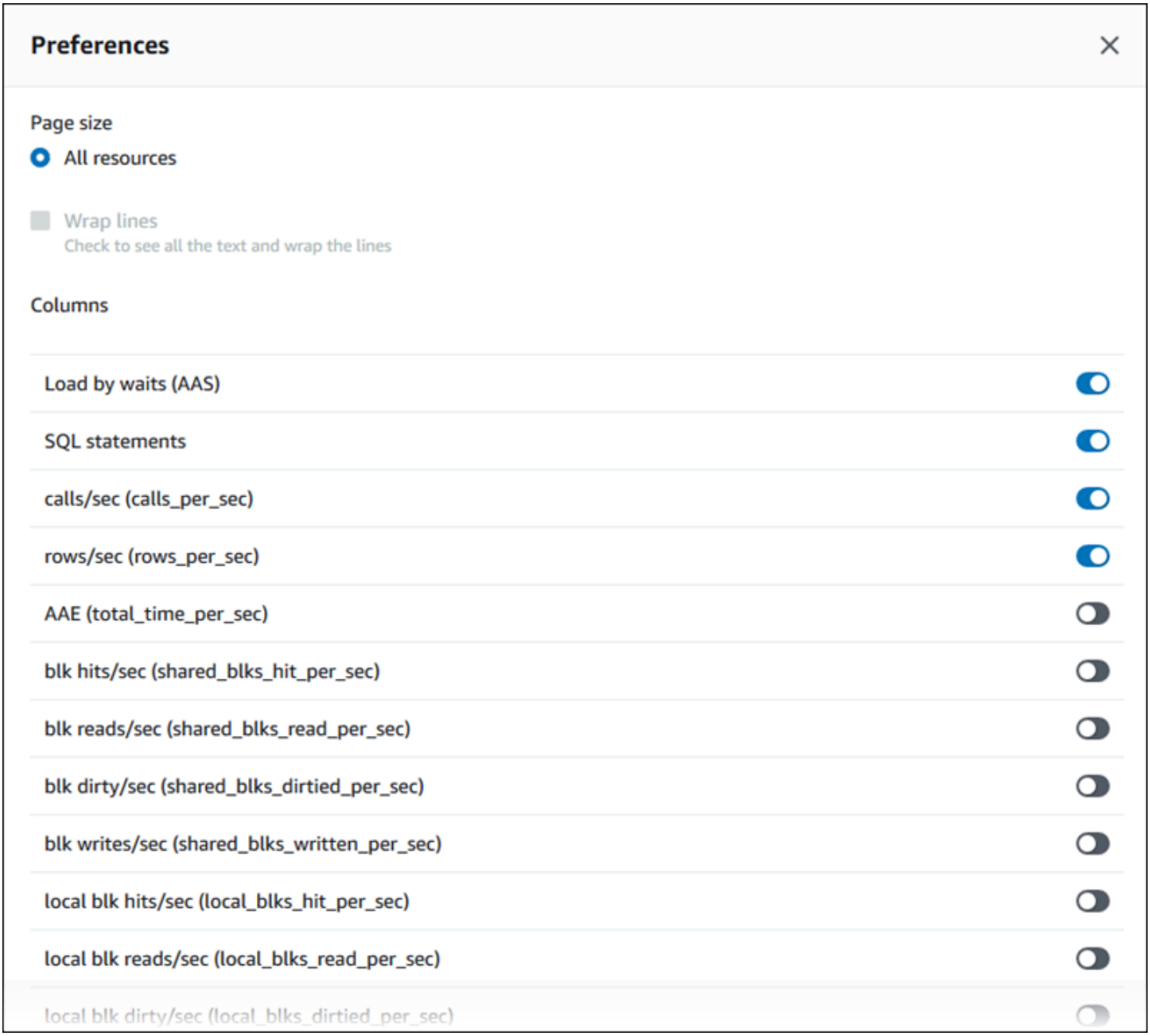
- ID SQL di supporto — Un valore hash dell'ID SQL. Questo valore serve solo come riferimento a un ID SQL quando si utilizza AWS Support. AWS Support non dispone dell'accesso agli ID SQL effettivi e al testo SQL.
- ID Digest di supporto – Un valore hash dell'ID Digest. Questo valore serve solo come riferimento a un ID Digest quando si utilizza AWS Support. AWS Support non dispone dell'accesso agli ID Digest effettivi e al testo SQL.

Scelta delle preferenze statistiche

È possibile controllare le statistiche visualizzate nella scheda Top SQL (Prime istruzioni SQL) scegliendo l'icona Preferenze.



Quando scegli l'icona Preferences (Preferenze), viene visualizzata la finestra Preferences (Preferenze). La schermata seguente è un esempio della finestra Preferences (Preferenze).



Abilitare le statistiche che si desidera visualizzare nella scheda Top SQL (Prime istruzioni SQL), utilizzare il mouse per scorrere fino alla fine della finestra, quindi scegliere Continua.

Per ulteriori informazioni sulle statistiche per secondo o per chiamata per i motori Aurora, consulta la sezione delle statistiche SQL specifiche del motore in Statistiche SQL per Performance Insights

Accesso a una maggiore quantità di testo SQL nel pannello di controllo di Performance Insights

Per impostazione predefinita, ciascuna riga nella tabella Top SQL (Prime istruzioni SQL) mostra 500 byte di testo SQL per ciascuna istruzione SQL.

Quando un'istruzione SQL supera i 500 byte, puoi visualizzare più testo nella sezione SQL text (Testo SQL) sotto la tabella Top SQL (Prime istruzioni SQL). In questo caso, la lunghezza massima per il testo visualizzato in SQL text (Testo SQL) è 4 KB. Questo limite viene introdotto dalla console ed è soggetto ai limiti impostati dal motore del database. Per salvare il testo visualizzato in SQL text (Testo SQL), scegli Download (Scarica).

Argomenti

- Limiti delle dimensioni del testo per i Aurora MySQL
- Impostazione del limite di testo SQL per le istanze database Aurora PostgreSQL
- Visualizzazione e download del testo SQL nella dashboard di Approfondimenti sulle prestazioni.

Limiti delle dimensioni del testo per i Aurora MySQL

Durante il download di testo SQL, il motore del database determina la sua lunghezza massima. Puoi scaricare il testo SQL fino ai seguenti limiti per motore.

Motore database	Lunghezza massima del testo scaricato
Aurora MySQL	La lunghezza è fissata a 4.096 byte.

La sezione SQL Text (Testo SQL) della console Performance Insights visualizza fino al massimo restituito dal motore. Ad esempio, se Aurora MySQL restituisce al massimo 1 KB a Performance Insights, può raccogliere e mostrare solo 1 KB, anche se la query originale è più grande. Pertanto, quando la query viene visualizzata in SQL text (Testo SQL) o scaricata, Performance Insights restituisce lo stesso numero di byte.

Se si utilizza AWS CLI o l'API, Approfondimenti sulle prestazioni non ha il limite di 4 KB applicato dalla console. `DescribeDimensionKeys` e `GetResourceMetrics` restituiscono al massimo 500 byte.

> **Note**
>
> `GetDimensionKeyDetails` restituisce la query completa, ma la dimensione è soggetta al limite del motore.

Impostazione del limite di testo SQL per le istanze database Aurora PostgreSQL

Aurora PostgreSQL gestisce il testo in modo diverso. È possibile impostare il limite delle dimensioni del testo con il parametro di istanza database `track_activity_query_size`. Questo parametro presenta le caratteristiche seguenti:

Dimensione di default del testo

Su Aurora Aurora PostgreSQL versione 9.6, l'impostazione di default per il parametro `track_activity_query_size` è 1.024 byte. Su Aurora PostgreSQL versione 10 o successive, l'impostazione di default per il parametro è 4.096 byte.

Dimensione massima del testo

Il limite per `track_activity_query_size` è 102.400 byte per Aurora PostgreSQL versione 12 e versioni precedenti. Il massimo è di 1 MB per la versione 13 e quelle successive.

Se il motore restituisce 1 MB a Performance Insights, la console visualizza solo i primi 4 KB. Se si scarica la query, si ottiene 1 MB per intero. In questo caso, la visualizzazione e il download restituiscono un numero diverso di byte. Per ulteriori informazioni sul parametro dell'istanza database `track_activity_query_size`, consulta Run-time Statistics nella documentazione di PostgreSQL.

Per aumentare la dimensione del testo SQL, aumenta il limite di `track_activity_query_size`. Per modificare il parametro, modifica l'impostazione del parametro nel gruppo di parametri associato all'istanza database Aurora PostgreSQL.

Modifica dell'impostazione quando l'istanza utilizza il gruppo di parametri di default

1. Crea un nuovo gruppo di parametri dell'istanza database per il motore del database e la versione del motore del database appropriati.
2. Imposta il parametro nel nuovo gruppo di parametri.
3. Associa il nuovo gruppo di parametri all'istanza database.

Per ulteriori informazioni sull'impostazione di un parametro dell'istanza database, consulta Modifica dei parametri in un gruppo di parametri database in Amazon Aurora.

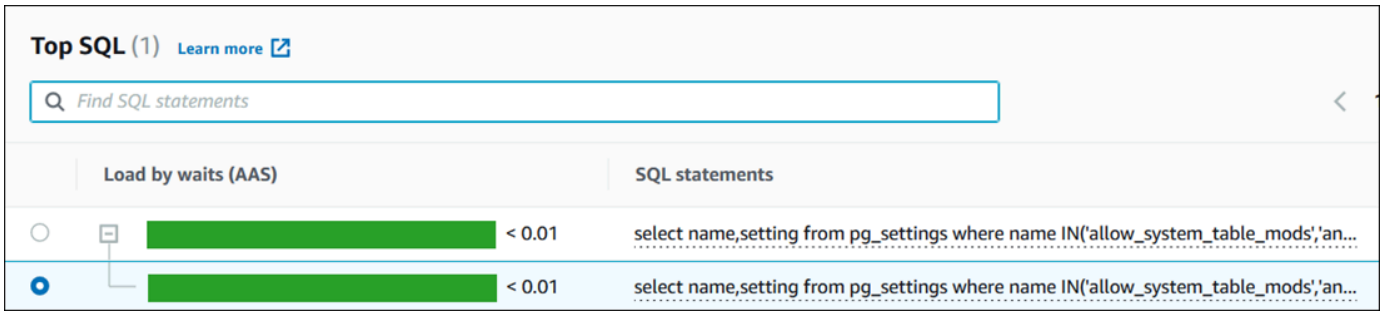
Visualizzazione e download del testo SQL nella dashboard di Approfondimenti sulle prestazioni.

Nel pannello di controllo di Performance Insights è possibile visualizzare e scaricare il testo SQL.

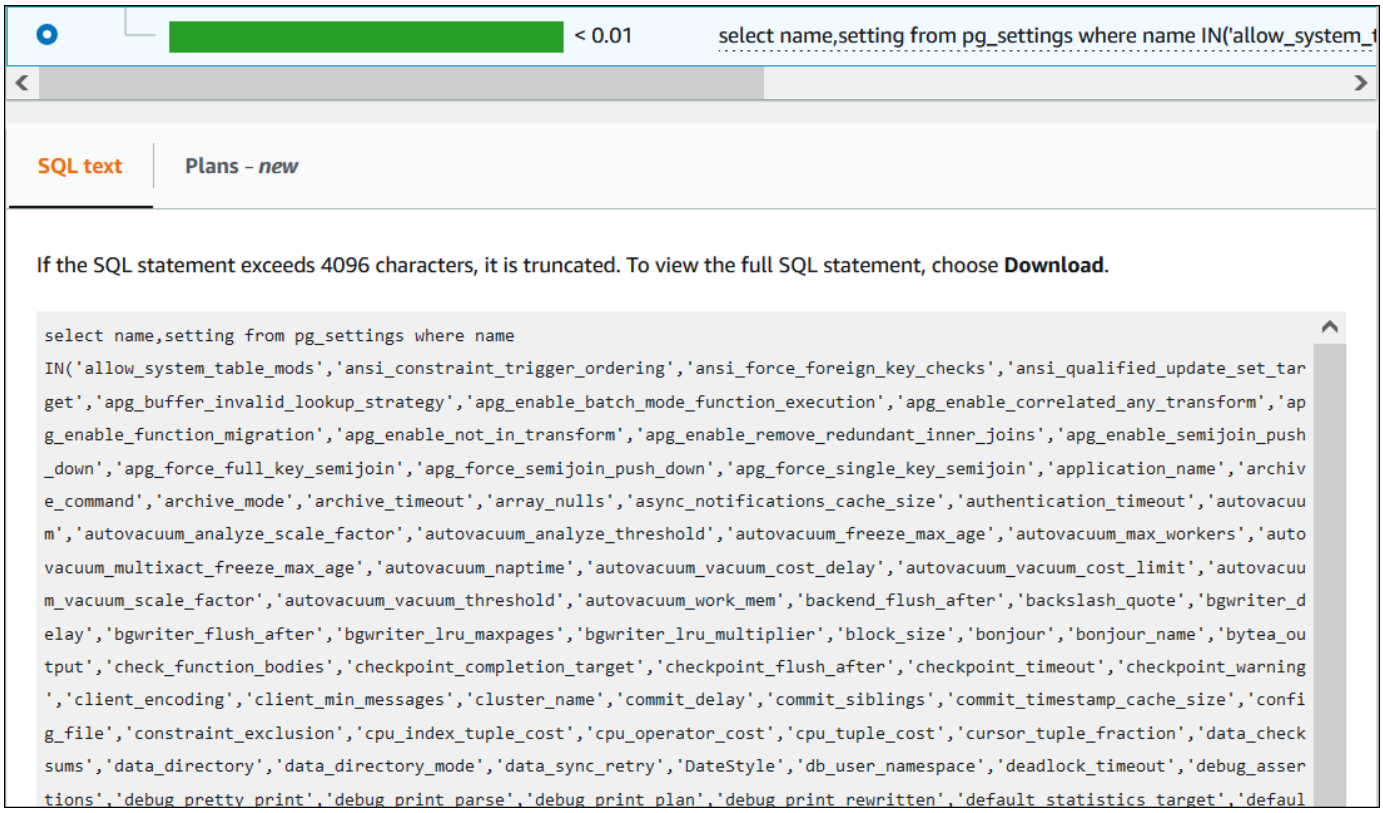
Per visualizzare una maggiore quantità di testo SQL nel pannello di controllo di Performance Insights

1. Apri la console Amazon RDS all'indirizzo https://console.aws.amazon.com/rds/.
2. Nel pannello di navigazione scegli Performance Insights.
3. Scegli un'istanza database.
4. Scorri verso il basso fino alla scheda SQL principale nella dashboard di Approfondimenti sulle prestazioni.
5. Scegli il segno più per espandere un digest SQL e scegli una delle query secondarie del digest.

   Le istruzioni SQL con testo superiore a 500 byte sono simili a quelle nell'immagine seguente.



6. Scorri verso il basso fino alla scheda Testo SQL.



Il pannello di controllo di Performance Insights può visualizzare fino a 4.096 byte per ciascuna istruzione SQL.

7. (Facoltativo) Scegliere Copia per copiare l'istruzione SQL visualizzata oppure scegliere Scarica per scaricare l'istruzione SQL e visualizzare il testo SQL fino al limite del motore database.

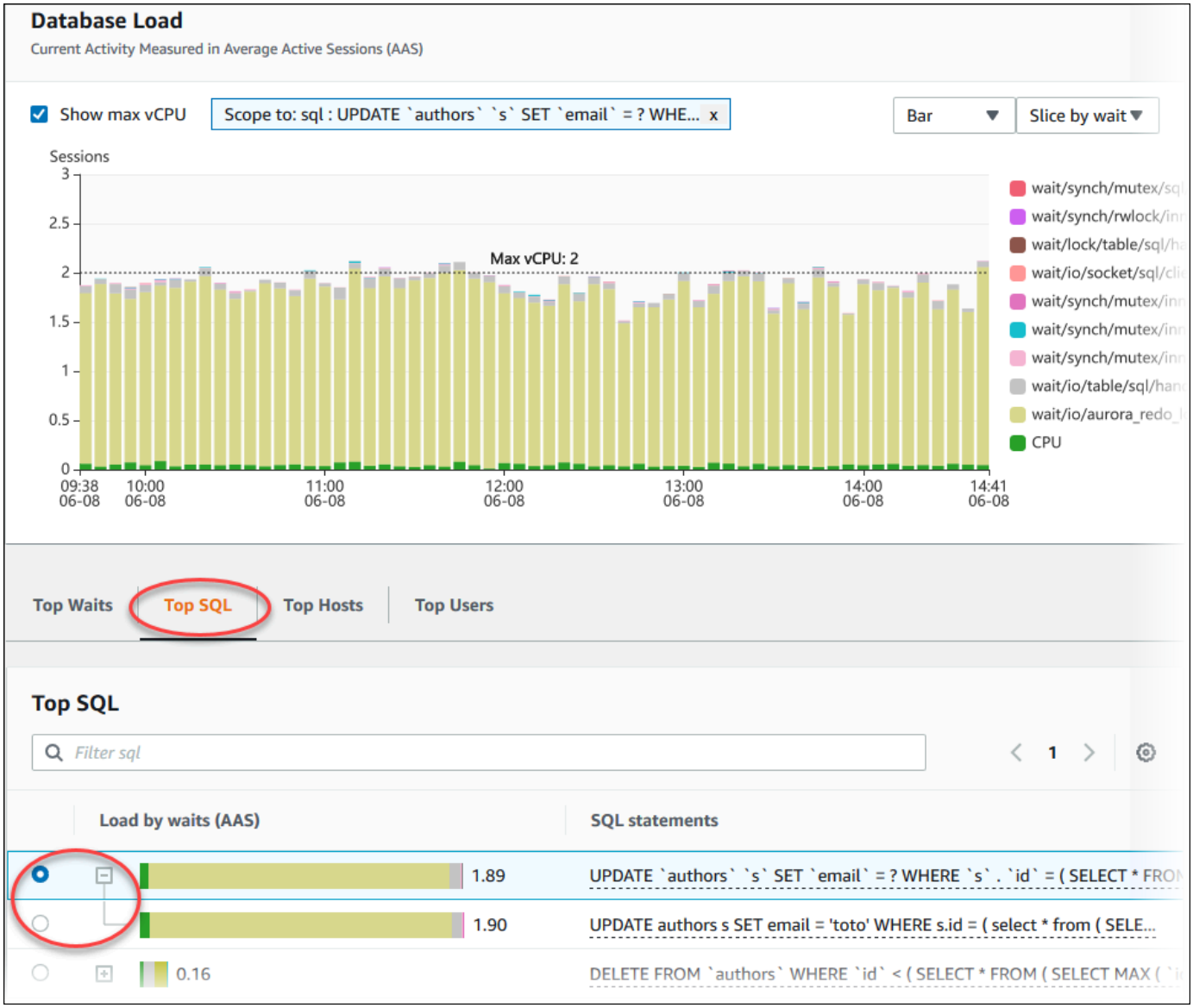
> **Note**
>
> Per copiare o scaricare l'istruzione SQL, disattiva i sistemi di blocco popup.

Visualizzazione delle statistiche SQL nel pannello di controllo di Performance Insights

Nel pannello di controllo di Performance Insights, le statistiche SQL sono disponibili nella scheda Top SQL (Prime istruzioni SQL) del grafico Database load (Carico database).

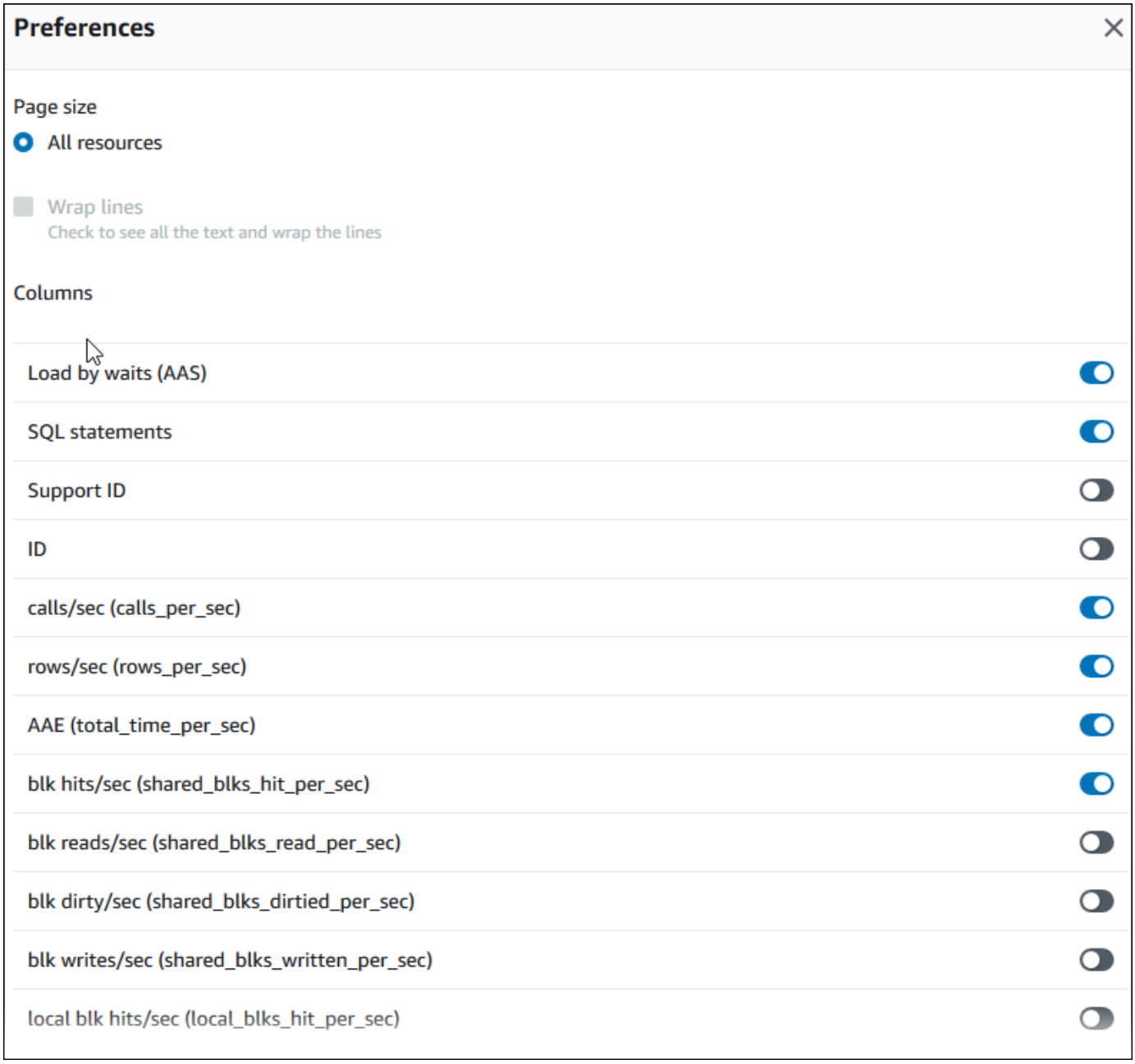
Per visualizzare le statistiche SQL

1. Apri la console Amazon RDS all'indirizzo https://console.aws.amazon.com/rds/.
2. Nel pannello di navigazione scegli Performance Insights.
3. Nella parte superiore della pagina, scegli il database di cui desideri visualizzare le statistiche SQL.
4. Scorrere fino alla parte inferiore della pagina e scegli Top SQL (Prime istruzioni SQL).
5. Scegli una specifica istruzione (solo Aurora MySQL) o un sunto di una query.

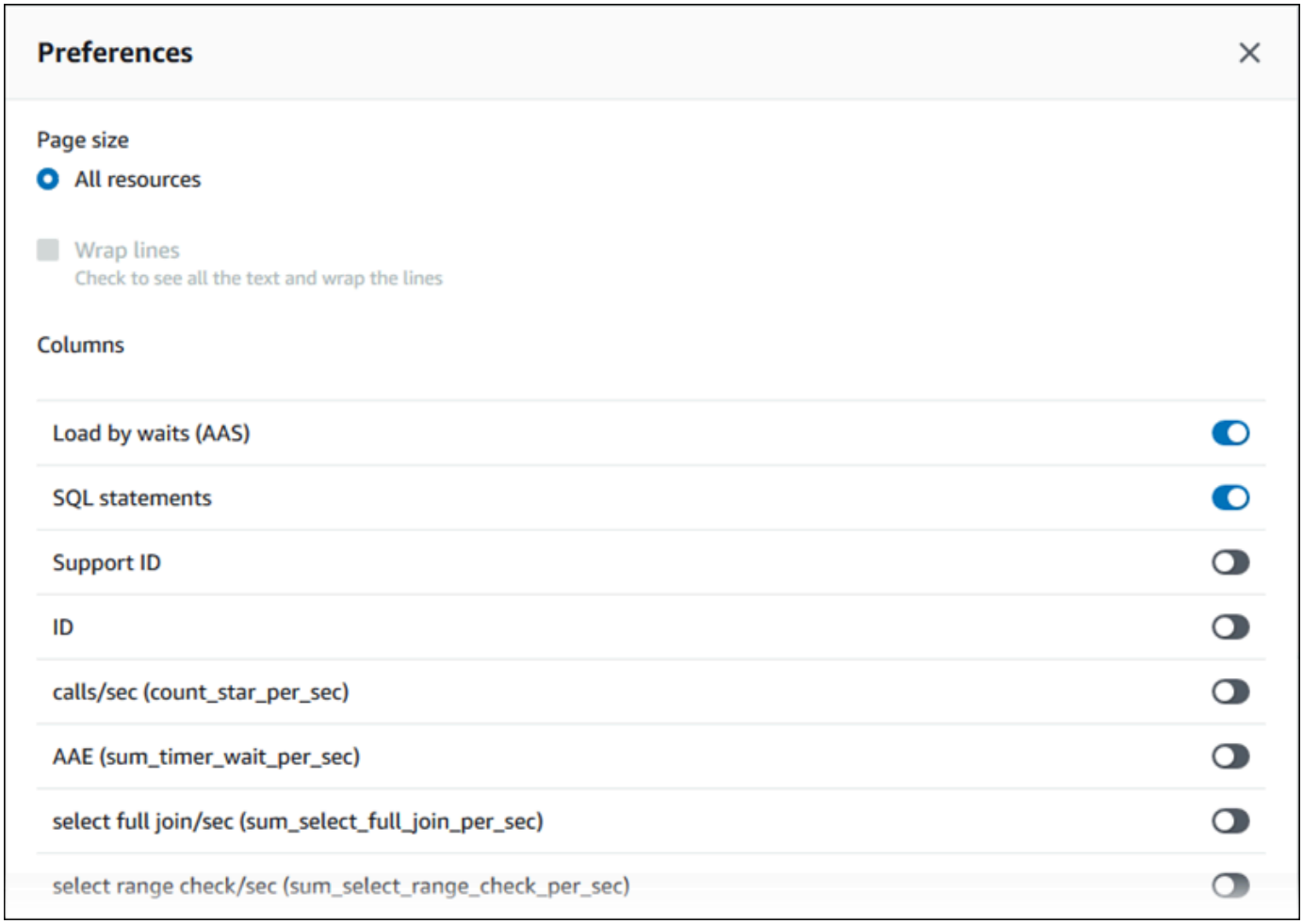


6. Scegliere le statistiche da visualizzare selezionando l'icona a forma di ingranaggio nell'angolo in alto a destra del grafico. Per le descrizioni delle statistiche SQL per i motori Amazon RDSAurora, consulta Statistiche SQL per Performance Insights.

   Il seguente esempio mostra le preferenze per PostgreSQL Aurora.



Il seguente esempio mostra le preferenze per le istanze database Aurora MySQL.



7. Per salvare le preferenze, scegli Save (Salva).

   La tabella Top SQL (Prime istruzioni SQL) si aggiorna.

Visualizzazione dei consigli proattivi di Approfondimenti sulle prestazioni

Approfondimenti sulle prestazioni di Amazon RDS monitora metriche specifiche e crea automaticamente soglie analizzando quali livelli potrebbero essere potenzialmente problematici per una specifica risorsa. Quando i nuovi valori delle metriche superano una soglia predefinita in un determinato periodo di tempo, Approfondimenti sulle prestazioni genera una raccomandazione proattiva. Questa raccomandazione aiuta a evitare un impatto sulle prestazioni del database in futuro. Per ricevere queste raccomandazioni proattive, è necessario attivare Approfondimenti sulle prestazioni con un periodo di conservazione del piano a pagamento.

Per ulteriori informazioni sull'attivazione di Performance Insights, consulta Attivazione e disattivazione di Approfondimenti sulle prestazioni per Aurora. Per informazioni sui prezzi e sulla conservazione dei

dati per Approfondimenti sulle prestazioni, consulta Prezzi e conservazione dei dati per Performance Insights.

Per determinare le Regioni, i motori di database e le classi di istanza supportate per le raccomandazioni proattive, consulta Supporto di classe di istanza, regione e motore di database Amazon Aurora per funzionalità Performance Insights.

È possibile visualizzare l'analisi dettagliata e le indagini consigliate delle raccomandazioni proattive nella pagina Dettagli della raccomandazione.

Per ulteriori informazioni sulle raccomandazioni, consulta Suggerimenti di Amazon Aurora.

Per visualizzare l'analisi dettagliata di una raccomandazione proattiva

1. Accedi alla Console di gestione AWS e apri la console Amazon RDS all'indirizzo https://console.aws.amazon.com/rds/.
2. Nel riquadro di navigazione, effettua una delle seguenti operazioni:
   - Scegli Raccomandazioni.

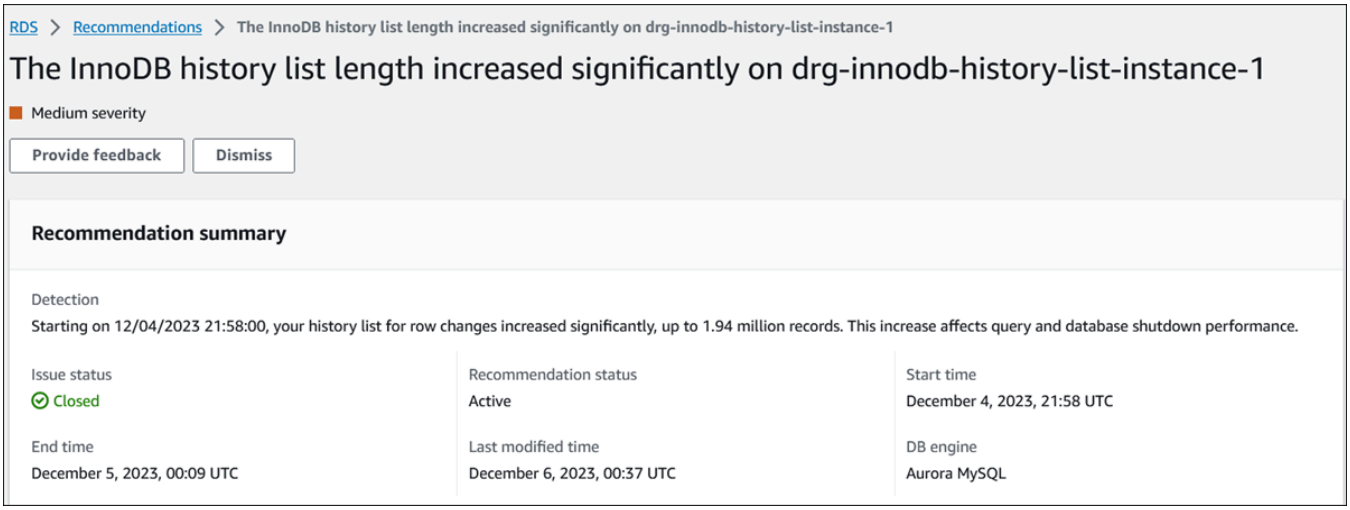
     La pagina Raccomandazioni mostra un elenco di raccomandazioni ordinate in base alla gravità per tutte le risorse dell'account.
   - Scegli Database, quindi scegli Raccomandazioni per una risorsa nella pagina dei database.

     La scheda Raccomandazioni mostra le raccomandazioni e i relativi dettagli per la risorsa selezionata.
3. Trova un consiglio proattivo e scegli Visualizza dettagli.

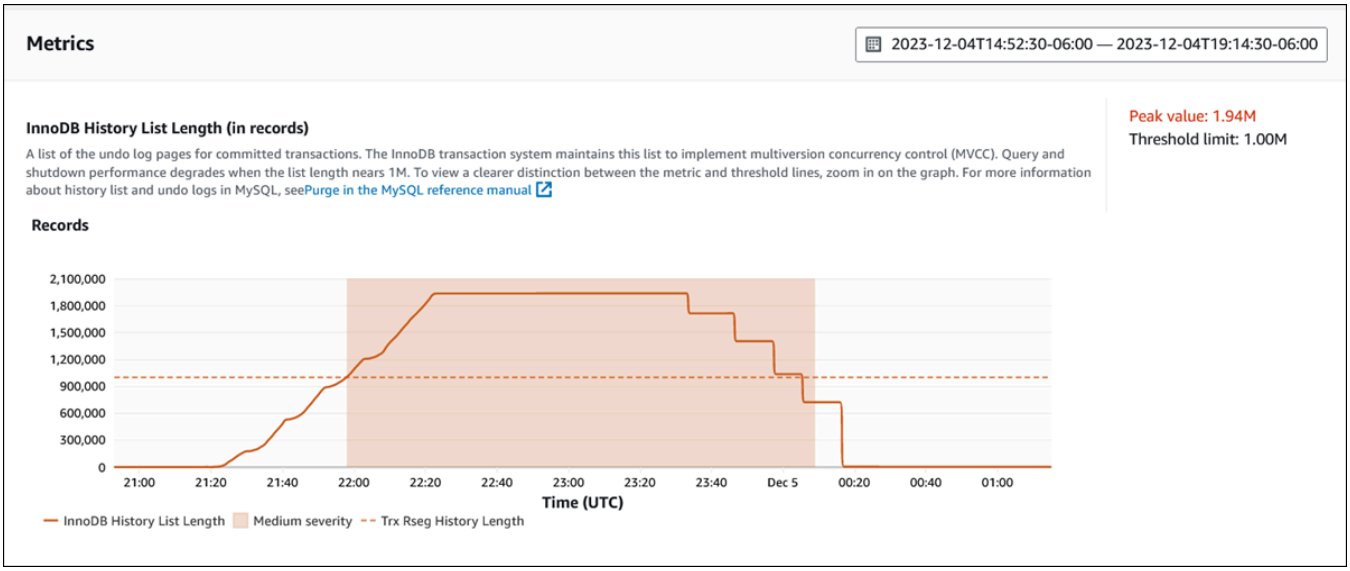
   Viene visualizzata la pagina Raccomandazioni. Il titolo fornisce il nome della risorsa interessata con il problema rilevato e la gravità.

   Di seguito sono riportati i componenti della pagina Dettagli della raccomandazione:

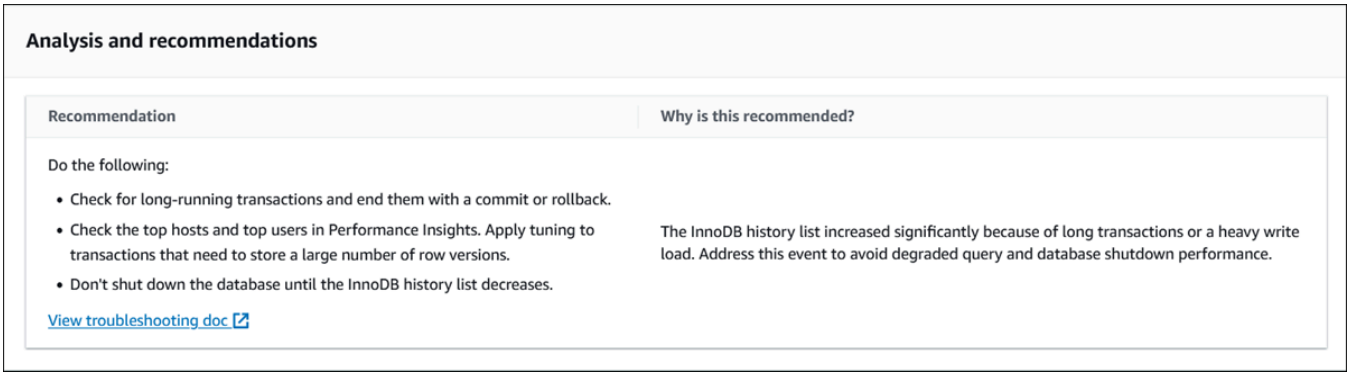
   - Riepilogo delle raccomandazioni: il problema rilevato, lo stato della raccomandazione e del problema, l'ora di inizio e di fine del problema, l'ora di modifica della raccomandazione e il tipo di motore.



- Metriche: i grafici del problema rilevato. Ogni grafico mostra una soglia determinata dal comportamento di base della risorsa e dai dati della metrica indicata dall'ora di inizio del problema.



- Analisi e raccomandazioni: la raccomandazione e il motivo della raccomandazione suggerita.



È possibile esaminare la causa del problema e quindi eseguire le azioni consigliate suggerite per risolvere il problema oppure scegliere Ignora in alto a destra per ignorare la raccomandazione.

Recupero delle metriche con l'API Approfondimenti sulle prestazioni per Aurora

Quando Performance Insights è attivato, l'API fornisce visibilità sulle prestazioni dell'istanza. Amazon CloudWatch Logs fornisce la fonte autorevole per i parametri di monitoraggio venduti per i servizi AWS.

Performance Insights offre una vista specifica del dominio del carico del database misurato come numero medio di sessioni attive (AAS). Questo parametro viene visualizzata dai consumer API come un set di dati temporali bidimensionali. La dimensione temporale dei dati fornisce i dati relativi al carico del database per ogni momento dell'intervallo di tempo in cui è stata eseguita la query. Ogni punto temporale scompone il carico complessivo in relazione alle dimensioni richieste, come `SQL`, `Wait-event`, `User` o `Host`, misurato in corrispondenza di quel punto temporale.

Amazon RDS Performance Insights monitora il cluster Amazon Aurora per consentire di analizzare e risolvere i problemi di performance del database. Un modo per visualizzare i dati di Performance Insights è disponibile nella Console di gestione AWS. Performance Insights fornisce inoltre un'API pubblica per eseguire query sui dati. Puoi usare l'API per effettuare quanto segue:

- Scaricamento dei dati in un database
- Aggiungi dati Performance Insights ai pannelli di controllo di monitoraggio esistenti
- Crea strumenti di monitoraggio

Per utilizzare l'API di Performance Insights, abilita Performance Insights su una delle istanze database Amazon RDS. Per informazioni sull'abilitazione di Performance Insights, consulta Attivazione e disattivazione di Approfondimenti sulle prestazioni per Aurora. Per ulteriori informazioni sull'API di Performance Insights, consulta la Documentazione di riferimento dell'API di Amazon RDS Performance Insights.

L'API di Performance Insights fornisce le seguenti operazioni.

Operazione di Performance Insights	AWS CLI command	Descrizione
CreatePerformanceAnalysisReport	aws pi create-performance-analysis-report	Crea un report di analisi delle prestazioni per un periodo di tempo specifico per l'istanza database. Il risultato è `AnalysisReportId` che è l'identificatore univoco del report.
DeletePerformanceAnalysisReport	aws pi delete-performance-analysis-report	Elimina un report di analisi delle prestazioni.
DescribeDimensionKeys	aws pi describe-dimension-keys	Recupera le prime N chiavi di dimensione per un parametro per un determinato periodo di tempo.
GetDimensionKeyDetails	aws pi get-dimension-key-details	Recupera gli attributi del gruppo di dimensioni specificato per un'istanza database o un'origine dati. Ad esempio, se si specifica un ID SQL e se i dettagli delle dimensioni sono disponibili, `GetDimensionKeyDetails` recupera il testo completo delle dimensioni `db.sql.statement` associate a questo ID. Questa operazione è utile perché `GetResourceMetrics` e `DescribeDimensionKeys` non supportano il recupero di testi

Operazione di Performance Insights	AWS CLI command	Descrizione
		di istruzioni SQL di grandi dimensioni.
GetPerformanceAnalysisReport	aws pi get-performance-analysis-report	Recupera il report, comprese le informazioni dettagliate relative al report. Il risultato include lo stato del report, l'ID del report, i dettagli sull'ora del report, le informazioni dettagliate e i suggerimenti.
GetResourceMetadata	aws pi get-resource-metadata	Recupera i metadati per diverse caratteristiche. Ad esempio, i metadati potrebbero indicare che una caratteristica è attivata o disattivata su un'istanza database specifica.
GetResourceMetrics	aws pi get-resource-metrics	Recupera parametri Performance Insights per un set di origini dati, su un periodo di tempo. Puoi fornire gruppi di dimensioni e dimensioni specifiche e fornire criteri di aggregazione e filtro per ogni gruppo.
ListAvailableResourceDimensions	aws pi list-available-resource-dimensions	Recupera le dimensioni su cui è possibile eseguire query per ogni tipo di parametro specificato su un'istanza specificata.

Operazione di Performance Insights	AWS CLI command	Descrizione
ListAvailableResourceMetrics	aws pi list-available-resource-metrics	Recupera tutti i parametri disponibili dei tipi di parametro specificati su cui è possibile eseguire query per un'istanza database specificata.
ListPerformanceAnalysisReports	aws pi list-performance-analysis-reports	Recupera tutti i report di analisi disponibili per l'istanza database. I report sono elencati in base all'ora di inizio di ciascun report.
ListTagsForResource	aws pi list-tags-for-resource	Elenca tutti i tag dei metadati aggiunti alla risorsa. L'elenco include il nome e il valore del tag.
TagResource	aws pi tag-resource	Aggiunge tag dei metadati alla risorsa Amazon RDS. Il tag include un nome e un valore.
UntagResource	aws pi untag-resource	Rimuove tag dei metadati dalla risorsa.

Per ulteriori informazioni sul recupero delle metriche di serie temporali e gli esempi di AWS CLI per Approfondimenti sulle prestazioni, consulta gli argomenti riportati di seguito.

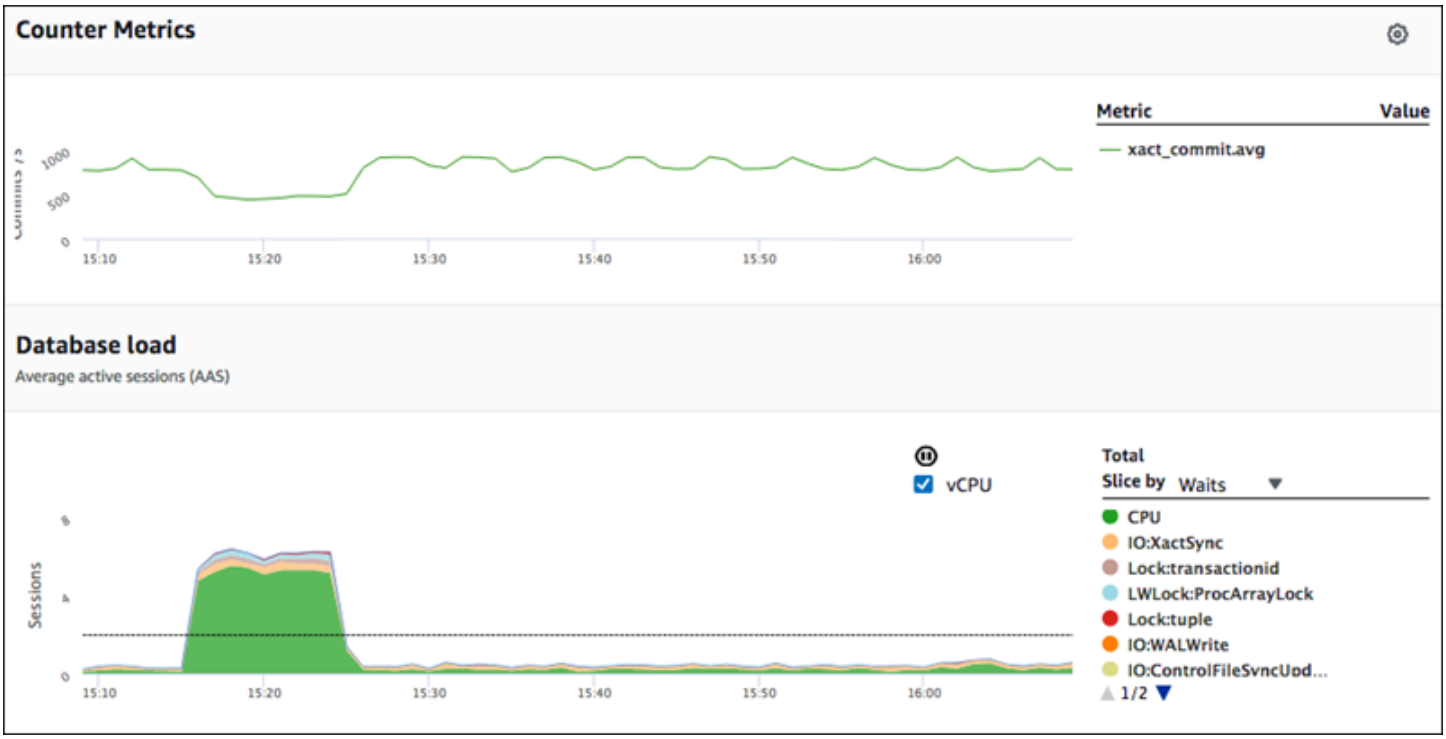
Argomenti

- Recupero delle metriche di serie temporali per Approfondimenti sulle prestazioni
- Esempi di utilizzo di AWS CLI per Performance Insights

Recupero delle metriche di serie temporali per Approfondimenti sulle prestazioni

L'operazione `GetResourceMetrics` recupera uno o più parametri di serie temporali dai dati di Performance Insights. `GetResourceMetrics` richiede un parametro e un periodo di tempo e restituisce una risposta con un elenco di punti di dati.

Ad esempio, la Console di gestione AWS utilizza `GetResourceMetrics` per popolare il grafico Counter Metrics (Parametri contatore) e il grafico Database Load (Carico del database), come illustrato nell'immagine seguente.



Tutti i parametri restituiti da `GetResourceMetrics` sono parametri di serie temporali standard ad eccezione di `db.load`. Questo parametro è visualizzato nel grafico Database Load (Carico del database). Il parametro `db.load` è diverso dagli altri parametri di serie temporali in quanto può essere suddiviso in sottocomponenti detti dimensioni. Nell'immagine precedente, `db.load` è suddiviso e raggruppato in base agli stati delle attese che formano il `db.load`.

> **Note**
>
> `GetResourceMetrics` può anche restituire il parametro `db.sampleload`, ma il parametro `db.load` è appropriato nella maggior parte dei casi.

Per informazioni sui parametri contatore restituiti da `GetResourceMetrics`, consulta Metriche contatore di Performance Insights.

I seguenti calcoli sono supportati per i parametri:

- Media: il valore medio per il parametro su un periodo di tempo. Aggiungi `.avg` al nome parametro.
- Minimo: il valore minimo per il parametro su un periodo di tempo. Aggiungi `.min` al nome parametro.
- Massimo: il valore massimo per il parametro su un periodo di tempo. Aggiungi `.max` al nome parametro.
- Somma: la somma dei valori dei parametri su un periodo di tempo. Aggiungi `.sum` al nome parametro.
- Conteggio di esempio: il numero di volte che il parametro è stato raccolto su un periodo di tempo. Aggiungi `.sample_count` al nome parametro.

Ad esempio, supponiamo che un parametro venga raccolto per 300 secondi (5 minuti) e che il parametro venga raccolto una volta al minuto. I valori per ogni minuto sono 1, 2, 3, 4 e 5. In questo caso. vengono restituiti i seguenti calcoli:

- Media: 3
- Minimo: 1
- Massimo: 5
- Somma: 15
- Conteggio del campione: 5

Per ulteriori informazioni sull'utilizzo del comando `get-resource-metrics` della AWS CLI, consulta `get-resource-metrics`.

Per l'opzione `--metric-queries`, specifica una o più query per cui ottenere risultati. Ciascuna query consiste di un parametro obbligatorio `Metric` e parametri facoltativi `GroupBy` e `Filter`. Di seguito è riportato un esempio della specifica di un'opzione `--metric-queries`.

```
{
   "Metric": "string",
   "GroupBy": {
     "Group": "string",
```

```
    "Dimensions": ["string", ...],
    "Limit": integer
  },
  "Filter": {"string": "string"
    ...}
```

Esempi di utilizzo di AWS CLI per Performance Insights

Nelle sezioni seguenti, viene illustrato l'utilizzo di AWS Command Line Interface (AWS CLI) per Approfondimenti sulle prestazioni ed esempi di utilizzo per AWS CLI.

Argomenti

- Guida integrata di AWS CLI per Approfondimenti sulle prestazioni
- Recupero dei parametri contatore
- Recupero della media del carico del database per i principali eventi di attesa
- Recupero della media del carico del database per il principale SQL
- Recupero della media del carico del database filtrata da SQL
- Recupero del testo completo di un'istruzione SQL
- Creazione di un report di analisi delle prestazioni per un periodo di tempo
- Recupero di un report di analisi delle prestazioni
- Elenco di tutti i report di analisi delle prestazioni per l'istanza database
- Eliminazione di un report di analisi delle prestazioni
- Aggiunta di un tag a un report di analisi delle prestazioni
- Elenco di tutti i tag per un report di analisi delle prestazioni
- Eliminazione di tag da un report di analisi delle prestazioni

Guida integrata di AWS CLI per Approfondimenti sulle prestazioni

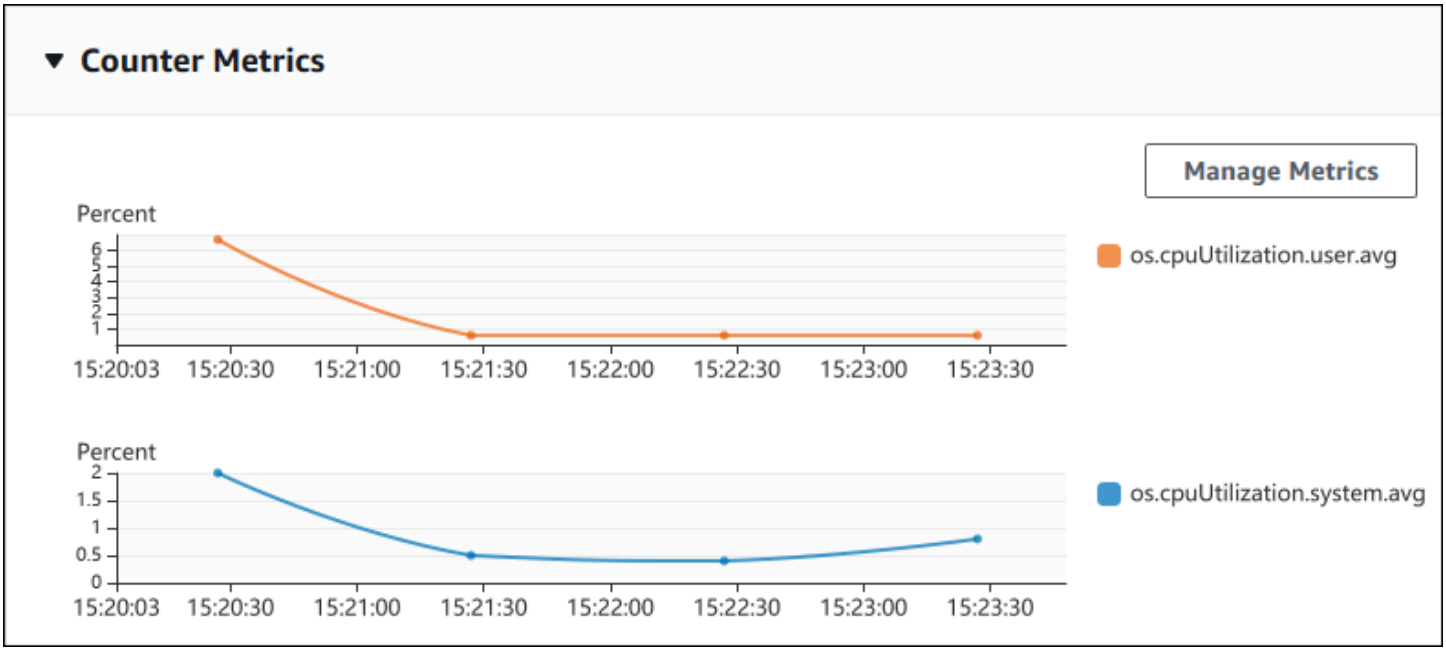
Puoi visualizzare i dati di Performance Insights utilizzando la AWS CLI. Puoi visualizzare la guida per i comandi AWS CLI per Performance Insights inserendo quanto segue nella riga di comando.

```
aws pi help
```

Se AWS CLI non è installata, consulta Installazione della AWS CLI nella Guida per l'utente di AWS CLI per informazioni sull'installazione.

Recupero dei parametri contatore

Lo screenshot seguente mostra due grafici dei parametri contatore nella Console di gestione AWS.



L'esempio seguente mostra come raccogliere gli stessi dati che utilizza la Console di gestione AWS per generare i due grafici dei parametri contatore.

Per Linux, macOS o Unix:

```
aws pi get-resource-metrics \
   --service-type RDS \
   --identifier db-ID \
   --start-time 2018-10-30T00:00:00Z \
   --end-time   2018-10-30T01:00:00Z \
   --period-in-seconds 60 \
   --metric-queries '[{"Metric": "os.cpuUtilization.user.avg"  },
                      {"Metric": "os.cpuUtilization.idle.avg"}]'
```

Per Windows:

```
aws pi get-resource-metrics ^
   --service-type RDS ^
   --identifier db-ID ^
   --start-time 2018-10-30T00:00:00Z ^
   --end-time   2018-10-30T01:00:00Z ^
   --period-in-seconds 60 ^
   --metric-queries '[{"Metric": "os.cpuUtilization.user.avg"  },
```

```
                    {"Metric": "os.cpuUtilization.idle.avg"}]'
```

Puoi agevolare la lettura del comando specificando un file per l'opzione `--metrics-query`. Il seguente esempio utilizza un file denominato query.json per l'opzione. Il file presenta i seguenti contenuti.

```
[
    {
        "Metric": "os.cpuUtilization.user.avg"
    },
    {
        "Metric": "os.cpuUtilization.idle.avg"
    }
]
```

Esegui il comando seguente per utilizzare il file.

Per Linux, macOS o Unix:

```
aws pi get-resource-metrics \
   --service-type RDS \
   --identifier db-ID \
   --start-time 2018-10-30T00:00:00Z \
   --end-time   2018-10-30T01:00:00Z \
   --period-in-seconds 60 \
   --metric-queries file://query.json
```

Per Windows:

```
aws pi get-resource-metrics ^
   --service-type RDS ^
   --identifier db-ID ^
   --start-time 2018-10-30T00:00:00Z ^
   --end-time   2018-10-30T01:00:00Z ^
   --period-in-seconds 60 ^
   --metric-queries file://query.json
```

L'esempio precedente specifica i seguenti valori per le opzioni:

- `--service-type` – RDS for Amazon RDS
- `--identifier` – L'ID risorsa per l'istanza database

- `--start-time` e `--end-time` – I valori ISO 8601 `DateTime` per il periodo su cui eseguire le query, con supporto di più formati

Esegue query per un intervallo di tempo di un'ora:

- `--period-in-seconds` – `60` per una query al minuto
- `--metric-queries` – Una serie di due query, ognuna solo per un parametro

  Il nome del parametro utilizza punti per classificare il parametro in una categoria utile, dove l'ultimo elemento è una funzione. Nell'esempio, la funzione è `avg` per ciascuna query. Come con Amazon CloudWatch, le funzioni supportate sono `min`, `max`, `total` e `avg`.

La risposta è simile a quella riportata di seguito.

```
{
    "Identifier": "db-XXX",
    "AlignedStartTime": 1540857600.0,
    "AlignedEndTime": 1540861200.0,
    "MetricList": [
        { //A list of key/datapoints
            "Key": {
                "Metric": "os.cpuUtilization.user.avg" //Metric1
            },
            "DataPoints": [
                //Each list of datapoints has the same timestamps and same number of
 items
                {
                    "Timestamp": 1540857660.0, //Minute1
                    "Value": 4.0
                },
                {
                    "Timestamp": 1540857720.0, //Minute2
                    "Value": 4.0
                },
                {
                    "Timestamp": 1540857780.0, //Minute 3
                    "Value": 10.0
                }
                //... 60 datapoints for the os.cpuUtilization.user.avg metric
            ]
        },
```

```
        {
            "Key": {
                "Metric": "os.cpuUtilization.idle.avg" //Metric2
            },
            "DataPoints": [
                {
                    "Timestamp": 1540857660.0, //Minute1
                    "Value": 12.0
                },
                {
                    "Timestamp": 1540857720.0, //Minute2
                    "Value": 13.5
                },
                //... 60 datapoints for the os.cpuUtilization.idle.avg metric
            ]
        }
    ] //end of MetricList
} //end of response
```

La risposta presenta un `Identifier`, un `AlignedStartTime` e un `AlignedEndTime`. Poiché il valore `--period-in-seconds` era `60`, l'ora di inizio e fine è stata allineata al minuto. Se `--period-in-seconds` fosse stato `3600`, l'ora di inizio e fine sarebbe stata allineata all'ora.

`MetricList` nella risposta ha una serie di voci, ciascuna con una voce `Key` e una voce `DataPoints`. Ciascun `DataPoint` ha un `Timestamp` e un `Value`. Ciascun elenco `Datapoints` ha 60 punti di dati in quanto le query sono per dati al minuto nell'arco di un'ora, con `Timestamp1/Minute1`, `Timestamp2/Minute2` e così via, fino a `Timestamp60/Minute60`.

Poiché la query è per due diversi parametri contatore, la risposta contiene due element `MetricList`.

Recupero della media del carico del database per i principali eventi di attesa

L'esempio seguente mostra la stessa query che utilizza la Console di gestione AWS per generare un grafico a linee ad area in pila. L'esempio recupera `db.load.avg` per l'ultima ora con carico diviso in base ai sette principali eventi di attesa. Il comando è come quello in Recupero dei parametri contatore. Tuttavia, il file query.json presenta i seguenti contenuti.

```
[
    {
        "Metric": "db.load.avg",
        "GroupBy": { "Group": "db.wait_event", "Limit": 7 }
    }
```

```
]
```

Eseguire il comando riportato qui di seguito.

Per Linux, macOS o Unix:

```
aws pi get-resource-metrics \
   --service-type RDS \
   --identifier db-ID \
   --start-time 2018-10-30T00:00:00Z \
   --end-time   2018-10-30T01:00:00Z \
   --period-in-seconds 60 \
   --metric-queries file://query.json
```

Per Windows:

```
aws pi get-resource-metrics ^
   --service-type RDS ^
   --identifier db-ID ^
   --start-time 2018-10-30T00:00:00Z ^
   --end-time   2018-10-30T01:00:00Z ^
   --period-in-seconds 60 ^
   --metric-queries file://query.json
```

L'esempio specifica il parametro di `db.load.avg` e un `GroupBy` dei sette principali eventi di attesa. Per i dettagli sui valori validi per questo esempio, consulta DimensionGroup nella Documentazione di riferimento dell'API di Performance Insights.

La risposta è simile a quella riportata di seguito.

```
{
    "Identifier": "db-XXX",
    "AlignedStartTime": 1540857600.0,
    "AlignedEndTime": 1540861200.0,
    "MetricList": [
        { //A list of key/datapoints
            "Key": {
                //A Metric with no dimensions. This is the total db.load.avg
                "Metric": "db.load.avg"
            },
            "DataPoints": [
```

```
                //Each list of datapoints has the same timestamps and same number of
 items
                {
                    "Timestamp": 1540857660.0, //Minute1
                    "Value": 0.5166666666666667
                },
                {
                    "Timestamp": 1540857720.0, //Minute2
                    "Value": 0.38333333333333336
                },
                {
                    "Timestamp": 1540857780.0, //Minute 3
                    "Value": 0.26666666666666666
                }
                //... 60 datapoints for the total db.load.avg key
            ]
        },
        {
            "Key": {
                //Another key. This is db.load.avg broken down by CPU
                "Metric": "db.load.avg",
                "Dimensions": {
                    "db.wait_event.name": "CPU",
                    "db.wait_event.type": "CPU"
                }
            },
            "DataPoints": [
                {
                    "Timestamp": 1540857660.0, //Minute1
                    "Value": 0.35
                },
                {
                    "Timestamp": 1540857720.0, //Minute2
                    "Value": 0.15
                },
                //... 60 datapoints for the CPU key
            ]
        },
        //... In total we have 8 key/datapoints entries, 1) total, 2-8) Top Wait Events
    ] //end of MetricList
} //end of response
```

In questa risposta, ci sono otto voci in `MetricList`. C'è una voce per il `db.load.avg` totale e sette voci ciascuno per il `db.load.avg` suddivise secondo uno dei sette principali eventi di attesa. A differenza del primo esempio, poiché era presente una dimensione di raggruppamento, deve esserci una chiave per ciascun raggruppamento del parametro. Può esserci una sola chiave per ciascun parametro, come nel caso d'uso del parametro contatore di base.

Recupero della media del carico del database per il principale SQL

L'esempio seguente raggruppa `db.wait_events` in base alle 10 principali istruzioni SQL. Ci sono due diversi gruppi per le istruzioni SQL:

- `db.sql` – L'istruzione SQL completa, come `select * from customers where customer_id = 123`
- `db.sql_tokenized` – L'istruzione SQL in formato token, come `select * from customers where customer_id = ?`

Quando si analizzano le prestazioni del database, può essere utile considerare le istruzioni SQL che si differenziano solo per i loro parametri come un unico elemento logico. Pertanto, puoi utilizzare `db.sql_tokenized` durante le query. Tuttavia, soprattutto se ti interessano piani explain, a volte è più utile esaminare le istruzioni SQL complete con parametri e raggruppamento di query per `db.sql`. Vi è una relazione padre-figlio tra SQL in formato token e completo, con più SQL completi (figli) raggruppati nello stesso SQL in formato token (padre).

Il comando in questo esempio è simile a quello in Recupero della media del carico del database per i principali eventi di attesa. Tuttavia, il file query.json presenta i seguenti contenuti.

```
[
    {
        "Metric": "db.load.avg",
        "GroupBy": { "Group": "db.sql_tokenized", "Limit": 10 }
    }
]
```

Nell'esempio seguente viene utilizzato `db.sql_tokenized`.

Per Linux, macOS o Unix:

```
aws pi get-resource-metrics \
   --service-type RDS \
   --identifier db-ID \
```

```
   --start-time 2018-10-29T00:00:00Z \
   --end-time   2018-10-30T00:00:00Z \
   --period-in-seconds 3600 \
   --metric-queries file://query.json
```

Per Windows:

```
aws pi get-resource-metrics ^
   --service-type RDS ^
   --identifier db-ID ^
   --start-time 2018-10-29T00:00:00Z ^
   --end-time   2018-10-30T00:00:00Z  ^
   --period-in-seconds 3600 ^
   --metric-queries file://query.json
```

In questo esempio vengono eseguite query su 24 ore, con un periodo in secondi di un'ora.

L'esempio specifica il parametro di `db.load.avg` e un `GroupBy` dei sette principali eventi di attesa. Per i dettagli sui valori validi per questo esempio, consulta DimensionGroup nella Documentazione di riferimento dell'API di Performance Insights.

La risposta è simile a quella riportata di seguito.

```
{
    "AlignedStartTime": 1540771200.0,
    "AlignedEndTime": 1540857600.0,
    "Identifier": "db-XXX",

    "MetricList": [ //11 entries in the MetricList
        {
            "Key": { //First key is total
                "Metric": "db.load.avg"
            }
            "DataPoints": [ //Each DataPoints list has 24 per-hour Timestamps and a
 value
                {
                    "Value": 1.6964980544747081,
                    "Timestamp": 1540774800.0
                },
                //... 24 datapoints
            ]
        },
        {
```

```
            "Key": { //Next key is the top tokenized SQL
                "Dimensions": {
                    "db.sql_tokenized.statement": "INSERT INTO authors (id,name,email)
 VALUES\n( nextval(?)  ,?,?)",
                    "db.sql_tokenized.db_id": "pi-2372568224",
                    "db.sql_tokenized.id": "AKIAIOSFODNN7EXAMPLE"
                },
                "Metric": "db.load.avg"
            },
            "DataPoints": [ //... 24 datapoints
            ]
        },
        // In total 11 entries, 10 Keys of top tokenized SQL, 1 total key
    ] //End of MetricList
} //End of response
```

Questa risposta ha 11 voci in `MetricList` (1 SQL totale, 10 SQL principali in formato token), dove ciascuna ha 24 `DataPoints` ogni ora.

Per SQL in formato token, ci sono tre voci in ciascun elenco di dimensioni:

- `db.sql_tokenized.statement` – L'istruzione SQL in formato token.
- `db.sql_tokenized.db_id` – L'ID database nativo utilizzato per fare riferimento a SQL o un ID sintetico che Performance Insights genera nel caso in cui l'ID database nativo non sia disponibile. Questo esempio restituisce l'ID sintetico `pi-2372568224`.
- `db.sql_tokenized.id` – L'ID della query all'interno di Performance Insights.

  Nella Console di gestione AWS, questo ID è detto Support ID (ID supporto). Si chiama così perché è l'ID dei dati che Supporto AWS può effettuare una verifica per facilitare la risoluzione di un problema relativo al database. AWS prende molto seriamente la sicurezza e la privacy dei dati e quasi tutti i dati vengono archiviati crittografati con la chiave AWS KMS. Pertanto, nessuno all'interno di AWS può accedere a tali dati. Nell'esempio precedente, sia `tokenized.statement` che `tokenized.db_id` vengono archiviati crittografati. Se riscontri un problema con il database, AWS Support può aiutarti facendo riferimento al Support ID (ID supporto).

Quando si eseguo query, potrebbe essere utile specificare un `Group` in `GroupBy`. Tuttavia, per un controllo più dettagliato dei dati restituiti, occorre specificare l'elenco delle dimensioni. Ad esempio, se tutto ciò di cui si necessita è `db.sql_tokenized.statement`, è possibile aggiungere l'attributo `Dimensions` al file query.json.

```
[
    {
        "Metric": "db.load.avg",
        "GroupBy": {
            "Group": "db.sql_tokenized",
            "Dimensions":["db.sql_tokenized.statement"],
            "Limit": 10
        }
    }
]
```

Recupero della media del carico del database filtrata da SQL

L'immagine precedente mostra che è stata selezionata una particolare query e che il grafico a linee ad area in pila con sessioni attive della media in alto è definito in base a tale query. Sebbene la query sia ancora per i sette principali eventi di attesa complessivi, il valore della risposta è filtrato. Il filtro fa sì che vengano prese in considerazione solo le sessioni che corrispondono al filtro specifico.

La query dell'API corrispondente in questo esempio è simile al comando in Recupero della media del carico del database per il principale SQL. Tuttavia, il file query.json presenta i seguenti contenuti.

```
[
```

```
 {
        "Metric": "db.load.avg",
        "GroupBy": { "Group": "db.wait_event", "Limit": 5  },
        "Filter": { "db.sql_tokenized.id": "AKIAIOSFODNN7EXAMPLE" }
    }
]
```

Per Linux, macOS o Unix:

```
aws pi get-resource-metrics \
   --service-type RDS \
   --identifier db-ID \
   --start-time 2018-10-30T00:00:00Z \
   --end-time   2018-10-30T01:00:00Z \
   --period-in-seconds 60 \
   --metric-queries file://query.json
```

Per Windows:

```
aws pi get-resource-metrics ^
   --service-type RDS ^
   --identifier db-ID ^
   --start-time 2018-10-30T00:00:00Z ^
   --end-time   2018-10-30T01:00:00Z ^
   --period-in-seconds 60 ^
   --metric-queries file://query.json
```

La risposta è simile a quella riportata di seguito.

```
{
    "Identifier": "db-XXX",
    "AlignedStartTime": 1556215200.0,
    "MetricList": [
        {
            "Key": {
                "Metric": "db.load.avg"
            },
            "DataPoints": [
                {
                    "Timestamp": 1556218800.0,
                    "Value": 1.4878117913832196
                },
```

```
                {
                    "Timestamp": 1556222400.0,
                    "Value": 1.192823803967328
                }
            ]
        },
        {
            "Key": {
                "Metric": "db.load.avg",
                "Dimensions": {
                    "db.wait_event.type": "io",
                    "db.wait_event.name": "wait/io/aurora_redo_log_flush"
                }
            },
            "DataPoints": [
                {
                    "Timestamp": 1556218800.0,
                    "Value": 1.1360544217687074
                },
                {
                    "Timestamp": 1556222400.0,
                    "Value": 1.058051341890315
                }
            ]
        },
        {
            "Key": {
                "Metric": "db.load.avg",
                "Dimensions": {
                    "db.wait_event.type": "io",
                    "db.wait_event.name": "wait/io/table/sql/handler"
                }
            },
            "DataPoints": [
                {
                    "Timestamp": 1556218800.0,
                    "Value": 0.16241496598639457
                },
                {
                    "Timestamp": 1556222400.0,
                    "Value": 0.05163360560093349
                }
            ]
        },
```

```
        {
            "Key": {
                "Metric": "db.load.avg",
                "Dimensions": {
                    "db.wait_event.type": "synch",
                    "db.wait_event.name": "wait/synch/mutex/innodb/
aurora_lock_thread_slot_futex"
                }
            },
            "DataPoints": [
                {
                    "Timestamp": 1556218800.0,
                    "Value": 0.11479591836734694
                },
                {
                    "Timestamp": 1556222400.0,
                    "Value": 0.013127187864644107
                }
            ]
        },
        {
            "Key": {
                "Metric": "db.load.avg",
                "Dimensions": {
                    "db.wait_event.type": "CPU",
                    "db.wait_event.name": "CPU"
                }
            },
            "DataPoints": [
                {
                    "Timestamp": 1556218800.0,
                    "Value": 0.05215419501133787
                },
                {
                    "Timestamp": 1556222400.0,
                    "Value": 0.05805134189031505
                }
            ]
        },
        {
            "Key": {
                "Metric": "db.load.avg",
                "Dimensions": {
                    "db.wait_event.type": "synch",
```

```
                    "db.wait_event.name": "wait/synch/mutex/innodb/lock_wait_mutex"
                }
            },
            "DataPoints": [
                {
                    "Timestamp": 1556218800.0,
                    "Value": 0.017573696145124718
                },
                {
                    "Timestamp": 1556222400.0,
                    "Value": 0.002333722287047841
                }
            ]
        }
    ],
    "AlignedEndTime": 1556222400.0
} //end of response
```

In questa risposta, tutti i valori sono filtrati in base al contributo di SQL in formato token AKIAIOSFODNN7EXAMPLE specificato nel file query.json. Le chiavi potrebbero inoltre seguire un ordine diverso rispetto a una query senza filtro, in quanto sono i cinque principali eventi di attesa che influenzano l'SQL filtrato.

Recupero del testo completo di un'istruzione SQL

L'esempio seguente recupera il testo completo di un'istruzione SQL per l'istanza database `db-10BCD2EFGHIJ3KL4M5NO6PQRS5`. `--group` è `db.sql`, e `--group-identifier` è `db.sql.id`. In questo esempio, *`my-sql-id`* rappresenta un ID SQL recuperato richiamando `pi get-resource-metrics` o `pi describe-dimension-keys`.

Eseguire il comando riportato qui di seguito.

Per Linux, macOS o Unix:

```
aws pi get-dimension-key-details \
   --service-type RDS \
   --identifier db-10BCD2EFGHIJ3KL4M5NO6PQRS5 \
   --group db.sql \
   --group-identifier my-sql-id \
   --requested-dimensions statement
```

Per Windows:

```
aws pi get-dimension-key-details ^
   --service-type RDS ^
   --identifier db-10BCD2EFGHIJ3KL4M5NO6PQRS5 ^
   --group db.sql ^
   --group-identifier my-sql-id ^
   --requested-dimensions statement
```

In questo esempio, sono disponibili i dettagli delle dimensioni. Pertanto, Performance Insights recupera il testo completo dell'istruzione SQL, senza troncarla.

```
{
    "Dimensions":[
    {
        "Value": "SELECT e.last_name, d.department_name FROM employees e, departments d
 WHERE e.department_id=d.department_id",
        "Dimension": "db.sql.statement",
        "Status": "AVAILABLE"
    },
    ...
    ]
}
```

Creazione di un report di analisi delle prestazioni per un periodo di tempo

L'esempio seguente crea un report di analisi delle prestazioni con l'ora di inizio `1682969503` e l'ora di fine `1682979503` per il database `db-loadtest-0`.

```
aws pi create-performance-analysis-report \
        --service-type RDS \
        --identifier db-loadtest-0 \
        --start-time 1682969503 \
        --end-time 1682979503 \
        --region us-west-2
```

La risposta è l'identificatore univoco `report-0234d3ed98e28fb17` per il report.

```
{
   "AnalysisReportId": "report-0234d3ed98e28fb17"
}
```

Recupero di un report di analisi delle prestazioni

L'esempio seguente recupera i dettagli del report di analisi per il report `report-0d99cc91c4422ee61`.

```
aws pi get-performance-analysis-report \
--service-type RDS \
--identifier db-loadtest-0 \
--analysis-report-id report-0d99cc91c4422ee61 \
--region us-west-2
```

La risposta fornisce lo stato, l'ID, i dettagli temporali e gli approfondimenti del report.

```
        {
    "AnalysisReport": {
        "Status": "Succeeded",
        "ServiceType": "RDS",
        "Identifier": "db-loadtest-0",
        "StartTime": 1680583486.584,
        "AnalysisReportId": "report-0d99cc91c4422ee61",
        "EndTime": 1680587086.584,
        "CreateTime": 1680587087.139,
        "Insights": [
           ... (Condensed for space)
        ]
    }
}
```

Elenco di tutti i report di analisi delle prestazioni per l'istanza database

L'esempio seguente elenca tutti i report di analisi delle prestazioni disponibili per il database `db-loadtest-0`.

```
aws pi list-performance-analysis-reports \
--service-type RDS \
--identifier db-loadtest-0 \
--region us-west-2
```

La risposta elenca tutti i report con i dettagli relativi all'ID, allo stato e al periodo di tempo del report.

```
{
```

```
    "AnalysisReports": [
        {
            "Status": "Succeeded",
            "EndTime": 1680587086.584,
            "CreationTime": 1680587087.139,
            "StartTime": 1680583486.584,
            "AnalysisReportId": "report-0d99cc91c4422ee61"
        },
        {
            "Status": "Succeeded",
            "EndTime": 1681491137.914,
            "CreationTime": 1681491145.973,
            "StartTime": 1681487537.914,
            "AnalysisReportId": "report-002633115cc002233"
        },
        {
            "Status": "Succeeded",
            "EndTime": 1681493499.849,
            "CreationTime": 1681493507.762,
            "StartTime": 1681489899.849,
            "AnalysisReportId": "report-043b1e006b47246f9"
        },
        {
            "Status": "InProgress",
            "EndTime": 1682979503.0,
            "CreationTime": 1682979618.994,
            "StartTime": 1682969503.0,
            "AnalysisReportId": "report-01ad15f9b88bcbd56"
        }
    ]
}
```

Eliminazione di un report di analisi delle prestazioni

L'esempio seguente elimina il report di analisi per il database `db-loadtest-0`.

```
aws pi delete-performance-analysis-report \
--service-type RDS \
--identifier db-loadtest-0 \
--analysis-report-id report-0d99cc91c4422ee61 \
--region us-west-2
```

Aggiunta di un tag a un report di analisi delle prestazioni

L'esempio seguente aggiunge un tag con una chiave `name` e un valore `test-tag` al report `report-01ad15f9b88bcbd56`.

```
aws pi tag-resource \
--service-type RDS \
--resource-arn arn:aws:pi:us-west-2:356798100956:perf-reports/RDS/db-loadtest-0/
report-01ad15f9b88bcbd56 \
--tags Key=name,Value=test-tag \
--region us-west-2
```

Elenco di tutti i tag per un report di analisi delle prestazioni

Nell'esempio seguente vengono elencati tutti i tag per il report `report-01ad15f9b88bcbd56`.

```
aws pi list-tags-for-resource \
--service-type RDS \
--resource-arn arn:aws:pi:us-west-2:356798100956:perf-reports/RDS/db-loadtest-0/
report-01ad15f9b88bcbd56 \
--region us-west-2
```

La risposta elenca il valore e la chiave per tutti i tag aggiunti al report:

```
{
    "Tags": [
        {
            "Value": "test-tag",
            "Key": "name"
        }
    ]
}
```

Eliminazione di tag da un report di analisi delle prestazioni

Nell'esempio seguente viene eliminato il tag `name` dal report `report-01ad15f9b88bcbd56`.

```
aws pi untag-resource \
--service-type RDS \
--resource-arn arn:aws:pi:us-west-2:356798100956:perf-reports/RDS/db-loadtest-0/
report-01ad15f9b88bcbd56 \
```

```
--tag-keys name \
--region us-west-2
```

Dopo che il tag è stato eliminato, se si chiama l'API `list-tags-for-resource` questo tag non viene elencato.

Registrazione di log delle chiamate di Performance Insights mediante AWS CloudTrail

Performance Insights viene eseguito con AWS CloudTrail, un servizio che fornisce un record delle azioni intraprese da un utente, un ruolo o un servizio AWS in Performance Insights. CloudTrail acquisisce tutte le chiamate API per Performance Insights come eventi. Questa acquisizione include chiamate dalla console Amazon RDS e dalle chiamate di codice alle operazioni API Performance Insights.

Se viene creato un trail, è possibile abilitare la distribuzione continua di eventi CloudTrail in un bucket Simple Storage Service (Amazon S3), inclusi gli eventi per Performance Insights. Se non configuri un trail, è comunque possibile visualizzare gli eventi più recenti nella console di CloudTrail in Event history (Cronologia eventi). Utilizzando le informazioni raccolte da CloudTrail è possibile determinare specifici dettagli. Queste informazioni includono la richiesta effettuata a Performance Insights, l'indirizzo IP da cui è stata eseguita la richiesta, l'autore della richiesta e il momento in cui è stata eseguita. Include anche dettagli aggiuntivi.

Per ulteriori informazioni su CloudTrail, consulta la Guida per l'utente di AWS CloudTrail.

Utilizzo delle informazioni di Performance Insights in CloudTrail

CloudTrail è abilitato sull'account AWS al momento della sua creazione. Quando si verifica un'attività in Performance Insights, tale attività viene registrata in un evento CloudTrail insieme ad altri eventi del servizio AWS nella console CloudTrail in Cronologia eventi. È possibile visualizzare, cercare e scaricare gli eventi recenti nell'account AWS. Per ulteriori informazioni, consulta Visualizzazione di eventi nella cronologia degli eventi di CloudTrail nella Guida per l'utente di AWS CloudTrail.

Per una registrazione di eventi nell'account AWS che includa eventi per Performance Insights, crea un trail. Un trail consente a CloudTrail di distribuire i file di log in un bucket Simple Storage Service (Amazon S3). Per impostazione predefinita, quando si crea un trail nella console, il trail sarà valido in tutte le Regioni AWS. Il percorso registra gli eventi da tutte le regioni AWSnella partizione AWS e distribuisce i file di log nel bucket Simple Storage Service (Amazon S3) specificato. Inoltre, è possibile configurare altri servizi AWS per analizzare con maggiore dettaglio e usare i dati evento raccolti nei

log CloudTrail. Per ulteriori informazioni, consulta gli argomenti seguenti nella Guida per l'utente di AWS CloudTrail:

- Panoramica della creazione di un trail
- Servizi e integrazioni CloudTrail supportati
- Configurazione delle notifiche Amazon SNS per CloudTrail
- Ricezione di file di log CloudTrail da più Regioni e Ricezione di file di log CloudTrail da più account

Tutte le operazioni di Performance Insights vengono registrate da CloudTrail e documentate nella Documentazione di riferimento dell'API di Performance Insights. Ad esempio, tutte le chiamate alle operazioni `DescribeDimensionKeys` e `GetResourceMetrics` generano voci nei file di log di CloudTrail.

Ogni evento o voce di log contiene informazioni sull'utente che ha generato la richiesta. Le informazioni di identità consentono di determinare quanto segue:

- Se la richiesta è stata effettuata con le credenziali dell'utente IAM o root.
- Se la richiesta è stata effettuata con le credenziali di sicurezza temporanee per un ruolo o un utente federato.
- Se la richiesta è stata effettuata da un altro servizio AWS.

Per ulteriori informazioni, vedi Elemento userIdentity di CloudTrail.

Voci del file di registro Performance Insights

Un percorso è una configurazione che consente la distribuzione di eventi come i file di log in un bucket Simple Storage Service (Amazon S3) specificato. I file di log di CloudTrail possono contenere una o più voci di log. Un evento rappresenta una singola richiesta da un'origine. Ogni evento include informazioni sull'operazione richiesta, la data e l'ora dell'azione, i parametri della richiesta e così via. I file di log CloudTrail non sono una traccia di stack ordinata delle chiamate API pubbliche e di conseguenza non devono apparire in base a un ordine specifico.

L'esempio seguente mostra una voce di log di CloudTrail che illustra l'operazione `GetResourceMetrics`:

```
{
    "eventVersion": "1.05",
```

```
    "userIdentity": {
        "type": "IAMUser",
         "principalId": "AKIAIOSFODNN7EXAMPLE",
        "arn": "arn:aws:iam::123456789012:user/johndoe",
        "accountId": "123456789012",
        "accessKeyId": "AKIAI44QH8DHBEXAMPLE",
        "userName": "johndoe"
    },
    "eventTime": "2019-12-18T19:28:46Z",
    "eventSource": "pi.amazonaws.com",
    "eventName": "GetResourceMetrics",
    "awsRegion": "us-east-1",
    "sourceIPAddress": "72.21.198.67",
    "userAgent": "aws-cli/1.16.240 Python/3.7.4 Darwin/18.7.0 botocore/1.12.230",
    "requestParameters": {
        "identifier": "db-YTDU5J5V66X7CXSCVDFD2V3SZM",
        "metricQueries": [
            {
                "metric": "os.cpuUtilization.user.avg"
            },
            {
                "metric": "os.cpuUtilization.idle.avg"
            }
        ],
        "startTime": "Dec 18, 2019 5:28:46 PM",
        "periodInSeconds": 60,
        "endTime": "Dec 18, 2019 7:28:46 PM",
        "serviceType": "RDS"
    },
    "responseElements": null,
    "requestID": "9ffbe15c-96b5-4fe6-bed9-9fccff1a0525",
    "eventID": "08908de0-2431-4e2e-ba7b-f5424f908433",
    "eventType": "AwsApiCall",
    "recipientAccountId": "123456789012"
}
```


API Approfondimenti sulle prestazioni ed endpoint VPC di interfaccia (AWS PrivateLink)

È possibile utilizzare AWS PrivateLink per creare una connessione privata tra il VPC e Approfondimenti sulle prestazioni di Amazon RDS. È possibile accedere ad Approfondimenti sulle prestazioni come se fosse nel VPC, senza utilizzare un gateway Internet, un dispositivo NAT, una

connessione VPN o una connessione Direct Connect. Le istanze nel VPC non richiedono indirizzi IP pubblici per l'accesso ad Approfondimenti sulle prestazioni.

Stabilisci questa connessione privata creando un endpoint di interfaccia attivato da AWS PrivateLink. In ciascuna sottorete viene creata un'interfaccia di rete endpoint da abilitare per l'endpoint di interfaccia. Queste sono interfacce di rete gestite dal richiedente che fungono da punto di ingresso per il traffico destinato ad Approfondimenti sulle prestazioni.

Per ulteriori informazioni, consulta la sezione Accesso a Servizi AWS tramite AWS PrivateLink nella Guida di AWS PrivateLink.

Considerazioni su Approfondimenti sulle prestazioni

Prima di configurare un endpoint VPC di interfaccia per Approfondimenti sulle prestazioni, consulta Considerations nella Guida di AWS PrivateLink.

Approfondimenti sulle prestazioni supporta l'esecuzione di chiamate a tutte le azioni API tramite l'endpoint dell'interfaccia.

Per impostazione predefinita, l'accesso completo ad Approfondimenti sulle prestazioni è consentito attraverso l'endpoint dell'interfaccia. Per controllare il traffico verso Approfondimenti sulle prestazioni tramite l'endpoint dell'interfaccia, associare un gruppo di sicurezza alle interfacce di rete dell'endpoint.

Disponibilità

L'API Approfondimenti sulle prestazioni attualmente supporta gli endpoint VPC in Regioni AWS che supportano Approfondimenti sulle prestazioni. Per informazioni su Approfondimenti sulle prestazioni, consulta Regioni e motori di database Aurora supportati per Approfondimenti sulle prestazioni.

Creare un endpoint dell'interfaccia per Approfondimenti sulle prestazioni

È possibile creare un endpoint dell'interfaccia per Approfondimenti sulle prestazioni utilizzando la console Amazon VPC o AWS Command Line Interface (AWS CLI). Per ulteriori informazioni, consulta la sezione Creazione di un endpoint di interfaccia nella Guida per l'utente di AWS PrivateLink.

Creare un endpoint dell'interfaccia per Approfondimenti sulle prestazioni utilizzando il seguente nome servizio:

Se si abilita il DNS privato per l'endpoint dell'interfaccia, è possibile effettuare richieste API ad Approfondimenti sulle prestazioni utilizzando il nome DNS predefinito per la Regione. Ad esempio, `pi.us-east-1.amazonaws.com`.

Creazione di una policy di endpoint VPC per l’API Approfondimenti sulle prestazioni

Una policy dell'endpoint è una risorsa IAM che è possibile allegare all'endpoint dell'interfaccia. La policy di endpoint predefinita permette l’accesso completo ad Approfondimenti sulle prestazioni tramite l’endpoint dell’interfaccia. Per controllare l’accesso consentito ad Approfondimenti sulle prestazioni dal VPC, associare una policy dell’endpoint personalizzata all’endpoint dell’interfaccia.

Una policy di endpoint specifica le informazioni riportate di seguito:

- I principali che possono eseguire azioni (Account AWS, utenti IAM e ruoli IAM).
- Le azioni che possono essere eseguite.
- Le risorse in cui è possibile eseguire le operazioni.

Per ulteriori informazioni, consulta la sezione Controllo dell'accesso ai servizi con policy di endpoint nella Guida di AWS PrivateLink.

Esempio: policy di endpoint VPC per le azioni di Approfondimenti sulle prestazioni

Di seguito è riportato l'esempio di una policy dell'endpoint personalizzata. Se associata a un endpoint dell’interfaccia, questa policy concede l’accesso alle azioni di Approfondimenti sulle prestazioni elencate per tutti i principali su tutte le risorse.

```
{
   "Statement":[
      {
         "Principal":"*",
         "Effect":"Allow",
         "Action":[
            "rds:CreatePerformanceAnalysisReport",
            "rds:DeletePerformanceAnalysisReport",
            "rds:GetPerformanceAnalysisReport"
         ],
         "Resource":"*"
      }
   ]
}
```

Esempio: policy VPC Endpoint che nega tutti gli accessi da un account AWS specificato

La seguente policy di endpoint VPC nega all'account AWS `123456789012` l'accesso completo alle risorse utilizzando l'endpoint. La policy consente tutte le operazioni da altri account.

```
{
  "Statement": [
    {
      "Action": "*",
      "Effect": "Allow",
      "Resource": "*",
      "Principal": "*"
    },
    {
      "Action": "*",
      "Effect": "Deny",
      "Resource": "*",
      "Principal": { "AWS": [ "123456789012" ] }
     }
   ]
}
```

Indirizzamento IP per Approfondimenti sulle prestazioni

Gli indirizzi IP permettono alle risorse nel VPC di comunicare tra loro e con le risorse su Internet. Approfondimenti sulle prestazioni supporta i protocolli di indirizzamento IPv4 e IPv6. Per impostazione predefinita, Approfondimenti sulle prestazioni e Amazon VPC utilizzano il protocollo di indirizzamento IPv4. Non puoi disattivare questo comportamento. Quando crei un VPC, assicurati di specificare un blocco CIDR IPv4 (un intervallo di indirizzi IPv4 privati).

È possibile scegliere di assegnare un intervallo CIDR IPv6 al VPC e alle sottoreti e di assegnare gli indirizzi IPv6 del blocco alle risorse RDS presenti nella sottorete. Il supporto del protocollo IPv6 espande il numero di indirizzi IP supportati. L'utilizzo del protocollo IPv6 consente di disporre di un numero sufficiente di indirizzi per adeguarsi alla futura espansione di Internet. Le risorse RDS nuove ed esistenti possono utilizzare indirizzi IPv4 e IPv6 all'interno del VPC. La configurazione, la protezione e la traduzione del traffico di rete tra i due protocolli utilizzati in diverse parti di un'applicazione può causare sovraccarico operativo. Puoi standardizzare il protocollo IPv6 per le risorse Amazon RDS per semplificare la configurazione di rete. Per ulteriori informazioni sugli endpoint e le quote del servizio, consulta Endpoint e quote di Amazon Relational Database Service.

Per ulteriori informazioni sull'indirizzamento IP di Aurora, consulta Indirizzamento IP Aurora.

Analisi delle anomalie prestazionali di Aurora con DevOps Amazon Guru per Amazon RDS

Amazon DevOps Guru è un servizio operativo completamente gestito che aiuta sviluppatori e operatori a migliorare le prestazioni e la disponibilità delle loro applicazioni. DevOpsGuru delega le attività associate all'identificazione dei problemi operativi in modo da poter implementare rapidamente i consigli per migliorare la tua applicazione. Per ulteriori informazioni, consulta Cos'è Amazon DevOps Guru? nella Guida per l'utente di Amazon DevOps Guru.

DevOpsGuru rileva, analizza e fornisce raccomandazioni per i problemi operativi esistenti per tutti i motori Amazon RDS DB. DevOpsGuru for RDS estende questa funzionalità applicando l'apprendimento automatico ai parametri di Performance Insights per i database Amazon Aurora per PostgreSQL. Queste funzionalità di monitoraggio consentono a DevOps Guru for RDS di rilevare e diagnosticare i rallentamenti delle prestazioni e consigliare azioni correttive specifiche. DevOpsGuru for RDS può anche rilevare condizioni problematiche nei database Aurora (database RDS) prima che si verifichino.

Ora puoi visualizzare questi consigli nella console RDS. Per ulteriori informazioni, consulta Suggerimenti di Amazon Aurora.

> **Important**
>
> Amazon DevOps Guru non è disponibile per i cluster Aurora Serverless DB.

Il video seguente è una panoramica di DevOps Guru for RDS.

Per un'analisi approfondita di questo argomento, consulta Amazon DevOps Guru for RDS under the hood.

Argomenti

- Vantaggi di DevOps Guru for RDS
- Come funziona DevOps Guru for RDS
- Configurazione di Guru per RDS DevOps

Vantaggi di DevOps Guru for RDS

Se sei responsabile di un database Amazon Aurora, potresti non sapere che si sta verificando un evento o una regressione che interessa il database. Quando scopri il problema, potresti non sapere perché si sta verificando o cosa fare al riguardo. Invece di rivolgerti a un amministratore di database (DBA) per ricevere assistenza o affidarti a strumenti di terze parti, puoi seguire i consigli di Guru for RDS. DevOps

L'analisi dettagliata di Guru for RDS consente di DevOps ottenere i seguenti vantaggi:

Diagnosi rapida

DevOpsGuru for RDS monitora e analizza continuamente la telemetria del database. DevOpsGuru for RDS utilizza tecniche statistiche e di apprendimento automatico per estrarre questi dati e rilevare anomalie. Per ulteriori informazioni sui dati di telemetria, consulta Monitoraggio del carico del DB con Approfondimenti sulle prestazioni su Amazon Aurora e Monitoraggio delle metriche del sistema operativo con monitoraggio avanzato nella Guida per l'utente di Amazon Aurora .

Risoluzione rapida

Ogni anomalia identifica il problema delle prestazioni e suggerisce strade di indagine o azioni correttive. Ad esempio, DevOps Guru for RDS potrebbe consigliare di esaminare specifici eventi di attesa. In alternativa, è consigliabile regolare le impostazioni del pool di applicazioni per limitare il numero di connessioni al database. Sulla base di questi consigli, è possibile risolvere i problemi di prestazioni più rapidamente rispetto alla risoluzione manuale dei problemi.

Approfondimenti proattivi

DevOpsGuru for RDS utilizza le metriche delle tue risorse per rilevare comportamenti potenzialmente problematici prima che diventino un problema più grave. Ad esempio, è in grado di rilevare quando il database utilizza un numero crescente di tabelle temporanee su disco, il che potrebbe iniziare a influire sulle prestazioni. DevOps Guru fornisce quindi consigli per aiutarti a risolvere i problemi prima che diventino problemi più grandi.

Conoscenza approfondita dei tecnici e del machine learning di Amazon

Per rilevare problemi di prestazioni e aiutarti a risolvere i problemi, DevOps Guru for RDS si affida all'apprendimento automatico (ML) e a formule matematiche avanzate. Gli ingegneri di database di Amazon hanno contribuito allo sviluppo dei risultati di DevOps Guru for RDS, che racchiudono molti anni di gestione di centinaia di migliaia di database. Attingendo a questa conoscenza collettiva, DevOps Guru for RDS può insegnarti le migliori pratiche.

Come funziona DevOps Guru for RDS

DevOpsGuru for RDS raccoglie dati sui database Aurora RDS per . La `DBLoad` metrica più importante è. DevOpsGuru for RDS utilizza le metriche di Performance Insights, le analizza con l'apprendimento automatico e pubblica le informazioni sulla dashboard.

Un'analisi è una raccolta di anomalie correlate rilevate da Guru. DevOps

Approfondimenti proattivi

Un approfondimento proattivo consente di individuare i comportamenti problematici prima che si verifichino. Contiene le anomalie accompagnate da suggerimenti e metriche correlati per aiutarti a risolvere le condizioni problematiche nei tuoi database Amazon Aurora prima che diventino problemi più seri. Questi approfondimenti sono pubblicati nella dashboard Guru. DevOps

Ad esempio, DevOps Guru potrebbe rilevare che il database Aurora PostgreSQL sta creando molte tabelle temporanee su disco. Se non affrontata per tempo, questa tendenza può causare problemi di prestazioni. Ogni approfondimento proattivo include i suggerimenti per i comportamenti correttivi e i collegamenti ad argomenti pertinenti in Ottimizzazione di Aurora MySQL con approfondimenti proattivi di Amazon DevOps Guru o Ottimizzazione di DevOps. Per ulteriori informazioni, consulta Working with Insights in DevOps Guru nella Amazon DevOps Guru User Guide.

Approfondimenti reattivi

Un approfondimento reattivo identifica un comportamento anomalo nel momento in cui si verifica. Se DevOps Guru for RDS rileva problemi di prestazioni nelle tue istanze DB Amazon Aurora , pubblica una panoramica reattiva nella dashboard Guru. DevOps Per ulteriori informazioni, consulta Working with Insights in DevOps Guru nella Amazon DevOps Guru User Guide.

Anomalie causali

Un'anomalia causale è un'anomalia di livello superiore all'interno di un approfondimento reattivo. Il caricamento del database (caricamento del DB) è l'anomalia causale di Guru for DevOps RDS.

Un'anomalia misura l'impatto sulle prestazioni assegnando un livello di gravità di Elevato,Medio, oppureBasso. Per ulteriori informazioni, consulta Concetti chiave per DevOps Guru for RDS nella Amazon DevOps Guru User Guide.

Se DevOps Guru rileva un'anomalia corrente sulla tua istanza DB, verrai avvisato nella pagina Databases della console RDS. La console ti avvisa anche delle anomalie che si sono verificate

nelle ultime 24 ore. Per andare alla pagina delle anomalie dalla console RDS, scegliere il link nel messaggio di avviso. La console RDS ti avvisa anche nella pagina del cluster di database Amazon Aurora .

Anomalie contestuali

Un'anomalia contestuale è un risultato del carico del database correlato a un approfondimento reattivo. Ogni anomalia contestuale descrive uno specifico problema di prestazioni di Amazon Aurora che richiede un'indagine. Ad esempio, DevOps Guru for RDS potrebbe consigliare di prendere in considerazione l'aumento della capacità della CPU o di esaminare gli eventi di attesa che contribuiscono al carico del DB.

> **Important**
>
> È consigliabile testare eventuali modifiche in un'istanza di test prima di modificare un'istanza di produzione. In questo modo, capisci l'impatto del cambiamento.

Per ulteriori informazioni, consulta la sezione Analyzing anomalies in Amazon RDS nella Amazon Guru User Guide. DevOps

Configurazione di Guru per RDS DevOps

Per consentire a DevOps Guru for Amazon RDS di pubblicare approfondimenti per un database Amazon Aurora , completa le seguenti attività.

Argomenti

- Configurazione delle politiche di accesso IAM per Guru for RDS DevOps
- Attivazione di Approfondimenti sulle prestazioni per le istanze database Aurora
- Attivare DevOps Guru e specificare la copertura delle risorse

Configurazione delle politiche di accesso IAM per Guru for RDS DevOps

Per visualizzare gli avvisi di DevOps Guru nella console RDS, il tuo utente o ruolo AWS Identity and Access Management (IAM) deve disporre di una delle seguenti politiche:

- La politica gestita AWS `AmazonDevOpsGuruConsoleFullAccess`

- La politica AWS gestita `AmazonDevOpsGuruConsoleReadOnlyAccess` e una delle seguenti politiche:
  - La politica AWS gestita `AmazonRDSFullAccess`
  - Una policy gestita dal cliente che include `pi:GetResourceMetrics` e `pi:DescribeDimensionKeys`

Per ulteriori informazioni, consulta Configurazione delle policy di accesso per Performance Insights.

Attivazione di Approfondimenti sulle prestazioni per le istanze database Aurora

DevOpsGuru for RDS si affida a Performance Insights per i suoi dati. Senza Performance Insights, DevOps Guru pubblica le anomalie, ma non include analisi e raccomandazioni dettagliate.

Quando crei un cluster database Aurora o modifichi un'istanza cluster, puoi attivare Performance Insights. Per ulteriori informazioni, consulta Attivazione e disattivazione di Approfondimenti sulle prestazioni per Aurora.

Attivare DevOps Guru e specificare la copertura delle risorse

Puoi attivare DevOps Guru per fargli monitorare i tuoi database Amazon PostgreSQL in uno dei seguenti modi.

Argomenti

- Attivazione di Guru nella console RDS DevOps
- Aggiungere risorse Aurora RDS DevOps
- Aggiungere risorse Aurora RDSCloudFormation

Attivazione di Guru nella console RDS DevOps

Puoi seguire più percorsi nella console Amazon RDS per attivare DevOps Guru.

Argomenti

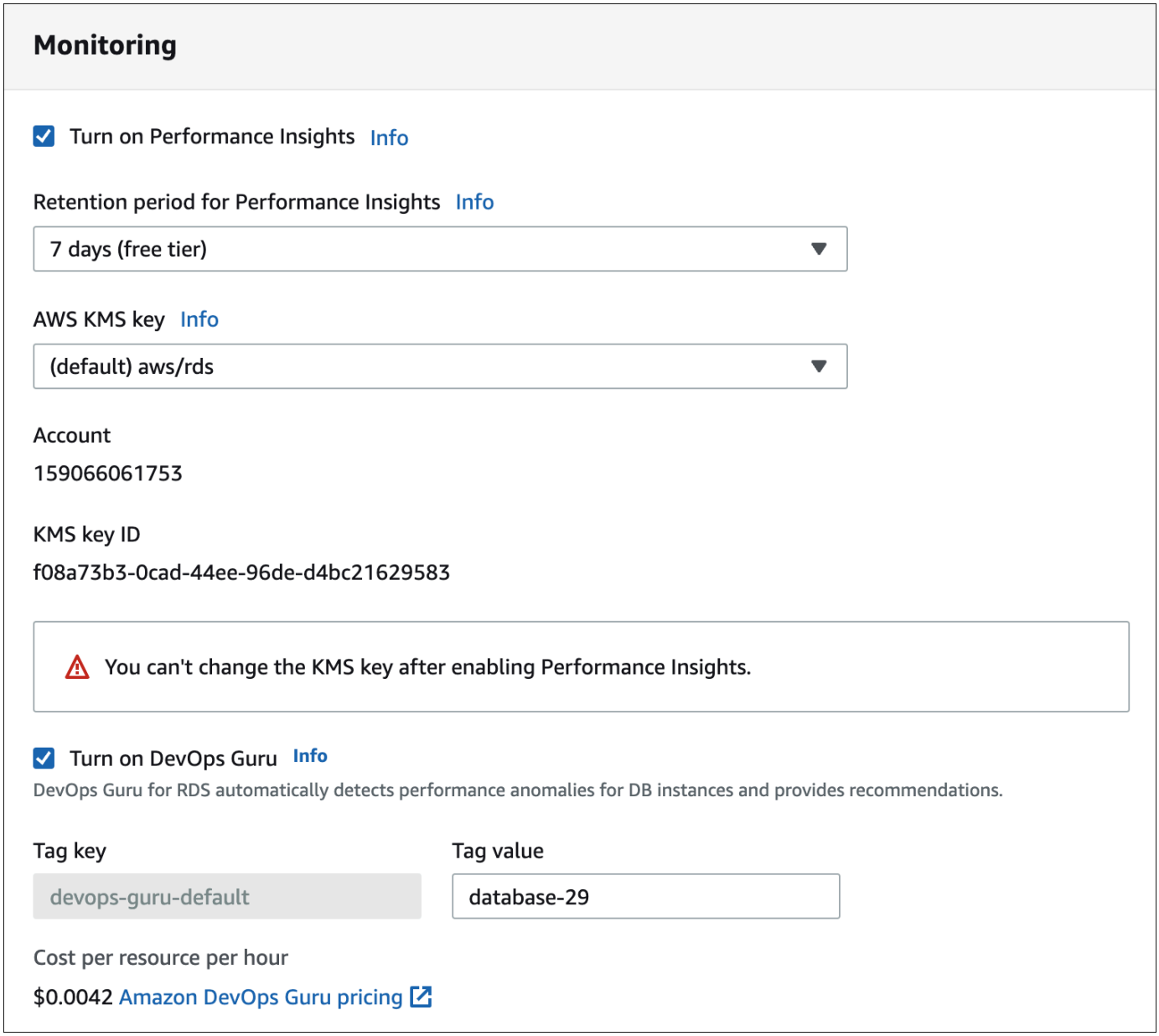
- Attivazione di DevOps Guru quando si crea un database per PostgreSQL
- Attivazione di DevOps Guru dal banner di notifica
- Risposta a un errore di autorizzazione quando attivi Guru DevOps

Attivazione di DevOps Guru quando si crea un database per PostgreSQL

Il flusso di lavoro di creazione include un'impostazione che attiva la copertura Guru per il database DevOps. Questa impostazione è abilitata per default quando scegli il modello Production (Produzione).

Per attivare DevOps Guru quando si crea un database per PostgreSQL

1. Accedi a Console di gestione AWS e apri la console Amazon RDS all'indirizzo https://console.aws.amazon.com/rds/.
2. Segui la procedura riportata in Creazione di un cluster di database, fino al passaggio, senza includerlo, in cui scegli le impostazioni di monitoraggio.
3. In Monitoring (Monitoraggio), scegli Turn on Performance Insights (Attiva Performance Insights). DevOpsAffinché Guru for RDS fornisca un'analisi dettagliata delle anomalie delle prestazioni, è necessario attivare Performance Insights.
4. Scegli Turn on Guru. DevOps



5. Crea un tag per il tuo database in modo che DevOps Guru possa monitorarlo. Esegui questa operazione:

   - Nel campo di testo per Tag key (Chiave tag), inserisci un nome che inizi con **`Devops-Guru-`**.
   - Nel campo di testo per Tag value (Valore tag), inserisci qualsiasi valore. Ad esempio, se specifichi **`rds-database-1`** come nome del database Aurora, puoi inserire anche **`rds-database-1`** come valore del tag.

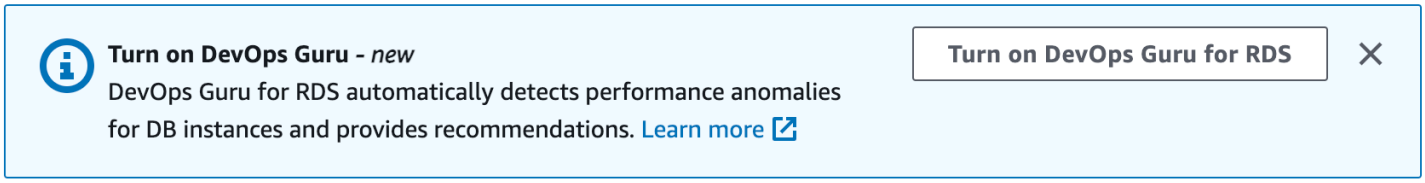
   Per ulteriori informazioni sui tag, consulta "Usa i tag per identificare le risorse nelle tue applicazioni DevOps Guru" nella Amazon DevOps Guru User Guide.

6. Completare le fasi restanti in Creazione di un cluster di database.

Attivazione di DevOps Guru dal banner di notifica

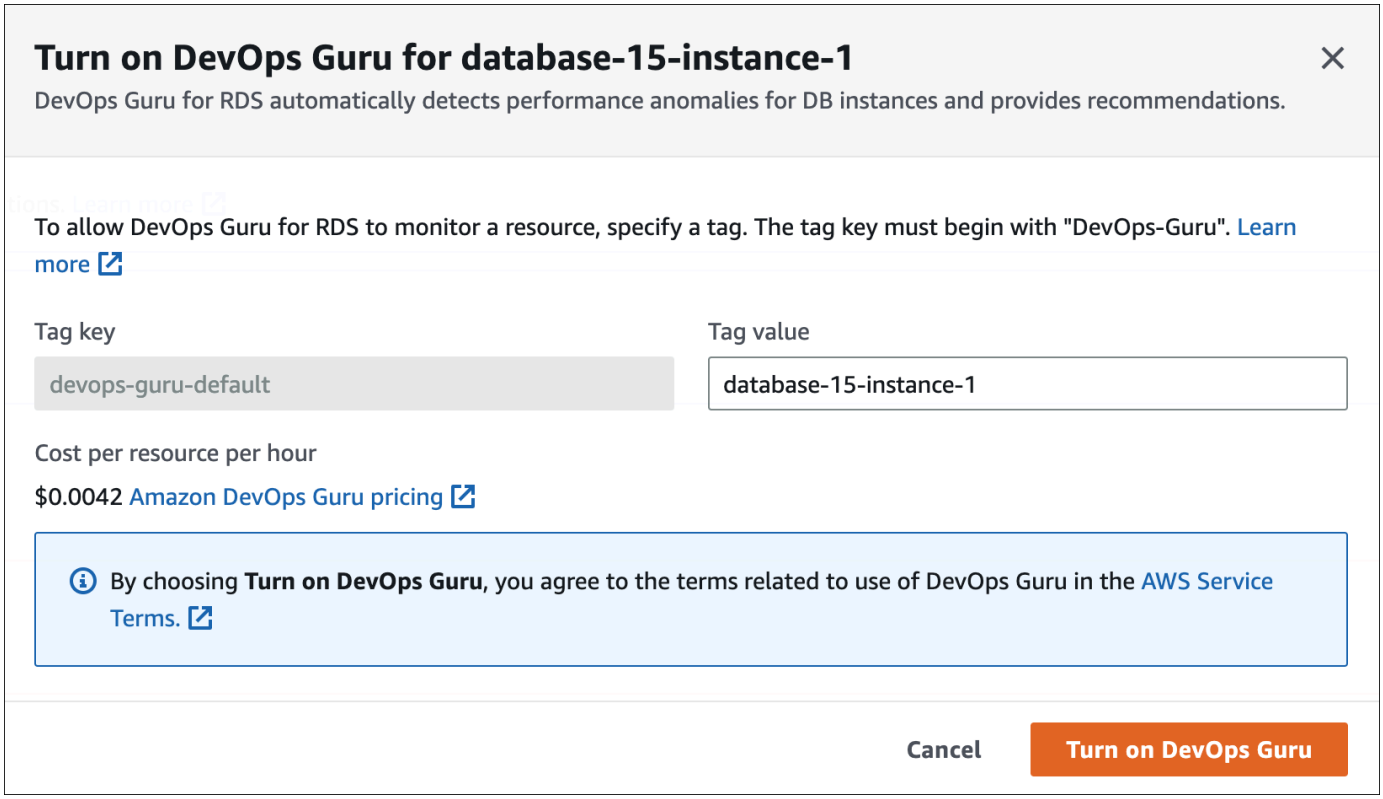
Se le tue risorse non sono coperte da DevOps Guru, Amazon RDS ti avvisa con un banner nelle seguenti posizioni:

- La scheda Monitoring (Monitoraggio) di un'istanza cluster database
- Pannello di controllo di Performance Insights



Per attivare DevOps Guru per il database per PostgreSQL

1. Nel banner, scegli Turn on Guru for RDS. DevOps
2. Immetti un nome e un valore della chiave tag. Per ulteriori informazioni sui tag, consulta "Usa i tag per identificare le risorse nelle tue applicazioni DevOps Guru" nella Amazon DevOps Guru User Guide.



3. Scegli Attiva Guru. DevOps

Risposta a un errore di autorizzazione quando attivi Guru DevOps

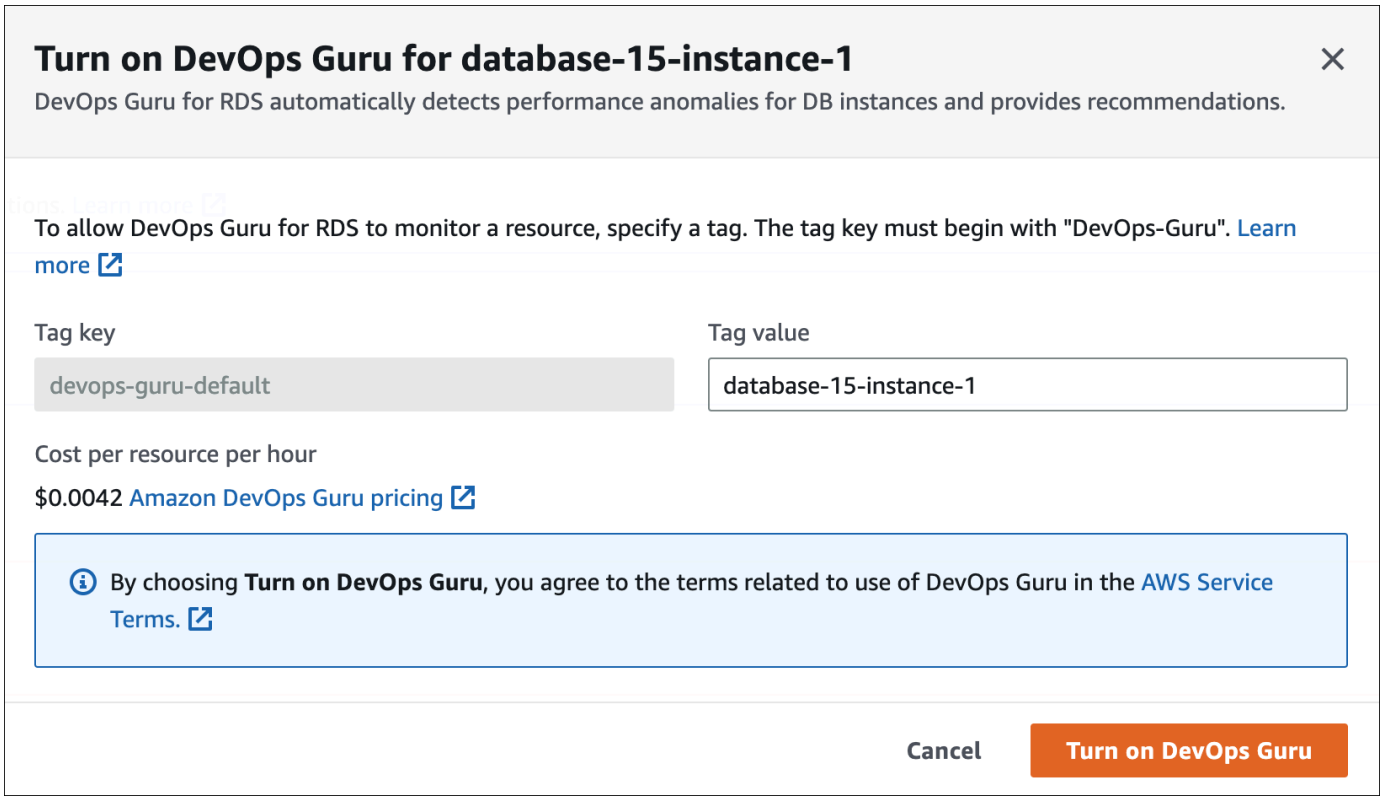
Se attivi DevOps Guru dalla console RDS quando crei un database, RDS potrebbe visualizzare il seguente banner relativo alle autorizzazioni mancanti.

Rispondere a un errore di autorizzazioni

1. Concedi all'utente o ruolo IAM il ruolo gestito dall'utente `AmazonDevOpsGuruConsoleFullAccess`. Per ulteriori informazioni, consulta Configurazione delle politiche di accesso IAM per Guru for RDS DevOps.
2. Aprire la console di RDS.
3. Nel pannello di navigazione scegli Approfondimenti sulle prestazioni.
4. Scegli un'istanza database nel cluster appena creato.
5. Scegli l'interruttore per attivare Guru for RDSDevOps.

DevOps Guru for RDS

6. Scegli un valore di tag. Per ulteriori informazioni, consulta "Usa i tag per identificare le risorse nelle tue applicazioni DevOps Guru" nella Amazon DevOps Guru User Guide.



7. Scegli Turn on Guru. DevOps

Aggiungere risorse Aurora RDS DevOps

È possibile specificare la copertura delle risorse Guru sulla console DevOps Guru. DevOps Segui la procedura descritta in Specificare la copertura delle risorse DevOps Guru nella Amazon DevOps Guru User Guide. Quando modifichi le risorse analizzate, scegli una delle opzioni seguenti:

- Scegli Tutte le risorse dell'account per analizzare tutte le risorse supportate, inclusi i database Aurora , nella tua regione e nella tua regione. Account AWS
- Scegli CloudFormation gli stack per analizzare i database Aurora RDS PostgreSQL che si trovano negli stack che preferisci. Per ulteriori informazioni, consulta Usa gli AWS CloudFormation stack per identificare le risorse nelle tue applicazioni DevOps Guru nella Amazon Guru DevOps User Guide.

- Scegli Tag per analizzare i database Aurora con tag. Per ulteriori informazioni, consulta Usa i tag per identificare le risorse nelle tue applicazioni DevOps Guru nella Amazon DevOps Guru User Guide.

Per ulteriori informazioni, consulta Enable DevOps Guru nella Amazon DevOps Guru User Guide.

Aggiungere risorse Aurora RDSCloudFormation

Puoi utilizzare i tag per aggiungere la copertura delle risorse Aurora ai tuoi modelli. CloudFormation La procedura seguente presuppone che si disponga di un CloudFormation modello sia per l'istanza DB Aurora RDS lo stack Guru. DevOps

Per specificare un'istanza DB Aurora un tag CloudFormation

1. Nel CloudFormation modello per l'istanza DB, definisci un tag utilizzando una coppia chiave/valore.

   L'esempio seguente assegna il valore `my-aurora-db-instance1` a `Devops-guru-cfn-default` per un'istanza database Aurora.

   ```
   MyAuroraDBInstance1:
     Type: "AWS::RDS::DBInstance"
     Properties:
       DBClusterIdentifier: my-aurora-db-cluster
       DBInstanceIdentifier: my-aurora-db-instance1
       Tags:
         - Key: Devops-guru-cfn-default
           Value: devopsguru-my-aurora-db-instance1
   ```

2. Nel CloudFormation modello per il tuo stack DevOps Guru, specifica lo stesso tag nel filtro di raccolta delle risorse.

   L'esempio seguente configura DevOps Guru per fornire una copertura alla risorsa con il valore del tag. `my-aurora-db-instance1`

   ```
   DevOpsGuruResourceCollection:
     Type: AWS::DevOpsGuru::ResourceCollection
     Properties:
       ResourceCollectionFilter:
         Tags:
           - AppBoundaryKey: "Devops-guru-cfn-default"
   ```

```
          TagValues:
          - "devopsguru-my-aurora-db-instance1"
```

Nell'esempio seguente si fornisce la copertura per tutte le risorse all'interno di `Devops-guru-cfn-default` del limite dell'applicazione.

```
DevOpsGuruResourceCollection:
  Type: AWS::DevOpsGuru::ResourceCollection
  Properties:
    ResourceCollectionFilter:
      Tags:
        - AppBoundaryKey: "Devops-guru-cfn-default"
          TagValues:
          - "*"
```

Per ulteriori informazioni, consulta AWS::DevOpsGuru::ResourceCollectione AWS: :RDS:: DBInstance nella Guida per l'CloudFormation utente.

Monitoraggio dei parametri del sistema operativo con il monitoraggio avanzato

Con il monitoraggio avanzato, potete monitorare il sistema operativo del vostro database in tempo reale. I parametri di monitoraggio avanzato sono utili quando si desidera vedere come viene utilizzata la CPU in un'istanza database dai diversi processi o thread.

Argomenti

- Panoramica sul monitoraggio avanzato
- Configurare e abilitare Monitoraggio avanzato
- Visualizzazione delle metriche nella console RDS
- Visualizzazione dell'utilizzo dei parametri del sistema operativo CloudWatch Logs

Panoramica sul monitoraggio avanzato

Amazon RDS fornisce parametri in tempo reale per il sistema operativo su cui è in esecuzione l'istanza di database. È possibile visualizzare tutti i parametri di sistema e le informazioni sui processi per le istanze del database RDS sulla console. È possibile gestire quali parametri si desidera monitorare per ogni istanza e personalizzare il pannello di controllo in base alle proprie esigenze. Per le descrizioni dei parametri di monitoraggio avanzato, consulta Metriche del sistema operativo nel monitoraggio avanzato.

RDS fornisce i parametri del monitoraggio avanzato nel tuo account Amazon CloudWatch Logs. Puoi creare filtri di parametri in CloudWatch da CloudWatch Logs e visualizzare i grafici nel pannello di controllo CloudWatch. È possibile utilizzare l'output JSON di monitoraggio avanzato da CloudWatch Logs in un sistema di monitoraggio a scelta. Per ulteriori informazioni, consulta Monitoraggio avanzato nelle domande frequenti su Amazon RDS.

Argomenti

- Differenze tra CloudWatch e parametri di monitoraggio avanzato
- Conservazione delle metriche di monitoraggio avanzato
- Costo di Enhanced Monitoring (monitoraggio avanzato)

Differenze tra CloudWatch e parametri di monitoraggio avanzato

Un hypervisor crea ed esegue macchine virtuali (VM). Utilizzando un hypervisor, un'istanza può supportare più VM ospiti condividendo virtualmente memoria e CPU. CloudWatch raccoglie i parametri sull'utilizzo della CPU dall'hypervisor per un'istanza database. Al contrario, Enhanced Monitoring raccoglie le metriche da un agente nell'istanza DB.

Di conseguenza, si potrebbero trovare differenze CloudWatch tra le misurazioni del monitoraggio avanzato, perché il livello dell'hypervisor esegue una piccola quantità di lavoro. Le differenze possono essere maggiori se le istanze DB utilizzano classi di istanza più piccole. In questo scenario, più macchine virtuali (VM) sono probabilmente gestite dal livello dell'hypervisor in una singola istanza fisica.

Per le descrizioni dei parametri di monitoraggio avanzato, consulta Metriche del sistema operativo nel monitoraggio avanzato. Per ulteriori informazioni sulle metriche di CloudWatch, consulta la Guida per l'utente di Amazon CloudWatch.

Conservazione delle metriche di monitoraggio avanzato

Per impostazione predefinita, i parametri di monitoraggio avanzato vengono memorizzati nei CloudWatch Logs per 30 giorni. Questo periodo di conservazione è diverso dalle metriche tipiche di CloudWatch.

Per modificare il periodo di tempo in cui i parametri vengono memorizzati nei CloudWatch Logs, modifica la conservazione per il gruppo di log `RDSOSMetrics` nella console CloudWatch. Per ulteriori informazioni, consulta Modifica della conservazione dei dati di log CloudWatch nella Amazon CloudWatch Logs User Guide.

Costo di Enhanced Monitoring (monitoraggio avanzato)

I parametri di monitoraggio avanzato vengono memorizzate nei registri di CloudWatch Logs anziché nei parametri di Cloudwatch. Il costo del monitoraggio avanzato dipende dai seguenti fattori:

- Verranno addebitati i costi per il monitoraggio avanzato solo se superi la quantità di dati trasferiti e di spazio di archiviazione forniti da Amazon CloudWatch Logs. Gli addebiti si basano sulle tariffe di trasferimento dati e archiviazione di CloudWatch Logs.
- La quantità di informazioni trasferite per un'istanza RDS è direttamente proporzionale alla granularità definita per la funzione di monitoraggio avanzato. Un intervallo di monitoraggio più piccolo comporta report più frequenti sulle metriche del sistema operativo e aumenta i costi di monitoraggio. Per gestire i costi, imposta granularità diverse per istanze diverse nei tuoi account.

- I costi di utilizzo per il Monitoraggio avanzato vengono applicati a ciascuna istanza database per cui il monitoraggio avanzato è abilitato. Il monitoraggio di un numero elevato di istanze DB è più costoso rispetto al monitoraggio solo di pochi.
- Le istanze database che supportano un carico di lavoro ad alta intensità di elaborazione hanno più attività di processo del sistema operativo per generare report e costi più elevati per il monitoraggio avanzato.

Per ulteriori informazioni sui prezzi, consulta Prezzi di Amazon CloudWatch.

Configurare e abilitare Monitoraggio avanzato

Per utilizzare Monitoraggio avanzato, è necessario creare un ruolo IAM e quindi abilitare Monitoraggio avanzato.

Argomenti

- Creazione di un ruolo IAM per Enhanced Monitoring
- Attivazione e disattivazione del monitoraggio avanzato
- Protezione dal problema del "confused deputy"

Creazione di un ruolo IAM per Enhanced Monitoring

Il monitoraggio avanzato richiede l'autorizzazione ad agire per conto dell'utente per inviare le informazioni sulle metriche del sistema operativo ai CloudWatch registri. Concedi le autorizzazioni di Enhanced Monitoring utilizzando un ruolo AWS Identity and Access Management (IAM). È possibile creare questo ruolo quando si abilita il monitoraggio avanzato o lo si crea in anticipo.

Argomenti

- Creazione del ruolo IAM quando si attiva Enhanced Monitoring
- Creazione del ruolo IAM prima di abilitare Enhanced Monitoring

Creazione del ruolo IAM quando si attiva Enhanced Monitoring

Quando si attiva Enhanced Monitoring nella console RDS, Amazon RDS è possibile creare il ruolo IAM necessario. Il ruolo è denominato `rds-monitoring-role`. RDS utilizza questo ruolo per l'istanza database specificata, la replica di lettura o il cluster di database Multi-AZ.

Per creare il ruolo IAM quando si attiva Enhanced Monitoring

1. Segui la procedura riportata in Attivazione e disattivazione del monitoraggio avanzato.
2. Imposta Ruolo di monitoraggio su Predefinito nel passaggio in cui si sceglie un ruolo.

Creazione del ruolo IAM prima di abilitare Enhanced Monitoring

È possibile creare il ruolo richiesto prima di abilitare Enhanced Monitoring. Quando si abilita Enhanced Monitoring, specifica il nome del nuovo ruolo. Si deve creare questo ruolo necessario se si abilita il monitoraggio avanzato utilizzando AWS CLI oppure l'API RDS.

L'utente che abilita il monitoraggio avanzato deve ricevere l'autorizzazione `PassRole`. Per ulteriori informazioni, consulta l'Esempio 2 in Concessione a un utente delle autorizzazioni per il trasferimento di un ruolo a un AWS servizio nella Guida per l'utente IAM.

Per creare un ruolo IAM per Amazon RDS Enhanced Monitoring

1. Apri la console IAM all'indirizzo. https://console.aws.amazon.com
2. Nel pannello di navigazione, seleziona Roles (Ruoli).
3. Selezionare Create role (Crea ruolo).
4. Scegliere la scheda Servizio AWS quindi seleziona RDS dall’elenco di servizi.
5. Scegli RDS - Enhanced Monitoring (RDS - Monitoraggio avanzato), quindi seleziona Next (Avanti).
6. Assicurati che le politiche di autorizzazione mostrino Amazon RDSEnhanced MonitoringRole, quindi scegli Avanti.
7. In Nome ruolo, immetti un nome per il ruolo. Ad esempio, specifica **`emaccess`**.

   L'entità affidabile per il tuo ruolo è il AWS servizio monitoring.rds.amazonaws.com.
8. Scegli Crea ruolo.

Attivazione e disattivazione del monitoraggio avanzato

Puoi gestire il monitoraggio avanzato utilizzando l'API, o RDS. Console di gestione AWS AWS CLI È possibile impostare granularità diverse per la raccolta delle metriche su ogni cluster di database. È inoltre possibile attivare il monitoraggio avanzato per un cluster di database esistente dalla console.

Console

È possibile attivare Monitoraggio avanzato quando si crea un cluster o una replica di lettura oppure quando si modifica un cluster. Se si modifica un'istanza database o un cluster di database per attivare il monitoraggio avanzato, non è necessario riavviare l'istanza database per rendere effettive le modifiche.

È possibile abilitare il monitoraggio avanzato nella console RDS quando si esegue una delle seguenti operazioni nella pagina Databases (Database):

- Creazione di un cluster: scegli Create database (Crea database).
- Creazione di una replica di lettura: scegli Actions (Operazioni), quindi Create read replica (Crea replica di lettura).
- Modifica di un'istanza database o di un cluster di database : scegli Modifica.

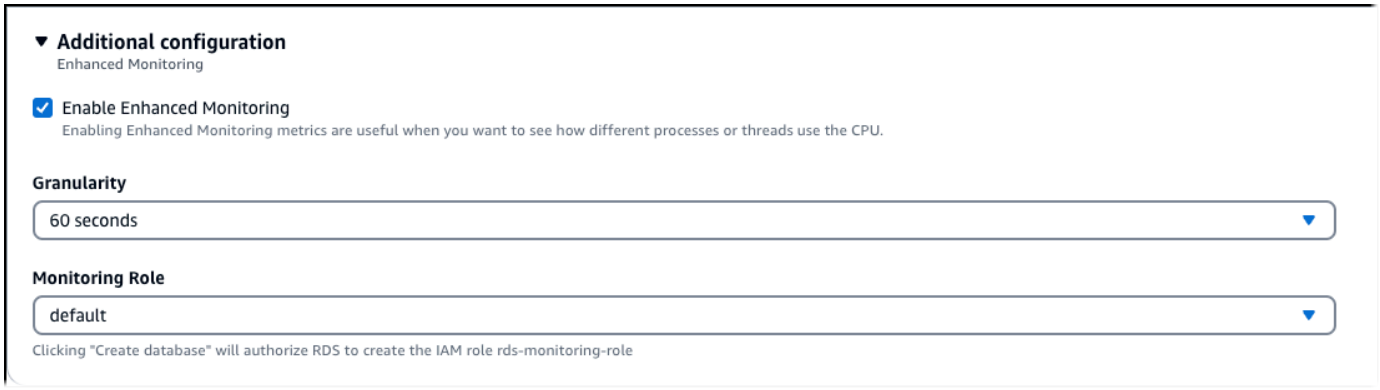
> **Note**
>
> Quando abiliti il monitoraggio avanzato per un cluster di database, non è possibile gestire il monitoraggio avanzato per singole istanze database all'interno del cluster.

Se gestisci il monitoraggio avanzato per singole istanze database in un cluster di database e la granularità è impostata su valori diversi per istanze diverse, il cluster di database è eterogeneo rispetto al monitoraggio avanzato. In questi casi, non è possibile modificare il cluster di database per gestire il monitoraggio avanzato a livello di cluster.

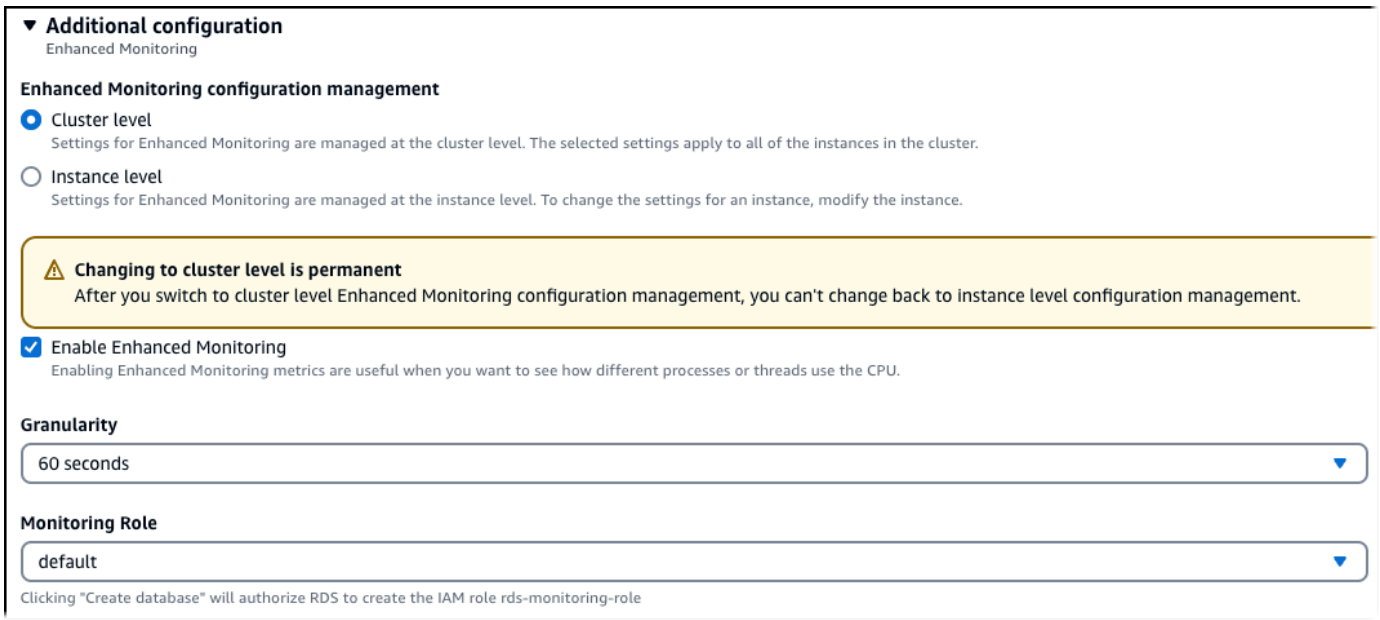
Per attivare o disattivare Monitoraggio avanzato nella console RDS

1. Scorri fino a Additional configuration (Configurazione aggiuntiva).
2. In Monitoraggio, scegli Abilita monitoraggio avanzato per , il cluster di database o la replica di lettura. L'abilitazione del monitoraggio avanzato a livello di cluster consente di gestire le impostazioni e le opzioni del monitoraggio avanzato a livello di cluster. Le impostazioni a livello di cluster si applicano a tutte le istanze database del cluster. Deseleziona l'opzione per disabilitare il monitoraggio avanzato a livello di cluster. Successivamente potrai modificare le impostazioni del monitoraggio avanzato per le singole istanze database del cluster.

   Nella pagina Crea database, puoi scegliere di attivare il monitoraggio avanzato a livello di cluster.



Se non abiliti il monitoraggio avanzato durante la creazione di un cluster, puoi modificare il cluster nella pagina Modifica il cluster DB.



> **Note**
>
> Non è possibile gestire il monitoraggio avanzato per una singola istanza database in un cluster di database che ha già il monitoraggio avanzato gestito a livello di cluster.

3. Non è possibile scegliere di gestire il monitoraggio avanzato a livello di cluster se il cluster è eterogeneo rispetto al monitoraggio avanzato. Per gestire il monitoraggio avanzato a livello di cluster, si modificano le impostazioni di monitoraggio avanzato per ogni istanza in modo che corrispondano. Ora è possibile scegliere di gestire il monitoraggio avanzato a livello di cluster quando si modifica il cluster.

4. Imposta la proprietà Monitoring Role sul ruolo IAM che hai creato per consentire ad Amazon RDS di comunicare con Amazon CloudWatch Logs per te, oppure scegli Default per fare in modo che RDS crei un ruolo per te denominato. `rds-monitoring-role`
5. Impostare la proprietà Granularità sull'intervallo, in secondi, tra i punti quando le metriche vengono raccolte per l'istanza database, il cluster di database, o la replica di lettura. La proprietà Granularity (Granularità) può essere impostata su uno dei valori seguenti: `1`, `5`, `10`, `15`, `30` oppure `60`.

   L'intervallo più veloce in cui la console RDS si aggiorna è ogni 5 secondi. Se si imposta la granularità su 1 secondo nella console RDS, vengono comunque visualizzati parametri aggiornati solo ogni 5 secondi. Puoi recuperare gli aggiornamenti dei parametri in 1 secondo utilizzando Logs. CloudWatch

AWS CLI

Per attivare il monitoraggio avanzato utilizzando i comandi seguentiAWS CLI, impostate l'`--monitoring-interval`opzione su un valore diverso da `0` e impostate l'`--monitoring-role-arn`opzione sul ruolo in cui avete creato. Creazione di un ruolo IAM per Enhanced Monitoring

- create-db-cluster
- modify-db-cluster
- create-db-instance
- create-db-instance-read-replica
- modify-db-instance

L'opzione `--monitoring-interval` specifica l'intervallo, in secondi, tra i punti quando vengono raccolti i parametri di monitoraggio avanzato. I valori validi per l'opzione sono `0`, `1`, `5`, `10`, `15`, `30` e `60`.

Per disattivare il monitoraggio avanzato utilizzando ilAWS CLI, imposta l'`--monitoring-interval`opzione su `0` in questi comandi.

Example

Nell'esempio seguente viene attivato il monitoraggio avanzato per un'istanza database:

Per Linux, macOS o Unix:

```
aws rds modify-db-instance \
    --db-instance-identifier mydbinstance \
    --monitoring-interval 30 \
    --monitoring-role-arn arn:aws:iam::123456789012:role/emaccess
```

Per Windows:

```
aws rds modify-db-instance ^
    --db-instance-identifier mydbinstance ^
    --monitoring-interval 30 ^
    --monitoring-role-arn arn:aws:iam::123456789012:role/emaccess
```

Example

Nell'esempio seguente si attiva il monitoraggio avanzato per un cluster di database:

Per Linux, macOS o Unix:

```
aws rds modify-db-cluster \
    --db-cluster-identifier mydbinstance \
    --monitoring-interval 30 \
    --monitoring-role-arn arn:aws:iam::123456789012:role/emaccess
```

Per Windows:

```
aws rds modify-db-cluster ^
    --db-cluster-identifier mydbinstance ^
    --monitoring-interval 30 ^
    --monitoring-role-arn arn:aws:iam::123456789012:role/emaccess
```

Example

Nell'esempio seguente viene attivato il monitoraggio avanzato per un cluster di database Multi-AZ:

Per Linux, macOS o Unix:

```
aws rds modify-db-cluster \
    --db-cluster-identifier mydbcluster \
    --monitoring-interval 30 \
```

```
    --monitoring-role-arn arn:aws:iam::123456789012:role/emaccess
```

Per Windows:

```
aws rds modify-db-cluster ^
    --db-cluster-identifier mydbcluster ^
    --monitoring-interval 30 ^
    --monitoring-role-arn arn:aws:iam::123456789012:role/emaccess
```

API RDS

Per attivare il monitoraggio avanzato utilizzando l'API RDS, imposta il parametro `MonitoringInterval` su un valore diverso da `0` e imposta il parametro `MonitoringRoleArn` sul ruolo creato in Creazione di un ruolo IAM per Enhanced Monitoring. Imposta questi parametri nelle seguenti operazioni:

- CreaDBCluster
- ModificaDBCluster
- CreaDBInstance
- CreaDBInstanceReadReplica
- ModificaDBInstance

Il parametro `MonitoringInterval` specifica l'intervallo, in secondi, tra i punti quando vengono raccolti i parametri di monitoraggio avanzato. I valori validi sono `0`, `1`, `5`, `10`, `15`, `30` e `60`.

Per disattivare il monitoraggio avanzato utilizzando l'API RDS, imposta `MonitoringInterval` su `0`.

Protezione dal problema del "confused deputy"

Il problema confused deputy è un problema di sicurezza in cui un'entità che non dispone dell'autorizzazione per eseguire un'azione può costringere un'entità maggiormente privilegiata a eseguire l'azione. NelAWS, l'impersonificazione tra servizi può portare alla confusione del vicesceriffo. La rappresentazione tra servizi può verificarsi quando un servizio (il servizio chiamante) effettua una chiamata a un altro servizio (il servizio chiamato). Il servizio chiamante può essere manipolato per utilizzare le proprie autorizzazioni e agire sulle risorse di un altro cliente, a cui normalmente non avrebbe accesso. Per evitare ciò, AWS fornisce strumenti per poterti a proteggere i tuoi dati per tutti i servizi con entità di servizio a cui è stato concesso l'accesso alle risorse dell'account. Per ulteriori informazioni, consulta Problema del "confused deputy".

Per limitare le autorizzazioni relative alle risorse che Amazon RDS può fornire a un altro servizio, si consiglia di utilizzare le chiavi di contesto delle condizioni globali `aws:SourceArn` e `aws:SourceAccount` in una policy di attendibilità per il tuo ruolo di monitoraggio avanzato. Se si utilizzano entrambe le chiavi di contesto delle condizioni globali, devono utilizzare lo stesso ID account.

Il modo più efficace per proteggersi dal problema "confused deputy" è quello di usare la chiave di contesto della condizione globale `aws:SourceArn` con l'ARN completo della risorsa. Per Amazon RDS, imposta `aws:SourceArn` su `arn:aws:rds:`*`Region`*`:`*`my-account-id`*`:db:`*`dbname`*.

L'esempio seguente usa le chiavi di contesto delle condizioni globali `aws:SourceArn` e `aws:SourceAccount` in una policy di affidabilità per prevenire il problema del "confused deputy".

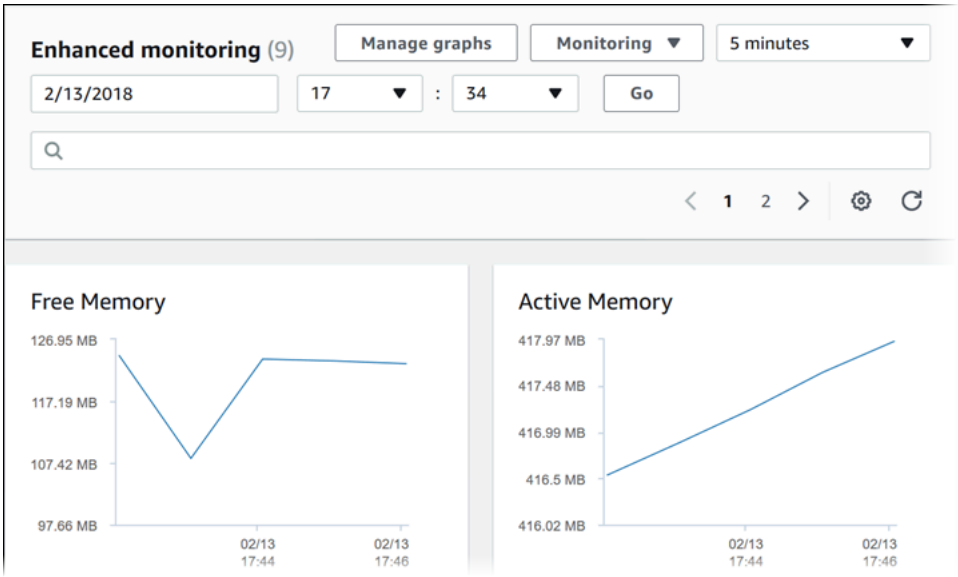
JSON

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": "monitoring.rds.amazonaws.com"
      },
      "Action": "sts:AssumeRole",
      "Condition": {
        "StringLike": {
          "aws:SourceArn": "arn:aws:rds:Region:my-account-id:db:dbname"
        },
        "StringEquals": {
          "aws:SourceAccount": "my-account-id"
        }
      }
    }
  ]
}
```

Visualizzazione delle metriche nella console RDS

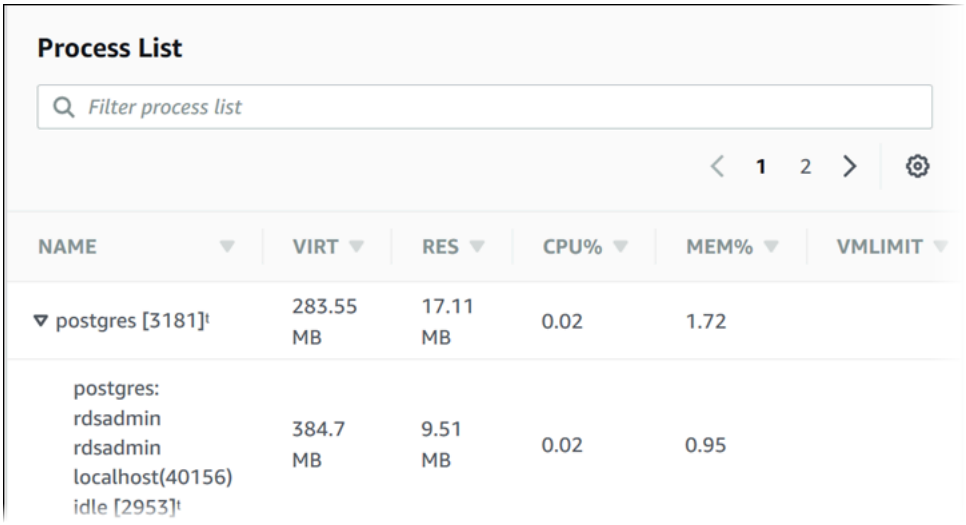
Puoi visualizzare i parametri del sistema operativo segnalati dal monitoraggio avanzato nella console RDS scegliendo Enhanced monitoring (Monitoraggio avanzato) per Monitoring (Monitoraggio).

L'esempio seguente mostra la pagina Enhanced Monitoring (Monitoraggio avanzato). Per le descrizioni dei parametri di monitoraggio avanzato, consulta Metriche del sistema operativo nel monitoraggio avanzato.



Se desideri vedere i dettagli per i processi in esecuzione nell'istanza database, scegli OS process list (Elenco processi sistema operativo) per Monitoring (Monitoraggio).

La vista Process List (Elenco processi) è mostrata di seguito.



I parametri del monitoraggio avanzato mostrati nella vista Process List (Elenco processi) sono organizzati nel seguente modo:

- RDS child processes (Processi figlio RDS) – Viene visualizzato un riepilogo dei processi RDS che supportano l'istanza database, ad esempio `aurora` per i cluster database di Amazon Aurora. I

thread del processo appaiono nidificati sotto il processo genitore. I thread del processo mostrano l'utilizzo della CPU solo quando gli altri parametri sono uguali per tutti i thread per il processo. La console visualizza un massimo di 100 processi e thread. I risultati sono una combinazione dei principali processi e thread CPU che consumano memoria. Se ci sono più di 50 processi e più di 50 thread, la console visualizza i primi 50 consumatori in ciascuna categoria. Questo display aiuta a identificare quali processi stanno avendo il maggiore impatto sulle prestazioni.

- Processi RDS: mostra un riepilogo delle risorse utilizzate dall'agente di gestione RDS, dei processi di monitoraggio della diagnostica e di altri AWS processi necessari per supportare le istanze DB RDS.
- OS processes (Processi del sistema operativo) – Viene visualizzato un riepilogo dei processi del kernel e di sistema, che generalmente hanno un impatto minimo sulle prestazioni.

Gli elementi elencati per ogni processo sono:

- VIRT – Indica la dimensione virtuale del processo.
- RES – Indica la memoria fisica effettiva utilizzata dal processo.
- CPU% – Indica la percentuale della larghezza di banda totale della CPU utilizzata dal processo.
- MEM% – Indica la percentuale della memoria totale utilizzata dal processo.

I dati di monitoraggio visualizzati nella console RDS vengono recuperati da Amazon CloudWatch Logs. Puoi anche recuperare le metriche per un'istanza DB come flusso di log da Logs. CloudWatch Per ulteriori informazioni, consulta Visualizzazione dell'utilizzo dei parametri del sistema operativo CloudWatch Logs.

I parametri di monitoraggio avanzato non vengono restituiti durante:

- Un failover dell'istanza database.
- Modifica della classe di istanza dell'istanza database (dimensionamento del calcolo).

I parametri del monitoraggio avanzato vengono restituiti durante un riavvio di un'istanza DB perché viene riavviato solo il motore del database. I parametri per il sistema operativo vengono ancora segnalati.

Visualizzazione dell'utilizzo dei parametri del sistema operativo CloudWatch Logs

Dopo aver abilitato il monitoraggio avanzato per l'istanza database o il cluster, è possibile visualizzare i relativi parametri utilizzando CloudWatch Logs, con ogni flusso di log che rappresenta una singola istanza database monitorata o cluster di database monitorato. L'identificatore del flusso di log è l'identificativo della risorsa (`DbiResourceId`) per l'istanza database o il cluster di database.

Per visualizzare i dati di log del Monitoraggio avanzato

1. Apri la console CloudWatch all'indirizzo https://console.aws.amazon.com/cloudwatch/.
2. Se necessario, scegliere la Regione AWS in cui si trova il cluster. Per ulteriori informazioni, consulta la pagina relativa a regioni ed endpoint nei Riferimenti generali di Amazon Web Services.
3. Selezionare Logs (Log) nel riquadro di navigazione.
4. Selezionare RDSOSMetrics nell'elenco di gruppi di log.
5. Scegliere il flusso di log che si desidera visualizzare dall´elenco dei flussi di log.

Riferimento per i parametri per Amazon Aurora

In questo riferimento, è possibile trovare descrizioni dei parametri di Amazon Aurora per Amazon CloudWatch, Performance Insights e monitoraggio avanzato.

Argomenti

- CloudWatch Parametri Amazon per Amazon Aurora
- CloudWatch Dimensioni Amazon per
- Disponibilità dei parametri Aurora nella console Amazon RDS
- Metriche Amazon CloudWatch per Approfondimenti sulle prestazioni di Amazon RDS
- Metriche contatore di Performance Insights
- Statistiche SQL per Performance Insights
- Metriche del sistema operativo nel monitoraggio avanzato

CloudWatch Parametri Amazon per Amazon Aurora

Lo spazio dei nomi `AWS/RDS` include i seguenti parametri che si applicano alle entità di database in esecuzione su Amazon Aurora. Alcuni parametri si applicano a Aurora MySQL, Aurora PostgreSQL o a entrambi. Inoltre, alcuni parametri sono specifici per un cluster di database, un'istanza database primaria, un'istanza database di replica o per tutte le istanze database.

Per i parametri del database globale Aurora, consulta Parametri Amazon CloudWatch per l'inoltro di scrittura in Aurora MySQL e Parametri Amazon CloudWatch per l'inoltro di scrittura in Aurora PostgreSQL. Per i parametri delle query parallele Aurora, consulta Monitoraggio di query in parallelo per Aurora MySQL.

Argomenti

- Parametri a livello di cluster per Amazon Aurora
- Parametri a livello di istanza per Amazon Aurora
- Metriche CloudWatch di utilizzo di

Parametri a livello di cluster per Amazon Aurora

Nella tabella seguente vengono descritti parametri specifici per i cluster Aurora.

Metrica	Descrizione	Si applica a	Unità
`AuroraGlobalDBDataTransferBytes`	In un database globale Aurora, la quantità di dati di redo log trasferiti dalla regione di origine a una AWS regione secondaria. AWS **Note** Questa metrica è disponibile solo nelle Regioni AWS secondarie.	Aurora MySQL e Aurora PostgreSQL	Byte
`AuroraGlobalDBProgressLag`	In un Database globale Aurora, la misura di quanto il cluster secondario si trova dietro il cluster primario sia per le transazioni utente che per le transazioni di sistema. **Note** Questa metrica è disponibile solo nelle Regioni AWS secondarie.	Aurora MySQL e Aurora PostgreSQL	Millisecondi
`AuroraGlobalDBReplicatedWriteIO`	In un database globale Aurora, il numero di I/O operazioni di scrittura replicate dalla AWS regione primaria al volume del cluster in una regione	Aurora MySQL e Aurora PostgreSQL	Conteggio

Metrica	Descrizione	Si applica a	Unità
	secondaria. AWS I calcoli di fatturazione per le AWS regioni secondarie in un database globale vengono utilizzati `VolumeWriteIOPs` per tenere conto delle scritture eseguite all'interno del cluster. I calcoli di fatturazione per la AWS regione principale in un database globale vengono utilizzati per `VolumeWriteIOPs` tenere conto dell'attività di scrittura all'interno di quel cluster e `AuroraGlobalDBReplicatedWriteIO` per tenere conto della replica tra regioni all'interno del database globale. **Note** Questa metrica è disponibile solo nelle Regioni AWS secondarie.		

Metrica	Descrizione	Si applica a	Unità
`AuroraGlobalDBReplicationLag`	Per un database globale Aurora, il ritardo durante la replica degli aggiornamenti dalla regione AWS primaria. **Note** Questa metrica è disponibile solo nelle Regioni AWS secondarie.	Aurora MySQL e Aurora PostgreSQL	Millisecondi
`AuroraGlobalDBRPOLag`	In un Database globale Aurora, il tempo di ritardo dell'obiettivo del punto di ripristino (RPO). Questo parametro misura la distanza del cluster secondario dietro il cluster primario per le transazioni utente. **Note** Questa metrica è disponibile solo nelle Regioni AWS secondarie.	Aurora MySQL e Aurora PostgreSQL	Millisecondi

Metrica	Descrizione	Si applica a	Unità
`AuroraVolumeBytesLeftTotal`	Spazio disponibile rimanente per il volume del cluster. All'aumentare del volume del cluster, questo valore diminuisce. Se raggiunge lo zero, il cluster segnala un errore. out-of-space Per rilevare se il cluster Aurora MySQL si sta avvicinando al limite di dimensione 128 tebibytes (TiB), questo valore è più semplice e più affidabile da monitorare rispetto a `VolumeBytesUsed`. `AuroraVolumeBytesLeftTotal` tiene conto dello spazio di archiviazione utilizzato per le la manutenzione interna e altre allocazioni che non influiscono sulla fatturazione dell'archiviazione.	Aurora MySQL	Byte
`BacktrackChangeRecordsCreationRate`	Il numero di record delle modifiche di backtrack create in cinque minuti per il cluster DB.	Aurora MySQL	Conteggio per 5 minuti
`BacktrackChangeRecordsStored`	Il numero di record delle modifiche di backtrack utilizzate per il cluster DB.	Aurora MySQL	Conteggio

Metrica	Descrizione	Si applica a	Unità
`BackupRetentionPeriodStorageUsed`	La quantità totale di storage di backup utilizzata per supportare la funzionalità di point-in-time ripristino all'interno della finestra di conservazione dei backup del cluster Aurora DB. Questa quantità è inclusa nel totale riportato dal parametro `TotalBackupStorageBilled` . Calcolato separatamente per ogni cluster Aurora. Per istruzioni, consulta Informazioni sull'utilizzo dello storage di backup Amazon Aurora.	Aurora MySQL e Aurora PostgreSQL	Byte
`ServerlessDatabaseCapacity`	Capacità corrente di un cluster di database Aurora Serverless.	Aurora MySQL e Aurora PostgreSQL	Conteggio

Metrica	Descrizione	Si applica a	Unità
`SnapshotStorageUsed`	Il totale dello storage di backup utilizzato da tutti gli snapshot Aurora per un cluster DB Aurora al di fuori della finestra di retention dei backup. Questa quantità è inclusa nel totale riportato dal parametro `TotalBackupStorageBilled` . Calcolato separatamente per ogni cluster Aurora. Per istruzioni, consulta Informazioni sull'utilizzo dello storage di backup Amazon Aurora.	Aurora MySQL e Aurora PostgreSQL	Byte
`TotalBackupStorageBilled`	Il totale, espresso in byte, dello storage di backup addebitato per un determinato cluster DB Aurora. Include lo storage di backup misurato dai parametri `BackupRetentionPeriodStorageUsed` e `SnapshotStorageUsed` . Questo parametro viene calcolato separatamente per ogni cluster Aurora. Per istruzioni, consulta Informazioni sull'utilizzo dello storage di backup Amazon Aurora.	Aurora MySQL e Aurora PostgreSQL	Byte

Metrica	Descrizione	Si applica a	Unità
`VolumeBytesUsed`	Quantità di spazio di archiviazione utilizzato dal cluster di database Aurora. Questo valore influisce sul costo del cluster database Aurora (per informazioni relative ai prezzi, consulta la pagina Prezzi di Amazon RDS). Questo valore non riflette alcune allocazioni di storage interne che non influenza no la fatturazione dello storage. Per Aurora MySQL è possibile prevedere i out-of-space problemi in modo più accurato verificando se si avvicina allo zero anziché `VolumeBytesUsed` confrontarli con `AuroraVolumeBytesLeftTotal` il limite di archiviazione di 128 TiB. Per i cluster che sono cloni, il valore di questa metrica dipende dalla quantità di dati aggiunti o modificati nel clone. La metrica può anche aumentare o diminuire quando il cluster originale viene eliminato o quando vengono aggiunti o eliminati	Aurora MySQL e Aurora PostgreSQL	Byte

Metrica	Descrizione	Si applica a	Unità
	nuovi cloni. Per maggiori dettagli, consultare Eliminazione di un volume cluster di origine. Tieni presente che non ha senso scegliere un valore `--period` piccolo, perché Amazon RDS raccoglie queste metriche a intervalli, non in modo continuo.		

Metrica	Descrizione	Si applica a	Unità
`VolumeReadIOPs`	Il numero di I/O operazioni di lettura fatturate da un volume del cluster in un intervallo di 5 minuti. Le operazioni di lettura fatturate sono calcolate a livello del volume del cluster, aggregate da tutte le istanze nel cluster di database di Aurora e, in seguito, indicate a intervalli di 5 minuti. Il valore è calcolato prendendo il valore del parametro Read operations per un periodo che supera i 5 minuti. Puoi determinare la quantità delle operazioni di lettura fatturate al secondo prendendo il valore del parametro Billed read operations e dividendo per 300 secondi. Ad esempio, se le Billed read operations restituiscono 13.686, allora le operazioni di lettura fatturate al secondo saranno 45 (13.686 / 300 = 45,62). Le operazioni di lettura fatturate si accumulano per query che richiedono pagine del database che non si trovano nella cache del buffer e, per questo, è	Aurora MySQL e Aurora PostgreSQL	Conteggio per 5 minuti

Metrica	Descrizione	Si applica a	Unità
	necessario caricarle dallo storage. Potresti vedere dei picchi nelle operazioni di lettura fatturate poiché i risultati della query vengono letti dallo storage e, in seguito, caricati nella cache del buffer. Tieni presente che non ha senso scegliere un valore `--period` piccolo, perché Amazon RDS raccoglie queste metriche a intervalli, non in modo continuo. **Tip** Se il tuo cluster Aurora MySQL utilizza una query parallela , potresti vedere un aumento dei valori di `VolumeReadIOPS` . Le query parallele non utilizzano il pool di buffer. Pertanto, sebbene le query siano veloci, questa elaborazione ottimizzata può comportare un aumento delle		

Metrica	Descrizione	Si applica a	Unità
	operazioni di lettura e degli addebiti associati.		
`VolumeWriteIOPs`	Il numero di I/O operazioni su disco di scrittura sul volume del cluster, riportate a intervalli di 5 minuti. Per una descrizione dettagliata del modo in cui vengono calcolate le operazioni di scrittura fatturate, consulta `VolumeReadIOPs` . Tieni presente che non ha senso scegliere un valore `--period` piccolo, perché Amazon RDS raccoglie queste metriche a intervalli, non in modo continuo.	Aurora MySQL e Aurora PostgreSQL	Conteggio per 5 minuti

Parametri a livello di istanza per Amazon Aurora

Le seguenti metriche CloudWatch Amazon specifiche dell'istanza si applicano a tutte le istanze Aurora MySQL e Aurora PostgreSQL, salvo diversa indicazione.

Metrica	Descrizione	Si applica a	unità
`AbortedClients`	Numero di connessioni client che non sono state chiuse correttamente.	Aurora MySQL	Conteggio
`ActiveTransactions`	Il numero medio delle attuali transazioni in esecuzione su	Aurora MySQL	Conteggio al secondo

Metrica	Descrizione	Si applica a	unità
	un'istanza database di Aurora al secondo. Per impostazione predefinita, Aurora non abilita questo parametro. Per iniziare a misurare questo valore, imposta `innodb_monitor_enable='all'` nel gruppo di parametri DB per una specifica istanza database.		
`ACUUtilization`	Valore del parametro `ServerlessDatabaseCapacity` diviso per il valore ACU massimo del cluster database. Questa metrica è applicabile solo per Aurora Serverless v2.	Aurora MySQL e Aurora PostgreSQL	Percentuale

Metrica	Descrizione	Si applica a	unità
`AuroraBinlogReplicaLag`	Il tempo di ritardo di un cluster di database di replica dei log binari in esecuzione su Aurora MySQL rispetto alla replica dei log binari di origine. Un ritardo indica che l'origine genera record più velocemente di quanto la replica possa applicarli. Questo parametro riporta valori diversi a seconda della versione del motore: Aurora MySQL versione 2 Il campo `Seconds_Behind_Master` del `SHOW SLAVE STATUS` MySQL Aurora MySQL versione 3 `SHOW REPLICA STATUS` È possibile utilizzare questo parametro per monitorare gli errori e il ritardo di replica in un cluster che funge da replica di log binario. Il valore del parametro indica quanto segue: Un valore elevato La replica sta ritardando l'origine della replica.	Primario per Aurora MySQL	Secondi

Metrica	Descrizione	Si applica a	unità
	`0` o un valore vicino a **`0`** Il processo di replica è attivo e attuale. **`-1`** Aurora non è in grado di determinare il ritardo, che può verificarsi durante la configurazione della replica o quando la replica si trova in uno stato di errore. Poiché la replica dei log binari si verifica solo sull'istanza di scrittura del cluster, consigliamo di utilizzare la versione del parametro associata al ruolo WRITER. Per ulteriori informazioni sull'amministrazione della replica, consulta Repliche di cluster di database Amazon Aurora MySQL tra Regioni AWS. Per ulteriori informazioni sulla risoluzione dei problemi, consulta Problemi di replica relativi a Amazon Aurora MySQL.		

Metrica	Descrizione	Si applica a	unità
`AuroraDMLRejectedMasterFull`	Il numero di query inoltrate che vengono rifiutate perché la sessione sull'istanza database di scrittura è piena.	Primario per Aurora MySQL versione 2	Conteggio
`AuroraDMLRejectedWriterFull`	Il numero di query inoltrate che vengono rifiutate perché la sessione sull'istanza database di scrittura è piena.	Primario per Aurora MySQL versione 3	Conteggio
`AuroraEstimatedSharedMemoryBytes`	La quantità stimata di memoria condivisa del buffer o del pool di buffer che è stata utilizzata attivamente durante l'ultimo intervallo di polling configurato.	Aurora PostgreSQL	Byte
`AuroraMemoryHealthState`	Indica lo stato di integrità della memoria. Un valore di `0` uguale a `NORMAL`. Un valore di `10` uguale a `RESERVED` significa che il server si sta avvicinando a un livello critico di utilizzo della memoria. Per ulteriori informazioni, consulta Risoluzione dei problemi di memoria insufficiente per i database Aurora MySQL.	Aurora MySQL versione 3.06.1 e versioni successive	Gauge

Metrica	Descrizione	Si applica a	unità
`AuroraMemoryNumDeclinedSqlTotal`	Il numero incrementale di query è diminuito come parte dell'elusione di (OOM). out-of-memory Per ulteriori informazioni, consulta Risoluzione dei problemi di memoria insufficiente per i database Aurora MySQL.	Aurora MySQL versione 3.06.1 e versioni successive	Conteggio
`AuroraMemoryNumKillConnTotal`	Il numero incrementale di connessioni chiuse per evitare l'esaurimento della memoria. Per ulteriori informazioni, consulta Risoluzione dei problemi di memoria insufficiente per i database Aurora MySQL.	Aurora MySQL versione 3.06.1 e versioni successive	Conteggio
`AuroraMemoryNumKillQueryTotal`	Il numero incrementale di query è terminato per evitare l'esaurimento della memoria. Per ulteriori informazioni, consulta Risoluzione dei problemi di memoria insufficiente per i database Aurora MySQL.	Aurora MySQL versione 3.06.1 e versioni successive	Conteggio

Metrica	Descrizione	Si applica a	unità
`AuroraMilliseconds SpentInOomRecovery`	Il tempo trascorso dallo stato di integrità della memoria è sceso al di sotto dello stato normale. Per ulteriori informazioni, consulta Risoluzione dei problemi di memoria insufficiente per i database Aurora MySQL.	Aurora MySQL versione 3.08.0 e versioni successive	Millisecondi
`AuroraNumOomRecoverySuccessful`	Il numero di volte in cui lo stato di integrità della memoria è stato ripristinato allo stato normale. Per ulteriori informazioni, consulta Risoluzione dei problemi di memoria insufficiente per i database Aurora MySQL.	Aurora MySQL versione 3.08.0 e versioni successive	Conteggio
`AuroraNumOomRecoveryTriggered`	Il numero di volte in cui lo stato di integrità della memoria è sceso al di sotto dello stato normale. Per ulteriori informazioni, consulta Risoluzione dei problemi di memoria insufficiente per i database Aurora MySQL.	Aurora MySQL versione 3.08.0 e versioni successive	Conteggio

Metrica	Descrizione	Si applica a	unità
`AuroraOptimizedReadsCacheHitRatio`	La percentuale di richieste servite dalla cache delle letture ottimizzate. Il valore viene calcolato utilizzando la formula seguente: `orcache_blks_hit/ (orcache_blks_hit + storage_blks_read)` Quando `AuroraOptimizedReadsCacheHitRatio` è al 100%, significa che tutte le pagine sono state lette dalla cache di lettura ottimizzata. Se `AuroraOptimizedReadsCacheHitRatio` è `0`, significa che nessuna pagina è stata letta dalla cache di lettura ottimizzata.	Primario per Aurora PostgreSQL	Percentuale
`AuroraReplicaLag`	Per una replica Aurora, la quantità di ritardo durante la replica degli aggiornamenti dall'istanza principale.	Replica per Aurora MySQL e Aurora PostgreSQL	Millisecondi

Metrica	Descrizione	Si applica a	unità
`AuroraReplicaLagMaximum`	Il ritardo massimo tra l'istanza primaria e ogni istanza database Aurora nel cluster di database. Quando le repliche di lettura vengono eliminate o rinominate, può verificarsi un picco temporaneo del ritardo di replica dovuto al processo di riciclaggio a cui è sottoposta la vecchia risorsa. Per ottenere una rappresentazione accurata del ritardo di replica durante quel periodo, si consiglia di monitorare la metrica `AuroraReplicaLag` su ogni istanza di replica di lettura.	Primario per Aurora MySQL e Aurora PostgreSQL	Millisecondi
`AuroraReplicaLagMinimum`	Il ritardo minimo tra l'istanza primaria e ogni istanza database Aurora nel cluster di database.	Primario per Aurora MySQL e Aurora PostgreSQL	Millisecondi
`AuroraSlowConnectionHandleCount`	Il numero di connessioni che hanno atteso due o più secondi per avviare l'handshake. Questa metrica si applica solo ad Aurora MySQL versione 3.	Aurora MySQL	Conteggio

Metrica	Descrizione	Si applica a	unità
`AuroraSlowHandshakeCount`	Il numero di connessioni che hanno impiegato 50 o più millisecondi per completare l'handshake. Questa metrica si applica solo ad Aurora MySQL versione 3.	Aurora MySQL	Conteggio
`BacktrackWindowActual`	La differenza tra la finestra target di backtrack e la finestra attuale di backtrack.	Primario per Aurora MySQL	Minutes (Minuti)
`BacktrackWindowAlert`	Il numero di volte in cui la finestra attuale di backtrack è più piccola della finestra di backtrack target in un intervallo temporale specifico.	Primario per Aurora MySQL	Conteggio
`BlockedTransactions`	Il numero medio di transazioni nel database bloccate ogni secondo.	Aurora MySQL	Conteggio al secondo
`BufferCacheHitRatio`	La percentuale di richieste gestite dalla cache del buffer.	Aurora MySQL e Aurora PostgreSQL	Percentuale
`CommitLatency`	La durata media impiegata dal motore e dall'archiviazione per completare le operazioni di commit.	Aurora MySQL e Aurora PostgreSQL	Millisecondi

Metrica	Descrizione	Si applica a	unità
`CommitThroughput`	Il numero medio di operazioni di conferma al secondo.	Aurora MySQL e Aurora PostgreSQL	Conteggio al secondo
`ConnectionAttempts`	Il numero di tentativi di connessione a un'istanza, a prescindere che vadano a buon fine.	Aurora MySQL	Conteggio

Metrica	Descrizione	Si applica a	unità
`CPUCreditBalance`	Il numero di crediti CPU accumulati da un'istanza, segnalati a intervalli di 5 minuti. Viene utilizzato per determinare per quanto tempo un'istanza database può superare il proprio livello di prestazioni di base a una determinata velocità. Questa metrica si applica solo alle seguenti classi di istanza: • Aurora MySQL: `db.t2.small` , `db.t2.medium` , `db.t3` e `db.t4g` • Aurora PostgreSQL: `db.t3` e `db.t4g` **Note** Consigliamo di utilizzare le classi di istanza database T solo per i server di sviluppo e test o altri server non di produzione. Per ulteriori informazioni sulle classi di istanza T, consulta Tipi di	Aurora MySQL e Aurora PostgreSQL	Conteggio

Metrica	Descrizione	Si applica a	unità
	classi di istanza database. I crediti di lancio funzionano allo stesso modo in Amazon RDS e in Amazon EC2. Per ulteriori informazioni, consulta Crediti di lancio nella Guida per l'utente di Amazon Elastic Compute Cloud per istanze Linux.		

Metrica	Descrizione	Si applica a	unità
`CPUCreditUsage`	Il numero di crediti CPU consumati durante il periodo specificato, segnalato a intervalli di 5 minuti. Questa metrica misura la quantità di tempo durante la quale i dati fisici CPUs sono stati utilizzati per l'elaborazione delle istruzioni mediante CPUs allocazioni virtuali all'istanza DB. Questa metrica si applica solo alle seguenti classi di istanza: • Aurora MySQL: `db.t2.small` , `db.t2.medium` , `db.t3` e `db.t4g` • Aurora PostgreSQL: `db.t3` e `db.t4g` **Note** Consigliamo di utilizzare le classi di istanza database T solo per i server di sviluppo e test o altri server non di produzione. Per ulteriori informazioni sulle classi di istanza T, consulta Tipi di	Aurora MySQL e Aurora PostgreSQL	Conteggio

Metrica	Descrizione	Si applica a	unità
	classi di istanza database.		
`CPUSurplusCreditBalance`	Il numero di crediti extra spesi da un'istanza illimitata quando il rispettivo valore `CPUCreditBalance` è pari a zero. Il valore `CPUSurplusCreditBalance` viene saldato con i crediti CPU ottenuti. Se il numero dei crediti extra va oltre il numero massimo di crediti che un'istanza può ottenere in un periodo di 24 ore, i crediti extra spesi, eccedenti il limite, incorreranno in costi aggiuntivi. I parametri di credito CPU sono disponibili solo con una frequenza di 5 minuti.	Aurora MySQL e Aurora PostgreSQL	Crediti (vCPU/minuti)

Metrica	Descrizione	Si applica a	unità
`CPUSurplusCreditsCharged`	Il numero di crediti extra spesi da un'istanza, che non sono saldati con i crediti CPU ottenuti e che pertanto incorrono in costi aggiuntivi. I crediti extra spesi subiscono costi aggiuntivi quando si verifica uno dei seguenti casi: • I crediti extra spesi vanno oltre il numero massimo di crediti che un'istanza può ottenere in un periodo di 24 ore. I crediti extra spesi, che eccedono il limite, subiscono costi aggiuntivi alla fine dell'ora; • l'istanza viene arrestata o terminata; • l'istanza passa da `unlimited` a `standard`. I parametri di credito CPU sono disponibili solo con una frequenza di 5 minuti.	Aurora MySQL e Aurora PostgreSQL	Crediti (vCPU/minuti)
`CPUUtilization`	La percentuale di CPU utilizzata da un'istanza database Aurora.	Aurora MySQL e Aurora PostgreSQL	Percentuale

Metrica	Descrizione	Si applica a	unità
`DatabaseConnections`	Il numero di connessioni di rete client all'istanza del database. Il numero di sessioni del database può essere superiore al valore del parametro perché il valore del parametro non include quanto segue: • Sessioni che non hanno più una connessione di rete ma che il database non ha ripulito • Sessioni create dal motore del database per i propri scopi • Sessioni create dalle funzionalità di esecuzione parallela del motore del database • Sessioni create dal pianificatore dei processi del motore del database • Connessioni Amazon Aurora	Aurora MySQL e Aurora PostgreSQL	Conteggio
`DDLLatency`	La durata media delle richieste, ad esempio, richieste di creazione, modifica e eliminazione.	Aurora MySQL	Millisecondi
`DDLThroughput`	Il numero medio di richieste DDL al secondo.	Aurora MySQL	Conteggio al secondo

Metrica	Descrizione	Si applica a	unità
`Deadlocks`	Il numero medio di deadlock nel database al secondo.	Aurora MySQL e Aurora PostgreSQL	Conteggio al secondo
`DeleteLatency`	La durata media delle operazioni di eliminazione.	Aurora MySQL	Millisecondi
`DeleteThroughput`	Il numero medio di query di eliminazione al secondo.	Aurora MySQL	Conteggio al secondo
`DiskQueueDepth`	Il numero di read/write richieste in sospeso in attesa di accesso al disco.	Aurora MySQL e Aurora PostgreSQL	Conteggio
`DMLLatency`	La durata media di inserimenti, aggiornamenti ed eliminazioni.	Aurora MySQL	Millisecondi
`DMLThroughput`	Il numero medio delle inserzioni, degli aggiornamenti e delle eliminazioni al secondo.	Aurora MySQL	Conteggio al secondo
`EngineUptime`	La durata di esecuzione dell'istanza.	Aurora MySQL e Aurora PostgreSQL	Secondi

Metrica	Descrizione	Si applica a	unità
`FreeableMemory`	La quantità di memoria RAM disponibile. Per i database Aurora MySQL e Aurora PostgreSQL, questa metrica segnala il valore del campo `MemAvailable` di `/proc/meminfo` .	Aurora MySQL e Aurora PostgreSQL	Byte
`FreeEphemeralStorage`	La quantità di spazio di archiviazione temporaneo disponibile. NVMe	Aurora PostgreSQL	Byte
`FreeLocalStorage`	La quantità di storage locale disponibile. A differenza di altri motori di database, per le istanze database Aurora questo parametro indica la quantità di storage disponibile per ogni istanza database. Questo valore dipende dalla classe dell'istanza database (per informazioni relative ai prezzi, consulta la pagina Prezzi di Amazon RDS). Puoi aumentare la quantità di spazio di storage gratuito per un'istanza scegliendo una classe di istanza database più ampia per l'istanza. (Non valido per Aurora Serverless v2).	Aurora MySQL e Aurora PostgreSQL	Byte

Metrica	Descrizione	Si applica a	unità
`InsertLatency`	La durata media delle operazioni di inserimento.	Aurora MySQL	Millisecondi
`InsertThroughput`	Il numero medio di operazioni di inserimento al secondo.	Aurora MySQL	Conteggio al secondo
`LoginFailures`	Il numero medio di tentativi di login non riusciti al secondo.	Aurora MySQL	Conteggio al secondo
`MaximumUsedTransactionIDs`	L'età dell'ID transazione non sottoposto al comando VACUUM meno recente, nelle transazioni. Se questo valore raggiunge 2.146.483 .648 (2^31 - 1.000.000), il database viene forzato in modalità di sola lettura, per evitare il wraparoun d dell'ID transazione. Per ulteriori informazioni, consulta Impedire gli errori wraparound dell'ID transazione nella documentazione di PostgreSQL.	Aurora PostgreSQL	Conteggio
`NetworkReceiveThroughput`	La quantità di throughput della rete ricevuto dai client da ogni istanza nel cluster di database Aurora. Questo throughput non include il traffico di rete tra le istanze nel cluster di database Aurora e il volume del cluster.	Aurora MySQL e Aurora PostgreSQL	Byte al secondo (la console mostra Megabyte al secondo)

Metrica	Descrizione	Si applica a	unità
`NetworkThroughput`	La quantità di throughput della rete ricevuto dai client e trasmessa agli stessi da ogni istanza nel cluster di database Aurora. Questo throughput non include il traffico di rete tra le istanze nel cluster di database Aurora e il volume del cluster.	Aurora MySQL e Aurora PostgreSQL	Byte al secondo
`NetworkTransmitThroughput`	La quantità di throughput della rete inviato ai client da ogni istanza nel cluster DB Aurora. Questo throughput non include il traffico di rete tra le istanze nel cluster di database e il volume del cluster.	Aurora MySQL e Aurora PostgreSQL	Byte al secondo (la console mostra Megabyte al secondo)
`NumBinaryLogFiles`	Il numero di file binlog generati.	Aurora MySQL	Conteggio
`OldestReplicationSlotLag`	L'entità del ritardo della replica più in ritardo in termini di dati WAL ricevuti.	Aurora PostgreSQL	Byte

Metrica	Descrizione	Si applica a	unità
`PurgeBoundary`	Numero di transazione fino al quale è consentita l'elimina zione da InnoDB. Se questa metrica non avanza per lunghi periodi di tempo, è una buona indicazione che l'eliminazione da InnoDB è bloccata da transazioni di lunga durata. Per indagare, controlla le transazioni attive sul cluster di database Aurora MySQL.	Aurora MySQL versione 2, versione 2.11 e successive Aurora MySQL versione 3, versione 3.08 e successive	Conteggio
`PurgeFinishedPoint`	Numero di transazione fino al quale viene eseguita l'eliminazione da InnoDB. Questa metrica può aiutarti a esaminare la velocità di eliminazione di InnoDB.	Aurora MySQL versione 2, versione 2.11 e successive Aurora MySQL versione 3, versione 3.08 e successive	Conteggio
`Queries`	Numero medio di query eseguite al secondo.	Aurora MySQL	Conteggio al secondo
`RDSToAuroraPostgreSQLReplicaLag`	La quantità di ritardo durante la replica degli aggiornamenti dall'istanza RDS PostgreSQL principale agli altri nodi presenti nel cluster.	Replica per Aurora PostgreSQL	Secondi

Metrica	Descrizione	Si applica a	unità
`ReadIOPS`	Il numero medio di I/O operazioni su disco al secondo, esclusi i report letti e scritti separatamente, a intervalli di 1 minuto.	Aurora MySQL e Aurora PostgreSQL	Conteggio al secondo
`ReadIOPSEphemeralStorage`	Il numero medio di I/O operazioni di lettura del disco su NVMe Ephemeral Storage.	Aurora PostgreSQL	Conteggio al secondo
`ReadLatency`	Il tempo medio impiegato per operazione su disco. I/O	Aurora MySQL e Aurora PostgreSQL	Secondi
`ReadLatencyEphemeralStorage`	Il tempo medio impiegato per ogni I/O operazione di lettura del disco per NVMe Ephemeral Storage.	Aurora PostgreSQL	Secondi
`ReadThroughput`	Il numero medio di byte letti dal disco al secondo.	Aurora MySQL e Aurora PostgreSQL	Byte al secondo
`ReadThroughputEphemeralStorage`	Il numero medio di byte letti dal disco al secondo per Ephemeral Storage. NVMe	Aurora PostgreSQL	Byte al secondo
`ReplicationSlotDiskUsage`	La quantità di spazio su disco occupata dai file degli slot di replica.	Aurora PostgreSQL	Byte

Metrica	Descrizione	Si applica a	unità
ResultSetCacheHitRatio	La percentuale di richieste servite dalla cache Resultset.	Aurora MySQL versione 2	Percentuale
RollbackSegmentHistoryListLength	I log di annullamento che registrano le transazioni sottoposte a commit con record contrassegnati per l'eliminazione. Questi record sono pianificati per essere elaborati dall'operazione di rimozione InnoDB.	Aurora MySQL	Conteggio
RowLockTime	Il tempo totale impiegato per l'acquisizione di blocchi di riga per le tabelle InnoDB.	Aurora MySQL	Millisecondi
SelectLatency	La durata media per le operazioni di selezione.	Aurora MySQL	Millisecondi
SelectThroughput	Numero medio di query di selezione al secondo.	Aurora MySQL	Conteggio al secondo
ServerlessDatabaseCapacity	Capacità corrente di un cluster di database Aurora Serverless.	Aurora MySQL e Aurora PostgreSQL	Conteggio
StorageNetworkReceiveThroughput	La quantità di velocità effettiva di rete ricevuta dal sottosistema di archiviazione Aurora mediante ogni istanza nel cluster di database.	Aurora MySQL e Aurora PostgreSQL	Byte al secondo

Metrica	Descrizione	Si applica a	unità
`StorageNetworkThroughput`	La quantità di throughput di rete ricevuto e inviato al sottosistema di archiviazione Aurora mediante ogni istanza nel cluster di database Aurora MySQL.	Aurora MySQL e Aurora PostgreSQL	Byte al secondo
`StorageNetworkTransmitThroughput`	La quantità di throughput di rete inviato al sottosistema di archiviazione Aurora mediante ogni istanza nel cluster di database Aurora.	Aurora MySQL e Aurora PostgreSQL	Byte al secondo
`SumBinaryLogSize`	La dimensione totale dei file binlog.	Aurora MySQL	Byte
`SwapUsage`	La quantità di spazio di scambio utilizzato. Questo parametro non è disponibile per le seguenti classi di istanza database: • db.r3.*, db.r4.* e db.r7g.* (Aurora MySQL) • db.r7g.* (Aurora PostgreSQL)	Aurora MySQL e Aurora PostgreSQL	Byte

Metrica	Descrizione	Si applica a	unità
`TempStorageIOPS`	Numero di IOPS per le operazioni sia di lettura che di scrittura eseguite sull'archiviazione locale collegata all'istanza database. Questo parametro rappresenta un conteggio e viene misurato una volta al secondo. Questa metrica è applicabile solo per Aurora Serverless v2.	Aurora MySQL e Aurora PostgreSQL	Conteggio al secondo
`TempStorageThroughput`	Quantità di dati trasferiti da e verso l'archiviazione locale associata all'istanza database. Questo parametro rappresenta i byte e viene misurato una volta al secondo. Questa metrica è applicabile solo per Aurora Serverless v2.	Aurora MySQL e Aurora PostgreSQL	Byte al secondo
`TransactionAgeMaximum`	L'età della più vecchia transazione attiva in esecuzione.	Aurora MySQL versione 3, versione 3.08 e successive	Secondi

Metrica	Descrizione	Si applica a	unità
`TransactionLogsDiskUsage`	La quantità di spazio su disco occupata dai log delle transazioni nell'istanza database di Aurora PostgreSQL. Questa metrica viene generata solo quando Aurora PostgreSQL utilizza la replica logica o. AWS Database Migration Service Per impostazione predefinita, Aurora PostgreSQL utilizza i record di log, non i log delle transazioni. Quando i log delle transazioni non sono in uso, il valore per questo parametro è `-1`.	Primario per Aurora PostgreSQL	Byte
`TruncateFinishedPoint`	L'identificatore della transazione fino al quale viene eseguito l'annullamento del troncamento.	Aurora MySQL versione 2, versione 2.11 e successive Aurora MySQL versione 3, versione 3.08 e successive	Conteggio
`UpdateLatency`	La durata media per le operazioni di aggiornamento.	Aurora MySQL	Millisecondi

Metrica	Descrizione	Si applica a	unità
`UpdateThroughput`	Il numero medio di aggiornamenti al secondo.	Aurora MySQL	Conteggio al secondo
`WriteIOPS`	Il numero di registri di scrittura di memoria Aurora generati al secondo. Questo è più o meno il numero di record di registro generati dal database. Questi non corrispondono a 8K pagine di scrittura e non corrispondono ai pacchetti di rete inviati.	Aurora MySQL e Aurora PostgreSQL	Conteggio al secondo
`WriteIOPSEphemeralStorage`	Il numero medio di operazioni di scrittura su disco su Ephemeral Storage. I/O NVMe	Aurora PostgreSQL	Conteggio al secondo
`WriteLatency`	Il tempo medio impiegato per operazione su disco. I/O	Aurora MySQL e Aurora PostgreSQL	Secondi
`WriteLatencyEphemeralStorage`	Il tempo medio impiegato per ogni I/O operazione di scrittura su disco per NVMe Ephemeral Storage.	Aurora PostgreSQL	Secondi
`WriteThroughput`	Il numero medio di byte scritti nello storage persistente ogni secondo.	Aurora MySQL e Aurora PostgreSQL	Byte al secondo
`WriteThroughputEphemeralStorage`	Il numero medio di byte scritti su disco al secondo per lo storage Ephemeral. NVMe	Aurora PostgreSQL	Byte al secondo

Metriche CloudWatch di utilizzo di

Il `AWS/Usage` namespace in Amazon CloudWatch include parametri di utilizzo a livello di account per le quote dei servizi Amazon RDS. CloudWatch raccoglie automaticamente i parametri di utilizzo per tutti. Regioni AWS

Per ulteriori informazioni, consulta i parametri di CloudWatch utilizzo nella Amazon CloudWatch User Guide. Per ulteriori informazioni sulle quote, consulta Quote e vincoli per Amazon Aurora e Requesting a quota increase nella Guida per l'utente di Service Quotas.

Metrica	Description	Unità*
`AuthorizationsPerDBSecurityGroup`	Il numero di regole di ingresso per gruppo di sicurezza database nell' Account AWS. Il valore utilizzato è il numero massimo di regole di ingresso in un gruppo di sicurezza database nell'account. Altri gruppi di sicurezza database nell'account potrebbero avere un numero inferiore di regole di ingresso.	Conteggio
`CustomEndpointsPerDBCluster`	Il numero di endpoint personalizzati per cluster di database nell' Account AWS. Il valore utilizzato è il numero massimo di endpoint personalizzati in un cluster di database dell'account. Altri cluster di database nell'account possono avere un numero inferiore di endpoint personalizzati.	Conteggio
`DBClusterParameterGroups`	Il numero di gruppi di parametri del cluster di database nel tuo Account AWS. I gruppi di parametri di default non vengono conteggiati.	Conteggio
`DBClusterRoles`	Il numero di ruoli associati AWS Identity and Access Management (IAM) per cluster DB nel tuo Account AWS. Il valore utilizzato è il numero massimo di ruoli IAM associati per un cluster di database nell'account. Altri cluster di database nell'account potrebbero avere un numero inferiore di ruoli IAM associati.	Conteggio
`DBClusters`	Il numero di cluster di database Amazon Aurora nel tuo Account AWS.	Conteggio

Metrica	Description	Unità*
DBInstanceRoles	Il numero di ruoli associati AWS Identity and Access Management (IAM) per istanza DB nel tuo Account AWS. Il valore utilizzato è il numero massimo di ruoli IAM associati per un'istanza database nell'account. Altre istanze database nell'account potrebbero avere un numero inferiore di ruoli IAM associati.	Conteggio
DBInstances	Il numero di istanze database nel tuo Account AWS.	Conteggio
DBParameterGroups	Il numero di gruppi di parametri database nel tuo Account AWS. I gruppi di parametri database di default non vengono conteggiati.	Conteggio
DBSubnetGroups	Il numero di gruppi di sottoreti nel tuo Account AWS. Il gruppo di sottoreti predefinito non viene conteggiato.	Conteggio
EventSubscriptions	Il numero di sottoscrizioni alle notifiche eventi presenti nell' Account AWS.	Conteggio
Integrations	Il numero di integrazioni Zero-ETL con Amazon Redshift nell' Account AWS.	Conteggio
ManualClusterSnapshots	Il numero di snapshot del cluster di database creati manualmente nel tuo Account AWS. Gli snapshot non validi non vengono conteggiati.	Conteggio
OptionGroups	Il numero di gruppi di opzioni nel tuo Account AWS. I gruppi di opzioni di default non vengono conteggiati.	Conteggio
Proxies	Il numero di proxy RDS presenti nel tuo account. AWS	Conteggio
ReadReplicasPerMaster	Il numero di repliche di lettura per istanza database nell'account. Il valore utilizzato è il numero massimo di repliche di lettura per un'istanza database nell'account. Altre istanze database nell'account potrebbero avere un numero inferiore di repliche di lettura.	Conteggio

Metrica	Description	Unità*
`ReservedDBInstances`	Il numero di istanze database riservate nel tuo Account AWS. Le istanze ritirate o rifiutate non vengono conteggiate.	Conteggio
`SubnetsPerDBSubnetGroup`	Il numero di sottoreti per gruppo di sottoreti di database nell' Account AWS. Il numero più alto di sottoreti per un gruppo di sottoreti di database nell'account. Altri gruppi di sottoreti di database nell'account potrebbero avere un numero inferiore di sottoreti.	Conteggio

> **Note**
>
> Amazon RDS non pubblica unità per i parametri di utilizzo. CloudWatch Le unità sono presenti solo nella documentazione.

CloudWatch Dimensioni Amazon per

Puoi filtrare i dati dei parametri di Aurora utilizzando qualsiasi dimensione riportata nella seguente tabella.

Dimensione	Filtra i dati richiesti per...
`DBInstanceIdentifier`	Un'istanza database specifica.
`DBClusterIdentifier`	Un cluster di database Aurora specifico.
`DBClusterIdentifier, Role`	Un cluster di database Aurora specifico, che aggrega il parametro per ruolo istanza (WRITER/READER). Ad esempio, puoi aggregare i parametri per tutte le istanze READER che appartengono a un cluster.
`DbClusterIdentifier, EngineName`	Una combinazione di nome specifica applicabile al cluster database Aurora e al motore Aurora. Ad esempio, è possibile

Dimensione	Filtra i dati richiesti per...
	visualizzare il parametro `VolumeReadIOPs` per il cluster `ams1` e il motore `aurora`.
`DatabaseClass`	Tutte le istanze di una classe di database. Ad esempio, puoi aggregare i parametri per tutte le istanze che appartengono alla classe database `db.r5.large` .
`EngineName`	Solo il nome del motore identificato. Ad esempio, puoi aggregare i parametri per tutte le istanze con nome del motore `aurora-postgresql` .
`SourceRegion`	Usa solo la regione specificata. Ad esempio, puoi aggregare i parametri per tutte le istanze database nella regione `us-east-1` .

Disponibilità dei parametri Aurora nella console Amazon RDS

Non tutti i parametri forniti da Amazon Aurora sono disponibili nella console Amazon RDS. Puoi comunque visualizzarli utilizzando altri strumenti, come la CLI AWS e l'API CloudWatch. Inoltre, alcuni dei parametri nella console Amazon RDS sono visualizzati solo per classi di istanze specifiche o con nomi e unità di misura diversi.

Argomenti

- Parametri Aurora disponibili nella vista Last Hour (Ultima ora)
- Aurora metriche disponibili in casi specifici
- Parametri Aurora non disponibili nella console

Parametri Aurora disponibili nella vista Last Hour (Ultima ora)

È possibile visualizzare un sottoinsieme di parametri Aurora nella vista Last Hour (Ultima ora) della console Amazon RDS. La tabella seguente elenca le categorie e i parametri associati visualizzati nella console Amazon RDS per un'istanza Aurora.

Categoria	Parametri
SQL	ActiveTransactions BlockedTransactions BufferCacheHitRatio CommitLatency CommitThroughput DatabaseConnections DDLLatency DDLThroughput Deadlocks DMLLatency DMLThroughput LoginFailures ResultSetCacheHitRatio SelectLatency SelectThroughput
System (Sistema)	AuroraReplicaLag AuroraReplicaLagMaximum AuroraReplicaLagMinimum CPUCreditBalance CPUCreditUsage CPUUtilization

Categoria	Parametri

	`FreeableMemory`
	`FreeLocalStorage` (Non valido per Aurora Serverless v2).
	`NetworkReceiveThroughput`

Distribuzione	`AuroraReplicaLag`
	`BufferCacheHitRatio`
	`ResultSetCacheHitRatio`
	`SelectThroughput`

Aurora metriche disponibili in casi specifici

Inoltre, alcuni dei parametri Aurora sono visualizzati solo per classi di istanze specifiche, solo per istanze database o con nomi e unità di misura diversi:

- I parametri `CPUCreditBalance` e `CPUCreditUsage` vengono visualizzati solo per le classi di istanza `db.t2` MySQL di Aurora e per le classi di istanza `db.t3` PostgreSQL di Aurora.
- I seguenti parametri che sono visualizzati con nomi diversi, come elencato:

Parametro	Display name (Nome visualizzato)
`AuroraReplicaLagMaximum`	Ritardo replica massimo
`AuroraReplicaLagMinimum`	Ritardo replica minimo
`DDLThroughput`	DDL
`NetworkReceiveThroughput`	Throughput di rete
`VolumeBytesUsed`	[Fatturati] Byte di volume utilizzati
`VolumeReadIOPs`	[Fatturati] IOPS lettura volume
`VolumeWriteIOPs`	[Fatturati] IOPS scrittura volume

- I seguenti parametri si applicano a un intero cluster di database Aurora, ma vengono visualizzati solo quando si visualizzano le istanze database per un cluster di database Aurora nella console Amazon RDS:
  - `VolumeBytesUsed`
  - `VolumeReadIOPs`
  - `VolumeWriteIOPs`
- I seguenti parametri sono visualizzati in megabyte, invece di byte, nella console Amazon RDS:
  - `FreeableMemory`
  - `FreeLocalStorage`
  - `NetworkReceiveThroughput`
  - `NetworkTransmitThroughput`
- Le seguenti metriche si applicano a un cluster di database Aurora PostgreSQL con Letture ottimizzate per Aurora:
  - `AuroraOptimizedReadsCacheHitRatio`
  - `FreeEphemeralStorage`
  - `ReadIOPSEphemeralStorage`
  - `ReadLatencyEphemeralStorage`
  - `ReadThroughputEphemeralStorage`
  - `WriteIOPSEphemeralStorage`
  - `WriteLatencyEphemeralStorage`
  - `WriteThroughputEphemeralStorage`

Parametri Aurora non disponibili nella console

I seguenti parametri Aurora non sono disponibili nella console Amazon RDS:

- `AuroraBinlogReplicaLag`
- `DeleteLatency`
- `DeleteThroughput`
- `EngineUptime`
- `InsertLatency`
- `InsertThroughput`

- `NetworkThroughput`
- `Queries`
- `UpdateLatency`
- `UpdateThroughput`

Metriche Amazon CloudWatch per Approfondimenti sulle prestazioni di Amazon RDS

Approfondimenti sulle prestazioni pubblica automaticamente le metriche in Amazon CloudWatch. Gli stessi dati possono essere sottoposti a query da Performance Insights, ma disporre delle metriche in CloudWatch agevola l'aggiunta di allarmi CloudWatch. Agevola inoltre l'aggiunta delle metriche ai pannelli di controllo di CloudWatch esistenti.

Parametro	Descrizione
DBLoad	Il numero di sessioni attive per il database. Generalmente, si richiedono i dati per il numero medio di sessioni attive. In Performance Insights, questi dati sono oggetto di query come `db.load.avg` .
DBLoadCPU	Il numero di sessioni attive in cui il tipo evento di attesa è CPU. In Performance Insights, questi dati sono oggetto di query come `db.load.avg` , filtrate per tipo di evento di attesa `CPU`.
DBLoadNonCPU	Il numero di sessioni attive in cui il tipo evento di attesa non è CPU.
DBLoadRelativeToNumVCPUs	Il rapporto tra il carico del database e il numero di CPU virtuali per il database.

> **Note**
>
> Questi parametri vengono pubblicati su CloudWatch solo in presenza di carico sull'istanza database.

Puoi esaminare queste metriche tramite la console CloudWatch, la AWS CLI o l'API di CloudWatch. Puoi anche esaminare altre metriche dei contatori di Approfondimenti sulle prestazioni utilizzando una speciale funzione matematica di metrica. Per ulteriori informazioni, consulta Esecuzione di query su altre metriche dei contatori di Approfondimenti sulle prestazioni in CloudWatch.

Ad esempio, puoi ottenere le statistiche per il parametro `DBLoad` eseguendo il comando get-metric-statistics.

```
aws cloudwatch get-metric-statistics \
    --region us-west-2 \
    --namespace AWS/RDS \
    --metric-name DBLoad  \
    --period 60 \
    --statistics Average \
    --start-time 1532035185 \
    --end-time 1532036185 \
    --dimensions Name=DBInstanceIdentifier,Value=db-loadtest-0
```

Questo esempio genera un output simile a quello riportato di seguito.

```
{
  "Datapoints": [
  {
  "Timestamp": "2021-07-19T21:30:00Z",
  "Unit": "None",
  "Average": 2.1
  },
  {
  "Timestamp": "2021-07-19T21:34:00Z",
  "Unit": "None",
  "Average": 1.7
  },
  {
  "Timestamp": "2021-07-19T21:35:00Z",
  "Unit": "None",
```

```
"Average": 2.8
},
{
"Timestamp": "2021-07-19T21:31:00Z",
"Unit": "None",
"Average": 1.5
},
{
"Timestamp": "2021-07-19T21:32:00Z",
"Unit": "None",
"Average": 1.8
},
{
"Timestamp": "2021-07-19T21:29:00Z",
"Unit": "None",
"Average": 3.0
},
{
"Timestamp": "2021-07-19T21:33:00Z",
"Unit": "None",
"Average": 2.4
}
],
"Label": "DBLoad"
}
```


Per ulteriori informazioni su CloudWatch, consulta Che cos'è Amazon CloudWatch? nella Guida per l'utente di Amazon CloudWatch.

Esecuzione di query su altre metriche dei contatori di Approfondimenti sulle prestazioni in CloudWatch

> **Note**
>
> Se si abilita la modalità avanzata di Database Insights, Amazon RDS pubblica le metriche dei contatori di Approfondimenti sulle prestazioni in Amazon CloudWatch. Con Database Insights, non è necessario utilizzare la funzione matematica di metrica `DB_PERF_INSIGHTS`. È possibile utilizzare la dashboard Database Insights di CloudWatch per effettuare ricerche, eseguire query e impostare allarmi per le metriche dei contatori di Approfondimenti sulle prestazioni.

È possibile eseguire query, allarmi e grafici sulle metriche di Approfondimenti sulle prestazioni di RDS da CloudWatch. È possibile accedere alle informazioni sul cluster di database utilizzando la funzione matematica di metrica `DB_PERF_INSIGHTS` per CloudWatch. Questa funzione consente di utilizzare le metriche di Approfondimenti sulle prestazioni che non vengono segnalate direttamente a CloudWatch per creare una nuova serie temporale.

È possibile utilizzare la nuova funzione matematica di metrica facendo clic sul menu a discesa Aggiungi formula nella schermata Seleziona metrica della console CloudWatch. È possibile utilizzarla per creare allarmi e grafici sulle metriche di Approfondimenti sulle prestazioni o su combinazioni di metriche CloudWatch e Approfondimenti sulle prestazioni, inclusi allarmi ad alta risoluzione per metriche inferiori al minuto. È inoltre possibile utilizzare la funzione a livello di codice includendo l'espressione matematica di metrica in una richiesta `get-metric-data`. Per ulteriori informazioni, consulta Sintassi e funzioni della matematica dei parametri e Creazione di un allarme sui parametri dei contatori di Approfondimenti sulle prestazioni da un database AWS.

Metriche contatore di Performance Insights

Le metriche contatore sono parametri prestazionali di sistema operativo e database nel pannello di controllo Performance Insights. Per agevolare l'individuazione e l'analisi di problemi legati alle prestazioni, è possibile correlare le metriche contatore ai carichi dei database. È necessario aggiungere una funzione statistica alla metrica per ottenere i valori della metrica. Ad esempio, le funzioni supportate per il parametro `os.memory.active` sono `.avg`, `.min`,`.max`, `.sum` e `.sample_count`.

I parametri dei contatori vengono raccolte una volta al minuto. La raccolta dei parametri del sistema operativo dipende dall'attivazione o dalla disattivazione della funzionalità Monitoraggio avanzato. Se la funzionalità Monitoraggio avanzato è disattivata, i parametri del sistema operativo vengono raccolte una volta al minuto. Se la funzionalità Monitoraggio avanzato è attivata, i parametri del sistema operativo vengono raccolte per il periodo di tempo selezionato. Per ulteriori informazioni sull'attivazione o sulla disattivazione della funzionalità Monitoraggio avanzato, consulta Attivazione e disattivazione del monitoraggio avanzato.

Argomenti

- Contatori del sistema operativo in Performance Insights
- Contatori di Performance Insights per Aurora MySQL
- Contatori di Performance Insights per Aurora PostgreSQL

Contatori del sistema operativo in Performance Insights

I seguenti contatori del sistema operativo, con prefisso `os`, sono disponibili in Approfondimenti sulle prestazioni per Aurora PostgreSQL e Aurora MySQL.

Puoi utilizzare l'API `ListAvailableResourceMetrics` per l'elenco dei parametri dei contatori disponibili per l'istanza database. Per ulteriori informazioni, consulta la guida ListAvailableResourceMetricsdi riferimento dell'API Amazon RDS Performance Insights.

Contatore	Tipo	Unità	Metrica	Description
Attivo	Memoria	Kilobyte	os.memory .active	La quantità di memoria assegnata, in kilobyte.
Buffer	Memoria	Kilobyte	os.memory .buffers	La quantità di memoria utilizzata per il buffering delle I/O richieste prima della scrittura sul dispositivo di storage, in kilobyte.
Cached	Memoria	Kilobyte	os.memory .cached	La quantità di memoria utilizzata per la memorizzazione nella cache dell'I/O basato sul file system, in kilobyte.
DB Cache	Memoria	Byte	os.memory .db.cache	La quantità di memoria utilizzata per la cache

Contatore	Tipo	Unità	Metrica	Description
				della pagina in base al processo del database, incluso tmpfs (shmem), in byte.
DB Resident Set Size	Memoria	Byte	os.memory.db.residentSetSize	La quantità di memoria utilizzata per la cache anonima e di swap in base al processo del database, escluso tmpfs (shmem), in byte.
DB Swap	Memoria	Byte	os.memory.db.swap	La quantità di memoria utilizzata per lo scambio dal processo del database, in byte.
Dirty	Memoria	Kilobyte	os.memory.dirty	La quantità di pagine di memoria nella RAM che sono state modificate ma non scritte nel relativo blocco di dati nello storage, in kilobyte.

Contatore	Tipo	Unità	Metrica	Description
Gratuito	Memoria	Kilobyte	os.memory.free	La quantità di memoria non assegnata, in kilobyte.
Huge Pages libere	Memoria	Pagine	os.memoria. hugePagesFree	Il numero di pagine di grandi dimensioni gratuite. Le pagine di grandi dimensioni sono una caratteri stica del kernel di Linux.
Huge Pages Rsvd	Memoria	Pagine	os.memoria. hugePagesRsvd	Il numero di pagine di grandi dimensioni impegnate.
Dimensioni Huge Pages	Memoria	Kilobyte	os.memoria. hugePagesSize	La dimensione per ogni unità delle pagine di grandi dimension i, in kilobyte.
Huge Pages Surp	Memoria	Pagine	os.memoria. hugePagesSurp	Il numero di pagine di grandi dimension i in eccesso disponibili sul totale.
Totale Huge Pages	Memoria	Pagine	os.memoria. hugePagesTotal	Il numero totale di Huge Pages.

Contatore	Tipo	Unità	Metrica	Description
Inattivo	Memoria	Kilobyte	os.memory .inactive	La quantità di pagine di memoria utilizzate meno frequentemente, in kilobyte.
Mapped	Memoria	Kilobyte	os.memory .mapped	La quantità totale di contenuti del sistema di file che è mappata in memoria all'interno di uno spazio di indirizzamento del processo, in kilobyte.
Out of Memory Kill Count	Memoria	Interruzioni	os.memoria. outOfMemoryKillCount	Il numero di interruzioni OOM avvenute nell'ultimo intervallo di raccolta.
Tabelle delle pagine	Memoria	Kilobyte	os.memory .pageTables	La quantità di memoria utilizzata dalle tabelle della pagina, in kilobyte.
Slab	Memoria	Kilobyte	os.memory.slab	La quantità di strutture dati del kernel riutilizzabili, in kilobyte.

Contatore	Tipo	Unità	Metrica	Description
Totale	Memoria	Kilobyte	os.memory.total	La quantità totale di memoria, in kilobyte.
Writeback	Memoria	Kilobyte	os.memory .writeback	La quantità di pagine sporche nella RAM che sono ancora scritte nello storage di backup, in kilobyte.
Guest	Utilizzo CPU	Percentuale	os.cpuUtilization. guest	La percentuale di CPU utilizzat a dai programmi guest.
Sospeso	Utilizzo CPU	Percentuale	os.cpuUtilization. idle	La percentuale di tempo CPU che è inattiva.
Irq	Utilizzo CPU	Percentuale	os.cpuUtilization. irq	La percentuale di CPU utilizzata dalle interruzioni dei software.
Nice	Utilizzo CPU	Percentuale	os.cpuUtilization. nice	La percentuale di CPU utilizzat a dai programmi in esecuzione con priorità più bassa.

Contatore	Tipo	Unità	Metrica	Description
Steal	Utilizzo CPU	Percentuale	os.cpuUtilization.steal	La percentuale di CPU utilizzata da altre macchine virtuali.
Sistema	Utilizzo CPU	Percentuale	os.cpuUtilization.system	La percentuale di CPU utilizzata dal kernel.
Totale	Utilizzo CPU	Percentuale	os.cpuUtilization.total	La percentuale totale del CPU utilizzata. Questo valore include il valore nice.
Utente	Utilizzo CPU	Percentuale	os.cpuUtilization.user	La percentuale di CPU utilizzata dai programmi utente.
Attendi	Utilizzo CPU	Percentuale	os.cpuUtilization.wait	La percentuale di CPU inutilizzata durante l'attesa dell' I/O accesso.
Aurora Storage Aurora Storage Bytes Rx	I/O del disco	Byte al secondo	os.diskio.AuroraStorage.auroraStorageBytesRx	Il numero di byte ricevuti al secondo dall’archiviazione Aurora.
Aurora Storage Aurora Storage Bytes Tx	I/O del disco	Byte al secondo	os.diskio.Aurora Storage.auroraStorageBytesTx	Il numero di byte caricati al secondo nell’archiviazione Aurora.

Contatore	Tipo	Unità	Metrica	Description
Aurora Storage Disk Queue Depth	I/O del disco	Richieste	os.diskio .Aurora Storage. diskQueueDepth	La lunghezza della coda del disco di archiviazione Aurora.
Aurora Storage Leggi PS IOs	I/O del disco	Richieste al secondo	os.diskio .AuroraStorage.Leggi PS IOs	Il numero di operazioni di lettura al secondo.
Aurora Storage Read Latency	I/O del disco	Millisecondi	os.diskIO .auroraStorage.readLatency	La latenza media di una I/O richiesta di lettura verso lo storage Aurora, in millisecondi.
Throughput di lettura dell'archiviazione Aurora	I/O del disco	Byte al secondo	os.diskIO .auroraStorage.readThroughput	La quantità di velocità effettiva di rete utilizzata dalle richieste al cluster di database, espressa in byte al secondo.
Aurora Storage Write PS IOs	I/O del disco	Richieste al secondo	os.diskio .AuroraStorage.Scrivi PS IOs	Il numero di operazioni di scrittura al secondo.

Contatore	Tipo	Unità	Metrica	Description
Aurora Storage Write Latency	I/O del disco	Millisecondi	os.diskIO.auroraStorage.writeLatency	La latenza media di una I/O richiesta di scrittura sullo storage Aurora, in millisecondi.
Aurora Storage Write Throughput	I/O del disco	Byte al secondo	os.diskIO.auroraStorage.writeThroughput	La quantità di throughput di rete effettivo utilizzato dalle risposte del cluster di database, espressa in byte al secondo.
Rdstemp Avg Queue Len	I/O del disco	Richieste	os.diskio.RDSTemp.avgQueueLen	Il numero di richieste in attesa nella coda del I/O dispositivo.
Restemp Avg Req Sz	I/O del disco	Richieste	os.diskio.RDSTemp.avgReqSz	Il numero di richieste in attesa nella coda del I/O dispositivo.

Contatore	Tipo	Unità	Metrica	Description
Restemp Await	I/O del disco	Millisecondi	os.diskIO .rdstemp.await	Il numero di millisecondi necessari per rispondere alle richieste , compreso il tempo della coda e il tempo del servizio.
Restemp Read PS IOs	I/O del disco	Richieste	os.diskio .rdstemp.Leggi PS IOs	Il numero di operazioni di lettura al secondo.
Rdstemp Read KB	I/O del disco	Kilobyte	os.diskIO .rdstemp.readKb	Il numero totale di kilobyte letti.
Rdstemp Read KB PS	I/O del disco	Kilobyte al secondo	os.diskIO .rdstemp. readKbPS	Il numero di kilobytes letti al secondo.
Rdstemp Rrqm PS	I/O del disco	Richieste al secondo	os.diskIO .rdstemp.rrqmPS	Il numero di richieste di lettura unite in coda al secondo.
Rdstemp TPS	I/O del disco	Transazioni al secondo	os.diskIO .rdstemp.tps	Il numero di I/ O transazioni al secondo.

Contatore	Tipo	Unità	Metrica	Description
Restemp Util	I/O del disco	Percentuale	os.diskIO .rdstemp.util	La percentuale di tempo della CPU durante il quale sono state emesse le richieste.
Restemp Write PS IOs	I/O del disco	Richieste al secondo	os.diskio .rdstemp.Write PS IOs	Il numero di operazioni di scrittura al secondo.
Rdstemp Write KB	I/O del disco	Kilobyte	os.diskIO .rdstemp.writeKb	Il numero totale di kilobyte scritti.
Rdstemp Write KB PS	I/O del disco	Kilobyte al secondo	os.diskIO .rdstemp. writeKbPS	Il numero di kilobytes scritti al secondo.
Rdstemp Wrqm PS	I/O del disco	Richieste al secondo	os.diskIO .rdstemp. wrqmPS	Il numero di richieste di scrittura unite in coda al secondo.
Bloccato	Processi	Processi	os.tasks.blocked	Il numero di attività che sono bloccate.
In esecuzione	Processi	Processi	os.tasks.running	Il numero di attività che sono in esecuzione.
Sleeping	Processi	Processi	os.tasks.sleeping	Il numero di attività che sono a riposo.

Contatore	Tipo	Unità	Metrica	Description
Arrestato	Processi	Processi	os.tasks.stopped	Il numero di attività che sono arrestate.
Totale	Processi	Processi	os.tasks.total	Il numero totale di attività.
Zombie	Processi	Processi	os.tasks.zombie	Il numero di attività secondari e che sono inattive con un'attività genitore attiva.
One	Media carico al minuto	Processes	sistema operativo. loadAvera geMinute.uno	Il numero di processi che richiedono l'ora della CPU nell'ultimo minuto.
Fifteen	Media carico al minuto	Processes	così. loadAvera geMinute. quindici	Il numero di processi che richiedono l'ora della CPU negli ultimi 15 minuti.
Cinque	Media carico al minuto	Processes	così. loadAvera geMinute.cinque	Il numero di processi che richiedono l'ora della CPU negli ultimi 5 minuti.

Contatore	Tipo	Unità	Metrica	Description
Cached	Swap	Kilobyte	os.swap.cached	La quantità di memoria di scambio, in kilobyte, utilizzata come memoria cache.
Gratuito	Swap	Kilobyte	os.swap.free	La quantità di memoria di scambio libera, in kilobyte.
In	Swap	Kilobyte	os.swap.in	Quantità di memoria, in kilobyte, scambiata in ingresso nel disco.
Out	Swap	Kilobyte	os.swap.out	Quantità di memoria, in kilobyte, scambiata in uscita dal disco.
Totale	Swap	Kilobyte	os.swap.total	La quantità totale di memoria di scambio disponibile, in kilobyte.

Contatore	Tipo	Unità	Metrica	Description
Max Files	File system	File	os.fileSys.maxFiles	Il numero massimo di file che è possibile creare per il file system su tutti i volumi di archiviazione.
Used Files	File system	File	os.fileSys.usedFiles	Il numero di file nel file system in tutti i volumi di archiviazione.
Used File Percent	File system	File	OS.Filesys.usedFilePercent	La percentuale di file disponibili in uso su tutti i volumi di archiviazione.
Used Percent	File system	Percentuale	os.fileSys.usedPercent	La percentuale di spazio su disco del file system in uso su tutti i volumi di archiviazione.
Used	File system	Kilobyte	os.fileSys.used	La quantità di spazio su disco utilizzata dai file nel file system in tutti i volumi di archiviazione, in kilobyte.

Contatore	Tipo	Unità	Metrica	Description
Totale	File system	Kilobyte	os.fileSys.total	Lo spazio totale su disco disponibile per il file system su tutti i volumi di archiviazione, in kilobyte.
Max Files	File system	File	OS.FileSys.<volumeName>.max Files	Il numero massimo di file che è possibile creare per il volume di archiviazione.
Used Files	File system	File	OS.Filesys.<volumeName>.File usati	Il numero di file nel volume di archiviazione.
Used File Percent	File system	File	OS.Filesys.<volumeName>.usedFilePercent	La percentuale di file disponibili in uso nel volume di archiviazione.
Used Percent	File system	Percentuale	OS.Filesys.<volumeName>.Percentuale di utilizzo	La percentuale di spazio su disco del volume di archiviazione in uso.
Used	File system	Kilobyte	OS.Filesys.<volumeName>.usato	La quantità di spazio su disco utilizzata dai file nel volume di archiviazione, in kilobyte.

Contatore	Tipo	Unità	Metrica	Description
Totale	File system	Kilobyte	OS.Filesys.<volumeName>.totale	Lo spazio totale su disco disponibile nel volume di archiviazione, in kilobyte.
Rx	Rete	Byte al secondo	os.network.rx	Il numero di bytes ricevuti al secondo.
Tx	Rete	Byte al secondo	os.network.tx	Il numero di bytes caricati al secondo.
Acu Utilization	Generali	Percentuale	os.general.acuUtilization	La percentuale di capacità attuale rispetto alla capacità massima configurata.
Max Configured Acu	Generali	ACUs	sistema operativo generale.maxConfiguredAcu	La capacità massima configurata dall'utente, in unità di capacità Aurora ()ACUs.
Min Configured Acu	Generali	ACUs	sistema operativo generale.minConfiguredAcu	La capacità minima configurata dall'utente, in ACUs.

Contatore	Tipo	Unità	Metrica	Description
Num VCPUs	Generali	v CPUs	os.general.num VCPUs	Il numero di virtual CPUs (vCPUs) per l'istanza DB.
Serverless Database Capacity	Generali	ACUs	sistema operativo generale. serverlessDatabase Capacity	La capacità attuale dell'istanza, in ACUs.

Contatori di Performance Insights per Aurora MySQL

I seguenti contatori del database sono disponibili in Performance Insights per Aurora MySQL.

Argomenti

- Contatori nativi per Aurora MySQL
- Contatori non nativi per Aurora MySQL

Contatori nativi per Aurora MySQL

I parametri nativi sono definiti dal motore del database e non da Amazon Aurora. Le definizioni di questi parametri nativi sono disponibili in Variabili dello stato del server nella documentazione di MySQL.

Contatore	Tipo	Unità	Parametro
Com_analyze	SQL	Query al secondo	db.SQL.Com_analyze
Com_optimize	SQL	Query al secondo	db.SQL.Com_optimize

Contatore	Tipo	Unità	Parametro
Com_select	SQL	Query al secondo	db.SQL.Com_select
Innodb_rows_deleted	SQL	Righe al secondo	db.SQL.Innodb_rows_deleted
Innodb_rows_inserted	SQL	Righe al secondo	db.SQL.Innodb_rows_inserted
Innodb_rows_read	SQL	Righe al secondo	db.SQL.Innodb_rows_read
Innodb_rows_updated	SQL	Righe al secondo	db.SQL.Innodb_rows_updated
Query	SQL	Query al secondo	db.SQL.Queries
Questions	SQL	Query al secondo	db.SQL.Questions
Select_full_join	SQL	Query al secondo	db.SQL.Select_full_join
Select_full_range_join	SQL	Query al secondo	db.SQL.Select_full_range_join
Select_range	SQL	Query al secondo	db.SQL.Select_range
Select_range_check	SQL	Query al secondo	db.SQL.Select_range_check
Select_scan	SQL	Query al secondo	db.SQL.Select_scan
Slow_queries	SQL	Query al secondo	db.SQL.Slow_queries

Contatore	Tipo	Unità	Parametro
Sort_merge_passes	SQL	Query al secondo	db.SQL.Sort_merge_passes
Sort_range	SQL	Query al secondo	db.SQL.Sort_range
Sort_rows	SQL	Query al secondo	db.SQL.Sort_rows
Sort_scan	SQL	Query al secondo	db.SQL.Sort_scan
Total_query_time	SQL	Millisecondi	db.SQL.Total_query_time
Table_locks_immediate	Locks	Richieste al secondo	db.Locks.Table_locks_immediate
Table_locks_waited	Locks	Richieste al secondo	db.Locks.Table_locks_waited
Innodb_row_lock_time	Locks	Millisecondi (media)	db.Locks.Innodb_row_lock_time
Aborted_clients	Utenti	Connessioni	db.Users.Aborted_clients
Aborted_connects	Utenti	Connessioni	db.Users.Aborted_connects
Connessioni	Utenti	Connessioni	db.Users.Connections

Contatore	Tipo	Unità	Parametro
External_threads_connected	Utenti	Connessioni	db.Users.External_threads_connected
max_connections	Utenti	Connessioni	db.Users.max_connections
Threads_connected	Utenti	Connessioni	db.Users.Threads_connected
Threads_created	Utenti	Connessioni	db.Users.Threads_created
Threads_running	Utenti	Connessioni	db.Users.Threads_running
Created_tmp_disk_tables	Temp	Tabelle al secondo	db.Temp.Created_tmp_disk_tables
Created_tmp_tables	Temp	Tabelle al secondo	db.Temp.Created_tmp_tables
Innodb_buffer_pool_pages_data	Cache	Pagine	db.Cache.Innodb_buffer_pool_pages_data
Innodb_buffer_pool_pages_total	Cache	Pagine	db.Cache.Innodb_buffer_pool_pages_total
Innodb_buffer_pool_read_requests	Cache	Pagine al secondo	db.Cache.Innodb_buffer_pool_read_requests
Innodb_buffer_pool_reads	Cache	Pagine al secondo	db.Cache.Innodb_buffer_pool_reads
Opened_tables	Cache	Tabelle	db.Cache.Opened_tables
Opened_table_definitions	Cache	Tabelle	db.Cache.Opened_table_definitions
Qcache_hits	Cache	Query	db.Cache.Qcache_hits

Contatori non nativi per Aurora MySQL

I parametri contatore non nativi sono contatori definiti da Amazon RDS. Un parametro non nativo può essere un parametro che si ottiene con una query specifica. Un parametro non nativo può essere un parametro derivato, dove vengono utilizzati due o più contatori nativi nei calcoli di rapporti, percentuali di riscontri o latenze.

Contatore	Tipo	Unità	Metrica	Descrizione	Definizione
active_transactions	Transa ni	db.Trar ctions.a tive_tra saction	Le transazio ni attive totali.	`SELECT COUNT(1) AS active_tr ansactions FROM INFOR MATION_SC HEMA.INNO DB_TRX`	
innodb_buffer_pool _hit_rate	Cache	db.Cac innoDB ffer_po _hit_rat	La percentua le di letture che InnoDB potrebbe soddisfar e dal pool di buffer.	`100 * innodb_bu ffer_pool _read_req uests / (innodb_ buffer_po ol_read_r equests + innodb_bu ffer_pool _reads)`	
innodb_buffer_pool _hits	Cache	Pagine al second	db.Cache. innoDB_bu ffer_pool _hits	Il numero di letture che InnoDB potrebbe soddisfare dal pool di buffer.	`innodb_bu ffer_pool _read_requests - innodb_bu ffer_pool _reads`
innodb_buffer_pool _usage	Cache	Percen le	db.Cache. innoDB_bu		`Innodb_bu ffer_pool`

Contatore	Tipo	Unità	Metrica	Descrizione	Definizione
			ffer_pool_usage	La percentuale del pool di buffer di InnoDB che contiene dati (pagine). **Note** Quando si utilizzano tabelle compresse, questo valore può variare. Per ulteriori dettagli, consulta le informazioni su `Innodb_buffer_pool_pages_data` e `Innodb_buffer_pool_pages_total` in Variabili dello stato del server nella documentazione di MySQL.	`_pages_data / Innodb_buffer_pool_pages_total * 100.0`

Contatore	Tipo	Unità	Metrica	Descrizione	Definizione
innodb_deadlocks	Locks	db.Lock.innodb_deadlocks	Il numero totale di deadlock.	`SELECT COUNT AS innodb_deadlocks FROM INFORMATION_SCHEMA.INNODB_METRICS WHERE NAME='lock_deadlocks'`	
innodb_lock_timeouts	Locks	db.Lock.innodb_lock_timeouts	Il numero totale di deadlock scaduti.	`SELECT COUNT AS innodb_lock_timeouts FROM INFORMATION_SCHEMA.INNODB_METRICS WHERE NAME='lock_timeouts'`	
innodb_row_lock_waits	Locks	db.Lock.innodb_row_lock_waits	Il numero totale di blocchi alle righe che ha determinato un'attesa.	`SELECT COUNT AS innodb_row_lock_waits FROM INFORMATION_SCHEMA.INNODB_METRICS WHERE NAME='lock_row_lock_waits'`	

Contatore	Tipo	Unità	Metrica	Descrizione	Definizione
innodb_rows_changed	SQL	db.SQL.Innodb_rows_changed	Il totale delle operazioni delle righe di InnoDB.	db.SQL.Innodb_rows_inserted + db.SQL.Innodb_rows_deleted + db.SQL.Innodb_rows_updated	
query_cache_hit_rate	Cache	Percentile	db.Cache.query_cache_hit_rate	La percentuale di riscontri della cache (cache query) del set di risultati MySQL.	Qcache_hits / (QCache_hits + Com_select) * 100
temp_disk_tables_percent	Temp	db.Temp.disk_tables_percent	La percentuale di tabelle temporanee create sul disco dal server durante l'esecuzione delle istruzioni.	(db.Temp.Created_tmp_disk_tables / db.Temp.Created_tmp_tables) * 100	

Contatore	Tipo	Unità	Metrica	Descrizione	Definizione
trx_rseg_history_len	Transazioni	Nessuno	db.Transactions.trx_rseg_history_len	L'elenco delle pagine di log degli annullamenti per le transazioni confermate che viene gestito dal sistema di transazioni InnoDB per implementare il controllo della concorrenza tra più versioni. Per ulteriori informazioni sui dettagli dei record dei log degli annullamenti, consulta https://dev.mysql.com/doc/refman/8.0/en/innodb-multi-versioning.html nella documentazione di MySQL.	`SELECT COUNT AS trx_rseg_history_len FROM INFORMATION_SCHEMA.INNODB_METRICS WHERE NAME='trx_rseg_history_len'`

Contatori di Performance Insights per Aurora PostgreSQL

I seguenti contatori del database sono disponibili in Performance Insights per Aurora PostgreSQL.

Argomenti

- Contatori nativi per Aurora PostgreSQL
- Contatori non nativi per Aurora PostgreSQL

Contatori nativi per Aurora PostgreSQL

I parametri nativi sono definiti dal motore del database e non da Amazon Aurora. Le definizioni di questi parametri nativi sono riportate in Viewing Statistics nella documentazione di PostgreSQL.

Contatore	Tipo	Unità	Parametro
tup_deleted	SQL	Tuple al secondo	db.SQL.tup_deleted
tup_fetched	SQL	Tuple al secondo	db.SQL.tup_fetched
tup_inserted	SQL	Tuple al secondo	db.SQL.tup_inserted
tup_returned	SQL	Tuple al secondo	db.SQL.tup_returned
tup_updated	SQL	Tuple al secondo	db.SQL.tup_updated
blks_hit	Cache	Blocchi al secondo	db.Cache.blks_hit
buffers_alloc	Cache	Blocchi al secondo	db.Cache.buffers_alloc
buffers_checkpoint	Checkpoint	Blocchi al secondo	db.Checkpoint.buffers_checkpoint
checkpoints_req	Checkpoint	Checkpoint al minuto	db.Checkpoint.checkpoints_req
checkpoint_sync_time	Checkpoint	Millisecondi per checkpoint	db.Checkpoint.checkpoint_sync_time
checkpoints_timed	Checkpoint	Checkpoint al minuto	db.Checkpoint.checkpoints_timed
checkpoint_write_time	Checkpoint	Millisecondi per checkpoint	db.Checkpoint.checkpoint_write_time
maxwritten_clean	Checkpoint	Interruzioni clean lettura in background al minuto	db.Checkpoint.maxwritten_clean
deadlocks	Concorrenza	Deadlock al minuto	db.Concurrency.deadlocks
blk_read_time	I/O	Millisecondi	db.IO.blk_read_time

Contatore	Tipo	Unità	Parametro
blks_read	I/O	Blocchi al secondo	db.IO.blks_read
buffers_backend	I/O	Blocchi al secondo	db.IO.buffers_backend
buffers_backend_fsync	I/O	Blocchi al secondo	db.IO.buffers_backend_fsync
buffers_clean	I/O	Blocchi al secondo	db.IO.buffers_clean
temp_bytes	Temp	Byte al secondo	db.Temp.temp_bytes
temp_files	Temp	File al minuto	db.Temp.temp_files
xact_commit	Transazioni	Commit al secondo	db.Transactions.xact_commit
xact_rollback	Transazioni	Rollback al secondo	db.Transactions.xact_rollback
numbackends	Utente	Connessioni	db.User.numbackends
archived_count	WAL	File al minuto	db.WAL.archived_count

Contatori non nativi per Aurora PostgreSQL

I parametri contatori non nativi sono contatori definiti da Amazon Aurora. Un parametro non nativo può essere un parametro che si ottiene con una query specifica. Un parametro non nativo può essere un parametro derivato, dove vengono utilizzati due o più contatori nativi nei calcoli di rapporti, percentuali di riscontri o latenze.

Contatore	Tipo	Unità	Metrica	Descrizione	Definizione
checkpoint_sync_latency	Checkpoint	Millisecondi	db.Checkpoint.checkpoint_sync_latency	La quantità totale di tempo impiegato nella parte dell'elaborazione dei checkpoint dove i file vengono sincronizzati sul disco.	`checkpoint_sync_time / (checkpoints_timed + checkpoints_req)`
checkpoint_write_latency	Checkpoint	Millisecondi	db.Checkpoint.checkpoint_write_latency	La quantità totale di tempo impiegato nella parte dell'elaborazione dei checkpoint dove i file vengono scritti sul disco.	`checkpoint_write_time / (checkpoints_timed + checkpoints_req)`
local_blks_read	I/O	Blocchi	db.IO.local_blks_read	Numero totale di blocchi locali letti.	Non applicabile
local_blk_read_time	I/O	Millisecondi	db.IO.local_blk_read_time	Se `track_io_timing` è abilitato, tiene traccia del tempo totale, in millisecondi, impiegato per leggere i blocchi di file di dati locali, altrimenti il valore è zero. Per ulteriori informazioni, consulta track_io_timing.	Non applicabile

Contatore	Tipo	Unità	Metrica	Descrizione	Definizione
num_blocked_sessions	Locks	db.Locks.num_blocked_sessions	Il numero di sessioni bloccate.	–	
orcache_blks_hit	I/O	Query	db.IO.orcache_blks_hit	Numero totale di blocchi condivisi provenienti dalla cache delle letture ottimizzate.	Non applicabile
orcache_blk_read_time	I/O	Millisecondi	db.IO.orcache_blk_read_time	Se `track_io_timing` è abilitato, tiene traccia del tempo totale, in millisecondi, impiegato per leggere i blocchi di file di dati dalla cache delle letture ottimizzate, altrimenti il valore è zero. Per ulteriori informazioni, consulta track_io_timing.	Non applicabile
read_latency	I/O	Millisecondi	db.IO.read_latency	Il tempo impiegato per la lettura dei blocchi di file di dati dai back-end in questa istanza.	`blk_read_time / blks_read`
storage_blks_read	I/O	Blocchi	db.IO.storage_blks_read	Numero totale di blocchi condivisi letti dall'archiviazione Aurora.	Non applicabile

Contatore	Tipo	Unità	Metrica	Descrizione	Definizione
storage_blk_read_time	I/O	Millisecondi	db.IO.storage_blk_read_time	Se `track_io_timing` è abilitato, tiene traccia del tempo totale, in millisecondi, impiegato per leggere i blocchi di file di dati dall'archiviazione Aurora, altrimenti il valore è zero. Per ulteriori informazioni, consulta track_io_timing.	Non applicabile
num_blocked_sessions	Locks	db.Locks.num_blocked_sessions	Il numero di sessioni bloccate.	–	
active_count	Stato	Sessioni	db.state.active_count	Il numero di sessioni nello stato `active`.	Non applicabile
idle_count	Stato	Sessioni	db.state.idle_count	Il numero di sessioni nello stato `idle`.	Non applicabile
idle_in_transaction_aborted_count	Stato	Sessioni	db.state.idle_in_transaction_aborted_count	Il numero di sessioni nello stato `idle in transaction (aborted)` .	Non applicabile

Contatore	Tipo	Unità	Metrica	Descrizione	Definizione
idle_in_transaction_count	Stato	Sessioni	db.state.idle_in_transaction_count	Il numero di sessioni nello stato `idle in transaction` .	Non applicabile
idle_in_transaction_max_time	Stato	Secondi	db.state.idle_in_transaction_max_time	La durata, espressa in secondi, della transazione più lunga nello stato `idle in transaction` .	Non applicabile
logical_reads	SQL	Blocchi	db.SQL.logical_reads	Il numero totale di occorrenze e letture di blocchi.	`blks_hit + blks_read`
queries_started	SQL	Query	db.SQL.queries	Il numero di query avviate.	Non applicabile
queries_finished	SQL	Query	db.SQL.queries	Il numero di query completate.	Non applicabile
total_query_time	SQL	Millisecondi	db.SQL.total_query_time	Il tempo totale impiegato per l'esecuzione delle istruzioni, espresso in millisecondi.	Non applicabile
active_transactions	Transazioni	Transazioni	db.Transactions.active_transactions	Il numero di transazioni attive.	Non applicabile
blocked_transactions	Transazioni	Transazioni	db.Transactions.blocked_transactions	Il numero di transazioni bloccate.	Non applicabile

Contatore	Tipo	Unità	Metrica	Descrizione	Definizione
commit_latency	Transazioni	Microsecondi	db.Transactions.commit_latency	La durata media delle operazioni di commit.	`db.Transactions.duration_commits / db.Transactions.xact_commit`
duration_commits	Transazioni	Millisecondi	db.Transactions.duration_commits	Il tempo totale della transazione trascorso nell'ultimo minuto, espresso in millisecondi.	Non applicabile
max_used_xact_ids	Transazioni	Transazioni	db.Transactions.max_used_xact_ids	Il numero di transazioni che non sono state sottoposte al vacuum.	Non applicabile
oldest_inactive_logical_replication_slot_xid_age	Transazioni	Lunghezza	db.Transactions.oldest_inactive_logical_replication_slot_xid_age	L'età della transazione meno recente in uno slot di replica logica inattivo.	Non applicabile
oldest_active_logical_replication_slot_xid_age	Transazioni	Lunghezza	db.Transactions.oldest_active_logical_replication_slot_xid_age	L'età della transazione meno recente in uno slot di replica logica attivo.	Non applicabile

Contatore	Tipo	Unità	Metrica	Descrizione	Definizione
oldest_reader_feedback_xid_age	Transazioni	Lunghezza	db.Transactions.oldest_reader_feedback_xid_age	L'età della transazione meno recente di lunga durata su un'istanza di lettura Aurora o un'istanza di lettura del database globale Aurora.	Non applicabile
oldest_prepared_transaction_xid_age	Transazioni	Lunghezza	db.Transactions.oldest_prepared_transaction_xid_age	L'età della transazione preparata meno recente.	Non applicabile
oldest_running_transaction_xid_age	Transazioni	Lunghezza	db.Transactions.oldest_running_transaction_xid_age	L'età della transazione in esecuzione meno recente.	Non applicabile
max_connections	Utenti	Utenti	db.User.max_connections	Il numero massimo di connessioni consentite per un database come configurato nel parametro `max_connections` .	Non applicabile
total_auth_attempts	Utenti	Utenti	db.User.total_auth_attempts	Il numero di tentativi di connessione all'istanza.	Non applicabile

Contatore	Tipo	Unità	Metrica	Descrizione	Definizione
archive_failed_count	WAL	File al minuto	db.WAL.archive_failed_count	Il numero di tentativi non riusciti di archiviazione di file WAL, espresso in file al minuto.	Non applicabile

Statistiche SQL per Performance Insights

Le statistiche SQL sono parametri relativi alle prestazioni delle query SQL raccolti da Performance Insights. Performance Insights raccoglie statistiche durante ogni secondo in cui è in esecuzione una query e per ogni chiamata SQL. Le statistiche SQL sono una media per l'intervallo di tempo selezionato.

Un SQL Digest è un composito di tutte le query con un determinato modello ma che non hanno necessariamente gli stessi valori letterali. Il digest sostituisce i valori letterali con un punto interrogativo. Ad esempio, `SELECT * FROM emp WHERE lname= ?`. Questo digest può essere costituito dalle seguenti query figlio:

```
SELECT * FROM emp WHERE lname = 'Sanchez'
SELECT * FROM emp WHERE lname = 'Olagappan'
SELECT * FROM emp WHERE lname = 'Wu'
```

Tutti i motori supportano le statistiche SQL delle query a livello di digest.

Per informazioni sull'assistenza alla regione, al motore di database e alla classe di istanza per questa funzionalità, consulta Supporto di classe di istanza, regione e motore di database Amazon Aurora per funzionalità Performance Insights

Argomenti

- Statistiche SQL per Aurora MySQL
- Statistiche SQL per Aurora PostgreSQL

Statistiche SQL per Aurora MySQL

Aurora MySQL raccolgono le statistiche SQL solo a livello di digest. Nessuna statistica viene mostrata a livello di istruzione.

Argomenti

- Statistiche digest per Aurora MySQL
- Statistiche per seconod per Aurora MySQL
- Statistiche per chiamata per Aurora MySQL
- Statistiche principali per Aurora MySQL

Statistiche digest per Aurora MySQL

Performance Insights raccoglie statistiche digest SQL dalla tabella `events_statements_summary_by_digest`. La tabella `events_statements_summary_by_digest` è gestita dal database.

La tabella digest non dispone di una policy di espulsione. Il seguente messaggio viene visualizzato in Console di gestione AWS quando la tabella è piena:

```
Performance Insights is unable to collect SQL Digest statistics on new queries because
 the table events_statements_summary_by_digest is full.
Please truncate events_statements_summary_by_digest table to clear the issue. Check the
 User Guide for more details.
```

In questa situazione, ,Aurora MySQL non esegue il tracciamento delle query SQL. Per risolvere questo problema, Performance Insights tronca automaticamente la tabella del digest quando sono soddisfatte entrambe le condizioni seguenti:

- La tabella è piena.
- Performance Insights gestisce automaticamente Performance Schema.

  Per la gestione automatica, `performance_schema` deve essere impostato su `0` e la Fonte non deve essere impostata su `user`. Se Performance Insights non gestisce automaticamente lo schemda delle prestazioni, vedi Panoramica dello schema di prestazioni per Approfondimenti sulle prestazioni su Aurora MySQL.

Nella AWS CLI, controllare l'origine di un valore di parametro eseguendo il comando describe-db-parameters.

Statistiche per seconod per Aurora MySQL

Le seguenti statistiche SQL sono disponibili per cluster di database Aurora MySQL.

Parametro	Unità
db.sql_tokenized.stats.count_star_per_sec	Chiamate al secondo
db.sql_tokenized.stats.sum_timer_wait_per_sec	Latenza media al secondo (in ms)
db.sql_tokenized.stats.sum_select_full_join_per_sec	Seleziona full join al secondo
db.sql_tokenized.stats.sum_select_range_check_per_sec	Seleziona controllo intervallo al secondo
db.sql_tokenized.stats.sum_select_scan_per_sec	Seleziona scansione al secondo
db.sql_tokenized.stats.sum_sort_merge_passes_per_sec	Ordina i pass di unione al secondo
db.sql_tokenized.stats.sum_sort_scan_per_sec	Ordina scansioni al secondo
db.sql_tokenized.stats.sum_sort_range_per_sec	Ordina intervalli al secondo
db.sql_tokenized.stats.sum_sort_rows_per_sec	Ordina righe al secondo
db.sql_tokenized.stats.sum_rows_affected_per_sec	Righe interessate al secondo
db.sql_tokenized.stats.sum_rows_examined_per_sec	Righe esaminate al secondo
db.sql_tokenized.stats.sum_rows_sent_per_sec	Righe inviate al secondo

Parametro	Unità
db.sql_tokenized.stats.sum_created_tmp_disk_tables_per_sec	Tabelle disco temporanee create al secondo
db.sql_tokenized.stats.sum_created_tmp_tables_per_sec	Tabelle temporanee create al secondo
db.sql_tokenized.stats.sum_lock_time_per_sec	Tempo di blocco al secondo (in ms)

Statistiche per chiamata per Aurora MySQL

I seguenti parametri forniscono le statistiche per chiamata di un'istruzione SQL.

Parametro	Unità
db.sql_tokenized.stats.sum_timer_wait_per_call	Latenza media per chiamata (in ms)
db.sql_tokenized.stats.sum_select_full_join_per_call	Seleziona full join per chiamata
db.sql_tokenized.stats.sum_select_range_check_per_call	Seleziona controllo intervallo per chiamata
db.sql_tokenized.stats.sum_select_scan_per_call	Seleziona scansioni per chiamata
db.sql_tokenized.stats.sum_sort_merge_passes_per_call	Ordina pass di unione per chiamata
db.sql_tokenized.stats.sum_sort_scan_per_call	Ordinare scansioni per chiamata
db.sql_tokenized.stats.sum_sort_range_per_call	Ordina intervalli per chiamata
db.sql_tokenized.stats.sum_sort_rows_per_call	Ordina righe per chiamata
db.sql_tokenized.stats.sum_rows_affected_per_call	Righe interessate per chiamata

Parametro	Unità
db.sql_tokenized.stats.sum_rows_examined_per_call	Righe esaminate per chiamata
db.sql_tokenized.stats.sum_rows_sent_per_call	Righe inviate per chiamata
db.sql_tokenized.stats.sum_created_tmp_disk_tables_per_call	Tabelle disco temporanee create per chiamata
db.sql_tokenized.stats.sum_created_tmp_tables_per_call	Tabelle temporanee create per chiamata
db.sql_tokenized.stats.sum_lock_time_per_call	Tempo di blocco per chiamata (in ms)

Statistiche principali per Aurora MySQL

Le seguenti statistiche SQL sono disponibili per cluster di database Aurora MySQL.

Parametro	Unità
db.sql_tokenized.stats.count_star	Chiamate
db.sql_tokenized.stats.sum_timer_wait	Tempo di attesa (in ms)
db.sql_tokenized.stats.sum_select_full_join	Seleziona full join
db.sql_tokenized.stats.sum_select_range_check	Seleziona i controlli dell’intervallo
db.sql_tokenized.stats.sum_select_scan	Seleziona scansioni
db.sql_tokenized.stats.sum_sort_merge_passes	Ordina i pass di unione
db.sql_tokenized.stats.sum_sort_scan	Ordina scansioni
db.sql_tokenized.stats.sum_sort_range	Ordina intervalli
db.sql_tokenized.stats.sum_sort_rows	Ordinare righe

Parametro	Unità
db.sql_tokenized.stats.sum_rows_affected	Righe interessate
db.sql_tokenized.stats.sum_rows_examined	Righe esaminate
db.sql_tokenized.stats.sum_rows_sent	Righe inviate
db.sql_tokenized.stats.sum_created_tmp_disk_tables	Tabelle disco temporanee create
db.sql_tokenized.stats.sum_created_tmp_tables	Tabelle temporanee create
db.sql_tokenized.stats.sum_lock_time	Tempo di blocco (in ms)

Statistiche SQL per Aurora PostgreSQL

Per ogni chiamata SQL e per ogni secondo di esecuzione di una query, Performance Insights raccoglie statistiche SQL. Tutti i motori Aurora raccolgono statistiche solo a livello di digest.

Di seguito sono disponibili informazioni sulle statistiche a livello di digest per Aurora PostgreSQL.

Argomenti

- Statistiche digest per Aurora PostgreSQL
- Statistiche digest al secondo per Aurora PostgreSQL
- Statistiche digest per chiamata per Aurora PostgreSQL
- Statistiche principali per Aurora PostgreSQL

Statistiche digest per Aurora PostgreSQL

Per visualizzare le statistiche digest SQL è necessario caricare la libreria `pg_stat_statements`. Per i cluster di database Aurora PostgreSQL che sono compatibili con PostgreSQL 10, questa libreria viene caricata per impostazione predefinita. Per i cluster di database Aurora PostgreSQL che sono compatibili con PostgreSQL 9.6, dovrai abilitare manualmente questa libreria. Per abilitarla manualmente, aggiungere `pg_stat_statements` a `shared_preload_libraries` nel gruppo parametri del database associati all'istanza database. Riavviare quindi l'istanza database. Per ulteriori informazioni, consulta Gruppi di parametri per Amazon Aurora.

> **Note**
>
> Performance Insights può raccogliere statistiche solo per le query non troncate in `pg_stat_activity`. Per impostazione predefinita, i database PostgreSQL troncano le query più lunghe di 1.024 byte. Per aumentare la dimensione della query, modificare il parametro `track_activity_query_size` nel gruppo di parametri database associato all'istanza database. Quando si modifica questo parametro, è necessario riavviare un'istanza database.

Statistiche digest al secondo per Aurora PostgreSQL

Le seguenti statistiche digest SQL sono disponibili per le istanze database Aurora PostgreSQL.

Parametro	Unità
db.sql_tokenized.stats.calls_per_sec	Chiamate al secondo
db.sql_tokenized.stats.rows_per_sec	Righe al secondo
db.sql_tokenized.stats.total_time_per_sec	Media delle esecuzioni attive al secondo (AAE, Average active executions)
db.sql_tokenized.stats.shared_blks_hit_per_sec	Richieste in blocco al secondo
db.sql_tokenized.stats.shared_blks_read_per_sec	Letture in blocco al secondo
db.sql_tokenized.stats.shared_blks_dirtied_per_sec	Blocchi sporchi al secondo
db.sql_tokenized.stats.shared_blks_written_per_sec	Scritture in blocco al secondo
db.sql_tokenized.stats.local_blks_hit_per_sec	Richieste in blocco locale al secondo
db.sql_tokenized.stats.local_blks_read_per_sec	Letture di blocchi locali al secondo
db.sql_tokenized.stats.local_blks_dirtied_per_sec	Blocco locale danneggiato al secondo

Parametro	Unità
db.sql_tokenized.stats.local_blks_written_per_sec	Scritture di blocchi locali al secondo
db.sql_tokenized.stats.temp_blks_written_per_sec	Scritture temporanee al secondo
db.sql_tokenized.stats.temp_blks_read_per_sec	Letture temporanee al secondo
db.sql_tokenized.stats.blk_read_time_per_sec	Letture medie simultanee al secondo
db.sql_tokenized.stats.blk_write_time_per_sec	Scritture medie simultanee al secondo

Statistiche digest per chiamata per Aurora PostgreSQL

I seguenti parametri forniscono le statistiche per chiamata di un'istruzione SQL.

Parametro	Unità
db.sql_tokenized.stats.rows_per_call	Righe per chiamata
db.sql_tokenized.stats.avg_latency_per_call	Latenza media per chiamata (in ms)
db.sql_tokenized.stats.shared_blks_hit_per_call	Richieste in blocco per chiamata
db.sql_tokenized.stats.shared_blks_read_per_call	Letture in blocco per chiamata
db.sql_tokenized.stats.shared_blks_written_per_call	Scritture in blocco per chiamata
db.sql_tokenized.stats.shared_blks_dirtied_per_call	Blocchi danneggiati per chiamata
db.sql_tokenized.stats.local_blks_hit_per_call	Richieste in blocco locale per chiamata
db.sql_tokenized.stats.local_blks_read_per_call	Letture di blocchi locali per chiamata

Parametro	Unità
db.sql_tokenized.stats.local_blks_dirtied_per_call	Blocco locale danneggiato per chiamata
db.sql_tokenized.stats.local_blks_written_per_call	Scritture di blocchi locali per chiamata
db.sql_tokenized.stats.temp_blks_written_per_call	Scritture temporanee di blocchi per chiamata
db.sql_tokenized.stats.temp_blks_read_per_call	Letture temporanee di blocchi per chiamata
db.sql_tokenized.stats.blk_read_time_per_call	Tempo di lettura per chiamata (in ms)
db.sql_tokenized.stats.blk_write_time_per_call	Tempo di scrittura per chiamata (in ms)

Statistiche principali per Aurora PostgreSQL

Le seguenti statistiche SQL sono disponibili per le istanze database Aurora PostgreSQL.

Parametro	Unità
db.sql_tokenized.stats.calls	Chiamate
db.sql_tokenized.stats.rows	Righe
db.sql_tokenized.stats.total_time	Tempo totale (in ms)
db.sql_tokenized.stats.shared_blks_hit	Richieste in blocco
db.sql_tokenized.stats.shared_blks_read	Blocchi letti
db.sql_tokenized.stats.shared_blks_dirtied	Blocchi sporchi
db.sql_tokenized.stats.shared_blks_written	Scritture di blocchi
db.sql_tokenized.stats.local_blks_hit	Hit in blocco locale
db.sql_tokenized.stats.local_blks_read	Letture di blocchi locali

Parametro	Unità
db.sql_tokenized.stats.local_blks_dirtied	Blocchi locali sporchi
db.sql_tokenized.stats.local_blks_written	Scritture di blocchi locali
db.sql_tokenized.stats.temp_blks_written	Scritture temporanee
db.sql_tokenized.stats.temp_blks_read	Letture temporanee
db.sql_tokenized.stats.blk_read_time	Letture simultanee medie (in ms)
db.sql_tokenized.stats.blk_write_time	Scritture simultanee medie (in ms)

Per ulteriori informazioni su questi parametri, consultare pg_stat_statements nella documentazione PostgreSQL.

Metriche del sistema operativo nel monitoraggio avanzato

Amazon Aurora fornisce parametri in tempo reale per il sistema operativo sul quale è in esecuzione il istanza database. Aurora fornisce i parametri di Enhanced Monitoring al tuo account Amazon CloudWatch Logs. Le tabelle seguenti elencano i parametri del sistema operativo disponibili utilizzando Amazon CloudWatch Logs.

Argomenti

- Parametri del sistema operativo per Aurora

Parametri del sistema operativo per Aurora

Gruppo	Parametro	Nome console	Descrizione
`General`	`engine`	Non applicabile	Il motore del database per l'istanza database.
	`instanceID`	Non applicabile	Identificatore istanze DB.

Gruppo	Parametro	Nome console	Descrizione
	`instanceResourceID`	Non applicabile	Un identificatore immutabile per l'istanza DB che è unico per una AWS regione, utilizzato anche come identificatore del flusso di log.
	`numVCPUs`	Non applicabile	Il numero di elementi virtuali CPUs per l'istanza DB.
	`timestamp`	Non applicabile	L'ora in cui sono stati presi i parametri.
	`uptime`	Non applicabile	Il periodo di esecuzione dell'istanza database è stato attivato.
	`version`	Non applicabile	La versione del formato JSON del flusso dei parametri del sistema operativo.
`cpuUtilization`	`guest`	Ospite CPU	La percentuale di CPU utilizzata dai programmi guest.
	`idle`	CPU inattiva	La percentuale di tempo CPU che è inattiva.
	`irq`	IRQ CPU	La percentuale di CPU utilizzata dalle interruzioni dei software.
	`nice`	CPU Nice	La percentuale di CPU utilizzata dai programmi in esecuzione con priorità più bassa.
	`steal`	Stealing della CPU	La percentuale di CPU utilizzata da altre macchine virtuali.
	`system`	Sistema CPU	La percentuale di CPU utilizzata dal kernel.
	`total`	Totale CPU	La percentuale totale del CPU utilizzata. Questo valore include il valore `nice`.

Gruppo	Parametro	Nome console	Descrizione
	`user`	Utente CPU	La percentuale di CPU utilizzata dai programmi utente.
	`wait`	Attesa CPU	La percentuale di CPU inutilizzata durante l'attesa dell' I/O accesso.
`diskIO`	`avgQueueLen`	Dimensione media coda	Il numero di richieste in attesa nella coda del I/O dispositivo.
	`avgReqSz`	Dimensione richiesta media	La dimensione della richiesta media, in kilobyte.
	`await`	Disco I/O in attesa	Il numero di millisecondi necessari per rispondere alle richieste, compreso il tempo della coda e il tempo del servizio.
	`device`	Non applicabile	L'identificatore del dispositivo del disco in uso.
	`readIOsPS`	IO/s di lettura	Il numero di operazioni di lettura al secondo.
	`readKb`	Totale lettura	Il numero totale di kilobyte letti.
	`readKbPS`	KB/s di lettura	Il numero di kilobytes letti al secondo.
	`readLatency`	Latenza di lettura	Il tempo trascorso tra l'invio di una I/O richiesta di lettura e il suo completamento, in millisecondi. Questo parametro è disponibile solo per Amazon Aurora.

Gruppo	Parametro	Nome console	Descrizione
	`readThroughput`	Throughput di lettura	La quantità di velocità effettiva di rete utilizzata dalle richieste al cluster di database, espressa in byte al secondo. Questo parametro è disponibile solo per Amazon Aurora.
	`rrqmPS`	Rqms	Il numero di richieste di lettura unite in coda al secondo.
	`tps`	TPS	Il numero di transazioni al secondo. I/O
	`util`	Disk I/O Util	La percentuale di tempo della CPU durante il quale sono state emesse le richieste.
	`writeIOsPS`	IO/s di scrittura	Il numero di operazioni di scrittura al secondo.
	`writeKb`	Totale scrittura	Il numero totale di kilobyte scritti.
	`writeKbPS`	KB/s di scrittura	Il numero di kilobytes scritti al secondo.
	`writeLatency`	Latenza di scrittura	Il tempo medio trascorso tra l'invio di una I/O richiesta di scrittura e il suo completamento, in millisecondi. Questo parametro è disponibile solo per Amazon Aurora.
	`writeThroughput`	Throughput di scrittura	La quantità di throughput di rete effettivo utilizzato dalle risposte del cluster di database, espressa in byte al secondo. Questo parametro è disponibile solo per Amazon Aurora.

Gruppo	Parametro	Nome console	Descrizione
	`wrqmPS`	Wrqms	Il numero di richieste di scrittura unite in coda al secondo.
`mountPoint`	Non applicabile	Il percorso del sistema di file.	
`fileSys`	`maxFiles`	Inode max	Il numero massimo di file che è possibile creare per il sistema di file.
	`mountPoint`	Non applicabile	Il percorso del sistema di file. In tal caso `/rdsdbdata*` , rappresenta l'aggregato di tutti i volumi di archiviazione.
	`name`	Non applicabile	Il nome del sistema di file.
	`total`	File system totale	Il numero totale di spazio su disco disponibile per il sistema di file, in kilobyte.
	`used`	Filesystem usato	La quantità totale di spazio su disco utilizzato dai file nel sistema di file, in kilobyte.
	`usedFilePercent`	Inode usati	La percentuale di file disponibili in uso.
	`usedFiles`	% di utilizzo	Il numero di file audio nel sistema di file.
	`usedPercent`	Filesystem usato	La percentuale dello spazio su disco del sistema di file in uso.
`loadAverageMinute`	`fifteen`	Carico medio 15 min	Il numero di processi che richiedono l'ora della CPU negli ultimi 15 minuti.

Gruppo	Parametro	Nome console	Descrizione
	`five`	Carico medio 5 min	Il numero di processi che richiedono l'ora della CPU negli ultimi 5 minuti.
	`one`	Carico medio 1 min	Il numero di processi che richiedono l'ora della CPU nell'ultimo minuto.
`memory`	`active`	Memoria attiva	La quantità di memoria assegnata, in kilobyte.
	`buffers`	Memoria bufferizzata	La quantità di memoria utilizzata per il buffering delle I/O richieste prima della scrittura sul dispositivo di storage, in kilobyte.
	`cached`	Memoria cache	La quantità di memoria utilizzata per la memorizzazione nella cache dell'I/O basato sul file system.
	`dirty`	Memoria sporca	La quantità di pagine di memoria nella RAM che sono state modificate ma non scritte nel relativo blocco di dati nello storage, in kilobyte.
	`free`	Memoria libera	La quantità di memoria non assegnata, in kilobyte.
	`hugePagesFree`	Huge Pages libere	Il numero di pagine di grandi dimensioni gratuite. Le pagine di grandi dimensioni sono una caratteristica del kernel di Linux.
	`hugePagesRsvd`	Huge Pages Rsvd	Il numero di pagine di grandi dimensioni impegnate.
	`hugePagesSize`	Dimensioni Huge Pages	La dimensione per ogni unità delle pagine di grandi dimensioni, in kilobyte.

Gruppo	Parametro	Nome console	Descrizione
	`hugePagesSurp`	Huge Pages Surp	Il numero di pagine di grandi dimensioni in eccesso disponibili sul totale.
	`hugePagesTotal`	Totale Huge Pages	Il numero totale di Huge Pages.
	`inactive`	Memoria inattiva	La quantità di pagine di memoria utilizzate meno frequentemente, in kilobyte.
	`mapped`	Memoria mappata	La quantità totale di contenuti del sistema di file che è mappata in memoria all'interno di uno spazio di indirizzamento del processo, in kilobyte.
	`pageTables`	Tabelle delle pagine	La quantità di memoria utilizzata dalle tabelle della pagina, in kilobyte.
	`slab`	Memoria slab	La quantità di strutture dati del kernel riutilizzabili, in kilobyte.
	`total`	Memoria totale	La quantità totale di memoria, in kilobyte.
	`writeback`	Memoria writeback	La quantità di pagine sporche nella RAM che sono ancora scritte nello storage di backup, in kilobyte.
`network`	`interface`	Non applicabile	L'identificatore per l'interfaccia di rete utilizzato per l'istanza database.
	`rx`	RX	Il numero di bytes ricevuti al secondo.
	`tx`	TX	Il numero di bytes caricati al secondo.

Gruppo	Parametro	Nome console	Descrizione
processList	cpuUsedPc	CPU %	La percentuale di CPU utilizzata dal processo.
	id	Non applicabile	L'identificatore del processo.
	memoryUsedPc	MEM%	Percentuale della memoria totale utilizzata dal processo.
	name	Non applicabile	Il nome del processo.
	parentID	Non applicabile	L'identificatore del processo per il processo genitore del processo.
	rss	RES	La quantità di RAM allocata al processo, in kilobyte.
	tgid	Non applicabile	L'identificatore del gruppo di thread, che è un numero che rappresenta l'ID del processo a cui appartiene un thread. Questo identificatore è utilizzato per raggruppare i thread dallo stesso processo.
	vss	VIRT	La quantità di memoria virtuale allocata al processo, in kilobyte.
swap	total	Swap	La quantità di memoria di scambio disponibile, in kilobyte.
	in	Swap in	Quantità di memoria, in kilobyte, scambiata in ingresso nel disco.
	out	Swap out	Quantità di memoria, in kilobyte, scambiata in uscita dal disco.
	free	Swap libera	La quantità di memoria di scambio libera, in kilobyte.

Gruppo	Parametro	Nome console	Descrizione
	`cached`	Swap occupata	La quantità di memoria di scambio, in kilobyte, utilizzata come memoria cache.
`tasks`	`blocked`	Attività bloccate	Il numero di attività che sono bloccate.
	`running`	Attività in esecuzione	Il numero di attività che sono in esecuzione.
	`sleeping`	Attività inattive	Il numero di attività che sono a riposo.
	`stopped`	Attività interrotte	Il numero di attività che sono arrestate.
	`total`	Totale attività	Il numero totale di attività.
	`zombie`	Attività Zombie	Il numero di attività secondarie che sono inattive con un'attività genitore attiva.

Monitoraggio di eventi, registri e flussi in un cluster di database Amazon Aurora

L'obiettivo del monitoraggio dei database Amazon Aurora e delle altre soluzioni AWS è mantenere quanto segue:

- Affidabilità
- Disponibilità
- Prestazioni
- Sicurezza

Monitoraggio dei parametri in un cluster di database Amazon Aurora spiega come monitorare il cluster usando i parametri. Una soluzione completa deve inoltre monitorare eventi, registri r flussi di attività del database. AWS fornisce i seguenti strumenti di monitoraggio:

- Amazon EventBridge è un servizio bus di eventi serverless che semplifica la connessione delle applicazioni ai dati provenienti da un'ampia gamma di origini. EventBridge fornisce un flusso di dati in tempo reale dalle tue applicazioni, dalle applicazioni Software-as-a-Service (SaaS) e dai servizi AWS. EventBridge instrada tali dati a destinazioni come AWS Lambda. In questo modo, puoi monitorare gli eventi che si verificano nei servizi e creare architetture basate su eventi. Per ulteriori informazioni, consulta la Guida per l'utente di Amazon EventBridge.
- Amazon CloudWatch Logs fornisce un modo per monitorare, archiviare e accedere ai file di registro dalle istanze Amazon Aurora, da AWS CloudTrail e da altre origini. Amazon CloudWatch Logs è in grado di monitorare le informazioni nei file di log e notificare quando vengono raggiunte determinate soglie. Puoi inoltre archiviare i dati del log in storage estremamente durevole. Per ulteriori informazioni, consulta la Guida per l'utente di Amazon CloudWatch Logs.
- AWS CloudTrail acquisisce chiamate API ed eventi correlati da parte di o per conto del tuo Account AWS. CloudTrail distribuisce i file di registro a un bucket Amazon S3 specificato. Puoi identificare quali utenti e account hanno richiamato AWS, l'indirizzo IP di origine da cui sono state effettuate le chiamate e quando sono avvenute. Per ulteriori informazioni, consultare la Guida per l'utente AWS CloudTrail.
- I flussi di attività di database sono una funzionalità di Amazon Aurora che fornisce un flusso quasi in tempo reale dell'attività nel cluster di database. Amazon Aurora inserisce le attività in Amazon Kinesis Data Streams. Il flusso Kinesis viene creato automaticamente. Da Kinesis, è possibile

configurare servizi AWS quali Amazon Data Firehose e AWS Lambda per consumare il flusso e archiviare i dati.

Argomenti

- Visualizzazione di registri, eventi e flussi nella console Amazon RDS
- Monitoraggio di eventi Amazon Aurora
- Monitoraggio dei file di log di Amazon Aurora
- Monitoraggio delle chamate API di Amazon Aurora in AWS CloudTrail
- Monitoraggio di Amazon Aurora tramite i flussi di attività del database
- Monitoraggio delle minacce con la protezione RDS Amazon GuardDuty per Amazon Aurora

Visualizzazione di registri, eventi e flussi nella console Amazon RDS

Amazon RDS si integra con Servizi AWS per visualizzare informazioni su registri, eventi e flussi di attività del database nella console RDS.

La scheda Logs & events (Registri ed eventi) per il cluster di database Aurora mostra le informazioni seguenti:

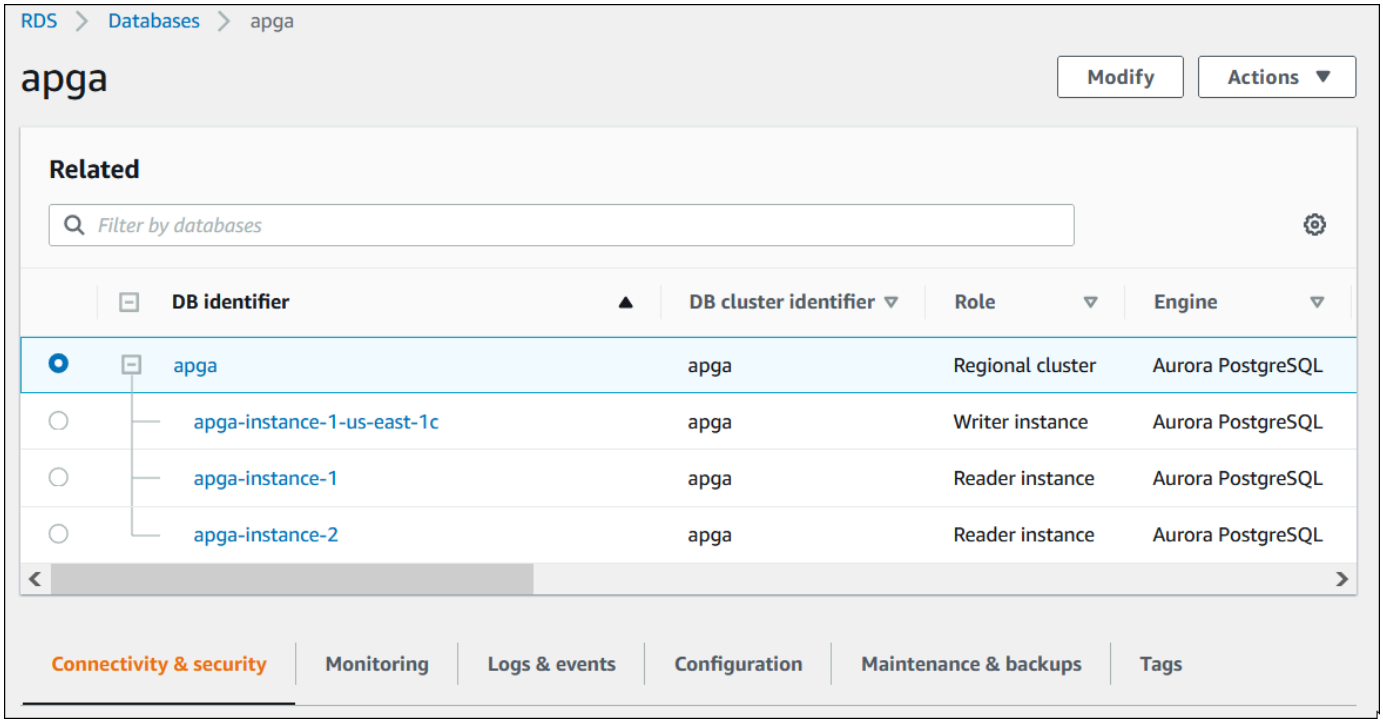
- Auto scaling policies and activities (Policy e attività di scalabilità automatica): mostra le policy e le attività relative alla funzione di scalabilità automatica di Aurora. Queste informazioni vengono visualizzate solo nella scheda Logs & events (Registri ed eventi) a livello di cluster.
- Amazon CloudWatch alarms (Allarmi di Amazon CloudWatch): mostra eventuali allarmi dei parametri configurati per l'istanza database nel cluster Aurora. Se non hai configurato allarmi, puoi crearli nella console di RDS.
- Recent events (Eventi recenti): mostra un riepilogo degli eventi (modifiche all'ambiente) per l'istanza o cluster di database Aurora. Per ulteriori informazioni, consulta Visualizzazione di eventi Amazon RDS.
- Log (Registro): mostra i file di log del database generati da un'istanza database nel cluster Aurora. Per ulteriori informazioni, consulta Monitoraggio dei file di log di Amazon Aurora.

La scheda Configuration (Configurazione) mostra le informazioni sui flussi di attività di database.

Per visualizzare registri, eventi e flussi per il cluster di database Aurora nella console RDS

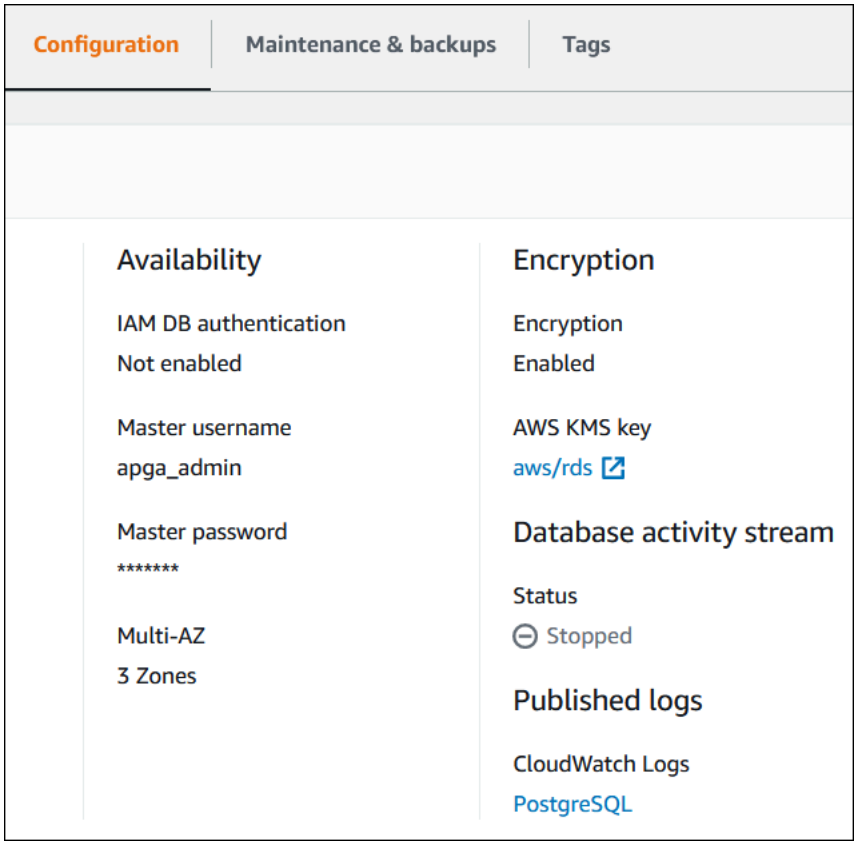
1. Accedi alla Console di gestione AWS e apri la console Amazon RDS all'indirizzo https://console.aws.amazon.com/rds/.
2. Nel pannello di navigazione, scegliere Databases (Database).
3. Scegliere il nome del cluster di Aurora che si desidera monitorare.

   Verrà visualizzata la pagina Databases (Database). L'esempio seguente mostra un cluster di database Amazon Aurora PostgreSQL denominato `apga`.



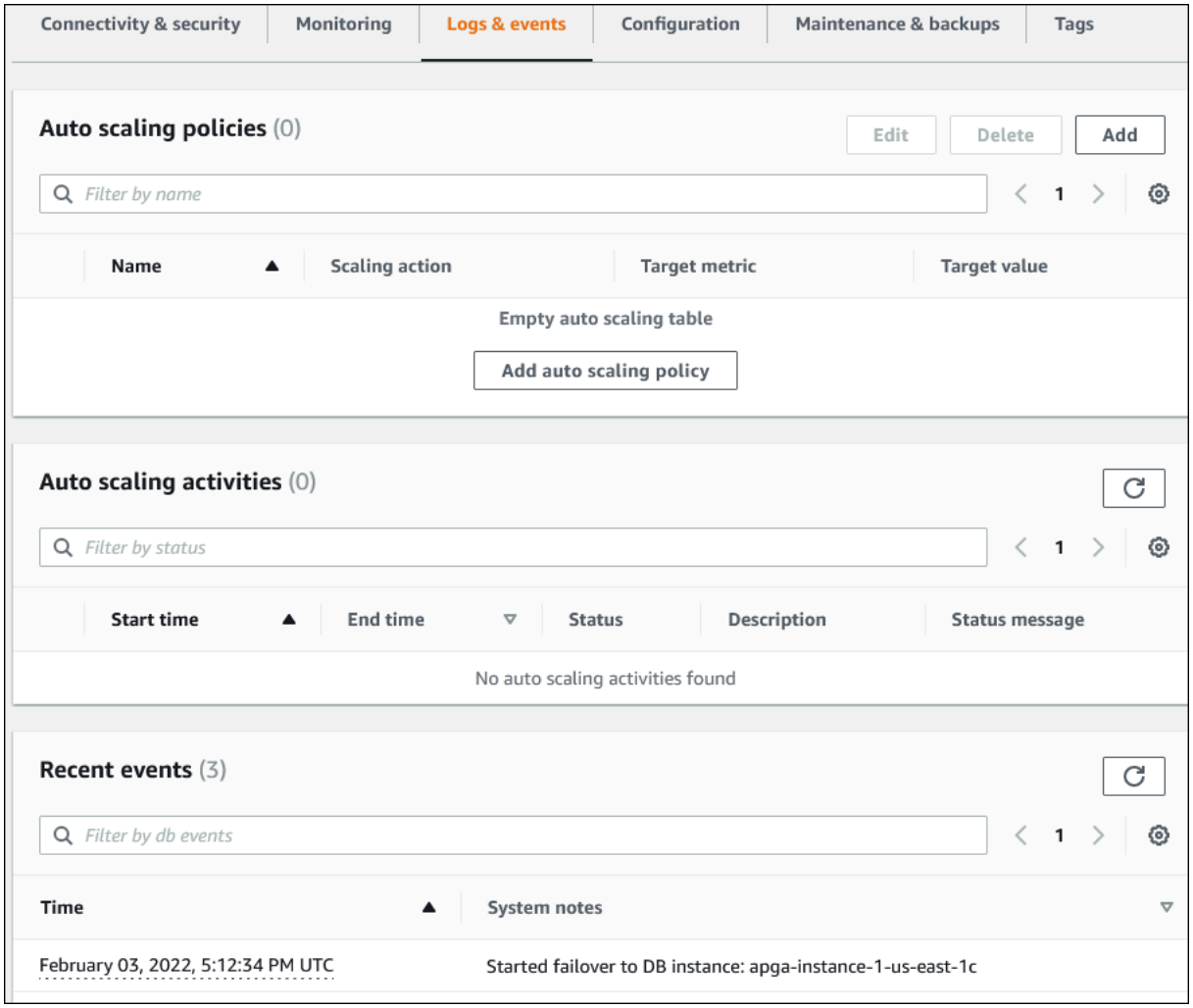
4. Scorrere verso il basso e selezionare Configuration (Configurazione).

   L'esempio seguente mostra lo stato dei flussi di attività di database per il cluster.



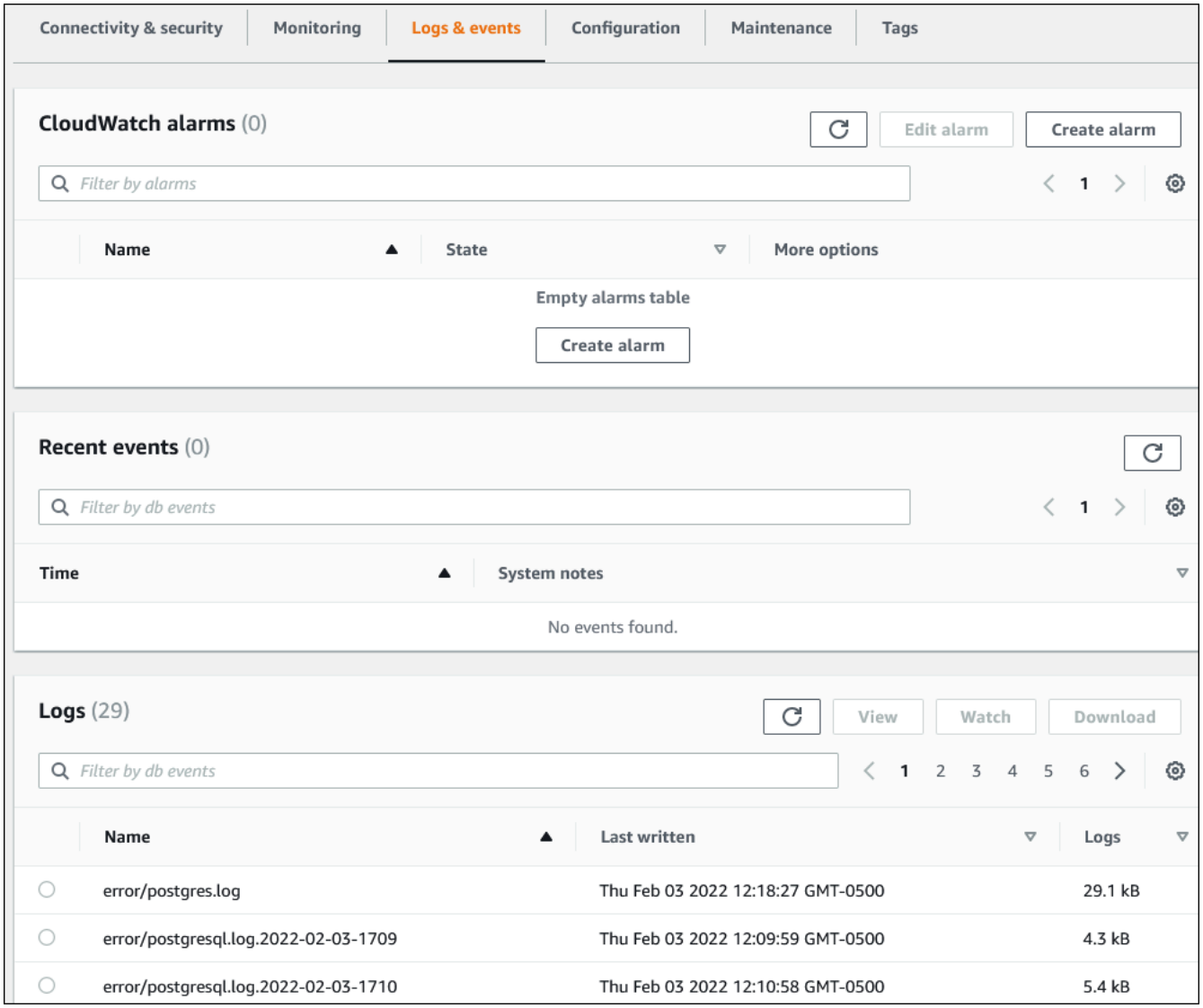
5. Scegliere Logs & events (Log ed eventi).

   Viene visualizzata la sezione Logs & events (Log ed eventi).



6. Scegliere un'istanza database nel cluster Aurora e quindi scegliere Logs & events (Registri ed eventi) per l'istanza.

   L'esempio seguente mostra che il contenuto è diverso tra la pagina dell'istanza database e la pagina del cluster di database. La pagina dell'istanza database mostra i registri e gli allarmi.


Connectivity & security
Monitoring
Logs & events
Configuration
Maintenance
Tags
CloudWatch alarms (0)
Edit alarm
Create alarm
Filter by alarms
1
Name
State
More options
Empty alarms table
Create alarm
Recent events (0)
Filter by db events
1
Time
System notes
No events found.
Logs (29)
View
Watch
Download
Filter by db events
1 2 3 4 5 6
Name
Last written
Logs
error/postgres.log
Thu Feb 03 2022 12:18:27 GMT-0500
29.1 kB
error/postgresql.log.2022-02-03-1709
Thu Feb 03 2022 12:09:59 GMT-0500
4.3 kB
error/postgresql.log.2022-02-03-1710
Thu Feb 03 2022 12:10:58 GMT-0500
5.4 kB

Monitoraggio di eventi Amazon Aurora

Un evento indica una modifica in un ambiente. Questo può essere un ambiente AWS, un'applicazione o un servizio partner SaaS o uno dei servizi o applicazioni personalizzati. Per le descrizioni degli eventi Aurora, consulta Categorie di eventi e messaggi di evento di Amazon RDS per Aurora.

Argomenti

- Panoramica degli eventi per Aurora
- Visualizzazione di eventi Amazon RDS
- Utilizzo della notifica degli eventi di Amazon RDS
- Creazione di una regola che si attiva su un evento Amazon Aurora
- Categorie di eventi e messaggi di evento di Amazon RDS per Aurora

Panoramica degli eventi per Aurora

Un evento RDS indica una modifica nell'ambiente Aurora. Ad esempio, di Amazon RDS genera un evento quando un cluster di database è riparato. Amazon Aurora distribuisce gli eventi a EventBridge quasi in tempo reale.

> **Note**
>
> Amazon RDS emette eventi sulla base del massimo sforzo. Si consiglia di non scrivere programmi che dipendono dall'ordine o dall'esistenza di eventi di notifica, poiché potrebbero essere fuori sequenza o mancanti.

Amazon RDS registra gli eventi correlati alle seguenti risorse:

- Cluster database

  Per un elenco degli eventi di un cluster, consulta Eventi di cluster di database.
- Istanze DB

  Per un elenco degli eventi dell'istanza database, consulta Eventi di istanza database.
- Gruppi di parametri database

Per un elenco degli eventi del gruppo parametri del database, consulta Eventi gruppo di parametri database.

- Gruppi di sicurezza DB

  Per un elenco di eventi del gruppo di sicurezza DB, consulta Eventi gruppo di sicurezza DB.

- Snapshot cluster DB

  Per un elenco di eventi degli snapshot del cluster database, consulta Eventi snapshot cluster di database.

- Eventi RDS Proxy

  Per un elenco degli eventi RDS Proxy, consulta Eventi RDS Proxy.

- Eventi di implementazione blu/verde

  Per l'elenco degli eventi di implementazione blu/verde, consulta Eventi di implementazione blu/verde.

Queste informazioni comprendono:

- La data e l'ora dell'evento
- Il nome di origine e il tipo di origine dell'evento
- Un messaggio associato all'evento
- Le notifiche degli eventi includono i tag che fanno riferimento al momento in cui il messaggio è stato inviato e potrebbero non riflettere i tag riferiti al momento in cui si è verificato l'evento.

Visualizzazione di eventi Amazon RDS

Puoi recuperare le seguenti informazioni sull'evento per le risorse Amazon Aurora:

- Nome risorsa
- Tipo di risorsa
- Ora dell'evento
- Riepilogo del messaggio dell'evento

È possibile accedere agli eventi nelle seguenti parti di Console di gestione AWS:

- La scheda Eventi, che mostra gli eventi delle ultime 24 ore.
- La tabella Eventi recenti nella sezione Registri ed eventi della scheda Database, che può mostrare gli eventi relativi alle ultime 2 settimane.

Puoi anche recuperare gli eventi usando il comando AWS CLI describe-events o l'operazione API RDS DescribeEvents. Se utilizzi la AWS CLI o l'API RDS per visualizzare gli eventi, puoi recuperare gli eventi fino agli ultimi 14 giorni.

> **Note**
>
> Se occorre archiviare eventi per periodi di tempo più lunghi, puoi inviare eventi Amazon RDS a EventBridge. Per ulteriori informazioni, consulta Creazione di una regola che si attiva su un evento Amazon Aurora

Per le descrizioni degli eventi Amazon Aurora, vedi Categorie di eventi e messaggi di evento di Amazon RDS per Aurora.

Per accedere a informazioni dettagliate relative agli eventi utilizzando AWS CloudTrail, inclusi i parametri della richiesta, consulta Eventi CloudTrail.

Console

Per visualizzare tutti gli eventi dell'istanza Amazon RDS delle ultime 24 ore

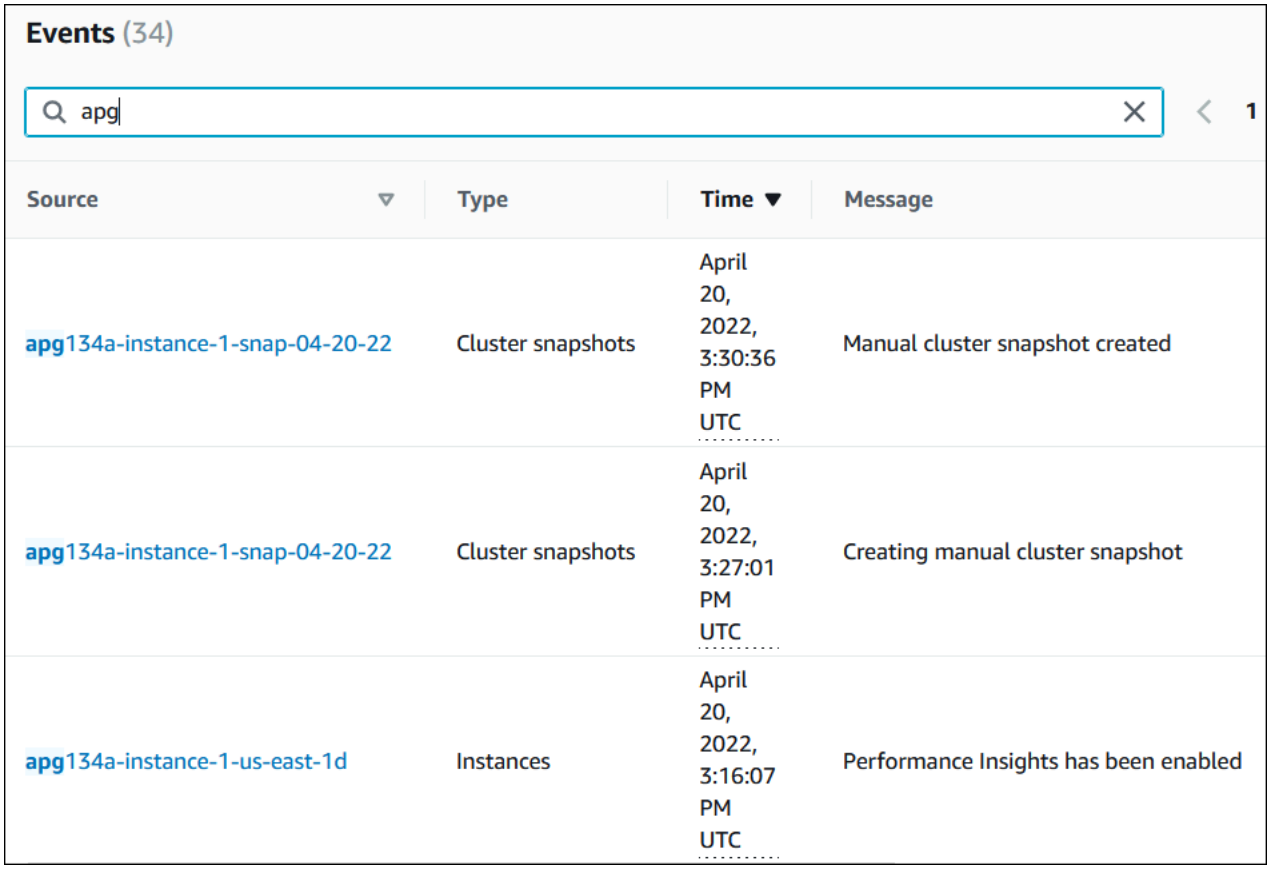
1. Accedi alla Console di gestione AWS e apri la console Amazon RDS all'indirizzo https://console.aws.amazon.com/rds/.

2. Nel pannello di navigazione selezionare Events (Eventi).

   Gli eventi disponibili sono indicati all'interno di un elenco.

3. (Opzionale) Inserisci un termine di ricerca per filtrare i risultati.

   Il seguente esempio mostra un elenco di eventi filtrati mediante i caratteri **apg**.



AWS CLI

Per visualizzare tutti gli eventi generati nell'ultima ora, invoca describe-events senza parametri.

```
aws rds describe-events
```

Il seguente output di esempio mostra che un'istanza del cluster database ha avviato il ripristino.

```
{
    "Events": [
        {
            "EventCategories": [
                "recovery"
```

```
            ],
            "SourceType": "db-instance",
            "SourceArn": "arn:aws:rds:us-east-1:123456789012:db:mycluster-instance-1",
            "Date": "2022-04-20T15:02:38.416Z",
            "Message": "Recovery of the DB instance has started. Recovery time will
 vary with the amount of data to be recovered.",
            "SourceIdentifier": "mycluster-instance-1"
        }, ...
```

Per visualizzare tutti gli eventi di Amazon RDS per gli ultimi 10.080 minuti (7 giorni) invoca il comando describe-events della AWS CLI e imposta il parametro `--duration` su `10080`.

```
aws rds describe-events --duration 10080
```

L'esempio seguente mostra gli eventi nell'intervallo di tempo specificato per l'istanza database *`test-instance`*.

```
aws rds describe-events \
    --source-identifier test-instance \
    --source-type db-instance \
    --start-time 2022-03-13T22:00Z \
    --end-time 2022-03-13T23:59Z
```

L'output di esempio seguente mostra lo stato di un backup.

```
{
    "Events": [
        {
            "SourceType": "db-instance",
            "SourceIdentifier": "test-instance",
            "EventCategories": [
                "backup"
            ],
            "Message": "Backing up DB instance",
            "Date": "2022-03-13T23:09:23.983Z",
            "SourceArn": "arn:aws:rds:us-east-1:123456789012:db:test-instance"
        },
        {
            "SourceType": "db-instance",
            "SourceIdentifier": "test-instance",
            "EventCategories": [
                "backup"
```

```
            ],
            "Message": "Finished DB Instance backup",
            "Date": "2022-03-13T23:15:13.049Z",
            "SourceArn": "arn:aws:rds:us-east-1:123456789012:db:test-instance"
        }
    ]
}
```

API

Puoi visualizzare tutti gli eventi dell'istanza Amazon RDS per gli ultimi 14 giorni chiamando l'operazione API RDS DescribeEvents e impostando il parametro `Duration` su `20160`.

Utilizzo della notifica degli eventi di Amazon RDS

Amazon RDS usa Amazon Simple Notification Service (Amazon SNS) per fornire una notifica quando si verifica un evento Amazon RDS. Queste notifiche possono essere in qualsiasi forma supportata da Amazon SNS per una regione AWS, ad esempio un'e-mail, un SMS o una chiamata a un endpoint HTTP.

Argomenti

- Panoramica delle notifiche eventi di Amazon RDS
- Concessione di autorizzazioni per pubblicare le notifiche in un argomento Amazon SNS
- Sottoscrizione alle notifiche eventi di Amazon RDS
- Tag e attributi delle notifiche eventi di Amazon RDS
- Elenco delle sottoscrizioni delle notifiche degli eventi Amazon RDS
- Modifica di una sottoscrizione alle notifiche eventi Amazon RDS
- Aggiunta di un identificatore di origine a una sottoscrizione alle notifiche eventi Amazon RDS
- Rimozione di un identificatore di origine da una sottoscrizione alle notifiche eventi Amazon RDS
- Creazione di un elenco delle categorie di notifiche eventi Amazon RDS
- Eliminazione di una sottoscrizione alle notifiche eventi Amazon RDS

Panoramica delle notifiche eventi di Amazon RDS

Amazon RDS raggruppa gli eventi in categorie che puoi sottoscrivere, per ricevere una notifica quando si verifica un evento di tale categoria.

Argomenti

- Risorse RDS idonee per la sottoscrizione di eventi
- Per sottoscrivere una notifica eventi di Amazon RDS, procedi come indicato di seguito:
- Consegna delle notifiche degli eventi RDS
- Fatturazione per le notifiche eventi Amazon RDS
- Esempi di eventi Aurora utilizzando Amazon EventBridge

Risorse RDS idonee per la sottoscrizione di eventi

Per Amazon Aurora, gli eventi si verificano sia a livello di cluster di database che di istanza database. È possibile sottoscrivere una categoria di eventi per le seguenti risorse:

- Istanza database
- Cluster DB
- Snapshot cluster di database
- DB parameter group (Gruppo di parametri database)
- Gruppo di sicurezza DB
- Server proxy per RDS
- Versioni personalizzate del motore

Ad esempio, sottoscrivendo la categoria Backup per una determinata istanza database, riceverai una notifica ogni volta che si verifica un evento relativo al backup che interessa l'istanza database. Sottoscrivendo una categoria Modifica della configurazione per un'istanza database, riceverai una notifica quando l'istanza database viene modificata. Riceverai una notifica anche quando viene modificata la sottoscrizione a una notifica eventi.

È possibile creare alcuni sottoscrizioni diverse. Per esempio, è possibile creare una sottoscrizione che riceve tutte le notifiche eventi per tutte le istanze database e un'altra sottoscrizione che include solo eventi critici per un sottoinsieme di istanze database. Per la seconda sottoscrizione, specifica una o più istanze database nel filtro.

Per sottoscrivere una notifica eventi di Amazon RDS, procedi come indicato di seguito:

Per sottoscrivere una notifica eventi Amazon RDS, procedi come indicato di seguito:

1. Puoi creare una sottoscrizione delle notifiche degli eventi Amazon RDS tramite la console Amazon RDS, AWS CLI o l'API.

   Amazon RDS usa l'ARN di un argomento Amazon SNS per identificare ogni sottoscrizione. La console Amazon RDS crea automaticamente l'ARN quando crei la sottoscrizione. Creare l'ARN utilizzando la console Amazon SNS, la AWS CLI o l'API Amazon SNS.
2. Amazon RDS invia un SMS o un'e-mail di approvazione all'indirizzo da te specificato nella sottoscrizione.
3. Puoi confermare la sottoscrizione selezionando il collegamento nella notifica ricevuta.

4. La console di Amazon RDS aggiorna la sezione My Event Subscriptions con lo stato della sottoscrizione.
5. Amazon RDS inizia inviando le notifiche agli indirizzi forniti al momento della creazione della sottoscrizione.

Per ulteriori informazioni su Identity and Access Management quando utilizzi Amazon SNS, consultaIdentity and Access Management in Amazon SNS nella Guida per sviluppatori di Amazon Simple Notification.

Puoi utilizzare AWS Lambda per elaborare le notifiche di eventi da un'istanza database. Per ulteriori informazioni, consulta Utilizzo di AWS Lambda con Amazon RDS Guida per gli sviluppatori di AWS Lambda.

Consegna delle notifiche degli eventi RDS

Amazon RDS invia le notifiche all'indirizzo fornito al momento della creazione della sottoscrizione. La notifica può inserire gli attributi del messaggio che forniscono i metadati strutturati sul messaggio. Per ulteriori informazioni sugli attributi del messaggio, consulta Categorie di eventi e messaggi di evento di Amazon RDS per Aurora.

La distribuzione delle notifiche degli eventi può richiedere fino a cinque minuti.

> **Important**
>
> Amazon RDS non garantisce l'ordine degli eventi inviato in un flusso di eventi. Tale ordine è soggetto a modifiche.

Quando Amazon SNS invia una notifica a un endpoint HTTP o HTTPS sottoscritto, il corpo del messaggio POST inviato all'endpoint contiene un documento JSON. Per ulteriori informazioni, consulta Formati di messaggio e JSON Amazon SNS nella Guida per gli sviluppatori di Amazon Simple Notification Service.

È possibile configurare SNS in modo che le notifiche vengano inviate tramite messaggi di testo. Per ulteriori informazioni, consulta Messaggistica SMS nella Guida per sviluppatori di Amazon Simple Notification Service.

Per disattivare le notifiche senza eliminare una sottoscrizione, scegliere No per Enabled (Abilitato) nella console Amazon RDS. In alternativa, puoi impostare il parametro `Enabled` a `false` usando la AWS CLI o API Amazon RDS.

Fatturazione per le notifiche eventi Amazon RDS

Fatturazione per la notifica degli eventi Amazon RDS avviene tramite Amazon SNS. L'uso della notifica degli eventi è soggetta alle tariffe di Amazon SNS. Per ulteriori informazioni sulla fatturazione di Amazon SNS, consulta prezzi di Amazon Simple Notification Service.

Esempi di eventi Aurora utilizzando Amazon EventBridge

Negli esempi seguenti vengono illustrati diversi tipi di eventi Aurora in formato JSON. Per un'esercitazione che illustra come acquisire e visualizzare eventi in formato JSON, consultare Tutorial: Registrazione delle modifiche dello stato di un'istanza database tramite Amazon EventBridge.

Argomenti

- Esempio di evento cluster di database
- Esempio di evento del gruppo parametri del database
- Esempio di evento snapshot cluster di database

Esempio di evento cluster di database

Di seguito è riportato un esempio di evento del cluster di database in formato JSON. L'evento mostra che al cluster denominato `my-db-cluster` è stata applicata la patch. L'ID evento è `RDS-EVENT-0173`.

```
{
  "version": "0",
  "id": "844e2571-85d4-695f-b930-0153b71dcb42",
  "detail-type": "RDS DB Cluster Event",
  "source": "aws.rds",
  "account": "123456789012",
  "time": "2018-10-06T12:26:13Z",
  "region": "us-east-1",
  "resources": [
    "arn:aws:rds:us-east-1:123456789012:cluster:my-db-cluster"
  ],
```

```
  "detail": {
    "EventCategories": [
      "notification"
    ],
    "SourceType": "CLUSTER",
    "SourceArn": "arn:aws:rds:us-east-1:123456789012:cluster:my-db-cluster",
    "Date": "2018-10-06T12:26:13.882Z",
    "Message": "Database cluster has been patched",
    "SourceIdentifier": "my-db-cluster",
    "EventID": "RDS-EVENT-0173"
  }
}
```

Esempio di evento del gruppo parametri del database

Di seguito è riportato un esempio di un evento gruppo parametri del database in formato JSON. L'evento mostra che il parametro `time_zone` è stato aggiornato nel gruppo di parametri `my-db-param-group`. L'ID evento è RDS-EVENT-0037.

```
{
  "version": "0",
  "id": "844e2571-85d4-695f-b930-0153b71dcb42",
  "detail-type": "RDS DB Parameter Group Event",
  "source": "aws.rds",
  "account": "123456789012",
  "time": "2018-10-06T12:26:13Z",
  "region": "us-east-1",
  "resources": [
    "arn:aws:rds:us-east-1:123456789012:pg:my-db-param-group"
  ],
  "detail": {
    "EventCategories": [
      "configuration change"
    ],
    "SourceType": "DB_PARAM",
    "SourceArn": "arn:aws:rds:us-east-1:123456789012:pg:my-db-param-group",
    "Date": "2018-10-06T12:26:13.882Z",
    "Message": "Updated parameter time_zone to UTC with apply method immediate",
    "SourceIdentifier": "my-db-param-group",
    "EventID": "RDS-EVENT-0037"
  }
}
```

Esempio di evento snapshot cluster di database

Di seguito è riportato un esempio di un evento snapshot del cluster di database in formato JSON. L'evento mostra la creazione dello snapshot denominato `my-db-cluster-snapshot`. L'ID evento è RDS-EVENT-0074.

```
{
  "version": "0",
  "id": "844e2571-85d4-695f-b930-0153b71dcb42",
  "detail-type": "RDS DB Cluster Snapshot Event",
  "source": "aws.rds",
  "account": "123456789012",
  "time": "2018-10-06T12:26:13Z",
  "region": "us-east-1",
  "resources": [
    "arn:aws:rds:us-east-1:123456789012:cluster-snapshot:rds:my-db-cluster-snapshot"
  ],
  "detail": {
    "EventCategories": [
      "backup"
    ],
    "SourceType": "CLUSTER_SNAPSHOT",
    "SourceArn": "arn:aws:rds:us-east-1:123456789012:cluster-snapshot:rds:my-db-
cluster-snapshot",
    "Date": "2018-10-06T12:26:13.882Z",
    "SourceIdentifier": "my-db-cluster-snapshot",
    "Message": "Creating manual cluster snapshot",
    "EventID": "RDS-EVENT-0074"
  }
}
```

Concessione di autorizzazioni per pubblicare le notifiche in un argomento Amazon SNS

Per concedere ad Amazon RDS le autorizzazioni per pubblicare notifiche su un argomento di Amazon Simple Notification Service (Amazon SNS), allega AWS Identity and Access Management una policy (IAM) all'argomento di destinazione. Per maggiori informazioni sulle autorizzazioni, consulta Esempi di casi per il controllo degli accessi Amazon SNS nella Guida per gli sviluppatori di Amazon Simple Notification Service.

Per impostazione predefinita, un argomento Amazon SNS dispone di una policy che consente a tutte le risorse Amazon RDS nello stesso account di pubblicare notifiche nello stesso. Puoi collegare una policy personalizzata per consentire notifiche tra account o per limitare l'accesso a determinate risorse.

Di seguito è riportato un esempio di policy IAM collegata all'argomento Amazon SNS di destinazione. Limita l'argomento alle istanze database con nomi corrispondenti al prefisso specificato. Per utilizzare questa policy, specifica i seguenti valori:

- `Resource` – Il nome della risorsa Amazon (ARN) per l'argomento Amazon SNS
- `SourceARN` – L'ARN della risorsa RDS
- `SourceAccount`— Il tuo ID Account AWS

Per visualizzare un elenco dei tipi di risorse e relativi ARNs, consulta Resources Defined by Amazon RDS nel Service Authorization Reference.

JSON

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": "events.rds.amazonaws.com"
      },
      "Action": [
        "sns:Publish"
      ],
      "Resource": "arn:aws:sns:us-east-1:123456789012:topic_name",
```

```
      "Condition": {
        "ArnLike": {
          "aws:SourceArn": "arn:aws:rds:us-east-1:123456789012:db:prefix-*"
        },
        "StringEquals": {
          "aws:SourceAccount": "123456789012"
        }
      }
    }
  ]
}
```

Sottoscrizione alle notifiche eventi di Amazon RDS

Il modo più semplice per creare una sottoscrizione è utilizzare la console RDS. Se scegli di creare sottoscrizioni delle notifiche degli eventi tramite la CLI o l'API, devi creare un argomento Amazon Simple Notification Service e sottoscrivere l'argomento con la console Amazon SNS o l'API di Amazon SNS. Dovrai inoltre annotare l'Amazon Resource Name (ARN) dell'argomento, in quanto viene utilizzato quando si inviano comandi CLI o operazioni API. Per informazioni sulla creazione di un argomento SNS e sull'abbonamento allo stesso, consulta Nozioni di base su Amazon SNS nella Guida per sviluppatori di Amazon Simple Notification Service.

Puoi specificare il tipo di origine per cui vuoi ricevere le notifiche e l'origine Amazon RDS che attiva l'evento:

Source type (Tipo di origine)

Il tipo di origine Ad esempio: Source Type (Tipo di origine) potrebbe essere Instances (Istanze). Devi scegliere un tipo di origine.

Risorse da includere

La risorsa Amazon RDS che genera gli eventi. Ad esempio, puoi scegliere Select specific instances (Seleziona istanze specifiche) e quindi myDBInstance1.

Nella tabella seguente viene illustrato il risultato quando si specificano o non si specificano ***Risorse*** da includere.

Risorse da includere	Descrizione	Esempio
Specificato	RDS invia notifiche relative a tutti gli eventi solo per la risorsa specificata.	Se Source type (Tipo di origine) è Instances (Istanze) e la risorsa è myDBInstance1, RDS invia notifiche relative a tutti gli eventi solo per myDBInstance1.
Non specificato	RDS invia notifiche relative agli eventi relativi al tipo di origine specificato per tutte le risorse Amazon RDS.	Se Source type (Tipo di origine) è Instances (Istanze), RDS invia notifiche relative a tutti gli eventi correlati alle istanze nell'account.

L'abbonato di un argomento Amazon SNS riceve per impostazione predefinita ogni messaggio pubblicato nell'argomento. Per ricevere solo un sottoinsieme dei messaggi, l'abbonato deve assegnare una policy di filtro all'abbonamento all'argomento. Per ulteriori informazioni, consulta Filtraggio messaggi di Amazon SNS nella Guida per gli sviluppatori di Amazon Simple Notification Service.

Console

Per sottoscrivere una notifica eventi RDS

1. Accedi alla Console di gestione AWS e apri la console Amazon RDS all'indirizzo https://console.aws.amazon.com/rds/.
2. Nel pannello di navigazione, scegliere Event subscriptions (Sottoscrizioni a eventi).
3. Nel riquadro Event subscriptions (Sottoscrizioni di eventi) scegliere Create event subscription (Crea sottoscrizione di eventi).
4. Inserisci i dettagli dell'abbonamento come segue:

   a. Per Name (Nome), immettere un nome per la sottoscrizione alle notifiche eventi.

   b. Nel campo Send notifications to (Invia notifica a), esegui una delle seguenti operazioni:

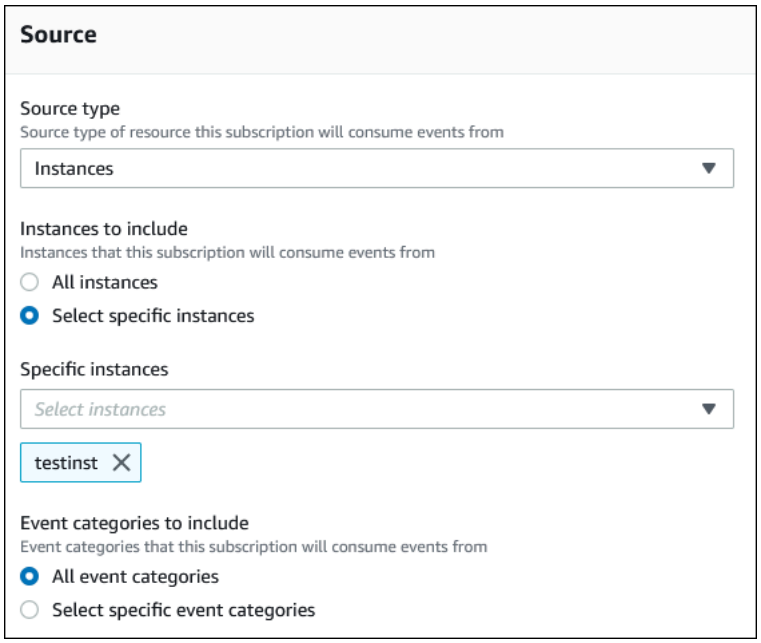
      - Scegli New email topic (Nuovo argomento e-mail). Inserisci un nome per l'argomento dell'email e un elenco di destinatari. Ti consigliamo di configurare le sottoscrizioni agli eventi con lo stesso indirizzo e-mail del contatto dell'account principale. I suggerimenti, gli eventi di assistenza e i messaggi personali vengono inviati utilizzando canali diversi. Le sottoscrizioni con lo stesso indirizzo e-mail assicurano che tutti i messaggi siano consolidati in un'unica posizione.
      - Scegli Amazon Resource Name (ARN) (Nome della risorsa Amazon (ARN)). Quindi scegli l'ARN Amazon SNS esistente per un argomento Amazon SNS.

        Se desideri utilizzare un argomento abilitato per la crittografia lato server (SSE), concedi ad Amazon RDS le autorizzazioni necessarie per accedere a AWS KMS key. Per ulteriori informazioni, consulta Abilitare la compatibilità tra le origini eventi dai servizi AWS e gli argomenti crittografati nella Guida per gli sviluppatori di Amazon Simple Notification Service.

   c. Per Source type (Tipo di origine) scegliere un tipo di origine. Ad esempio, scegli Clusters (Cluster) o Cluster snapshots (Snapshot cluster).

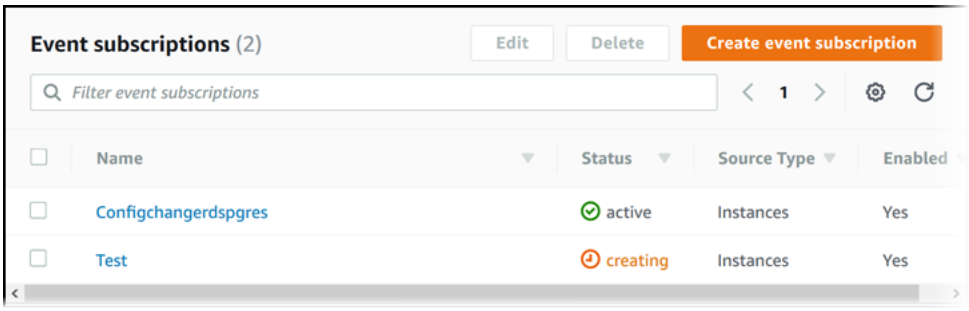
   d. Scegli le categorie di eventi e le risorse per i quali desideri ricevere notifiche eventi.

L'esempio seguente configura le notifiche degli eventi per l'istanza database denominata `testinst`.



e. Scegli Crea.

La console Amazon RDS indica che è in corso la creazione della sottoscrizione.



AWS CLI

Per sottoscrivere notifiche degli eventi RDS, utilizzare il comando AWS CLI `create-event-subscription`. Includi i parametri obbligatori seguenti:

- `--subscription-name`
- `--sns-topic-arn`

Example

Per Linux, macOS o Unix:

```
aws rds create-event-subscription \
    --subscription-name myeventsubscription \
    --sns-topic-arn arn:aws:sns:us-east-1:123456789012:myawsuser-RDS \
    --enabled
```

Per Windows:

```
aws rds create-event-subscription ^
    --subscription-name myeventsubscription ^
    --sns-topic-arn arn:aws:sns:us-east-1:123456789012:myawsuser-RDS ^
    --enabled
```

API

Per sottoscrivere le notifiche degli eventi Amazon RDS, invoca la funzione API Amazon RDS `CreateEventSubscription`. Includi i parametri obbligatori seguenti:

- `SubscriptionName`
- `SnsTopicArn`

Tag e attributi delle notifiche eventi di Amazon RDS

Quando Amazon RDS invia una notifica di evento ad Amazon Simple Notification Service (SNS) o Amazon EventBridge, la notifica contiene gli attributi dei messaggi e i tag degli eventi. RDS invia gli attributi del messaggio separatamente insieme al messaggio, mentre i tag degli eventi sono inclusi nel corpo del messaggio. Usa gli attributi del messaggio e i tag di Amazon RDS per aggiungere metadati alle risorse. Puoi modificare questi tag con notazioni personalizzate relative alle istanze database , cluster Aurora e così via. Per ulteriori informazioni sul tagging delle risorse di Amazon RDS, consulta Applicazione di tag alle risorse Amazon Aurora e Amazon RDS.

Per impostazione predefinita, Amazon SNS e Amazon EventBridge ricevono tutti i messaggi a loro inviati. SNS ed EventBridge possono filtrare il messaggio e inviare le notifiche alla modalità di comunicazione preferita, ad esempio tramite e-mail, SMS o una chiamata a un endpoint HTTP.

> **Note**
>
> La notifica inviata tramite e-mail o SMS non includerà tag di evento.

La tabella seguente mostra gli attributi dei messaggi per gli eventi RDS inviati agli abbonati all'argomento.

Attributo per gli eventi Amazon RDS	Descrizione
EventID	Identificatore del messaggio dell'evento RDS, ad esempio RDS-EVENT-0006.
Risorsa	L'identificatore ARN della risorsa che emette l'evento, ad esempio `arn:aws:rds:ap-southeast-2:123456789012:db:database-1` .

I tag RDS forniscono i dati sulla risorsa interessata dall'evento del servizio. RDS aggiunge lo stato corrente dei tag nel corpo del messaggio quando la notifica viene inviata a SNS o EventBridge.

Per ulteriori informazioni sull'applicazioni di filtri agli attributi dei messaggi per SNS, consulta Filtraggio messaggi di Amazon SNS nella Guida per gli sviluppatori di Amazon Simple Notification Service.

Per ulteriori informazioni sull'applicazione di filtri ai tag degli eventi per EventBridge, consulta Comparison operators for use in event patterns in Amazon EventBridge nella Guida per l'utente di Amazon EventBridge.

Per ulteriori informazioni sull'applicazione di filtri ai tag basati sul payload per SNS, consulta Introducing payload-based message filtering for Amazon SNS.

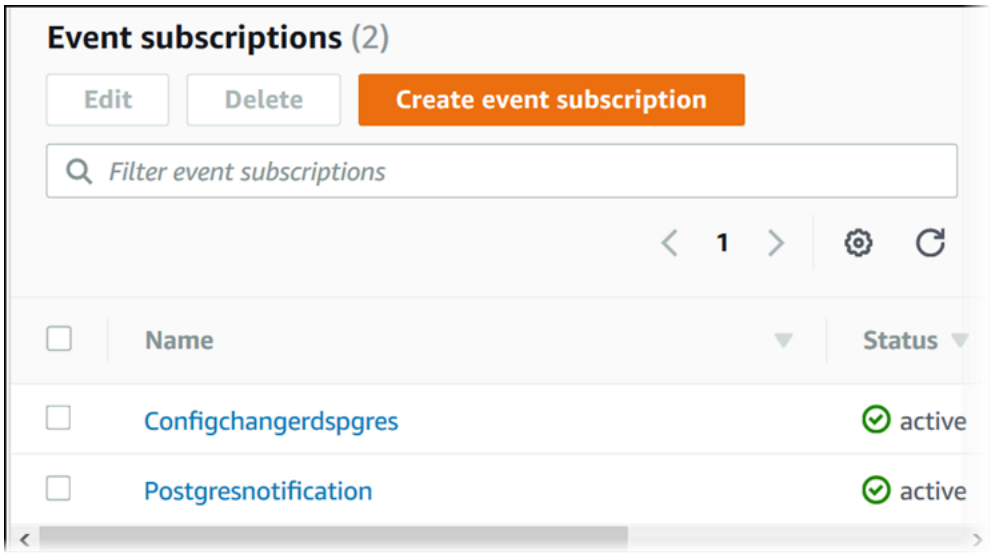
Elenco delle sottoscrizioni delle notifiche degli eventi Amazon RDS

Puoi creare un elenco delle sottoscrizioni correnti alle notifiche eventi Amazon RDS.

Console

Per creare un elenco delle sottoscrizioni correnti alle notifiche eventi Amazon RDS

1. Accedi alla Console di gestione AWS e apri la console Amazon RDS all'indirizzo https://console.aws.amazon.com/rds/.
2. Nel pannello di navigazione selezionare Event subscriptions (Sottoscrizioni di eventi). Il riquadro Event subscriptions (Sottoscrizioni di eventi) mostra tutte le sottoscrizioni delle notifiche degli eventi.



AWS CLI

Per visualizzare un elenco delle sottoscrizioni delle notifiche degli eventi Amazon RDS, utilizza il comando della AWS CLI `describe-event-subscriptions`.

Example

L'esempio seguente illustra tutte le sottoscrizioni a eventi.

```
aws rds describe-event-subscriptions
```

L'esempio seguente illustra `myfirsteventsubscription`.

```
aws rds describe-event-subscriptions --subscription-name myfirsteventsubscription
```

API

Per creare un elenco delle sottoscrizioni delle notifiche degli eventi Amazon RDS, chiamare l'operazione API Amazon RDS `DescribeEventSubscriptions`.

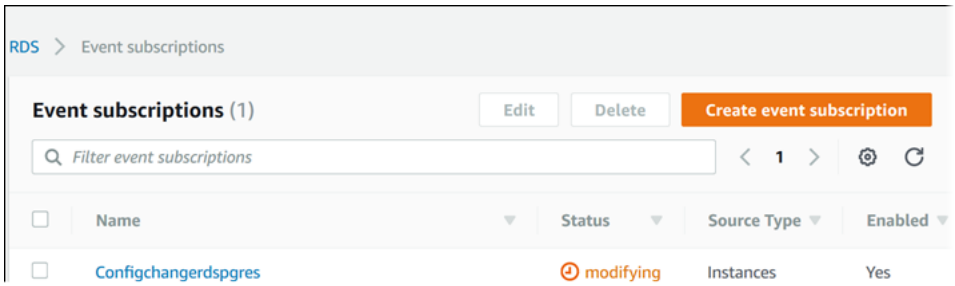
Modifica di una sottoscrizione alle notifiche eventi Amazon RDS

Dopo aver creato un abbonamento, puoi modificarne il nome, l'identificatore di origine, le categorie o l'ARN dell'argomento.

Console

Per modificare una sottoscrizione alle notifiche eventi Amazon RDS

1. Accedi alla Console di gestione AWS e apri la console Amazon RDS all'indirizzo https://console.aws.amazon.com/rds/.
2. Nel pannello di navigazione selezionare Event subscriptions (Sottoscrizioni di eventi).
3. Nel riquadro Event subscriptions (Sottoscrizioni di eventi) scegliere la sottoscrizione da modificare e selezionare Edit (Modifica).
4. Apportare le modifiche alla sottoscrizione nella sezione Target (Destinazione) o Source (Origine).
5. Scegli Modifica. La console Amazon RDS indica che è in corso la modifica della sottoscrizione.



AWS CLI

Per modificare una sottoscrizione delle notifiche degli eventi Amazon RDS, utilizza il comando della AWS CLI `modify-event-subscription`. Includi il seguente parametro obbligatorio:

- `--subscription-name`

Example

Il codice seguente abilita `myeventsubscription`.

Per Linux, macOS o Unix:

```
aws rds modify-event-subscription \
```

```
    --subscription-name myeventsubscription \
    --enabled
```

Per Windows:

```
aws rds modify-event-subscription ^
    --subscription-name myeventsubscription ^
    --enabled
```

API

Per modificare un evento Amazon RDS, chiamare l'operazione API Amazon RDS `ModifyEventSubscription`. Includi il seguente parametro obbligatorio:

- `SubscriptionName`

Aggiunta di un identificatore di origine a una sottoscrizione alle notifiche eventi Amazon RDS

Puoi aggiungere un identificatore di origine Amazon RDS che genera l'evento a una sottoscrizione esistente.

Console

Puoi aggiungere o rimuovere facilmente gli identificatori di origine tramite la console Amazon RDS, selezionandoli o deselezionandoli quando modifichi una sottoscrizione. Per ulteriori informazioni, consulta Modifica di una sottoscrizione alle notifiche eventi Amazon RDS.

AWS CLI

Per aggiungere un identificatore di origine a una sottoscrizione delle notifiche degli eventi Amazon RDS, utilizza il comando della AWS CLI `add-source-identifier-to-subscription`. Includi i parametri obbligatori seguenti:

- `--subscription-name`
- `--source-identifier`

Example

L'esempio seguente aggiunge l'identificatore di origine `mysqldb` alla sottoscrizione `myrdseventsubscription`.

Per Linux, macOS o Unix:

```
aws rds add-source-identifier-to-subscription \
    --subscription-name myrdseventsubscription \
    --source-identifier mysqldb
```

Per Windows:

```
aws rds add-source-identifier-to-subscription ^
    --subscription-name myrdseventsubscription ^
    --source-identifier mysqldb
```

API

Per aggiungere un identificatore di origine a una sottoscrizione delle notifiche degli eventi Amazon RDS, chiamare l'operazione API Amazon RDS `AddSourceIdentifierToSubscription`. Includi i parametri obbligatori seguenti:

- `SubscriptionName`
- `SourceIdentifier`

Rimozione di un identificatore di origine da una sottoscrizione alle notifiche eventi Amazon RDS

Per smettere di ricevere notifiche relative agli eventi di un'origine, puoi rimuovere un identificatore di origine Amazon RDS che genera l'evento da una sottoscrizione.

Console

Puoi aggiungere o rimuovere facilmente gli identificatori di origine tramite la console Amazon RDS, selezionandoli o deselezionandoli quando modifichi una sottoscrizione. Per ulteriori informazioni, consulta Modifica di una sottoscrizione alle notifiche eventi Amazon RDS.

AWS CLI

Per rimuovere un identificatore di origine da una sottoscrizione delle notifiche degli eventi Amazon RDS, utilizza il comando della AWS CLI `remove-source-identifier-from-subscription`. Includi i parametri obbligatori seguenti:

- `--subscription-name`
- `--source-identifier`

Example

L'esempio seguente rimuove l'identificatore dell'origine `mysqldb` dalla sottoscrizione `myrdseventsubscription`.

Per Linux, macOS o Unix:

```
aws rds remove-source-identifier-from-subscription \
    --subscription-name myrdseventsubscription \
    --source-identifier mysqldb
```

Per Windows:

```
aws rds remove-source-identifier-from-subscription ^
    --subscription-name myrdseventsubscription ^
    --source-identifier mysqldb
```

API

Per rimuovere un identificatore di origine da una sottoscrizione delle notifiche degli eventi Amazon RDS, utilizzare l'operazione API Amazon RDS `RemoveSourceIdentifierFromSubscription`. Includi i parametri obbligatori seguenti:

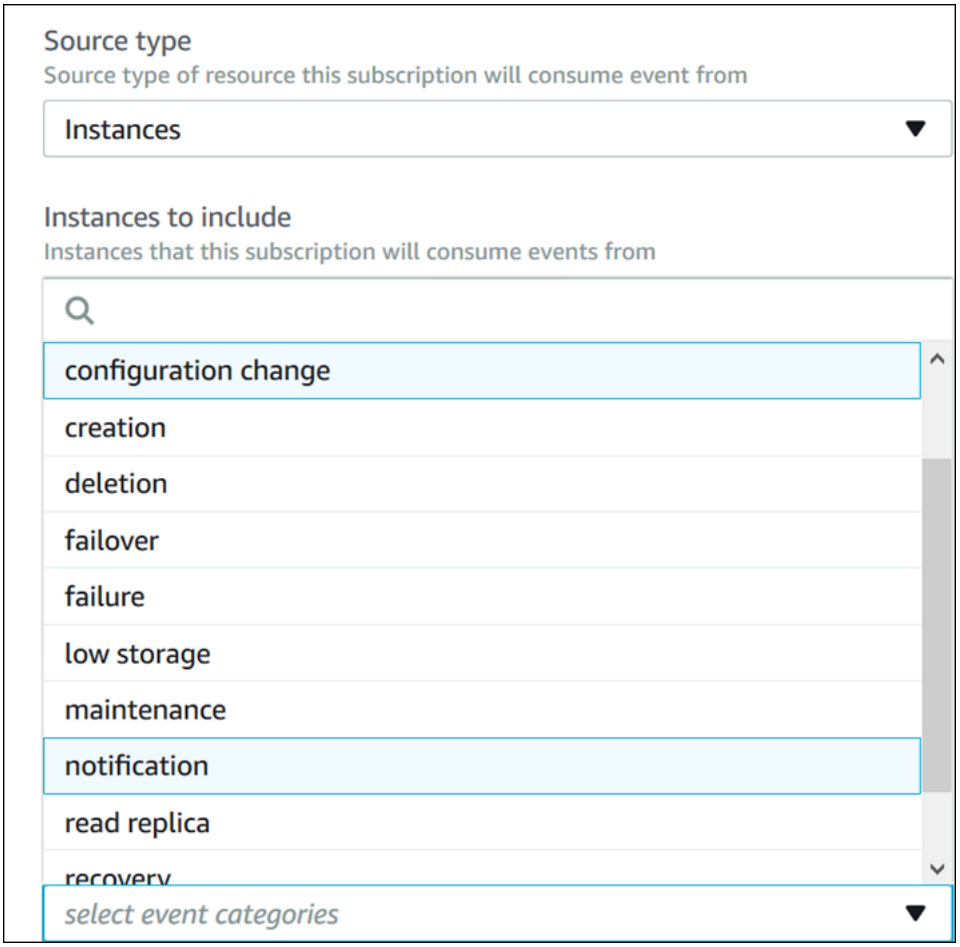
- `SubscriptionName`
- `SourceIdentifier`

Creazione di un elenco delle categorie di notifiche eventi Amazon RDS

Tutti gli eventi per un tipo di risorsa sono raggruppati in categorie. Per visualizzare l'elenco delle categorie disponibili, utilizza le procedure riportate di seguito.

Console

Quando crei o modifichi una sottoscrizione alle notifiche eventi, le categorie di eventi vengono visualizzate nella console Amazon RDS. Per ulteriori informazioni, consulta Modifica di una sottoscrizione alle notifiche eventi Amazon RDS.



AWS CLI

Per visualizzare un elenco delle categorie delle notifiche degli eventi Amazon RDS, utilizza il comando della AWS CLI `describe-event-categories`. Questo comando non prevede parametri obbligatori.

Example

```
aws rds describe-event-categories
```

API

Per elencare le categorie delle notifiche degli eventi Amazon RDS, utilizzare l'operazione API Amazon RDS `DescribeEventCategories`. Questo comando non prevede parametri obbligatori.

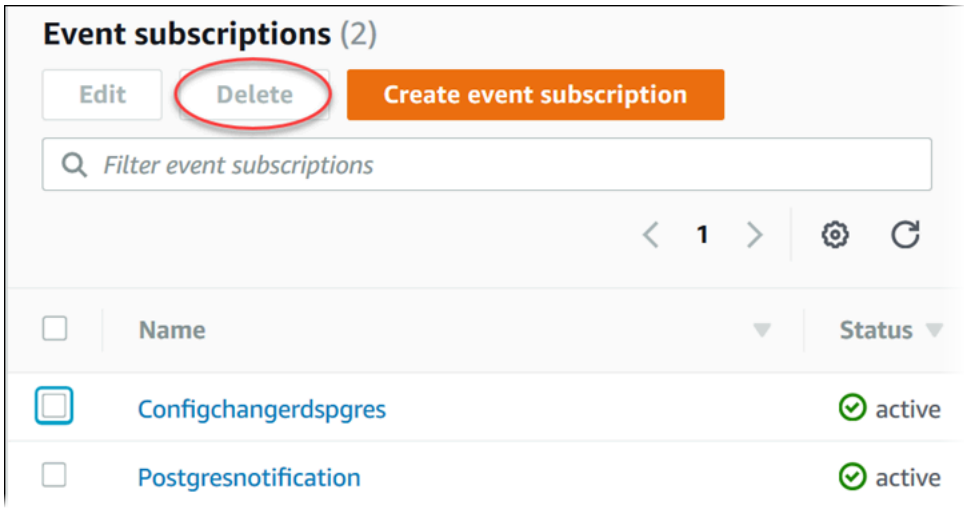
Eliminazione di una sottoscrizione alle notifiche eventi Amazon RDS

Puoi eliminare un abbonamento quando questo non è più necessario. Tutti gli abbonati all'argomento non riceveranno più le notifiche di eventi specificate dall'abbonamento.

Console

Per eliminare una sottoscrizione alle notifiche eventi Amazon RDS

1. Accedi alla Console di gestione AWS e apri la console Amazon RDS all'indirizzo https://console.aws.amazon.com/rds/.
2. Nel pannello di navigazione scegliere DB Event Subscriptions (Sottoscrizioni di eventi database).
3. Nel riquadro My DB Event Subscriptions (Sottoscrizioni di eventi database personali) scegliere la sottoscrizione che si vuole eliminare.
4. Scegliere Delete (Elimina).
5. La console Amazon RDS indica che è in corso l'eliminazione della sottoscrizione.



AWS CLI

Per eliminare una sottoscrizione delle notifiche degli eventi Amazon RDS, utilizza il comando AWS CLI `delete-event-subscription`. Includi il seguente parametro obbligatorio:

- `--subscription-name`

Example

L'esempio seguente elimina la sottoscrizione `myrdssubscription`.

```
aws rds delete-event-subscription --subscription-name myrdssubscription
```

API

Per eliminare una sottoscrizione delle notifiche degli eventi Amazon RDS, utilizzare l'operazione API RDS `DeleteEventSubscription`. Includi il seguente parametro obbligatorio:

- `SubscriptionName`

Creazione di una regola che si attiva su un evento Amazon Aurora

Amazon EventBridge consente di automatizzare i servizi AWS e di rispondere a eventi di sistema, come i problemi relativi alla disponibilità delle applicazioni o alle modifiche delle risorse.

Argomenti

- Tutorial: Registrazione delle modifiche dello stato di un'istanza database tramite Amazon EventBridge

Tutorial: Registrazione delle modifiche dello stato di un'istanza database tramite Amazon EventBridge

In questo tutorial, crei una funzione AWS Lambda che registri le modifiche dello stato per un'istanza . Successivamente crei una regola che esegua la funzione ogni volta che si verifica un cambiamento di stato di un'istanza database RDS esistente. Il tutorial presuppone che si dispone di una piccola istanza di test in esecuzione che è possibile arrestare temporaneamente.

> **Important**
>
> Non eseguire questo tutorial su un'istanza database di produzione in esecuzione.

Argomenti

- Fase 1. Creazione di una funzione AWS Lambda
- Fase 2: Creazione di una regola
- Fase 3: Test della regola

Fase 1. Creazione di una funzione AWS Lambda

Crea una funzione Lambda per registrare gli eventi di modifica dello stato. È necessario specificare questa funzione alla creazione della regola.

Per creare una funzione Lambda

1. Apri la console AWS Lambda all'indirizzo https://console.aws.amazon.com/lambda/.
2. Se è la prima volta che utilizzi Lambda, verrà visualizzata una pagina di benvenuto. Selezionare Get Started Now (Inizia subito). Altrimenti, scegliere Create function (Crea funzione).

3. Scegli Author from scratch (Crea da zero).
4. Nella pagina Create function (Crea funzione), procedere come segue:
   a. Digitare un nome e una descrizione per la funzione Lambda. Ad esempio, denomina la funzione **`RDSInstanceStateChange`**.
   b. In Runtime, seleziona Node.js 16x.
   c. In Architecture (Architettura), scegli x86_64.
   d. In Execution role (Ruolo di esecuzione), esegui una delle operazioni seguenti:
      - Scegliere Create a new role with basic Lambda permissions (Crea un nuovo ruolo con le autorizzazioni Lambda di base).
      - In Execution role (Ruolo di esecuzione), sceglie Use an existing role (Utilizza un ruolo esistente). Scegli il ruolo che desideri usare.
   e. Selezionare Create function (Crea funzione).
5. Sulla pagina RDSInstanceStateChange, effettua le seguenti operazioni:
   a. In Origine codice, seleziona index.js.
   b. Nel riquadro di index.js, elimina il codice esistente.
   c. Immetti il seguente codice:

```
console.log('Loading function');

exports.handler = async (event, context) => {
    console.log('Received event:', JSON.stringify(event));
};
```

   d. Selezionare Deploy (Distribuisci).

Fase 2: Creazione di una regola

Crea una regola per l'esecuzione della funzione Lambda ogni volta che avvii un'istanza Amazon RDS.

Per creare la regola EventBridge

1. Aprire la console Amazon EventBridge all'indirizzo https://console.aws.amazon.com/events/.
2. Nel pannello di navigazione, scegli Regole.

3. Scegli Create rule (Crea regola).
4. Immettere un nome e una descrizione per la regola. Ad esempio, specifica **`RDSInstanceStateChangeRule`**.
5. Scegli Rule with an event pattern (Regola con un modello di eventi), quindi seleziona Next (Successivo).
6. Per Event source (Origine evento), scegli AWS events or EventBridge partner events (Eventi o eventi di partner EventBridge).
7. Scorri verso il basso fino alla sezione Event pattern (Modello di eventi).
8. In Event source (Origine eventi), selezionare Servizi AWS.
9. In AWS service (Servizio AWS), scegli Relational Database Service (RDS).
10. Per Tipo di evento, seleziona Evento istanza database RDS.
11. Lascia il modello di eventi predefinito. Quindi scegli Successivo.
12. Per Target types (Tipi di destinazione), scegli AWS service (Servizio).
13. Per Select a target (Seleziona destinazione), scegli Lambda function (Funzione Lambda).
14. In Function (Funzione), seleziona la funzione Lambda che hai creato. Quindi scegli Successivo.
15. In Configure tags (Configura tag), scegli Next (Successivo).
16. Esamina i passaggi nella regola. Quindi scegli Create rule (Crea regola).

Fase 3: Test della regola

Per verificare la regola, arresta un'istanza database RDS. Dopo aver atteso alcuni minuti perché l'istanza venga avviata e inizializzata, verifica che la funzione Lambda sia stata richiamata.

Per effettuare il test della regola arrestando un'istanza database

1. Apri la console Amazon RDS all’indirizzo https://console.aws.amazon.com/rds/.
2. Arresta un'istanza database RDS.
3. Aprire la console Amazon EventBridge all'indirizzo https://console.aws.amazon.com/events/.
4. Nel pannello di navigazione, seleziona Regole, scegli il nome della regola creata.
5. In Dettagli della regola scegli Monitoraggio.

   Sarai reindirizzato alla console Amazon CloudWatch. Se non vieni reindirizzato, fai clic su Visualizza i parametri in CloudWatch.
6. In Tutti i parametri, seleziona il nome della regola creata.

Il grafico deve indicare che la regola è stata richiamata.

7. Nel pannello di navigazione, selezionare Log groups (Gruppi di log).
8. Scegli il nome del gruppo di log per la funzione Lambda (/aws/lambda/***nome-funzione***).
9. Scegliere il nome del flusso di log per visualizzare i dati forniti dalla funzione per l'istanza avviata. Sarà visualizzato un evento ricevuto simile a quello seguente:

```
{
    "version": "0",
    "id": "12a345b6-78c9-01d2-34e5-123f4ghi5j6k",
    "detail-type": "RDS DB Instance Event",
    "source": "aws.rds",
    "account": "111111111111",
    "time": "2021-03-19T19:34:09Z",
    "region": "us-east-1",
    "resources": [
        "arn:aws:rds:us-east-1:111111111111:db:testdb"
    ],
    "detail": {
        "EventCategories": [
            "notification"
        ],
        "SourceType": "DB_INSTANCE",
        "SourceArn": "arn:aws:rds:us-east-1:111111111111:db:testdb",
        "Date": "2021-03-19T19:34:09.293Z",
        "Message": "DB instance stopped",
        "SourceIdentifier": "testdb",
        "EventID": "RDS-EVENT-0087"
    }
}
```

Per altri esempi di eventi RDS in formato JSON, vedere Panoramica degli eventi per Aurora.

10. (Facoltativo) Al termine, puoi aprire la console Amazon RDS e avviare l'istanza terminata.

Categorie di eventi e messaggi di evento di Amazon RDS per Aurora

Amazon RDS genera un numero significativo di eventi in categorie a cui puoi abbonarti utilizzando la console Amazon RDS o AWS CLI l'API.

Argomenti

- Eventi di cluster di database
- Eventi snapshot cluster di database
- Eventi di istanza database
- Eventi gruppo di parametri database
- Eventi gruppo di sicurezza DB
- Eventi gruppo shard database
- Eventi RDS Proxy
- Eventi di implementazione blu/verde

Eventi di cluster di database

La tabella seguente riporta la categoria di eventi e un elenco di eventi applicabili quando il tipo di origine è un cluster database.

> **Note**
>
> Non esiste alcuna categoria di eventi per Aurora Serverless nel tipo di evento cluster database. Gli eventi serverless Aurora vanno da RDS-EVENT-0141 a RDS-EVENT-0149.

Categoria	ID evento RDS	Messaggio	Note
modifica della configurazione	RDS-EVENT-0016	Reimpostare le credenziali master.	Nessuno
modifica della configurazione	RDS-EVENT-0179	I flussi di attività del database vengono avviati nel cluster di database.	Per ulteriori informazioni, consulta Monitoraggio di Amazon Aurora

Categoria	ID evento RDS	Messaggio	Note
			tramite i flussi di attività del database.
modifica della configurazione	RDS-EVENT-0180	I flussi di attività del database vengono arrestati nel cluster di database.	Per ulteriori informazioni consulta Monitoraggio di Amazon Aurora tramite i flussi di attività del database.
creazione	RDS-EVENT-0170	Cluster di database creato.	Nessuno
eliminazione	RDS-EVENT-0171	Cluster di database eliminato.	Nessuno
failover	RDS-EVENT-0069	Failover del cluster non riuscito. Verificare lo stato delle istanze del cluster e riprovare.	Nessuno
failover	RDS-EVENT-0070	Promuovere nuovamente le primarie precedenti:. *name*	Nessuno
failover	RDS-EVENT-0071	Failover completato sull'istanza DB:*name*.	Nessuno
failover	RDS-EVENT-0072	È stato avviato il failover dello stesso tipo AZ sull'istanza DB:. *name*	Nessuno
failover	RDS-EVENT-0073	Avviato il failover cross-AZ verso un'istanza DB:. *name*	Nessuno

Categoria	ID evento RDS	Messaggio	Note
errore	RDS-EVENT-0083	Amazon RDS non è riuscito a creare credenziali per accedere al tuo bucket Amazon S3 per il tuo cluster DB. *name* Possibili cause: il ruolo IAM di acquisizione di snapshot S3 non è configurato correttamente nel tuo account o non è stato trovato il bucket Amazon S3 specificato. Per ulteriori dettagli, consulta la sezione relativa alla risoluzione dei problemi nella documenta zione di Amazon RDS.	Per ulteriori informazioni, consulta Migrazione fisica da MySQL utilizzando XtraBackup Percona e Amazon S3 .
errore	RDS-EVENT-0143	Il cluster DB non è riuscito a scalare da *units* a *units* per questo motivo:. *reason*	Scaling non riuscito per il cluster di database Aurora Serverless.
errore	RDS-EVENT-0354	Non è possibile creare il cluster DB a causa di risorse incompatibili. *message*.	*message*Include dettagli sull'errore.
errore	RDS-EVENT-0355	Il cluster DB non può essere creato a causa di limiti di risorse insufficienti. *message*.	*message*Include dettagli sull'errore.

Categoria	ID evento RDS	Messaggio	Note
errore	RDS-EVENT-0387	Impossibile partizionare le istanze DB nel cluster DB *name* per l'applicazione delle patch.	Gli aggiornamenti del sistema operativo per le istanze database nel cluster di database non sono riusciti.
Failover globale	RDS-EVENT-0181	È iniziato il passaggio globale al cluster DB nella regione. *name name*	Questo evento riguarda un'operazione di switchover (precedentemente denominata "failover pianificato gestito"). Il processo può essere ritardato perché altre operazioni sono in esecuzione sul cluster di database.
Failover globale	RDS-EVENT-0182	Il vecchio cluster DB primario *name* nella regione è *name* stato chiuso con successo.	Questo evento riguarda un'operazione di switchover (precedentemente denominata "failover pianificato gestito"). La vecchia istanza principale nel database globale non accetta scritture. Tutti i volumi sono sincronizzati.

Categoria	ID evento RDS	Messaggio	Note
Failover globale	RDS-EVENT-0183	In attesa della sincroniz zazione dei dati tra i membri del cluster globale. Ritardi attuali del cluster di database primario: *reason*.	Questo evento riguarda un'operazione di switchove r (precedentemente denominata "failover pianificato gestito"). Si verifica un ritardo di replica durante la fase di sincronizzazione del failover globale del database.
Failover globale	RDS-EVENT-0184	Il nuovo cluster DB primario *name* nella regione *name* è stato promosso con successo.	Questo evento riguarda un'operazione di switchove r (precedentemente denominata "failover pianificato gestito"). La topologia del volume del database globale viene ristabilita con il nuovo volume primario.

Categoria	ID evento RDS	Messaggio	Note
Failover globale	RDS-EVENT-0185	Il passaggio globale al cluster DB *name* nella regione *name* è terminato.	Questo evento riguarda un'operazione di switchover (precedentemente denominata "failover pianificato gestito"). Lo switchover globale del database è stato completato nel cluster database primario. Le repliche potrebbero richiedere molto tempo per arrivare online dopo il completamento del failover.
Failover globale	RDS-EVENT-0186	Il passaggio globale al cluster DB nella regione è stato *name* annullato. *name*	Questo evento riguarda un'operazione di switchover (precedentemente denominata "failover pianificato gestito").
Failover globale	RDS-EVENT-0187	Il passaggio globale al cluster DB nella regione non è riuscito. *name name*	Questo evento riguarda un'operazione di switchover (precedentemente denominata "failover pianificato gestito").
Failover globale	RDS-EVENT-0238	Il failover globale verso il cluster DB *name* nella regione è stato completato. *name*	Nessuno
Failover globale	RDS-EVENT-0239	Il failover globale sul cluster DB *name* nella regione *name* non è riuscito.	Nessuno

Categoria	ID evento RDS	Messaggio	Note
Failover globale	RDS-EVENT-0240	È iniziata la risincron izzazione dei membri del cluster DB *name* nella regione *name* dopo il failover globale.	Nessuno
Failover globale	RDS-EVENT-0241	È stata completata la risincronizzazione dei membri del cluster DB nella regione dopo il failover globale. *name name*	Nessuno
Failover globale	RDS-EVENT-0397	Aurora ha terminato la modifica del nome DNS in cui si risolve l'endpoint di scrittura globale.	Nessuno
Failover globale	RDS-EVENT-0423	In attesa della sincroniz zazione dei dati con il cluster di database di destinazione. Il cluster di database di destinazione corrente è in ritardo rispetto al cluster di database primario: *%s*.	Questo evento riguarda un'operazione di switchove r (precedentemente denominata "failover pianificato gestito"). Si verifica un ritardo di replica durante la fase di sincronizzazione del failover globale del database.
manutenzi one	RDS-EVENT-0156	È disponibile un aggiornam ento della versione secondaria del motore del database per il cluster di database.	Nessuno

Categoria	ID evento RDS	Messaggio	Note
manutenzione	RDS-EVENT-0173	La versione del motore del cluster database è stata aggiornata.	L'applicazione di patch del cluster database è stata completata.
manutenzione	RDS-EVENT-0174	Il cluster di database si trova in uno stato che non consente l'aggiornamento.	Nessuno
manutenzione	RDS-EVENT-0176	La versione principale del motore del cluster database è stata aggiornata.	Nessuno
manutenzione	RDS-EVENT-0177	L'aggiornamento del cluster di database è in corso.	Nessuno
manutenzione	RDS-EVENT-0286	È iniziato l'aggiornamento della versione del motore del cluster di *version_number* database. Il cluster rimane online.	Nessuno
manutenzione	RDS-EVENT-0287	Requisito di aggiornamento del sistema operativo rilevato.	Nessuno
manutenzione	RDS-EVENT-0288	Avvio dell'aggiornamento del sistema operativo del cluster in corso.	Nessuno
manutenzione	RDS-EVENT-0289	Aggiornamento del sistema operativo del cluster completato.	Nessuno

Categoria	ID evento RDS	Messaggio	Note
manutenzione	RDS-EVENT-0290	Il cluster di database è stato patchato: versione sorgente *version_number new_version_number* =>.	Nessuno
manutenzione	RDS-EVENT-0363	Preparazione dell'aggiornamento in corso: *cluster_name*	Sono stati avviati i controlli preliminari per l'aggiornamento per il cluster di database.
manutenzione	RDS-EVENT-0388	Avvio di patch offline alle istanze DB nella partizione e*name*/*name*per il cluster DB:. *name partition_n*	Avvio degli aggiornamenti del sistema operativo per le istanze database nel cluster di database.
manutenzione	RDS-EVENT-0389	Non è stato possibile aggiornare il sistema operativo del cluster di database. Attendere la finestra di manutenzione successiva oppure aggiornare il sistema operativo del cluster di database manualmente.	Nessuno

Categoria	ID evento RDS	Messaggio	Note
manutenzione	RDS-EVENT-0424	La versione in esecuzione del cluster *name* DB è superiore alla versione *version* di aggiornamento di destinazione *version* per il cluster globale. Non è consigliabile che nel cluster secondario sia in esecuzione una versione successiva rispetto a quella del cluster globale, poiché potrebbero verificarsi problemi durante il failover o lo switchover. Valutare la possibilità di aggiornare il cluster globale alla versione corrispondente.	Nessuno
notification	RDS-EVENT-0076	Migrazione da *name* a *name* non riuscita. Motivo:*reason*.	La migrazione a un cluster di database Aurora non è riuscita.
notifica	RDS-EVENT-0077	Conversione non riuscita*name*. *name*a InnoDB. Motivo:*reason*.	Un tentativo di convertire una tabella dal database di origine a InnoDB non è riuscito durante la migrazione a un cluster di database Aurora.

Categoria	ID evento RDS	Messaggio	Note
notifica	RDS-EVENT-0085	Impossibile aggiornare il cluster DB *name* perché lo stato dell'istanza *name* è di*name*. Risolvere il problema o eliminare l'istanza e riprovare.	Si è verificato un errore durante il tentativo di patch al cluster Aurora DB. Controlla lo stato dell'istanza, risolvi il problema e riprova. Per ulteriori informazioni, consulta Manutenzione di un cluster di database Amazon Aurora.
notifica	RDS-EVENT-0141	Scalabilità del cluster DB da *units* a *units* per questo motivo:*reason*.	Scaling avviato per il cluster di database Aurora Serverless.
notifica	RDS-EVENT-0142	Il cluster DB è stato scalato da a*units*. *units*	Scaling completato per il cluster di database Aurora Serverless.
notifica	RDS-EVENT-0144	Il cluster database verrà messo in pausa.	È stata avviata una pausa automatica per il cluster database Aurora Serverless.
notification	RDS-EVENT-0145	Il cluster database è in pausa.	Il cluster database Aurora Serverless è stato messo in pausa.
notification	RDS-EVENT-0146	Pausa annullata per il cluster database.	Pausa per il cluster database Aurora Serverless annullata.
notification	RDS-EVENT-0147	Il cluster database è in fase di riattivazione.	Operazione di riattivazione avviata per il cluster database Aurora Serverless.

Categoria	ID evento RDS	Messaggio	Note
notification	RDS-EVENT-0148	Il cluster database è stato riattivato.	Operazione di riattivazione per il cluster database Aurora Serverless completata.
notification	RDS-EVENT-0149	Il cluster DB è stato scalato da *units* a*units*, ma la scalabilità non è stata perfetta per questo motivo:. *reason*	Scaling continuo completato con l'opzione forzata per il cluster di database Aurora Serverless. Le connessioni potrebbero essere state interrotte come richiesto.
notifica	RDS-EVENT-0150	Cluster database arrestato.	Nessuno
notification	RDS-EVENT-0151	Cluster database avviato.	Nessuno
notification	RDS-EVENT-0152	Arresto del cluster database non riuscito.	Nessuno
notification	RDS-EVENT-0153	Il cluster di database è in fase di avvio perché è stato superato il tempo massimo concesso per l'arresto.	Nessuno
notification	RDS-EVENT-0172	Cluster rinominato da a. *name name*	Nessuno
notification	RDS-EVENT-0234	Attività di esportazione non riuscita.	L'attività di esportazione del cluster database non è riuscita.
notification	RDS-EVENT-0235	Attività di esportazione annullata.	L'attività di esportazione del cluster database è stata annullata.

Categoria	ID evento RDS	Messaggio	Note
notification	RDS-EVENT-0236	Attività di esportazione completata.	L'attività di esportazione del cluster database è stata completata.
notification	RDS-EVENT-0386	Le istanze DB nel cluster DB *name* sono state partizionate:. *list_of_partitions* Il cluster di database è online.	Gli aggiornamenti del sistema operativo per le istanze database nel cluster di database sono riusciti.
notification	RDS-EVENT-0426	RDS non può configurare il cluster DB *name* come fonte di replica a causa di transazioni inattive o di lunga durata. Attendere il completamento delle transazioni o annullarle e riprovare.	Nessuno

Eventi snapshot cluster di database

La tabella seguente riporta la categoria di eventi e un elenco di eventi applicabili quando il tipo di origine è uno snapshot del cluster database.

Categoria	ID evento RDS	Messaggio	Note
backup	RDS-EVENT-0074	Creazione di uno snapshot del cluster manuale in corso.	Nessuno
backup	RDS-EVENT-0075	Snapshot del cluster manuale creato.	Nessuno

Categoria	ID evento RDS	Messaggio	Note
notification	RDS-EVENT-0162	Attività di esportazione dello snapshot del cluster non riuscita.	Nessuno
notification	RDS-EVENT-0163	Attività di esportazione dello snapshot del cluster annullata.	Nessuno
notification	RDS-EVENT-0164	Attività di esportazione dello snapshot del cluster completata.	Nessuno
backup	RDS-EVENT-0168	Creazione snapshot cluster automatizzato.	Nessuno
backup	RDS-EVENT-0169	Snapshot di cluster automatizzato creato.	Nessuno
errore	RDS-EVENT-0489	Attività di esportazi one*name*: l'estrazione della tabella non è riuscita a causa di. *name message*	Questo evento non è supportato da un'esport azione più rapida.
notification	RDS-EVENT-0491	Attività di esportazi one*name*: la tabella *name* verrà esportata in una modalità a thread singolo più lenta. Attendi un ritardo nel completamento dell'esportazione.	Questo evento non è supportato da un'esport azione più rapida.

Categoria	ID evento RDS	Messaggio	Note
notification	RDS-EVENT-0492	Attività di esportazione*name*: la tabella *name* è in corso e la dimensione esportata è in GB. *number*	Questo evento verrà inviato solo per le tabelle che richiedono più di 1 ora per essere completate. Questo evento non è supportato da un'esportazione più rapida.
notification	RDS-EVENT-0493	L'attività di esportazione è in corso. *name name*	Nessuno
		Attività di esportazione*name*: l'avanzamento dell'esportazione è in*number*% con *number* le tabelle in corso e le *number* tabelle completate. La dimensione totale delle tabelle finite è di *number* GB.	Questo evento non è supportato da un'esportazione più rapida.

Eventi di istanza database

La tabella seguente riporta la categoria di eventi e un elenco di eventi applicabili quando il tipo di origine è un'istanza database.

Categoria	ID evento RDS	Messaggio	Note
disponibilità	RDS-EVENT-0004	L'istanza database è stata arrestata.	Nessuno
availability	RDS-EVENT-0006	L'istanza database è stata riavviata.	Nessuno

Categoria	ID evento RDS	Messaggio	Note
disponibilità	RDS-EVENT-0022	Errore durante il riavvio di mysql:. *message*	Si è verificato un errore durante il riavvio di Aurora MySQL o RDS per MariaDB.
disponibilità	RDS-EVENT-0419	Amazon RDS non è riuscito ad accedere alla chiave di KMS crittografia per l'istanza *name* del database. Il database verrà posto in uno stato inaccessibile. Per ulteriori dettagli, consulta la sezione relativa alla risoluzione dei problemi nella documentazione di Amazon RDS.	Nessuno
backtrack	RDS-EVENT-0131	La finestra di backtrack effettiva è più piccola della finestra di backtrack target specificata. Prendi in considerazione di ridurre il numero di ore della finestra di backtrack target.	Per ulteriori informazioni sul backtrack, consulta Backtrack di un cluster database Aurora.
backtrack	RDS-EVENT-0132	La finestra di backtrack effettiva è uguale alla finestra di backtrack target specificata.	Nessuno
modifica della configurazione	RDS-EVENT-0011	Aggiornato per utilizzare DBParameter Group*name*.	Nessuno

Categoria	ID evento RDS	Messaggio	Note
modifica della configurazione	RDS-EVENT-0012	Vengono applicate le modifiche alla classe di istanza database.	Nessuno
modifica della configurazione	RDS-EVENT-0014	Applicazione delle modifiche alla classe di istanza database completata.	Nessuno
modifica della configurazione	RDS-EVENT-0017	Applicazione delle modifiche all'archiviazione allocata completata.	Nessuno
modifica della configurazione	RDS-EVENT-0025	Applicazione delle modifiche per la conversione in un'istanza database Multi-AZ completata.	Nessuno
modifica della configurazione	RDS-EVENT-0029	Applicazione delle modifiche per la conversione in un'istanza database standard (Single-AZ) completata.	Nessuno
modifica della configurazione	RDS-EVENT-0033	Ci sono *`number`* utenti che corrispondono al nome utente principale; viene reimpostato solo quello non legato a un host specifico.	Nessuno
modifica della configurazione	RDS-EVENT-0067	Impossibile reimpostare la password. Informazioni sull'errore:. *`message`*	Nessuno

Categoria	ID evento RDS	Messaggio	Note
modifica della configurazione	RDS-EVENT-0078	Intervallo di monitoraggio modificato in. *number*	La configurazione Enhanced Monitoring è stata modificata.
modifica della configurazione	RDS-EVENT-0092	Aggiornamento gruppo di parametri DB terminato.	Nessuno
creazione	RDS-EVENT-0005	Istanza database creata.	Nessuno
eliminazione	RDS-EVENT-0003	Istanza database eliminata.	Nessuno
errore	RDS-EVENT-0035	Istanza di database inserita in*state*. *message*.	L'istanza database include parametri non validi. Ad esempio, se non è stato possibile avviare l'istanza database perché un parametro relativo alla memoria è troppo elevato per questa classe di istanze, è necessario modificare il parametro relativo alla memoria e riavviare l'istanza database.
errore	RDS-EVENT-0036	Istanza di database in*state*. *message*.	L'istanza database si trova in una rete non compatibile. Alcune delle sottoreti specificate non IDs sono valide o non esistono.

Categoria	ID evento RDS	Messaggio	Note
errore	RDS-EVENT-0079	Amazon RDS non è stato in grado di creare le credenziali per un monitoraggio avanzato e questa funzionalità è stata disattivata. Ciò è probabilmente dovuto al fatto che rds-monitoring-role non è presente e non è configurato correttamente nel tuo account. Per ulteriori dettagli, consulta la sezione relativa alla risoluzione dei problemi nella documentazione di Amazon RDS.	Impossibile abilitare la funzionalità Monitoraggio avanzato senza il ruolo IAM di monitoraggio avanzato. Per ulteriori informazioni sulla creazione del ruolo IAM, consulta Per creare un ruolo IAM per Amazon RDS Enhanced Monitoring.
errore	RDS-EVENT-0080	Amazon RDS non è stato in grado di configurare il monitoraggio avanzato sulla tua istanza *name* e questa funzionalità è stata disabilitata. Ciò è probabilmente dovuto al fatto che rds-monitoring-role non è presente e non è configurato correttamente nel tuo account. Per ulteriori dettagli, consulta la sezione relativa alla risoluzione dei problemi nella documentazione di Amazon RDS.	La funzionalità Monitoraggio avanzato è stata disabilitata a causa di un errore che ha provocato la modifica della configurazione. È possibile che il ruolo IAM di monitoraggio avanzato non sia configurato correttamente. Per informazioni sulla creazione del ruolo IAM di monitoraggio avanzato, consulta Per creare un ruolo IAM per Amazon RDS Enhanced Monitoring.

Categoria	ID evento RDS	Messaggio	Note
errore	RDS-EVENT-0082	Amazon RDS non è riuscito a creare credenziali per accedere al tuo Amazon S3 Bucket per la tua istanza DB. *name* Possibili cause: il ruolo IAM di acquisizione di snapshot S3 non è configurato correttamente nel tuo account o non è stato trovato il bucket Amazon S3 specificato. Per ulteriori dettagli, consulta la sezione relativa alla risoluzione dei problemi nella documentazione di Amazon RDS.	Aurora non è riuscito a copiare i dati di backup da un bucket Simple Storage Service (Amazon S3). È probabile che le autorizzazioni necessarie perché Aurora possa accedere al bucket Simple Storage Service (Amazon S3) non siano configurate correttamente. Per ulteriori informazioni, consulta Migrazione fisica da MySQL utilizzando XtraBackup Percona e Amazon S3 .
errore	RDS-EVENT-0353	L'istanza DB non può essere creata a causa di limiti di risorse insufficienti. *message*.	*message*Include dettagli sull'errore.

Categoria	ID evento RDS	Messaggio	Note
errore	RDS-EVENT-0418	Amazon RDS non è in grado di accedere alla chiave di KMS crittografia per l'istanza *name* del database. È possibile che la chiave sia disabilitata o che Amazon RDS non riesca ad accedervi. Se il problema persiste, il database verrà posto in uno stato inaccessibile. Per ulteriori dettagli, consulta la sezione relativa alla risoluzione dei problemi nella documentazione di Amazon RDS.	Nessuno
errore	RDS-EVENT-0420	Amazon RDS ora può accedere correttamente alla chiave di KMS crittografia per l'istanza *name* di database.	Nessuno

Categoria	ID evento RDS	Messaggio	Note
storage insufficiente	RDS-EVENT-0007	Lo spazio di archiviazione allocato è stato esaurito. Allocare spazio di archiviazione aggiuntivo per risolvere il problema.	Lo storage allocato per l'istanza database è esaurito. Per risolvere questo problema, assegna ulteriore storage per l'istanza database. Per ulteriori informazioni, consulta le domande frequenti su RDS. Puoi monitorare lo spazio di storage per un'istanza database usando il parametro Free Storage Space (Spazio di storage libero).
storage insufficiente	RDS-EVENT-0089	La capacità di storage gratuita per l'istanza DB: *name* è bassa rispetto allo *percentage* storage fornito [Provisioned Storage:*size*, Free Storage:]. *size* È possibile aumentare lo spazio di archiviazione assegnato per risolvere questo problema.	L'istanza database ha utilizzato oltre il 90% dello storage allocato. Puoi monitorare lo spazio di storage per un'istanza database usando il parametro Free Storage Space (Spazio di storage libero).
storage insufficiente	RDS-EVENT-0227	Lo storage del cluster Aurora è pericolosamente basso e rimangono solo terabyte. *amount* Prendere misure idonee per ridurre il carico di archiviazione sul cluster.	Il sottosistema di archiviazione di Aurora sta esaurendo lo spazio.

Categoria	ID evento RDS	Messaggio	Note
manutenzione	RDS-EVENT-0026	Applicazione di patch offline all'istanza database in corso.	È in corso la manutenzione offline dell'istanza database. L'istanza database non è attualmente disponibile.
manutenzione	RDS-EVENT-0027	Applicazione di patch offline all'istanza database terminata.	La manutenzione offline dell'istanza database è stata completata. L'istanza database è ora disponibile.
manutenzione	RDS-EVENT-0047	Applicazione delle patch all'istanza database completata.	Nessuno
manutenzione	RDS-EVENT-0155	È disponibile un aggiornamento della versione secondaria del motore di database per l'istanza database.	Nessuno
manutenzione	RDS-EVENT-0178	L’aggiornamento dell’istanza database è in corso.	Nessuno
manutenzione	RDS-EVENT-0422	RDS sostituirà l'host dell'istanza DB a *name* causa di un'azione di manutenzione in sospeso.	Nessuno
notifica	RDS-EVENT-0044	*message*	Si tratta di una notifica emessa dall'operatore. Per ulteriori informazioni, consulta il messaggio di evento.

Categoria	ID evento RDS	Messaggio	Note
notifica	RDS-EVENT-0048	Ritardo dell'aggiornamento del motore di database in corso perché questa istanza contiene repliche di lettura che prima devono essere aggiornate.	L'applicazione di patch all'istanza database è stata ritardata.
notifica	RDS-EVENT-0087	Istanza database arrestata.	Nessuno
notification	RDS-EVENT-0088	Istanza database creata.	Nessuno
notification	RDS-EVENT-0365	I file del fuso orario sono stati aggiornati. Riavviare l’istanza database affinché vengano applicate le modifiche.	Nessuno
notifica, serverless	RDS-EVENT-0370	Sospensione avviata per l’istanza database.	È stato avviato un nuovo tentativo di sospendere un’istanza database Aurora Serverless v2 inattiva.
notifica, serverless	RDS-EVENT-0371	Sospensione annullata per l’istanza database.	Un tentativo di sospendere un’istanza database Aurora Serverless v2 inattiva ha avuto esito negativo a causa del carico di lavoro.
notifica, serverless	RDS-EVENT-0372	La sospensione dell’istanza database è riuscita.	L’istanza database Aurora Serverless v2 è stata sospesa.

Categoria	ID evento RDS	Messaggio	Note
notifica, serverless	RDS-EVENT-0373	Ripresa avviata per l'istanza database.	È stata avviata la ripresa dell'istanza database Aurora Serverless v2 a seguito di un nuovo carico di lavoro o di attività amministrative o di manutenzione.
notifica, serverless	RDS-EVENT-0374	L'istanza database è stata ripresa.	L'istanza database Aurora Serverless v2 è stata ripresa.
notification	RDS-EVENT-0385	La topologia del cluster è stata aggiornata.	Sono state apportate modifiche DNS al cluster di database per l'istanza database. Sono inclusi i casi di aggiunta o eliminazi one di nuove istanze database oppure di failover.
notifica, database globale	RDS-EVENT-0390	Il tentativo di bloccare le scritture per il cluster DB *`cluster_id`* nella regione *`region_id`* è riuscito.	Aurora ha iniziato a bloccare le scritture a livello di archiviazione in preparazione dello switchover o del failover di un database globale Aurora.

Categoria	ID evento RDS	Messaggio	Note
notifica, database globale	RDS-EVENT-0391	Il tentativo di bloccare le scritture per il cluster DB *cluster_id* nella regione è *region_id* scaduto.	Aurora non ha potuto bloccare le scritture a livello di archiviazione in preparazione dello switchover o del failover di un database globale Aurora. Lo switchover o il failover procederanno, ma potrebbe essere necessari o recuperare i dati scritti di recente dallo snapshot del cluster primario originale.
replica di lettura	RDS-EVENT-0045	La replica è stata interrotta.	Questo messaggio viene visualizzato quando si verifica un errore durante la replica. Per determina re il tipo di errore, consulta Risoluzione dei problemi relativi a una replica di lettura MySQL.
replica di lettura	RDS-EVENT-0046	Replica della replica di lettura ripristinata.	Questo messaggio viene visualizzato quando crei per la prima volta una replica di lettura o come messaggio di monitorag gio che conferma il corretto funzionamento della replica. Se il messaggio segue una notifica `RDS-EVENT-0045` , la replica viene ripristinata in seguito a un errore o dopo l'arresto della replica.

Categoria	ID evento RDS	Messaggio	Note
replica di lettura	RDS-EVENT-0057	Lo streaming della replica è stata interrotta.	Nessuno
recupero	RDS-EVENT-0020	È stato avviato il recupero dell'istanza database. La durata del recupero varia in funzione della quantità di dati da recuperare.	Nessuno
recupero	RDS-EVENT-0021	È stato completato il recupero dell'istanza database.	Nessuno
recupero	RDS-EVENT-0023	Richiesta istantanea emergente:. *message*	È stato richiesto un backup manuale, ma in Amazon RDS è in corso la creazione di uno snapshot DB. Invia di nuovo la richiesta quando Amazon RDS avrà completato lo snapshot DB.
recupero	RDS-EVENT-0052	Ripristino dell'istanza Multi-AZ avviato.	La durata del recupero varia in funzione della quantità di dati da recuperare.
recupero	RDS-EVENT-0053	Ripristino dell'istanza Multi-AZ completato. Failover o attivazione in sospeso.	Questo messaggio indica che Amazon RDS ha preparato l'istanza database per avviare un failover sull'istanza secondaria, se necessario.

Categoria	ID evento RDS	Messaggio	Note
recupero	RDS-EVENT-0361	Il recupero dell'istanza database in standby è stato avviato.	L'istanza database in standby viene ricompila ta durante il processo di ripristino. Il processo di ripristino influisce sulle prestazioni del database.
recupero	RDS-EVENT-0362	Il recupero dell'istanza database in standby è stato completato.	L'istanza database in standby viene ricompila ta durante il processo di ripristino. Il processo di ripristino influisce sulle prestazioni del database.
ripristino	RDS-EVENT-0019	Ripristinato da un'istanza DB a. *name name*	L'istanza DB è stata ripristinata da un point-in-time backup.
applicazione di patch di sicurezza	RDS-EVENT-0230	È disponibile un aggiornam ento del sistema per l'istanza database. Per informazioni sull'appl icazione degli aggiornam enti, consulta "Manutenz ione di un'istanza database" nella Guida per l'utente di RDS.	È disponibile una nuova versione secondaria della patch del sistema operativo per l'istanza database. Per informazioni sull'appl icazione degli aggiornam enti, consulta Aggiornam enti del sistema operativo per i cluster Aurora DB.

Categoria	ID evento RDS	Messaggio	Note
manutenzione	RDS-EVENT-0425	Amazon RDS non è in grado di eseguire l’aggiornamento del sistema operativo perché non sono disponibili indirizzi IP nelle sottoreti specificate. Scegliere le sottoreti con indirizzi IP disponibili e riprovare.	Nessuno
manutenzione	RDS-EVENT-0429	Amazon RDS non può eseguire l'aggiornamento del sistema operativo a causa della capacità insufficiente disponibile per il tipo di *type* istanza nella zona di *zone* disponibilità	Nessuno

Categoria	ID evento RDS	Messaggio	Note
manutenzione	RDS-EVENT-0501	Il certificato del server dell'istanza database Amazon RDS richiede la rotazione attraverso un'azione di manutenzione in sospeso.	Il certificato del server dell'istanza DB richiede la rotazione attraverso un'azione di manutenzione in sospeso. Amazon RDS riavvia il database durante questa manutenzione per completare la rotazione dei certificati. Per pianificare questa manutenzione, vai alla scheda Manutenzione e backup e scegli Applica ora o Pianifica per la prossima finestra di manutenzione. Se la modifica non è pianificata, Amazon RDS la applica automaticamente nella finestra di manutenzione alla data di applicazione automatica indicata nell'azione di manutenzione.
manutenzione	RDS-EVENT-0502	Amazon RDS ha pianificato una rotazione dei certificati del server per l'istanza DB durante la prossima finestra di manutenzione. Questa manutenzione richiederà il riavvio del database.	Nessuno

Eventi gruppo di parametri database

La tabella seguente riporta la categoria di eventi e un elenco di eventi applicabili quando il tipo di origine è un gruppo dei parametri database.

Categoria	ID evento RDS	Messaggio	Note
modifica della configurazione	RDS-EVENT-0037	Parametro aggiornato *name* al metodo *value method* with apply.	Nessuno

Eventi gruppo di sicurezza DB

La tabella seguente riporta la categoria di eventi e un elenco di eventi applicabili quando il tipo di origine è un gruppo di sicurezza DB.

> **Note**
>
> I gruppi di sicurezza DB sono risorse per EC2 -Classic. EC2-Classic è stato ritirato il 15 agosto 2022. Se non hai effettuato la migrazione da EC2 -Classic a un VPC, ti consigliamo di migrare il prima possibile. Per ulteriori informazioni, consulta Migrare da EC2 -Classic a un VPC nella Amazon EC2 User Guide e nel blog EC2-Classic Networking is Retiring — Here's How to Prepare.

Categoria	ID evento RDS	Messaggio	Note
modifica della configurazione	RDS-EVENT-0038	Modifica al gruppo di sicurezza applicata.	Nessuno
errore	RDS-EVENT-0039	*user*Revoca dell'autorizzazione come.	Il gruppo di sicurezza di proprietà di *user* non esiste. L'autorizzazione per gruppo di sicurezza è stata revocata perché non è valida.

Eventi gruppo shard database

La tabella seguente riporta la categoria di eventi e un elenco di eventi applicabili quando il tipo di origine è un gruppo di shard del database.

Categoria	ID evento RDS	Messaggio	Note
errore	RDS-EVENT-0324	Processo di caricamento dati non riuscito.	Il processo di caricamento dei dati non è riuscito per il motivo indicato nel messaggio di errore.
manutenzione	RDS-EVENT-0302	L'azione è necessaria per finalizzare un job split shard in sospeso *job_ID* per l'ID del subcluster *number* nel gruppo di shard DB. *name* Per completare l'operazione, invoca il codice SQL: `SELECT rds_aurora.limitless_finalize_split_shard( job_ID);`	Nessuno
manutenzione	RDS-EVENT-0303	Il processo di finalizzazione dello split shard *job_ID* è iniziato per l'ID del subcluster nel gruppo di shard DB. *number name*	Nessuno
manutenzione	RDS-EVENT-0304	Il processo di split shard *job_ID* è stato completato correttamente per l'ID del subcluster nel gruppo di shard DB. *number name* Un nuovo shard con ID di	Nessuno

Categoria	ID evento RDS	Messaggio	Note
		subcluster *number* è stato aggiunto al gruppo di shard DB. *name*	
manutenzione	RDS-EVENT-0305	Il processo di split shard non *job_ID* è riuscito per l'ID *number* del subcluster nel gruppo di shard DB. *name*	Nessuno
manutenzione	RDS-EVENT-0366	Il processo di split shard *job_ID* è stato avviato per l'ID *number* del subcluster nel gruppo di shard DB. *name*	Nessuno
manutenzione	RDS-EVENT-0367	Il processo di aggiunta del router *job_ID* è iniziato nel gruppo di shard DB. *name*	Nessuno
manutenzione	RDS-EVENT-0368	Il processo di aggiunta del router *job_ID* è stato completato correttamente. Un nuovo router con ID di subcluster *number* è stato aggiunto al gruppo di shard DB. *name*	Nessuno
manutenzione	RDS-EVENT-0369	Il processo di aggiunta del router non *job_ID* è riuscito nel gruppo di shard DB. *name*	Nessuno

Categoria	ID evento RDS	Messaggio	Note
manutenzione	RDS-EVENT-0394	Il processo di aggiunta del router *job_ID* è stato annullato nel gruppo di shard DB. *name*	Nessuno
manutenzione	RDS-EVENT-0395	Il lavoro Split Shard *job_ID* è stato annullato nel gruppo di shard DB. *name*	Nessuno
notification	RDS-EVENT-0321	Inizializzazione dell'infrastruttura per il processo di caricamento dei dati.	Nessuno
notification	RDS-EVENT-0322	Il processo di caricamento dei dati è in corso.	Nessuno
notification	RDS-EVENT-0323	Il processo di caricamento dei dati è stato completato.	Nessuno
notification	RDS-EVENT-0325	Annullamento del processo di caricamento dei dati in base alla richiesta del cliente.	Nessuno
notification	RDS-EVENT-0326	Il processo di caricamento dei dati è stato annullato.	Nessuno

Eventi RDS Proxy

La tabella seguente riporta la categoria di eventi e un elenco di eventi applicabili quando il tipo di origine è un proxy RDS.

Categoria	ID evento RDS	Messaggio	Note
modifica della configurazione	RDS-EVENT-0204	Proxy *name* DB modificato da RDS.	Nessuno
modifica della configurazione	RDS-EVENT-0207	RDS ha modificato il punto finale del proxy DB. *name*	Nessuno
modifica della configurazione	RDS-EVENT-0213	RDS ha rilevato l'aggiunta dell'istanza DB e l'ha aggiunta automaticamente al gruppo di destinazione del proxy DB. *name*	Nessuno
modifica della configurazione	RDS-EVENT-0214	RDS ha rilevato l'eliminazione dell'istanza DB *name* e l'ha rimossa automaticamente dal gruppo target *name* del proxy DB. *name*	Nessuno
modifica della configurazione	RDS-EVENT-0215	RDS ha rilevato l'eliminazione del cluster DB *name* e lo ha rimosso automaticamente dal gruppo *name* di destinazione del proxy DB. *name*	Nessuno
creazione	RDS-EVENT-0203	Proxy DB creato da RDS. *name*	Nessuno
creazione	RDS-EVENT-0206	Endpoint creato da RDS *name* per il proxy DB. *name*	Nessuno
eliminazione	RDS-EVENT-0205	RDS ha eliminato il proxy DB. *name*	Nessuno

Categoria	ID evento RDS	Messaggio	Note
eliminazione	RDS-EVENT-0208	RDS ha eliminato l'endpoint *name* per il proxy DB. *name*	Nessuno
errore	RDS-EVENT-0243	RDS non è riuscito a fornire la capacità per il proxy *name* perché non ci sono abbastanza indirizzi IP disponibili nelle sottoreti :. *name* Per risolvere il problema, assicurarsi che le sottoreti abbiano il numero minimo di indirizzi IP non utilizzati come consigliato nella documentazione di Server proxy per Amazon RDS.	Per determinare il numero consigliato per la classe di istanza, consulta Pianificazione della capacità degli indirizzi IP.
errore	RDS-EVENT-0275	RDS ha limitato alcune connessioni al proxy DB. *name* Il numero di richieste di connessione simultane e dal client al proxy ha superato il limite.	Nessuno

Eventi di implementazione blu/verde

La tabella seguente mostra la categoria di eventi e un elenco di eventi quando una blue/green distribuzione è il tipo di origine.

Per ulteriori informazioni sulle blue/green distribuzioni, vedere. Utilizzo di Amazon Blue/Green Aurora Deployments per gli aggiornamenti del database

Categoria	ID evento Amazon RDS	Messaggio	Note
creazione	RDS-EVENT-0244	Le attività di implement azione blu/verde sono state completate. È possibile apportare ulteriori modifiche ai database dell'ambiente verde o passare all'imple mentazione.	Nessuno
errore	RDS-EVENT-0245	La creazione della blue/ green distribuzione non è riuscita perché. *reason*	Nessuno
eliminazione	RDS-EVENT-0246	L'implementazione blu/ verde è stata eliminata.	Nessuno
notification	RDS-EVENT-0247	Passaggio da *blue* a avviato. *green*	Nessuno
notification	RDS-EVENT-0248	Switchover completat o al momento dell'impl ementazione. blue/green	Nessuno
errore	RDS-EVENT-0249	Switchover annullato al momento dell'impl ementazione. blue/green	Nessuno
notification	RDS-EVENT-0259	Passaggio dal cluster DB di replica lettura all'avvio. *blue green*	Nessuno
notification	RDS-EVENT-0260	*blue green blueblue-old*Rinominato *green* in *blue* e in.	Nessuno
errore	RDS-EVENT-0261	Il passaggio dal cluster DB di a è stato annullato	Nessuno

Categoria	ID evento Amazon RDS	Messaggio	Note
		a causa di. *blue green reason*	
notification	RDS-EVENT-0311	È iniziata la sincroniz zazione della sequenza per il passaggio del cluster DB a. *blue green* Quando si utilizzano le sequenze lo switchover può comportare tempi di inattività prolungati.	Nessuno
notification	RDS-EVENT-0312	La sincronizzazione della sequenza per il passaggio del cluster DB a è stata completata. *blue green*	Nessuno
errore	RDS-EVENT-0314	La sincronizzazione della sequenza per il passaggio del cluster DB *blue* a *green* è stata annullata perché le sequenze non sono state sincronizzate.	Nessuno
notification	RDS-EVENT-0409	*message*	Nessuno

Monitoraggio dei file di log di Amazon Aurora

Ogni motore di database RDS genera registri cui è possibile accedere per il controllo e la risoluzione dei problemi. Il tipo di registri dipende dal motore di database.

È possibile accedere ai log del database per le istanze database utilizzando la Console di gestione AWS, AWS Command Line Interface (AWS CLI) o l'API Amazon RDS. Non puoi visualizzare, controllare o scaricare registri delle transazioni.

> **Note**
>
> In alcuni casi, i log contengono dati nascosti. Quindi, Console di gestione AWS potrebbe mostrare i contenuti in un file di log, ma questo potrebbe essere vuoto quando lo scarichi.

Argomenti

- Visualizzazione ed elenco dei file di log del database
- Download di un file di log di database
- Controllo di un file di log di database
- Pubblicazione di log di database su Amazon CloudWatch Logs
- Lettura dei contenuti del file di log con REST
- File di log del database Aurora MySQL
- File di log del database Aurora PostgreSQL

Visualizzazione ed elenco dei file di log del database

Puoi visualizzare i file di log del database per il motore DB Amazon Aurora utilizzando la Console di gestione AWS. Puoi elencare i file di log disponibili per il download o il monitoraggio tramite AWS CLI o l'API di Amazon RDS.

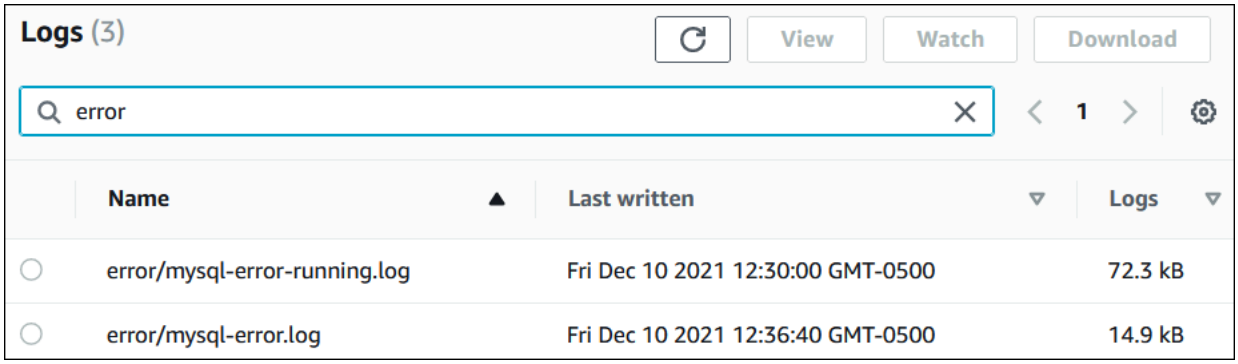
> **Note**
>
> Non è possibile visualizzare i file di registro per cluster database Aurora Serverless v1 nella console RDS. Tuttavia, puoi visualizzarli nella console Amazon CloudWatch all'indirizzo https://console.aws.amazon.com/cloudwatch/.

Console

Per visualizzare un file di log di database

1. Apri la console Amazon RDS all'indirizzo https://console.aws.amazon.com/rds/.
2. Nel pannello di navigazione, scegliere Databases (Database).
3. Scegliere il nome dell'istanza di database che ha il file di log che si desidera visualizzare.
4. Scegliere la scheda Logs & events (Log ed eventi).
5. Scorrere fino alla sezione Logs (Log).
6. (Opzionale) Inserisci un termine di ricerca per filtrare i risultati.

   Nell'esempio seguente vengono elencati i registri filtrati dal testo **`error`**.



7. Scegli il log che desideri visualizzare, quindi seleziona View (Visualizza).

AWS CLI

Per elencare i file di log del database disponibili per un'istanza database, utilizza il comando AWS CLI `describe-db-log-files`.

Il seguente esempio restituisce un elenco di file di log per un'istanza database denominata `my-db-instance`.

Example

```
aws rds describe-db-log-files --db-instance-identifier my-db-instance
```

API RDS

Per elencare i file di log del database disponibili per un'istanza database usa l'operazione API Amazon RDS `DescribeDBLogFiles`.

Download di un file di log di database

Puoi usare la Console di gestione AWS, AWS CLI o l'API per scaricare un file di log del database.

Console

Per scaricare un file di log di database

1. Apri la console Amazon RDS all'indirizzo https://console.aws.amazon.com/rds/.
2. Nel pannello di navigazione, scegliere Databases (Database).
3. Scegliere il nome dell'istanza di database che ha il file di log che si desidera visualizzare.
4. Scegliere la scheda Logs & events (Log ed eventi).
5. Scorrere fino alla sezione Logs (Log).
6. Nella sezione Logs (Log), selezionare il pulsante accanto al log che si desidera scaricare, quindi selezionare Download (Scarica).
7. Aprire il menu contestuale (clic con il tasto destro del mouse) per il collegamento fornito, quindi scegliere Save Link As (Salva collegamento come). Immettere l'ubicazione in cui si intende salvare il file di log, quindi scegliere Save (Salva).

AWS CLI

Per scaricare un file di log del database, utilizzare il comando AWS CLI `download-db-log-file-portion`. Per impostazione predefinita, questo comando scarica la porzione più recente di un file di log. Tuttavia, puoi scaricare un intero file specificando il parametro `--starting-token 0`.

L'esempio seguente mostra come scaricare tutto il contenuto di un file di log denominato log/ERROR.4 e come archiviarlo in un file locale denominato errorlog.txt.

Example

Per Linux, macOS o Unix:

```
aws rds download-db-log-file-portion \
    --db-instance-identifier myexampledb \
    --starting-token 0 --output text \
    --log-file-name log/ERROR.4 > errorlog.txt
```

Per Windows:

```
aws rds download-db-log-file-portion ^
    --db-instance-identifier myexampledb ^
    --starting-token 0 --output text ^
    --log-file-name log/ERROR.4 > errorlog.txt
```

API RDS

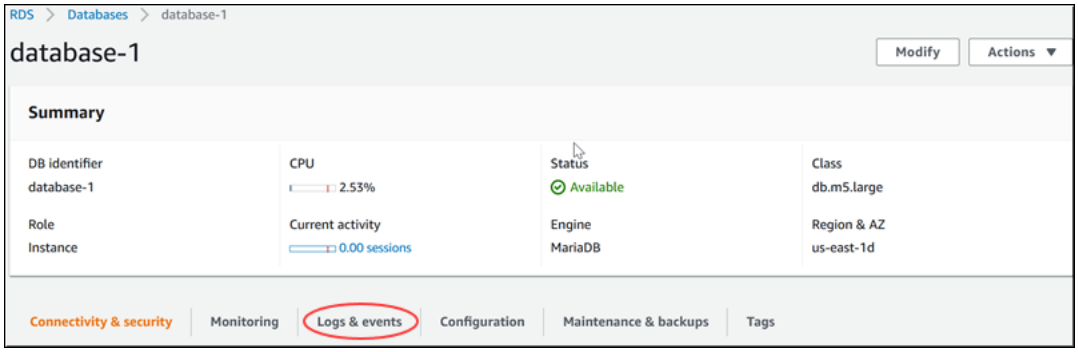
Per scaricare un file di log del database, utilizzare l'operazione API Amazon RDS `DownloadDBLogFilePortion`.

Controllo di un file di log di database

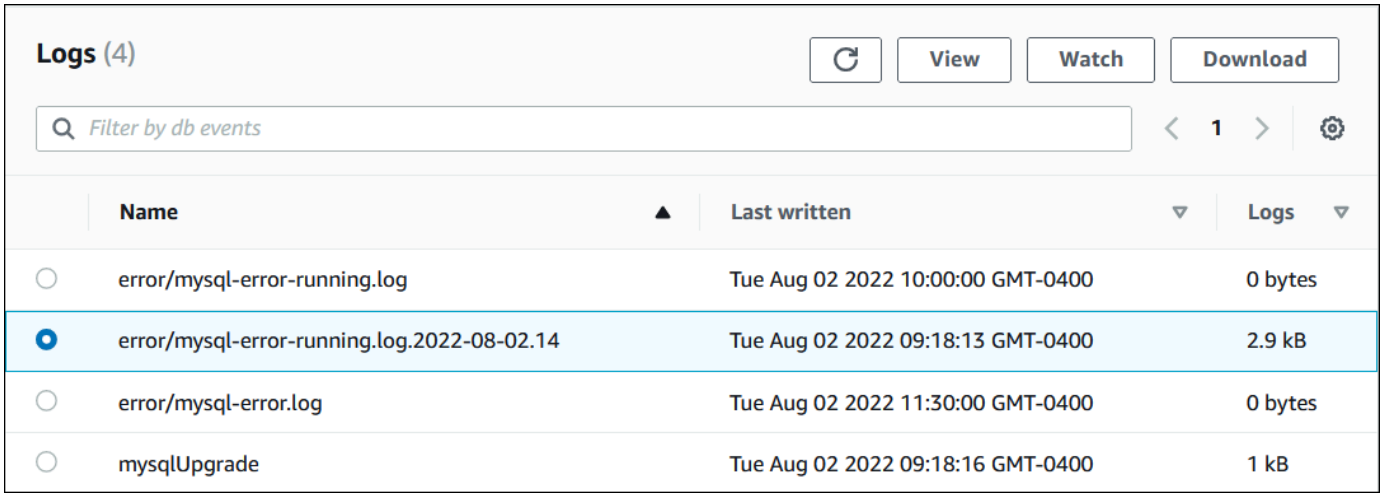
Controllare un file di registro del database equivale a eseguire l'accodamento del file su un sistema UNIX o Linux. Puoi controllare un file di registro usando la Console di gestione AWS. RDS aggiorna la coda del registro ogni 5 secondi.

Per controllare un file di log di database

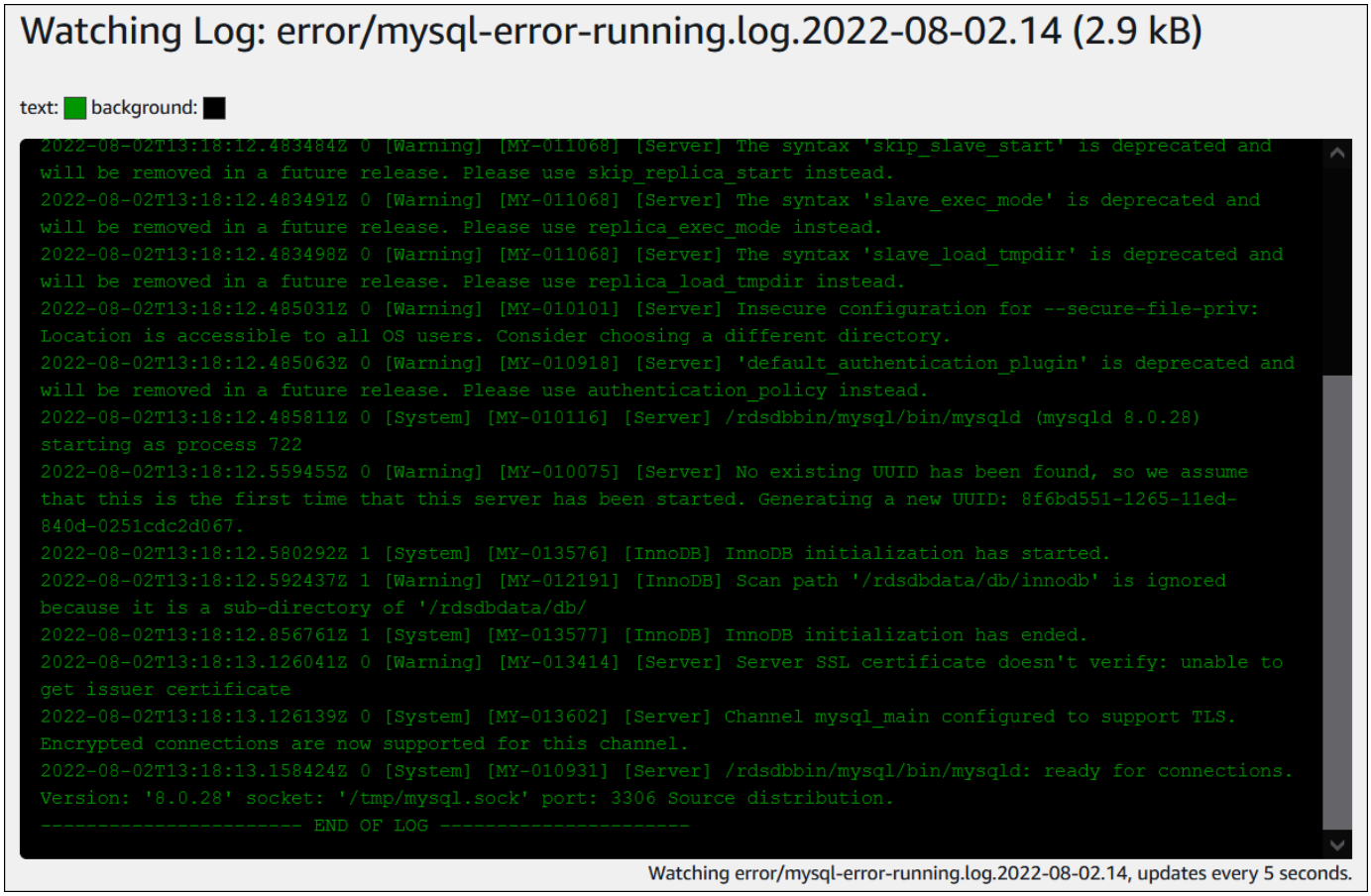
1. Aprire la console Amazon RDS all'indirizzo https://console.aws.amazon.com/rds/.
2. Nel pannello di navigazione, scegliere Databases (Database).
3. Scegliere il nome dell'istanza di database che ha il file di log che si desidera visualizzare.
4. Scegliere la scheda Logs & events (Log ed eventi).



5. Nella sezione Logs (Log), scegliere un file di log, quindi selezionare Watch (Controlla).



RDS mostra la coda del registro, come nel seguente esempio MySQL.



Pubblicazione di log di database su Amazon CloudWatch Logs

In un database on-premise, i registri del database risiedono nel file system. Amazon RDS non fornisce accesso host ai registri del database sul file system del cluster database. Per questo motivo, Amazon RDS consente di esportare i registri del database nei file di log Amazon CloudWatch. Con File di log CloudWatch, puoi eseguire analisi in tempo reale dei dati dei registri. Puoi anche archiviare i dati in un archivio estremamente durevole e gestirli con l'agente File di log CloudWatch.

Argomenti

- Panoramica dell'integrazione RDS con i file di log CloudWatch
- Decidere quali registri pubblicare nei file di log CloudWatch
- Specifica dei registri da pubblicare nei file di log CloudWatch
- Ricerca e filtraggio dei registri nei file di log CloudWatch

Panoramica dell'integrazione RDS con i file di log CloudWatch

Nei file di log CloudWatch, un flusso di log è una sequenza di eventi di log che condividono la stessa origine. Ciascuna origine di log in CloudWatch Logs costituisce un flusso di log distinto. Un gruppo di log è un gruppo di flussi di log che condividono le stesse impostazioni di conservazione, monitoraggio e controllo degli accessi.

Amazon Aurora esegue lo streaming continuo dei record di log del cluster database in un gruppo di log. Ad esempio, disponi di un gruppo di registri `/aws/rds/cluster/`*`cluster_name`*`/`*`log_type`* per ogni tipo di registro che pubblichi. Questo gruppo di log si trova nella stessa regione AWS dell'istanza database che genera il log.

AWS conserva i dati di registro pubblicati nei file di log CloudWatch per un periodo di tempo indefinito a meno che non venga specificato un periodo di conservazione. Per ulteriori informazioni, consulta la pagina relativa alla modifica del periodo di conservazione dei dati dei log in CloudWatch Logs.

Decidere quali registri pubblicare nei file di log CloudWatch

Ogni motore di database RDS supporta il proprio set di registri. Per informazioni sulle opzioni per il motore di database, consulta i seguenti argomenti:

- the section called “Pubblicazione di log Aurora MySQL in CloudWatch Logs”
- the section called “Pubblicazione di log Aurora PostgreSQL in CloudWatch Logs”

Specifica dei registri da pubblicare nei file di log CloudWatch

Puoi specificare quali registri pubblicare nella console. Assicurati di disporre di un ruolo collegato al servizio in AWS Identity and Access Management (IAM). Per ulteriori informazioni sui ruoli collegati al servizio, consulta Utilizzo di ruoli collegati ai servizi per Amazon Aurora.

Per specificare i registri da pubblicare

1. Apri la console Amazon RDS all'indirizzo https://console.aws.amazon.com/rds/.
2. Nel pannello di navigazione, scegliere Databases (Database).
3. Esegui una delle operazioni seguenti:
   - Scegliere Crea database.
   - Scegli un database dall'elenco, quindi scegli Modify (Modifica).

4. In Logs exports (Esportazioni di log), scegli quali registri pubblicare.

   L'esempio seguente specifica il log di audit, i log degli errori, il log generale, il log delle istanze, il log degli errori di autenticazione del database IAM e il log delle query lente per un cluster di database Aurora MySQL.

Ricerca e filtraggio dei registri nei file di log CloudWatch

Puoi cercare voci di registro che soddisfino un criterio specificato utilizzando la console File di log CloudWatch. Puoi accedere ai registri tramite la console RDS, che porta alla console File di log CloudWatch, o direttamente dalla console File di log CloudWatch.

Per cercare registri RDS utilizzando la console RDS

1. Apri la console Amazon RDS all'indirizzo https://console.aws.amazon.com/rds/.
2. Nel pannello di navigazione, scegliere Databases (Database).
3. Scegli un cluster database o un'istanza database.
4. Scegliere Configuration (Configurazione).
5. In Published logs (Log pubblicati), scegli il registro del database che desideri visualizzare.

Per cercare i registri RDS utilizzando la console File di log CloudWatch

1. Apri la console CloudWatch all'indirizzo https://console.aws.amazon.com/cloudwatch/.
2. Nel pannello di navigazione, seleziona Log groups (Gruppi di log).
3. Nella casella del filtro, immetti **`/aws/rds`**.
4. In Log Groups (Gruppi di log), seleziona il nome del gruppo di log contenente il flusso di log da cercare.
5. In Log Streams (Flussi di log), seleziona il nome del flusso di log da cercare.
6. In Eventi di log, immettere la sintassi del filtro da utilizzare.

Per ulteriori informazioni, consulta Ricerca e filtraggio dei dati di registro nella Guida per l'utente di File di log Amazon CloudWatch. Per un blog tutorial su come monitorare i registri RDS, consulta la sezione relativa alla creazione di un monitoraggio proattivo del database per Amazon RDS con File di log Amazon CloudWatch, AWS Lambda e Amazon SNS.

Lettura dei contenuti del file di log con REST

Amazon RDS fornisce un endpoint REST che consente l'accesso ai file di log dell'istanza database. Questo è utile se hai necessità di scrivere un'applicazione per eseguire lo streaming dei contenuti del file di log Amazon RDS.

La sintassi è:

```
GET /v13/downloadCompleteLogFile/DBInstanceIdentifier/LogFileName HTTP/1.1
Content-type: application/json
host: rds.region.amazonaws.com
```

I parametri seguenti sono obbligatori:

- *DBInstanceIdentifier* — Nome dell'istanza database che contiene il file di log che vuoi scaricare.
- *LogFileName* — Nome del file di log da scaricare.

La risposta contiene i contenuti del file di log richiesto, come stream.

L'esempio seguente scarica il file di log denominato log/ERROR.6 per l'istanza database denominata sample-sql nella regione us-west-2.

```
GET /v13/downloadCompleteLogFile/sample-sql/log/ERROR.6 HTTP/1.1
host: rds.us-west-2.amazonaws.com
X-Amz-Security-Token: AQoDYXdzEIH//////////
wEa0AIXLhngC5zp9CyB1R6abwKrXHVR5efnAVN3XvR7IwqKYalFSn6UyJuEFTft9nObglx4QJ+GXV9cpACkETq=
X-Amz-Date: 20140903T233749Z
X-Amz-Algorithm: AWS4-HMAC-SHA256
X-Amz-Credential: AKIADQKE4SARGYLE/20140903/us-west-2/rds/aws4_request
X-Amz-SignedHeaders: host
X-Amz-Content-SHA256: e3b0c44298fc1c229afbf4c8996fb92427ae41e4649b934de495991b7852b855
X-Amz-Expires: 86400
X-Amz-Signature: 353a4f14b3f250142d9afc34f9f9948154d46ce7d4ec091d0cdabbcf8b40c558
```

Se specifichi un'istanza database non esistente, otterrai il seguente errore:

- `DBInstanceNotFound`—*DBInstanceIdentifier* non fa riferimento a un'istanza database esistente. (Codice di stato HTTP: 404)

File di log del database Aurora MySQL

Puoi monitorare i log Aurora MySQL direttamente tramite la console Amazon RDS, l'API di Amazon RDS, AWS CLI o gli SDK AWS. Puoi anche eseguire l'accesso ai log MySQL indirizzando i log a una tabella del database nel database principale e facendo una ricerca in tale tabella. Puoi utilizzare la utility mysqlbinlog per scaricare un log binario.

Per ulteriori informazioni sulla visualizzazione, il download e la visione di log di database basati su file, consulta Monitoraggio dei file di log di Amazon Aurora.

Argomenti

- Panoramica dei registri di database Aurora MySQL
- Invio dell'output log di Aurora MySQL alle tabelle
- Configurazione della registrazione di log binari Aurora MySQL per database Single-AZ
- Accesso ai log binari MySQL

Panoramica dei registri di database Aurora MySQL

Puoi monitorare i seguenti tipi di file di registro Aurora MySQL:

- Log di errori
- Log delle query lente
- Log generale
- Log di audit
- Log delle istanze
- Log degli errori di autenticazione del database IAM

Il registro degli errori Aurora MySQL viene generato per impostazione predefinita. È possibile generare la query lenta e i log generali impostando i parametri nel gruppo di parametri di database.

Argomenti

- Registri degli errori Aurora MySQL
- Registri generali e delle query lente di Aurora MySQL
- Registro di verifica di Aurora MySQL
- Log delle istanze Aurora MySQL

- Rotazione e conservazione dei registri per Autora MySQL
- Pubblicazione di log Aurora MySQL in Amazon CloudWatch Logs

Registri degli errori Aurora MySQL

Aurora MySQL scrive errori nel file `mysql-error.log`. Ogni file di log ha l'ora di creazione (in UTC) accodata al nome. I file di log hanno anche un timestamp che ti aiuta a determinare quando le voci del log sono state scritte.

Aurora MySQL scrive nel registro degli errori solo durante l'avvio, l'arresto e quando si verificano errori. Un'istanza database può andare avanti ore senza che ci siano nuove voci scritte nel file di log degli errori. Se non vedi voci recenti, significa che il server non ha riscontrato errori che generano una voce di registro.

In base alla progettazione, i registri degli errori vengono filtrati in modo da visualizzare solo eventi imprevisti come errori. Tuttavia, i registri degli errori contengono anche altre informazioni sul database, ad esempio l'avanzamento della query, che non vengono visualizzate. Pertanto, anche senza errori effettivi, la dimensione dei registri degli errori potrebbe aumentare a causa delle attività del database in corso. E anche quando presentano una dimensione in byte o kilobyte nella Console di gestione AWS, i log degli errori potrebbero avere 0 byte quando li scarichi.

Aurora MySQL scrive `mysql-error.log` su disco ogni 5 minuti. Aggiunge il contenuto del registro a `mysql-error-running.log`.

Aurora MySQL ruota il file `mysql-error-running.log` ogni ora.

> **Note**
>
> Il periodo di conservazione dei log è diverso tra Amazon RDS e Aurora.

Registri generali e delle query lente di Aurora MySQL

Il registro delle query lente e il registro generale di Aurora MySQL possono essere scritti in un file o una tabella di database impostando i parametri nel gruppo parametri del database. Per informazioni sulla creazione e la modifica di un gruppo di parametri database, consulta Gruppi di parametri per Amazon Aurora. Devi impostare questi parametri prima di poter visualizzare il log delle query lente o il log generale nella console Amazon RDS o tramite l'API di Amazon RDS, la CLI di Amazon RDS o gli SDK AWS.

Puoi controllare la registrazione di Aurora MySQL utilizzando i parametri in questo elenco:

- `slow_query_log`: per creare il log delle query lente, imposta su 1. Il valore predefinito è 0.
- `general_log`: per creare il log generale, imposta su 1. Il valore predefinito è 0.
- `long_query_time`: per evitare che le query a esecuzione rapida vengano registrate nel registro delle query lente, specifica in secondi un valore per il runtime di query più breve da registrare. Il valore predefinito è 10 secondi, il minimo è 0 secondi. Se log_output = FILE, puoi specificare un valore in virgola mobile con risoluzione al microsecondo. Se log_output = TABLE, devi specificare un valore intero con risoluzione al secondo. Vengono registrate solo le query con runtime che supera il valore `long_query_time`. Ad esempio, impostando `long_query_time` su 0,1 si impedisce a tutte le query con tempo di esecuzione inferiore a 100 millisecondi di essere registrate.
- `log_queries_not_using_indexes`: per registrare tutte le query che non usano un indice sul log delle query lente, imposta su 1. Le query che non utilizzano un indice vengono registrate anche se il runtime è inferiore al valore del parametro `long_query_time`. Il valore predefinito è 0.
- `log_output` *`option`*: puoi specificare una delle seguenti opzioni per il parametro `log_output`.
  - TABLE `mysql.general_log` Scrive le query generali nella tabella – e le query lente nella tabella `mysql.slow_log`.
  - FILE – Scrive sia i log generali sia i log delle query lente nel file system.
  - NONE – Disabilita il logging.

  Per Aurora MySQL versione 2 e 3, il valore predefinito di `log_output` è `FILE`.

Per garantire la visualizzazione dei dati delle query lente in Amazon CloudWatch Logs, è necessario che siano soddisfatte le seguenti condizioni:

- CloudWatch Logs deve essere configurato per includere i log delle query lente.
- `slow_query_log` deve essere abilitato.
- `log_output` deve essere impostato su `FILE`.
- La query deve richiedere più tempo di quanto configurato per `long_query_time`.

Per ulteriori informazioni sui log delle query lente e i log generali, consulta i seguenti argomenti nella documentazione di MySQL:

- Log delle query lente
- Log delle query generali

Registro di verifica di Aurora MySQL

La registrazione di controllo per Aurora MySQL è denominata Advanced Auditing. Per attivare Advanced Auditing, imposta alcuni parametri del cluster database. Per ulteriori informazioni, consulta Utilizzo dell'audit avanzato con un cluster di database Amazon Aurora MySQL.

Log delle istanze Aurora MySQL

Aurora crea un file di log separato per le istanze database con la funzionalità di pausa automatica abilitata. Questo file instance.log registra tutti i motivi per cui le istanze database non possono essere messe in pausa quando previsto. Per ulteriori informazioni sul comportamento dei file di log delle istanze e sulla funzionalità di pausa automatica di Aurora, consulta Monitoraggio delle attività di pausa e ripresa di Aurora Serverless v2.

Rotazione e conservazione dei registri per Autora MySQL

Quando la registrazione è abilitata, Amazon Aurora ruota o elimina i file di registro Amazon a intervalli regolari. Questa è una misura preventiva per ridurre l'eventualità che un file di log molto grande comprometta l'uso del database o la performance. Autora MySQL gestisce la rotazione e l'eliminazione come segue:

- Le dimensioni del file di registro degli errori di Aurora MySQL sono limitate a un massimo del 15 per cento dello storage locale per un'istanza database. Per mantenere questa soglia, i log vengono ruotati automaticamente ogni ora. Aurora MySQL rimuove i registri dopo 30 giorni o quando è stato occupato il 15% dello spazio su disco. Se le dimensioni del file di log combinato superano tale soglia dopo la rimozione dei file di log più vecchi, i file di log più grandi vengono eliminati fino a che le dimensioni del file di log non rimangono inferiori alla soglia.
- Aurora MySQL rimuove i log di audit, generali, di query lente dopo 24 ore o quando è stato occupato il 15% dello spazio di archiviazione.
- Quando la registrazione `FILE` è abilitata, i file di registro generale e delle query lente vengono esaminati ogni ora e quelli più vecchi di 24 ore vengono eliminati. In alcuni casi, la dimensione del file di registro combinato restante dopo l'eliminazione supera la soglia del 15 per cento di spazio locale di un'istanza database. In questi casi, i file di log più vecchi vengono eliminati fino a che le dimensioni del file di log non rimangono inferiori alla soglia.
- Quando la registrazione `TABLE` è abilitata, le tabelle di registro non vengono ruotate o eliminate. Le tabelle di registro vengono troncate quando la dimensione di tutti i registri combinati è troppo grande. È possibile abbonarsi alla categoria di eventi `low storage` per ricevere una notifica quando le tabelle di log devono essere ruotate o eliminate manualmente per liberare spazio. Per ulteriori informazioni, consulta Utilizzo della notifica degli eventi di Amazon RDS.

Puoi ruotare la tabella `mysql.general_log` chiamando manualmente la procedura `mysql.rds_rotate_general_log`. Puoi ruotare la tabella `mysql.slow_log` chiamando la procedura `mysql.rds_rotate_slow_log`.

Quando ruoti manualmente le tabelle di registro, la tabella di registro corrente viene copiata in una tabella di registro di backup e le voci nella tabella di registro corrente vengono eliminate. Se esiste già una tabella di log di backup, questa viene eliminata prima che la tabella di log corrente sia copiata nel backup. Puoi eseguire una query sulla tabella di log di backup, se necessario. La tabella di log di backup per la tabella `mysql.general_log` è denominata `mysql.general_log_backup`. La tabella di log di backup per la tabella `mysql.slow_log` è denominata `mysql.slow_log_backup`.

- I registri di controllo Aurora MySQL vengono ruotati quando la dimensione del file raggiunge i 100 MB e rimossi dopo 24 ore.
- Amazon RDS ruota file di log degli errori di autenticazione del database IAM di dimensioni superiori a 10 MB. Amazon RDS rimuove file di log degli errori di autenticazione del database IAM più vecchi di cinque giorni o più grandi di 100 MB.

Per usare i log dalla console Amazon RDS, dall'API di Amazon RDS, dalla CLI di Amazon RDS o dagli SDK AWS, imposta il parametro `log_output` su FILE. Analogamente al registro degli errori di Aurora MySQL, questi file di registro vengono ruotati ogni ora. I file di log che sono stati generati durante le precedenti 24 ore vengono conservati. Il periodo di conservazione è diverso tra Amazon RDS e Aurora.

Pubblicazione di log Aurora MySQL in Amazon CloudWatch Logs

Puoi configurare il cluster database Aurora MySQL per la pubblicazione dei dati di log in un gruppo di log in Amazon CloudWatch Logs. Con CloudWatch Logs puoi eseguire analisi in tempo reale dei dati di log e usare CloudWatch per creare allarmi e visualizzare parametri. Puoi utilizzare CloudWatch Logs per archiviare i record dei log in uno storage estremamente durevole. Per ulteriori informazioni, consulta Pubblicazione di log Amazon Aurora MySQL in Amazon CloudWatch Logs.

Invio dell’output log di Aurora MySQL alle tabelle

Puoi indirizzare il log generale e il log delle query lente alle tabelle sull'istanza database creando un gruppo di parametri del database e impostando il parametro server `log_output` su `TABLE`. Le query generali vengono quindi registrate sulla tabella `mysql.general_log`, mentre le query lente vengono registrate sulla tabella `mysql.slow_log`. Puoi eseguire query sulle tabelle per avere

accesso alle informazioni di log. L'abilitazione di questa registrazione aumenta il numero di dati scritti sul database, il che potrebbe compromettere le performance.

Sia il log generale che quello delle query lente sono disattivati per impostazione predefinita. Per abilitare la registrazione sulle tabelle devi impostare anche i parametri server `general_log` e `slow_query_log` su `1`.

Le tabelle di log continuano a crescere fino a che le rispettive attività di registrazione non vengono disattivate eseguendo la reimpostazione del parametro appropriato su `0`. Spesso nel corso del tempo si accumulano grandi quantità di dati che possono usare una percentuale considerevole dello spazio di archiviazione assegnato. Amazon Aurora non consente di troncare le tabelle dei registri, ma è possibile spostarne il contenuto. La rotazione delle tabelle ne salva il contenuto in una tabella di backup e crea una nuova tabella di log vuota. Puoi ruotare manualmente le tabelle di log con le seguenti procedure a riga di comando, nelle quali il prompt dei comandi è indicato da `PROMPT>`:

```
PROMPT> CALL mysql.rds_rotate_slow_log;
PROMPT> CALL mysql.rds_rotate_general_log;
```

Per rimuovere completamente i dati vecchi e recuperare lo spazio del disco, chiama la procedura adeguata due volte in successione.

Configurazione della registrazione di log binari Aurora MySQL per database Single-AZ

Il log binario è un insieme di file di log che contengono informazioni sulle modifiche apportate ai dati di un'istanza server Aurora MySQL. Il log binario contiene informazioni come le seguenti:

- Eventi che descrivono le modifiche al database come la creazione di tabelle o la modifica di righe
- Informazioni sulla durata di ogni istruzione che ha aggiornato i dati
- Eventi per istruzioni che avrebbero potuto aggiornare i dati ma non l'hanno fatto

Il log binario registra le istruzioni inviate durante la replica. È inoltre necessario per alcune operazioni di ripristino. Per ulteriori informazioni, consulta la pagina relativa al log binario nella documentazione di MySQL.

I registri binari sono accessibili solo dall'istanza database principale, non dalle repliche.

MySQL su Amazon Aurora supporta i formati di logging binario basati su righe, basati su istruzioni e misti. Si consiglia il formato misto a meno che non sia necessario un formato binlog specifico.

Per dettagli sui diversi formati di log binari Aurora MySQL, consulta Binary logging formats nella documentazione MySQL.

Se pianifichi di utilizzare la replica, il formato di logging binario è importante in quanto determina il record delle modifiche dei dati che viene registrato nella sorgente e inviato ai target della replica. Per ulteriori informazioni sui vantaggi e sugli svantaggi dei vari formati di logging binario per la replica, consulta la pagina relativa a vantaggi e svantaggi della replica basata su istruzioni e basata su riga nella documentazione di MySQL.

> **Important**
>
> Con MySQL 8.0.34, MySQL ha reso obsoleto il parametro `binlog_format`. Nelle versioni successive di MySQL, MySQL prevede di rimuovere il parametro e supportare solo la replica basata su righe. Di conseguenza, consigliamo di utilizzare la registrazione dei log basata su righe per le nuove configurazioni di replica MySQL. Per ulteriori informazioni, consulta binlog_format nella documentazione MySQL.
> Le versioni 8.0 e 8.4 di MySQL accettano il parametro `binlog_format`. Quando si utilizza questo parametro, MySQL emette un avviso che indica che è obsoleto. In una futura versione principale, MySQL rimuoverà il parametro `binlog_format`.
> La replica basata sulle istruzioni può causare incoerenze tra il cluster database di origine e una replica di lettura. Per ulteriori informazioni, consulta la pagina relativa alla determinazione delle istruzioni sicure e non sicure nel logging binario nella documentazione MySQL.
> Abilitando la registrazione binaria, aumenta il numero delle operazioni I/O di scrittura sul disco nel cluster di database. Puoi monitorare l'utilizzo degli IOPS con la metrica CloudWatch `VolumeWriteIOPs`.

Per impostare il formato di registrazione binaria MySQL

1. Apri la console Amazon RDS all'indirizzo https://console.aws.amazon.com/rds/.
2. Nel riquadro di navigazione scegliere Parameter groups (Gruppi di parametri).
3. Scegliere il gruppo di parametri del cluster di database associato al cluster di database da modificare.

   Non è consentito modificare un gruppo di parametri predefinito. Se l'cluster database è usata da un gruppo di parametri predefinito, creare un nuovo gruppo di parametri e associarlo all'cluster database.

Per ulteriori informazioni sui gruppi di parametri, consulta Gruppi di parametri per Amazon Aurora.

4. In Actions (Operazioni), scegliere Edit (Modifica).
5. Imposta il parametro `binlog_format` sul formato di registrazione binaria scelto (`ROW`, `STATEMENT` o `MIXED`). Puoi anche utilizzare il valore `OFF` per disattivare la registrazione binaria.

> **Note**
>
> L'impostazione di `binlog_format` su `OFF` nel gruppo di parametri del cluster di database disabilita la variabile di sessione `log_bin`. Questo comportamento disabilita la registrazione dei log binari sul cluster di database Aurora MySQL, che a sua volta reimposta la variabile di sessione `binlog_format` sul valore predefinito `ROW` nel database.

6. Scegliere Salva modifiche per salvare gli aggiornamenti applicati al gruppo di parametri cluster database.

Dopo aver eseguito questi passaggi, è necessario riavviare l'istanza di scrittura nel cluster database per applicare le modifiche. In Aurora MySQL versione 2.09 e precedenti, quando si riavvia l'istanza di scrittura, vengono riavviate anche tutte le istanze di lettura nel cluster database. In Aurora MySQL versione 2.10 e successive, è necessario riavviare tutte le istanze di lettura manualmente. Per ulteriori informazioni, consulta Riavvio di un cluster Amazon Aurora DB o di un'istanza Amazon Aurora DB.

> **Important**
>
> La modifica di un gruppo di parametri cluster di database influisce su tutti i cluster di database che utilizzano tale gruppo di parametri. Se si desidera specificare diversi formati di logging binario per diversi cluster database Aurora MySQL in una regione AWS, i cluster di database devono utilizzare gruppi di parametri del cluster database diversi. Questi gruppi di parametri identificano diversi formati di logging. Assegnare il gruppo di parametri cluster di database appropriato a ciascun cluster di database. Per ulteriori informazioni sui parametri Aurora MySQL, consulta Parametri di configurazione Aurora MySQL.

Accesso ai log binari MySQL

Puoi utilizzare la utility mysqlbinlog per il download o lo streaming di log binari dalle istanze database RDS for MySQL. Il log binario viene scaricato sul computer locale dove è possibile eseguire operazioni come la riproduzione del log tramite utility mysql. Per ulteriori informazioni sull'uso dell'utilità mysqlbinlog, consulta Utilizzo di mysqlbinlog per il backup di file di log binari nella documentazione di MySQL.

Per eseguire la utility mysqlbinlog su un'istanza Amazon RDS usa le seguenti opzioni:

- `--read-from-remote-server`: obbligatorio
- `--host`: il nome DNS dall'endpoint dell'istanza.
- `--port`: la porta utilizzata dall'istanza.
- `--user`: un utente MySQL al quale è stata concessa l'autorizzazione `REPLICATION SLAVE`.
- `--password`: la password dell'utente MySQL oppure ometti un valore di password affinché l'utilità richieda una password.
- `--raw`: scarica il file in formato binario.
- `--result-file`: il file locale per riceve l'output raw.
- `--stop-never`: trasmette in streaming i file di log binari.
- `--verbose`: quando utilizzi il formato binlog `ROW`, includi questa opzione per visualizzare gli eventi di riga come istruzioni pseudo-SQL. Per ulteriori informazioni sull'opzione `--verbose`, consulta Visualizzazione degli eventi di riga di mysqlbinlog nella documentazione di MySQL.
- Specifica il nome di uno o più file di log binari. Per ottenere l'elenco dei log disponibili, utilizza il comando SQL `SHOW BINARY LOGS`.

Per ulteriori informazioni sulle opzioni di mysqlbinlog, consulta Utilità mysqlbinlog per l'elaborazione di file di log binari nella documentazione di MySQL.

Gli esempi seguenti mostrano come utilizzare l'utilità mysqlbinlog.

Per Linux, macOS o Unix:

```
mysqlbinlog \
    --read-from-remote-server \
    --host=MySQLInstance1.cg034hpkmmjt.region.rds.amazonaws.com \
    --port=3306  \
    --user ReplUser \
```

```
    --password \
    --raw \
    --verbose \
    --result-file=/tmp/ \
    binlog.00098
```

Per Windows:

```
mysqlbinlog ^
    --read-from-remote-server ^
    --host=MySQLInstance1.cg034hpkmmjt.region.rds.amazonaws.com ^
    --port=3306  ^
    --user ReplUser ^
    --password ^
    --raw ^
    --verbose ^
    --result-file=/tmp/ ^
    binlog.00098
```

I log binari devono rimanere disponibili sull'istanza database affinché l'utilità mysqlbinlog possa accedervi. Per garantirne la disponibilità, usare la stored procedure mysql.rds_set_configuration e specificare un periodo abbastanza lungo che consenta di scaricare i log. Se questa configurazione non è impostata, Amazon RDS elimina i log binari il prima possibile, causando lacune nei log binari recuperati dall'utilità mysqlbinlog.

L'esempio seguente imposta il periodo di conservazione su 1 giorno.

```
call mysql.rds_set_configuration('binlog retention hours', 24);
```

Per visualizzare l'impostazione attuale, utilizza la procedura archiviata mysql.rds_show_configuration.

```
call mysql.rds_show_configuration;
```

File di log del database Aurora PostgreSQL

È possibile monitorare i seguenti tipi di file di log Aurora PostgreSQL:

- Log di PostgreSQL
- Log delle istanze
- Log degli errori di autenticazione del database IAM

> **Note**
>
> Per abilitare i log degli errori di autenticazione del database IAM, è necessario prima abilitare l'autenticazione del database IAM per il cluster di database Aurora PostgreSQL. Per ulteriori informazioni sull'abilitazione database IAM, consulta Abilitazione e disabilitazione dell'autenticazione database IAM.

Aurora PostgreSQL registra le attività del database nel file di log PostgreSQL predefinito. Per un'istanza database PostgreSQL on-premise, questi messaggi vengono archiviati localmente in `log/postgresql.log`. Per un cluster database Aurora PostgreSQL, il file di log è disponibile nel cluster Aurora. Questi log sono accessibili anche tramite la Console di gestione AWS, in cui è possibile visualizzarli o scaricarli. Il livello di registrazione predefinito rileva gli errori di accesso, gli errori irreversibili del server, i deadlock e gli errori delle query.

Per ulteriori informazioni su come visualizzare, scaricare e guardare i registri di database basati su file, consulta Monitoraggio dei file di log di Amazon Aurora. Per ulteriori informazioni sui registri PostgreSQL, consulta Working with Amazon RDS and Aurora PostgreSQL logs: Part 1 (Utilizzo dei registri RDS e Aurora PostgreSQL: parte 1) e Working with Amazon RDS and Aurora PostgreSQL logs: Part 2 (Utilizzo dei registri RDS e Aurora PostgreSQL: parte 2).

Oltre ai log PostgreSQL standard trattati in questo argomento, Aurora PostgreSQL supporta anche l'estensione di audit PostgreSQL (`pgAudit`). La maggior parte dei settori regolamentati e degli enti governativi deve mantenere un log di audit o un audit trail delle modifiche apportate ai dati per conformità ai requisiti legali. Per informazioni sull'installazione e sull'utilizzo di pgAudit, consulta Utilizzo di pgAudit per registrare l'attività del database.

Aurora crea un file di log separato per le istanze database con la funzionalità di pausa automatica abilitata. Questo file instance.log registra tutti i motivi per cui le istanze database non possono essere messe in pausa quando previsto. Per ulteriori informazioni sul comportamento dei file di log delle

istanze e sulla funzionalità di pausa automatica di Aurora, consulta Monitoraggio delle attività di pausa e ripresa di Aurora Serverless v2.

Argomenti

- Parametri per la registrazione di log in Aurora PostgreSQL
- Attivazione della registrazione delle query per il cluster database Aurora PostgreSQL

Parametri per la registrazione di log in Aurora PostgreSQL

È possibile personalizzare il comportamento di registrazione per il cluster database Aurora PostgreSQL modificando vari parametri. Nella tabella seguente sono riportati, tra le altre impostazioni, i parametri che stabiliscono la durata di archiviazione dei log, quando ruotarli e se l'output del log è in formato CSV (valori separati da virgole). Puoi anche trovare l'output di testo inviato a STDERR, tra le altre impostazioni. Per modificare le impostazioni per i parametri modificabili, utilizza un gruppo di parametri del·cluster database personalizzato per il cluster database Aurora PostgreSQL. Per ulteriori informazioni, consulta Gruppi di parametri per Amazon Aurora.

Parametro	Predefinita	Descrizione
log_destination	stderr	Imposta il formato di output per il registro. L'impostazione predefinita è `stderr`, ma puoi anche specificare il formato CSV aggiungendo `csvlog` all'impostazione. Per ulteriori informazioni, consulta Impostazione della destinazione del registro (`stderr`, `csvlog`).
log_filename	postgresql.log.%Y-%m-%d-%H%M	Specifica il modello per il nome del file di log. Oltre al valore predefinito, questo parametro supporta `postgresql.log.%Y-%m-%d` e `postgresql.log.%Y-%m-%d-%H` per il modello del nome del file. Per Aurora PostgreSQL versione 17.4 e successive, non è possibile modificare questo parametro.
log_line_prefix	%t:%r:%u@%d:[%p]:	Definisce il prefisso per ogni riga di log che viene scritta in `stderr`, per annotare l'ora (%t),

Parametro	Predefinita	Descrizione
		l'host remoto (%r), l'utente (%u), il database (%d) e l'ID del processo (%p).
log_rotation_age	60	I minuti dopo i quali il file di log viene ruotato automaticamente. È possibile modificarli con un valore compreso tra 1 e 1440 minuti. Per ulteriori informazioni, consulta Impostazione della rotazione dei file di log.
log_rotation_size	–	La dimensione (KB) che stabilisce la rotazione automatica del log. È possibile modificare impostando un valore compreso nell'intervallo compreso tra 50.000 e 1.000.000 di kilobyte. Per ulteriori informazioni, consulta Impostazione della rotazione dei file di log.
rds.log_retention_period	4320	I registri PostgreSQL più vecchi del numero di minuti specificato vengono eliminati. Il valore di default di 4.320 minuti elimina i file di log dopo 3 giorni. Per ulteriori informazioni, consulta Impostazione del periodo di retention dei log.

Per identificare i problemi dell'applicazione, puoi cercare fallimenti di query, errori di accesso, deadlock ed errori irreversibili del server nel registro. Ad esempio, supponi di convertire un'applicazione legacy da Oracle ad Aurora PostgreSQL, ma non tutte le query sono state convertite correttamente. Queste query formattate in modo errato generano messaggi di errore nei registri che puoi utilizzare per identificare i problemi. Per ulteriori informazioni sulla registrazione delle query, consulta Attivazione della registrazione delle query per il cluster database Aurora PostgreSQL.

Negli argomenti seguenti sono disponibili informazioni su come impostare vari parametri che controllano i dettagli di base dei log PostgreSQL.

Argomenti

- Impostazione del periodo di retention dei log
- Impostazione della rotazione dei file di log

- Impostazione della destinazione del registro (stderr, csvlog)
- Informazioni sul parametro log_line_prefix

Impostazione del periodo di retention dei log

Il parametro `rds.log_retention_period` specifica per quanto tempo il cluster database Aurora PostgreSQL conserva i file di log. L'impostazione predefinita è 3 giorni (4.320 minuti), ma è possibile impostare qualsiasi valore compreso tra 1 giorno (1.440 minuti) e 7 giorni (10.080 minuti). Assicurati che il cluster database Aurora PostgreSQL abbia spazio di archiviazione sufficiente per contenere i file di log per il periodo di tempo specificato.

Ti consigliamo di pubblicare regolarmente i tuoi log su File di log Amazon CloudWatch, in modo da poter visualizzare e analizzare i dati di sistema anche molto tempo dopo che i log sono stati rimossi dal cluster database Aurora PostgreSQL. Per ulteriori informazioni, consulta Pubblicazione di log Aurora PostgreSQL in Amazon CloudWatch Logs. Dopo aver impostato la pubblicazione su CloudWatch, Aurora non elimina un log finché non viene pubblicato in CloudWatch Logs.

Amazon Aurora comprime i log PostgreSQL meno recenti quando lo spazio di archiviazione per l'istanza database raggiunge una determinata soglia. Aurora comprime i file utilizzando l'utilità di compressione gzip. Per ulteriori informazioni, consulta il sito web di gzip.

Quando lo spazio di archiviazione per l'istanza database si riduce e tutti i log disponibili sono compressi, si riceve un avviso del tipo seguente:

```
Warning: local storage for PostgreSQL log files is critically low for
this Aurora PostgreSQL instance, and could lead to a database outage.
```

Se lo spazio di archiviazione non è sufficiente, Aurora potrebbe eliminare i log PostgreSQL compressi prima della fine del periodo di conservazione specificato. Se ciò accade, viene visualizzato un messaggio simile al seguente:

```
The oldest PostgreSQL log files were deleted due to local storage constraints.
```

Impostazione della rotazione dei file di log

Per impostazione predefinita, nuovi file di log vengono creati da Aurora ogni ora. La tempistica è controllata dal parametro `log_rotation_age`. Questo parametro ha un valore predefinito di 60 (minuti), ma è possibile impostarlo su qualsiasi valore tra 1 minuto e 24 ore (1.440 minuti). Al

momento della rotazione, viene creato un nuovo file di log distinto. Il file è denominato in base al modello specificato dal parametro `log_filename`.

I file di log possono anche essere ruotati in base alle loro dimensioni, come specificato dal parametro `log_rotation_size`. Questo parametro specifica che il log deve essere ruotato quando raggiunge la dimensione specificata (in kilobyte). Il valore di timeout predefinito per `log_rotation_size` è di 100.000 kB (kilobyte) per un cluster di database Aurora PostgreSQL, ma puoi impostare questo parametro a qualsiasi valore compreso tra 50.000 e 1.000.000 kilobyte.

I nomi dei file di registro si basano sul modello di nome di file specificato nel parametro `log_filename`. Le impostazioni disponibili per questo parametro sono le seguenti:

- `postgresql.log.%Y-%m-%d` : formato predefinito per il nome del file di registro. Include l'anno, il mese e la data nel nome del file di log.
- `postgresql.log.%Y-%m-%d-%H`: include l'ora nel formato del nome del file di registro.
- `postgresql.log.%Y-%m-%d-%H%M`: include ora:minuto nel formato del nome del file di log.

Se imposti il parametro `log_rotation_age` su un valore inferiore a 60 minuti, imposta il parametro `log_filename` sul formato minuti.

Per ulteriori informazioni, consulta `log_rotation_age` e `log_rotation_size` nella documentazione di PostgreSQL.

Impostazione della destinazione del registro (**`stderr`**, **`csvlog`**)

Per impostazione predefinita, Aurora PostgreSQL genera i log in formato errore standard (stderr). Questo formato è l'impostazione predefinita per il parametro `log_destination`. Ogni messaggio ha un prefisso che utilizza il modello specificato nel parametro `log_line_prefix`. Per ulteriori informazioni, consulta Informazioni sul parametro log_line_prefix.

Aurora PostgreSQL può anche generare log in formato `csvlog`. Il formato `csvlog` è utile per analizzare i dati dei registri in formato CSV. Ad esempio, supponi di utilizzare l'estensione `log_fdw` per lavorare con i log come tabelle esterne. La tabella esterna creata sui file di log di `stderr` contiene una singola colonna con i dati degli eventi di log. Aggiungendo `csvlog` al parametro `log_destination`, ottieni il file di log in formato CSV con le demarcazioni per le diverse colonne della tabella esterna. In tal modo puoi ordinare e analizzare i log più facilmente.

Se specifichi `csvlog` per questo parametro, tieni presente che vengono generati entrambi i file `stderr` e `csvlog`. Ti consigliamo di monitorare lo spazio di archiviazione consumato

dai registri tenendo conto di `rds.log_retention_period` e delle altre impostazioni che influiscono sull'archiviazione e sulla rotazione dei registri. Utilizzando `stderr` e `csvlog` lo spazio di archiviazione consumato dai registri aumenta più del doppio.

Se aggiungi `csvlog` a `log_destination` e vuoi ripristinare solo `stderr`, devi reimpostare il parametro. Per farlo, nella console Amazon RDS apri il gruppo di parametri del·cluster database personalizzato per la tua istanza. Scegli il parametro `log_destination`, seleziona Edit parameter (Modifica parametro), quindi Reset (Reimposta).

Per ulteriori informazioni sulla configurazione dei registri, consulta Utilizzo dei log Amazon RDS e Aurora PostgreSQL: Parte 1.

Informazioni sul parametro log_line_prefix

Il formato di log `stderr` applica il prefisso a ogni messaggio di log con i dettagli specificati dal parametro `log_line_prefix`. Il valore predefinito è:

```
%t:%r:%u@%d:[%p]:t
```

A partire da Aurora PostgreSQL versione 16, è anche possibile scegliere:

```
%m:%r:%u@%d:[%p]:%l:%e:%s:%v:%x:%c:%q%a
```

Ogni voce del log inviata a stderr include le seguenti informazioni in base al valore selezionato:

- `%t` - Ora della voce di log senza millisecondi
- `%m` - Ora della voce di log con millisecondi
- `%r` - Indirizzo dell'host remoto
- `%u@%d` - Nome utente @ nome del database
- `[%p]` - ID del processo, se disponibile
- `%l` - Numero di riga di log per sessione
- `%e` - Codice di errore SQL
- `%s` - Timestamp di inizio del processo
- `%v` - ID della transazione virtuale
- `%x` - ID della transazione
- `%c` - ID della sessione

- %q - Terminatore non di sessione
- %a - Nome dell'applicazione

Attivazione della registrazione delle query per il cluster database Aurora PostgreSQL

È possibile raccogliere informazioni più approfondite sulle attività dei database, tra cui query, query in attesa di blocchi, checkpoint e molti altri dettagli impostando alcuni parametri elencati nella tabella seguente. Questo argomento illustra la registrazione delle query.

Parametro	Predefinita	Descrizione
log_connections	–	Registra ogni connessione riuscita. Per informazioni su come utilizzare questo parametro con `log_disconnections` per rilevare l'abbandono della connessio ne, consulta Gestione dell'abbandono della connessione Aurora PostgreSQL con pooling .
log_disconnections	–	Registra il momento in cui termina ciascuna sessione e la relativa durata. Per informazi oni su come utilizzare questo parametro con `log_connections` per rilevare l'abbandono della connessione, consulta Gestione dell'abba ndono della connessione Aurora PostgreSQL con pooling .
log_checkpoints	–	Non applicabile per Aurora PostgreSQL
log_lock_waits	–	Registra lunghe attese di lock. Per impostazi one predefinita, questo parametro non è impostato.
log_min_duration_s ample	–	Imposta il tempo (ms) minimo di esecuzione oltre il quale viene registrato un campione di istruzioni. La dimensione del campione viene impostata utilizzando il parametro `log_state ment_sample_rate` .

Parametro	Predefinita	Descrizione
log_min_duration_statement	–	Viene registrata qualsiasi istruzione SQL che viene eseguita per il periodo specificato o per più tempo. Per impostazione predefinita, questo parametro non è impostato. L'attivazione di questo parametro può aiutarti a trovare query non ottimizzate.
log_statement	–	Imposta il tipo di istruzioni registrate. Per impostazione predefinita, questo parametro non è impostato, ma puoi modificarlo in `all`, `ddl` o `mod` per specificare i tipi di istruzioni SQL che vuoi registrare. Se specifichi un valore diverso da `none` per questo parametro, dovrai adottare ulteriori misure per evitare l'esposizione delle password nei file di log. Per ulteriori informazioni, consulta Riduzione del rischio di esposizione delle password quando si utilizza la registrazione delle query.
log_statement_sample_rate	–	La percentuale di istruzioni che superano il tempo specificato in `log_min_duration_sample` da registrare, espressa come valore in virgola mobile compreso tra 0,0 e 1,0.
log_statement_stats	–	Scrive le statistiche cumulative sulla prestazione nel registro del server.

Utilizzo della registrazione per trovare query lente

È possibile registrare istruzioni e query SQL per trovare le query con prestazioni lente. Puoi attivare questa funzionalità modificando le impostazioni nei parametri `log_statement` e `log_min_duration` come descritto in questa sezione. Prima di attivare la registrazione delle query per il cluster database Aurora PostgreSQL, è necessario essere consapevoli della possibile esposizione delle password nei registri e di come mitigare i rischi. Per ulteriori informazioni, consulta Riduzione del rischio di esposizione delle password quando si utilizza la registrazione delle query.

Di seguito sono disponibili informazioni di riferimento sui parametri `log_statement` e `log_min_duration`.

log_statement

Questo parametro specifica il tipo di istruzioni SQL che devono essere inviate al registro. Il valore predefinito è `none`. Se modifichi questo parametro in `all`, `ddl` o `mod`, esegui le azioni consigliate per ridurre il rischio di esporre le password nei log. Per ulteriori informazioni, consulta Riduzione del rischio di esposizione delle password quando si utilizza la registrazione delle query.

tutto

Registra tutte le istruzioni. Questa impostazione è consigliata per il debug.

ddl

Registra tutte le istruzioni DDL (Data Definition Language), come CREATE, ALTER, DROP e così via.

mod

Registra tutte le istruzioni DDL e DML (Data Manipulation Language), come INSERT, UPDATE e DELETE, che modificano i dati.

nessuno

Nessuna istruzione SQL viene registrata. Consigliamo questa impostazione per evitare il rischio di esporre le password nei registri.

log_min_duration_statement

Viene registrata qualsiasi istruzione SQL che viene eseguita per il periodo specificato o per più tempo. Per impostazione predefinita, questo parametro non è impostato. L'attivazione di questo parametro può aiutarti a trovare query non ottimizzate.

–1–2147483647

Il numero di millisecondi (ms) di runtime durante il quale un'istruzione viene registrata.

Per configurare la registrazione delle query

Questi passaggi presuppongono che il cluster database Aurora PostgreSQL utilizzi un gruppo di parametri cluster database personalizzato.

1. Imposta il parametro `log_statement` su `all`. L'esempio seguente mostra le informazioni scritte nel file `postgresql.log` con questa impostazione del parametro.

```
2022-10-05 22:05:52 UTC:52.95.4.1(11335):postgres@labdb:[3639]:LOG: statement:
 SELECT feedback, s.sentiment,s.confidence
FROM support,aws_comprehend.detect_sentiment(feedback, 'en') s
ORDER BY s.confidence DESC;
2022-10-05 22:05:52 UTC:52.95.4.1(11335):postgres@labdb:[3639]:LOG: QUERY
 STATISTICS
2022-10-05 22:05:52 UTC:52.95.4.1(11335):postgres@labdb:[3639]:DETAIL: ! system
 usage stats:
! 0.017355 s user, 0.000000 s system, 0.168593 s elapsed
! [0.025146 s user, 0.000000 s system total]
! 36644 kB max resident size
! 0/8 [0/8] filesystem blocks in/out
! 0/733 [0/1364] page faults/reclaims, 0 [0] swaps
! 0 [0] signals rcvd, 0/0 [0/0] messages rcvd/sent
! 19/0 [27/0] voluntary/involuntary context switches
2022-10-05 22:05:52 UTC:52.95.4.1(11335):postgres@labdb:[3639]:STATEMENT: SELECT
 feedback, s.sentiment,s.confidence
FROM support,aws_comprehend.detect_sentiment(feedback, 'en') s
ORDER BY s.confidence DESC;
2022-10-05 22:05:56 UTC:52.95.4.1(11335):postgres@labdb:[3639]:ERROR: syntax error
 at or near "ORDER" at character 1
2022-10-05 22:05:56 UTC:52.95.4.1(11335):postgres@labdb:[3639]:STATEMENT: ORDER BY
 s.confidence DESC;
---------------------- END OF LOG ----------------------
```

2. Impostare il parametro `log_min_duration_statement`. L'esempio seguente mostra le informazioni scritte nel file `postgresql.log` quando il parametro è impostato su `1`.

   Le query che superano la durata specificata nel parametro `log_min_duration_statement` vengono registrate. Di seguito viene riportato un esempio. Puoi visualizzare il file di log per il cluster database Aurora PostgreSQL nella console Amazon RDS.

```
2022-10-05 19:05:19 UTC:52.95.4.1(6461):postgres@labdb:[6144]:LOG: statement: DROP
 table comments;
2022-10-05 19:05:19 UTC:52.95.4.1(6461):postgres@labdb:[6144]:LOG: duration:
 167.754 ms
2022-10-05 19:08:07 UTC::@:[355]:LOG: checkpoint starting: time
2022-10-05 19:08:08 UTC::@:[355]:LOG: checkpoint complete: wrote 11 buffers
 (0.0%); 0 WAL file(s) added, 0 removed, 0 recycled; write=1.013 s, sync=0.006 s,
```

```
 total=1.033 s; sync files=8, longest=0.004 s, average=0.001 s; distance=131028 kB,
 estimate=131028 kB
----------------------- END OF LOG ----------------------
```

Riduzione del rischio di esposizione delle password quando si utilizza la registrazione delle query

Ti consigliamo di mantenere `log_statement` impostato su `none` per evitare di esporre le password. Se imposti `log_statement` su `all`, `ddl` o `mod`, ti consigliamo di eseguire una o più delle seguenti operazioni.

- Per il client, applica la crittografia delle informazioni sensibili. Per ulteriori informazioni, consulta Encryption Options (Opzioni di crittografia) nella documentazione di PostgreSQL. Usa le opzioni `ENCRYPTED` (e `UNENCRYPTED`) delle istruzioni `CREATE` e `ALTER`. Per ulteriori informazioni, consulta CREATE USER nella documentazione di PostgreSQL.
- Per il cluster database Aurora PostgreSQL, configura e usa l'estensione di audit PostgreSQL (pgAudit). Questa estensione oscura le informazioni sensibili nelle istruzioni CREATE e ALTER inviate al registro. Per ulteriori informazioni, consulta Utilizzo di pgAudit per registrare l'attività del database.
- Limita l'accesso ai log CloudWatch.
- Utilizza meccanismi di autenticazione più efficaci come IAM.

Monitoraggio delle chiamate API di Amazon Aurora in AWS CloudTrail

AWS CloudTrail è un servizio AWS che ti aiuta a controllare il tuo account AWS. AWS CloudTrail è attivato sul tuo account AWS quando lo crei. Per ulteriori informazioni su CloudTrail, consulta la AWS CloudTrailGuida per l'utente di .

Argomenti

- Integrazione di CloudTrail con Amazon Aurora
- Voci del file di log Amazon Aurora

Integrazione di CloudTrail con Amazon Aurora

Tutte le operazioni Amazon Aurora sono registrate da CloudTrail. CloudTrail fornisce un record delle operazioni eseguite da un utente, un ruolo o un servizio AWS in Amazon Aurora.

Eventi CloudTrail

CloudTrail acquisisce le chiamate API per Amazon Aurora come eventi. Un evento rappresenta una singola richiesta da un'origine e include informazioni sull'operazione richiesta, data e ora dell'operazione, parametri della richiesta e così via. Gli eventi includono le chiamate della console Amazon RDS e le chiamate del codice alle operazioni API Amazon RDS.

L'attività Amazon Aurora viene registrata in un evento CloudTrail nella cronologia eventi. Puoi utilizzare la console CloudTrail per visualizzare gli ultimi 90 giorni di attività API ed eventi registrati in una regione AWS. Per ulteriori informazioni, consulta Visualizzazione di eventi mediante la cronologia eventi di CloudTrail.

Trail CloudTrail

Per una registrazione continua degli eventi nell'account AWS che includa gli eventi per Amazon Aurora, crea un percorso. Un percorso è una configurazione che consente la consegna di eventi a un bucket Simple Storage Service (Amazon S3) specificato. CloudTrail in genere consegna i file di log entro 15 minuti dall'attività dell'account.

> **Note**
>
> Se non configuri un trail, è comunque possibile visualizzare gli eventi più recenti nella console di CloudTrail in Event history (Cronologia eventi).

È possibile creare due tipi di trail per un account AWS: un trail che si applica a tutte le regioni o un trail che si applica a una regione. Per impostazione predefinita, quando si crea un trail nella console, il trail sarà valido in tutte le Regioni .

Inoltre, è possibile configurare altri servizi AWS per analizzare con maggiore dettaglio e usare i dati evento raccolti nei log CloudTrail. Per ulteriori informazioni, consulta:

- Panoramica della creazione di un percorso
- Servizi e integrazioni CloudTrail supportati
- Configurazione delle notifiche Amazon SNS per CloudTrail
- Ricezione di file di log CloudTrail da più Regioni e Ricezione di file di log CloudTrail da più account

Voci del file di log Amazon Aurora

I file di log di CloudTrail possono contenere una o più voci di log. I file di log di CloudTrail non sono una traccia stack ordinata delle chiamate pubbliche dell'API, quindi non vengono visualizzati in un ordine specifico.

L'esempio seguente mostra una voce di log di CloudTrail che illustra l'operazione `CreateDBInstance`.

```
{
    "eventVersion": "1.04",
    "userIdentity": {
        "type": "IAMUser",
        "principalId": "AKIAIOSFODNN7EXAMPLE",
        "arn": "arn:aws:iam::123456789012:user/johndoe",
        "accountId": "123456789012",
        "accessKeyId": "AKIAI44QH8DHBEXAMPLE",
        "userName": "johndoe"
    },
    "eventTime": "2018-07-30T22:14:06Z",
```

```
    "eventSource": "rds.amazonaws.com",
    "eventName": "CreateDBInstance",
    "awsRegion": "us-east-1",
    "sourceIPAddress": "192.0.2.0",
    "userAgent": "aws-cli/1.15.42 Python/3.6.1 Darwin/17.7.0 botocore/1.10.42",
    "requestParameters": {
        "enableCloudwatchLogsExports": [
            "audit",
            "error",
            "general",
            "slowquery"
        ],
        "dBInstanceIdentifier": "test-instance",
        "engine": "mysql",
        "masterUsername": "myawsuser",
        "allocatedStorage": 20,
        "dBInstanceClass": "db.m1.small",
        "masterUserPassword": "****"
    },
    "responseElements": {
        "dBInstanceArn": "arn:aws:rds:us-east-1:123456789012:db:test-instance",
        "storageEncrypted": false,
        "preferredBackupWindow": "10:27-10:57",
        "preferredMaintenanceWindow": "sat:05:47-sat:06:17",
        "backupRetentionPeriod": 1,
        "allocatedStorage": 20,
        "storageType": "standard",
        "engineVersion": "8.0.28",
        "dbInstancePort": 0,
        "optionGroupMemberships": [
            {
                "status": "in-sync",
                "optionGroupName": "default:mysql-8-0"
            }
        ],
        "dBParameterGroups": [
            {
                "dBParameterGroupName": "default.mysql8.0",
                "parameterApplyStatus": "in-sync"
            }
        ],
        "monitoringInterval": 0,
        "dBInstanceClass": "db.m1.small",
        "readReplicaDBInstanceIdentifiers": [],
```

```
        "dBSubnetGroup": {
            "dBSubnetGroupName": "default",
            "dBSubnetGroupDescription": "default",
            "subnets": [
                {
                    "subnetAvailabilityZone": {"name": "us-east-1b"},
                    "subnetIdentifier": "subnet-cbfff283",
                    "subnetStatus": "Active"
                },
                {
                    "subnetAvailabilityZone": {"name": "us-east-1e"},
                    "subnetIdentifier": "subnet-d7c825e8",
                    "subnetStatus": "Active"
                },
                {
                    "subnetAvailabilityZone": {"name": "us-east-1f"},
                    "subnetIdentifier": "subnet-6746046b",
                    "subnetStatus": "Active"
                },
                {
                    "subnetAvailabilityZone": {"name": "us-east-1c"},
                    "subnetIdentifier": "subnet-bac383e0",
                    "subnetStatus": "Active"
                },
                {
                    "subnetAvailabilityZone": {"name": "us-east-1d"},
                    "subnetIdentifier": "subnet-42599426",
                    "subnetStatus": "Active"
                },
                {
                    "subnetAvailabilityZone": {"name": "us-east-1a"},
                    "subnetIdentifier": "subnet-da327bf6",
                    "subnetStatus": "Active"
                }
            ],
            "vpcId": "vpc-136a4c6a",
            "subnetGroupStatus": "Complete"
        },
        "masterUsername": "myawsuser",
        "multiAZ": false,
        "autoMinorVersionUpgrade": true,
        "engine": "mysql",
        "cACertificateIdentifier": "rds-ca-2015",
        "dbiResourceId": "db-ETDZIIXHEWY5N7GXVC4SH7H5IA",
```

```
        "dBSecurityGroups": [],
        "pendingModifiedValues": {
            "masterUserPassword": "****",
            "pendingCloudwatchLogsExports": {
                "logTypesToEnable": [
                    "audit",
                    "error",
                    "general",
                    "slowquery"
                ]
            }
        },
        "dBInstanceStatus": "creating",
        "publiclyAccessible": true,
        "domainMemberships": [],
        "copyTagsToSnapshot": false,
        "dBInstanceIdentifier": "test-instance",
        "licenseModel": "general-public-license",
        "iAMDatabaseAuthenticationEnabled": false,
        "performanceInsightsEnabled": false,
        "vpcSecurityGroups": [
            {
                "status": "active",
                "vpcSecurityGroupId": "sg-f839b688"
            }
        ]
    },
    "requestID": "daf2e3f5-96a3-4df7-a026-863f96db793e",
    "eventID": "797163d3-5726-441d-80a7-6eeb7464acd4",
    "eventType": "AwsApiCall",
    "recipientAccountId": "123456789012"
}
```


Come illustrato nell'elemento `userIdentity` nell'esempio precedente, ogni voce di evento o di registro contiene informazioni su chi ha generato la richiesta. Le informazioni di identità consentono di determinare quanto segue:

- Se la richiesta è stata effettuata con le credenziali dell'utente IAM o root.
- Se la richiesta è stata effettuata con le credenziali di sicurezza temporanee per un ruolo o un utente federato.
- Se la richiesta è stata effettuata da un altro servizio AWS.

Per ulteriori informazioni su `userIdentity`, consultare Elemento userIdentity CloudTrail. Per ulteriori informazioni su `CreateDBInstance` e altre operazioni di Amazon Aurora, consulta la Documentazione di riferimento delle API di Amazon RDS.

Monitoraggio di Amazon Aurora tramite i flussi di attività del database

Con Flussi di attività del database puoi monitorare pressoché in tempo reale i flussi di attività del database.

Argomenti

- Panoramica dei flussi di attività di database
- Prerequisiti di rete per Database Activity Streams Aurora MySQL
- Avvio di un flusso di attività di database
- Recupero dello stato di un flusso di attività del database
- Arresto di un flusso di attività di database
- Monitoraggio di flussi di attività di database
- Esempi di policy IAM per flussi di attività del database

Panoramica dei flussi di attività di database

Come amministratore del database Amazon Aurora, devi proteggere il database e soddisfare i requisiti normativi e di conformità. Una strategia consiste nell'integrare i flussi di attività del database con gli strumenti di monitoraggio. In questo modo, puoi monitorare e impostare gli allarmi per l'attività di audit nel cluster Amazon Aurora.

Le minacce alla sicurezza sono sia esterne che interne. Per proteggersi dalle minacce interne, è possibile controllare l'accesso degli amministratori ai flussi di dati configurando la funzionalità flussi di attività del database. I DBA non hanno accesso alla raccolta, alla trasmissione, all'archiviazione e all'elaborazione dei flussi.

Indice

- Come funzionano i flussi di attività del database
- Modalità asincrona e sincrona per flussi di attività di database
- Requisiti e limitazioni per flussi di attività del database
- Disponibilità di regioni e versioni
- Classi di istanza database supportate per i flussi di attività di database

Come funzionano i flussi di attività del database

In Amazon Aurora, puoi avviare un flusso di attività del database a livello di cluster. Tutte le istanze database all'interno del cluster hanno i flussi di attività del database abilitati.

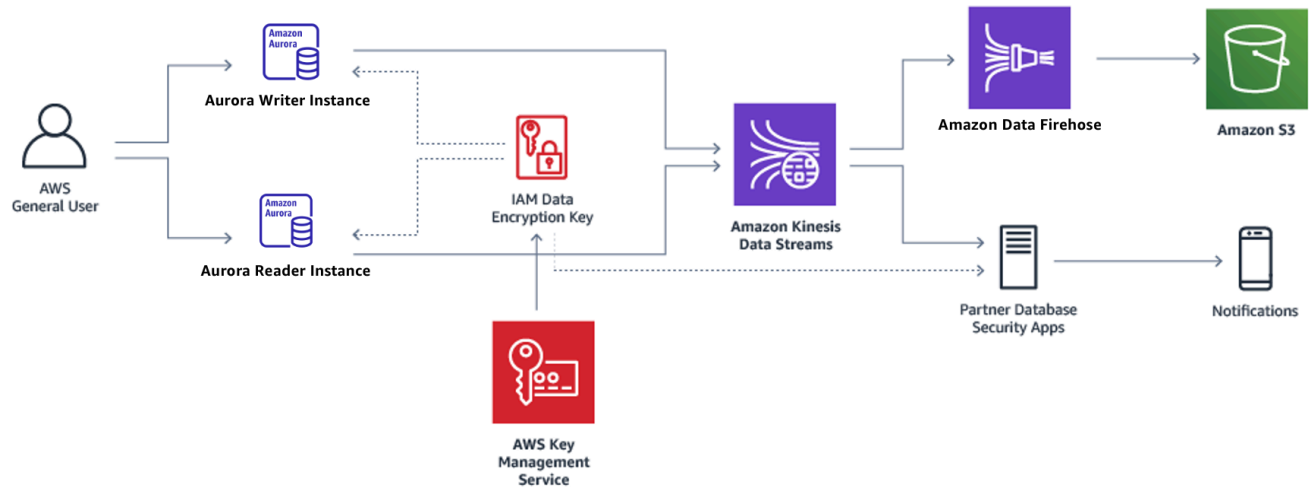
Il cluster di database Aurora inserisce le attività in un flusso di dati Amazon Kinesis pressoché in tempo reale. Il flusso Kinesis viene creato automaticamente. Da Kinesis, è possibile configurare servizi AWS quali Amazon Data Firehose e AWS Lambda per consumare il flusso e archiviare i dati.

> **Important**
>
> L'utilizzo della funzionalità Flussi di attività del database in Amazon Aurora è gratuito, ma Amazon Kinesis addebita i costi del flusso di dati. Per ulteriori informazioni, consulta Prezzi di Amazon Kinesis Data Streams.

Se utilizzi un database globale Aurora, avvia un flusso di attività del database su ciascun cluster di database separatamente. Ciascun cluster fornisce i dati di audit al proprio flusso Kinesis all'interno della propria Regione AWS. I flussi di attività non funzionano in modo diverso durante un failover. Continuano a verificare il database globale come al solito.

Puoi configurare le applicazioni per la gestione della conformità affinché attingano dai flussi di attività del database. Per Aurora PostgreSQL, le applicazioni di conformità includono Security Guardium di IBM e SecureSphere Database Audit and Protection di Imperva. Tali applicazioni possono utilizzare il flusso per generare avvisi e attività di verifica per il cluster di database Aurora.

Il grafico riportato di seguito mostra un cluster di database Aurora configurato con Amazon Data Firehose.



Modalità asincrona e sincrona per flussi di attività di database

Puoi decidere di impostare la sessione del database per gestire gli eventi dell'attività di database in una delle seguenti modalità:

- Modalità asincrona: quando una sessione di database genera un evento del flusso di attività, vengono ripristinate immediatamente le normali attività della sessione. In background, l'evento di flusso di attività è reso un record durevole. Se si verifica un errore nell'attività in background, viene inviato une evento RDS. Questo indica l'inizio e la fine di qualsiasi finestra temporale in cui i record dell'evento del flusso di attività potrebbero essere stati persi.

  La modalità asincrona favorisce le prestazioni del database rispetto alla precisione del flusso di attività.

  > **Note**
  >
  > La modalità asincrona è disponibile per entrambi Aurora PostgreSQL e Aurora MySQL.

- Modalità sincrona: quando una sessione di database genera un evento del flusso di attività, la sessione si blocca finché l'evento non viene reso durevole. Se per qualche motivo non è possibile rendere l'evento durevole, vengono ripristinate le normali attività della sessione di database. Tuttavia, viene inviato un evento RDS che indica che i record del flusso di attività possono essere persi per qualche secondo. Un secondo evento RDS viene inviato dopo che viene ripristinato lo stato di integrità del sistema.

La modalità sincrona favorisce la precisione del flusso di attività sulle prestazioni del database.

> **Note**
>
> La modalità sincrona è disponibile per Aurora PostgreSQL. Non è possibile utilizzare la modalità sincrona con Aurora MySQL.

Requisiti e limitazioni per flussi di attività del database

In Aurora, i flussi di attività del database hanno i requisiti e le limitazioni riportati di seguito:

- I flussi di attività del database richiedono l'utilizzo di Amazon Kinesis.
- L'utilizzo di AWS Key Management Service (AWS KMS) è obbligatorio perché i flussi di attività del database sono sempre crittografati.
- L'applicazione di una crittografia aggiuntiva al flusso di dati di Amazon Kinesis non è compatibile con i flussi di attività del database, già crittografati con la tua chiave AWS KMS.
- Avvia il flusso di attività del database a livello di cluster di database. Se aggiungi un'istanza database al cluster, non è necessario avviare un flusso di attività sull'istanza: questa viene controllata automaticamente.
- In un database globale Aurora, avvia un flusso di attività del database su ciascun cluster di database separatamente. Ciascun cluster fornisce i dati di audit al proprio flusso Kinesis all'interno della propria Regione AWS.
- In Aurora PostgreSQL, assicurati di interrompere il flusso di attività del database prima di eseguire un aggiornamento a una versione principale. È possibile avviare il flusso di attività del database al termine dell'aggiornamento.

Disponibilità di regioni e versioni

La disponibilità e il supporto della funzionalità varia a seconda delle versioni specifiche di ciascun motore di database Aurora e in tutte le Regioni AWS. Per ulteriori informazioni sulla disponibilità di versioni e Regioni con Aurora e i flussi di attività del database, consulta Regioni e motori di database Aurora supportati per flussi di attività del database.

Classi di istanza database supportate per i flussi di attività di database

Per Aurora MySQL, è possibile utilizzare flussi di attività di database con le seguenti classi di istanza database:

- db.r8g.*large
- db.r7g.*large
- db.r7i.*large
- db.r6g.*large
- db.r6i.*large
- db.r5.*large
- db.x2g.*

Per Aurora PostgreSQL, è possibile utilizzare flussi di attività di database con le seguenti classi di istanza database:

- db.r8g.*large
- db.r7i.*large
- db.r7g.*large
- db.r6g.*large
- db.r6i.*large
- db.r6id.*large
- db.r5.*large
- db.r4.*large
- db.x2g.*

Prerequisiti di rete per Database Activity Streams Aurora MySQL

Nella sezione seguente, puoi scoprire come configurare il cloud privato virtuale (VPC) per l'utilizzo con flussi di attività di database.

> **Note**
>
> I prerequisiti di rete di Aurora MySQL si applicano alle seguenti versioni del motore:

- Aurora MySQL versione 2, fino alla 2.11.3
- Aurora MySQL versione 2.12.0
- Aurora MySQL versione 3, fino alla 3.04.2

Argomenti

- Prerequisiti per endpoint AWS KMS
- Prerequisiti per la disponibilità pubblica
- Prerequisiti per la disponibilità privata

Prerequisiti per endpoint AWS KMS

Le istanze in un cluster Aurora MySQL che utilizzano flussi di attività devono essere in grado di accedere endpoint AWS KMS. Assicurarsi che questo requisito sia soddisfatto prima di abilitare i flussi di attività del database per il cluster Aurora MySQL. Se il cluster Aurora è disponibile al pubblico, questo requisito viene soddisfatto automaticamente.

Important

Se il cluster database Aurora MySQL non è in grado di accedere all'endpoint AWS KMS, il flusso di attività smette di funzionare. In tal caso, Aurora notifica questo problema utilizzando eventi RDS.

Prerequisiti per la disponibilità pubblica

Affinché un cluster di database Aurora sia pubblico, deve soddisfare i seguenti requisiti:

- Publicly Accessible (Accessibile pubblicamente) è Yes (Sì) nella pagina dei dettagli del cluster Console di gestione AWS.
- Inoltre, il cluster di database è in una sottorete pubblica di Amazon VPC. Per ulteriori informazioni sulle istanze database accessibili pubblicamente, consulta Uso di un cluster database in un VPC. Per ulteriori informazioni sulle sottoreti pubbliche Amazon VPC, consulta VPC e sottoreti.

Prerequisiti per la disponibilità privata

Se il cluster database Aurora si trova in una sottorete pubblica VPC e non è accessibile pubblicamente, significa che è privato. Per mantenere privato il cluster e utilizzarlo con flussi di attività del database, sono disponibili le opzioni seguenti:

- Configurare Network Address Translation (NAT) nel VPC. Per ulteriori informazioni, consulta Gateway NAT.
- Creare un endpoint AWS KMS nel VPC Questa opzione è consigliata perché è più semplice da configurare.

Creare un endpoint AWS KMS nel VPC

1. Accedere alla console Amazon VPC all'indirizzo https://console.aws.amazon.com/vpc/.
2. Nel pannello di navigazione, seleziona Endpoints (Endpoint).
3. Scegliere Create Endpoint (Crea endpoint).

   La pagina Creazione endpoint viene visualizzata.
4. Esegui questa operazione:
   - In Service category (Categoria dei servizi), scegliere AWS services (Servizi AWS).
   - In Service Name (Nome servizio), scegliere com.amazonaws.***regione***.kms, dove *regione* è la Regione AWS in cui si trova il cluster.
   - Per VPC, scegliere il VPC in cui si trova il cluster.
5. Scegliere Create Endpoint (Crea endpoint).

Per ulteriori informazioni sulla configurazione degli endpoint VPC, consulta Endpoint VPC.

Avvio di un flusso di attività di database

Per monitorare l'attività del database per tutte le istanze del cluster database Aurora, avvia un flusso di attività Aurora livello di cluster. Le eventuali istanze database aggiunte al cluster vengono anche monitorate automaticamente. Se utilizzi un database globale Aurora, avvia un flusso di attività del database su ciascun cluster di database separatamente. Ciascun cluster fornisce i dati di audit al proprio flusso Kinesis all'interno della propria Regione AWS.

Quando avvii un flusso di attività, ogni evento di attività del database configurato nella policy di audit, genera un evento di flusso di attività. Gli eventi di accesso vengono generati da comandi SQL quali `CONNECT` e `SELECT`. Gli eventi di modifica vengono generati da comandi SQL quali `CREATE` e `INSERT`.

Console

Come avviare un flusso di attività di database

1. Apri la console Amazon RDS all'indirizzo https://console.aws.amazon.com/rds/.
2. Nel pannello di navigazione, scegliere Databases (Database).
3. Seleziona il cluster di database per cui desideri abilitare un flusso di attività.
4. In Actions (Operazioni), scegliere Start activity stream (Avvia flusso di attività).

   Viene visualizzata la finestra Avvia flusso di attività di database: *nome*, dove *nome* è la tua cluster di database.
5. Specificare le seguenti impostazioni:

   - In AWS KMS key, scegliere una chiave dall'elenco di AWS KMS keys.

     > **Note**
     >
     > Se il cluster Aurora MySQL non è in grado di accedere alle chiavi KMS, seguire le istruzioni riportate in Prerequisiti di rete per Database Activity Streams Aurora MySQL per attivare tale accesso.

     Aurora utilizza la chiave KMS per crittografare la chiave che a sua volta esegue la crittografia dell'attività del database. Scegliere una chiave KMS diversa dalla chiave di default. Per ulteriori informazioni sulle chiavi di crittografia e AWS KMS, consulta Che cos'è AWS Key Management Service? nella Guida per gli sviluppatori di AWS Key Management Service.
   - In Database activity stream mode (Modalità flusso di attività di database), scegliere Asynchronous (Asincrona) o Synchronous (Sincrona).

> **Note**
>
> Questa scelta si applica solo a Aurora PostgreSQL. Per Aurora MySQL, è possibile utilizzare solo la modalità asincrona.

- Scegliere Immediatamente.

  Selezionando Subito, il cluster database viene riavviata immediatamente. Se si sceglie Durante la finestra di manutenzione successiva, il cluster database non si riavvia subito. In questo caso, il flusso di attività del database non viene avviato fino alla finestra di manutenzione successiva.

6. Scegli Start database activity stream (Avvia flusso di attività di database).

   Lo stato del cluster di database mostra che il flusso di attività è in fase di avvio.

> **Note**
>
> Se ricevi l'errore `You can't start a database activity stream in this configuration`, controlla Classi di istanza database supportate per i flussi di attività di database per vedere se il cluster database utilizza una classe di istanza supportata.

AWS CLI

Per avviare flussi di attività del database per un cluster di database, configura il cluster di database utilizzando il comando AWS CLI start-activity-stream.

- `--resource-arn` *`arn`*: specifica l'Amazon Resource Name (ARN) del cluster database.
- `--mode` *`sync-or-async`*: specifica la modalità sincrona (`sync`) o asincrona (`async`). Per Aurora PostgreSQL, è possibile scegliere uno dei due valori. Per Aurora MySQL, specifica `async`.
- `--kms-key-id` *`key`*: specifica l'identificatore della chiave KMS per la crittografia dei messaggi nel flusso di attività del database. L'identificatore di chiave AWS KMS è l'ARN della chiave, l'ID chiave, l'ARN dell'alias o il nome alias per la AWS KMS key.

L'esempio seguente avvia un flusso di attività del database per un cluster di database in modalità asincrona.

Per Linux, macOS o Unix:

```
aws rds start-activity-stream \
    --mode async \
    --kms-key-id my-kms-key-arn \
    --resource-arn my-cluster-arn \
    --apply-immediately
```

Per Windows:

```
aws rds start-activity-stream ^
    --mode async ^
    --kms-key-id my-kms-key-arn ^
    --resource-arn my-cluster-arn ^
    --apply-immediately
```

Amazon RDS API

Per avviare flussi di attività del database per un cluster di database, configura il cluster utilizzando l'operazione StartActivityStream.

Richiamare l'operazione con i parametri seguenti:

- `Region`
- `KmsKeyId`
- `ResourceArn`
- `Mode`

> **Note**
>
> Se ricevi un errore che indica che non è possibile avviare un flusso di attività del database con la versione corrente del database Aurora PostgreSQL, applica la patch più recente per Aurora PostgreSQL prima di avviare un flusso di attività del database. Per informazioni sulle versioni del database Aurora PostgreSQL, consulta Aggiornamento dei cluster di database Amazon Aurora.

Di seguito sono riportate le versioni minime della patch necessarie per avviare i flussi di attività del database con Aurora PostgreSQL.

- 3.4.15 (11.9.15), 11.21.10
- 12.9.15, 12.15.9, 12.16.10, 12.17.7, 12.18.5, 12.19.4, 12.20.3, 12.22.3
- 13.9.12, 13.11.9, 13.12.10, 13.13.7, 13.14.5, 13.15.4, 13.16.3, 13.18.3
- 14.6.12, 14.8.9, 14.9.10, 14.10.7, 14.11.5, 14.12.4, 14.13.3, 14.15.3
- 15.3.9, 15.4.10, 15.5.7, 15.6.5, 15.7.4, 15.8.3, 15.10.3
- 16.1.7, 16.2.5, 16.3.4, 16.4.3, 16.6.3

Recupero dello stato di un flusso di attività del database

Puoi recuperare lo stato di un flusso di attività tramite la console o la AWS CLI.

Console

Come recuperare lo stato di un flusso di attività del database

1. Apri la console Amazon RDS all'indirizzo https://console.aws.amazon.com/rds/.
2. Nel pannello di navigazione, scegliere Database e selezionare il link del cluster database.
3. Scegliere la scheda Configurazione e selezionare Flusso di attività di database per lo stato.

AWS CLI

È possibile ottenere la configurazione del flusso di attività per un cluster di database come risposta a una richiesta della CLI describe-db-clusters .

Nell'esempio seguente viene illustrato *my-cluster*.

```
aws rds --region my-region describe-db-clusters --db-cluster-identifier my-cluster
```

Il seguente esempio mostra una risposta in formato JSON: Sono visualizzati i seguenti campi:

- `ActivityStreamKinesisStreamName`
- `ActivityStreamKmsKeyId`
- `ActivityStreamStatus`

- `ActivityStreamMode`
-

Questi campi sono gli stessi per Aurora PostgreSQL e Aurora MySQL, tranne che `ActivityStreamMode` è sempre `async` per Aurora MySQL, mentre per Aurora PostgreSQL esso potrebbe essere `sync` o `async`.

```
{
    "DBClusters": [
        {
      "DBClusterIdentifier": "my-cluster",
            ...
            "ActivityStreamKinesisStreamName": "aws-rds-das-cluster-
A6TSYXITZCZXJHIRVFUBZ5LTWY",
            "ActivityStreamStatus": "starting",
            "ActivityStreamKmsKeyId": "12345678-abcd-efgh-ijkl-bd041f170262",
            "ActivityStreamMode": "async",
            "DbClusterResourceId": "cluster-ABCD123456"
            ...
        }
    ]
}
```

API RDS

È possibile ottenere la configurazione del flusso di attività per un cluster di database come risposta a un'operazione DescribeDBClusters .

Arresto di un flusso di attività di database

Puoi interrompere un flusso di attività utilizzando la console o AWS CLI.

Se elimini il cluster di database, il flusso di attività viene arrestato e il flusso Amazon Kinesis sottostante viene eliminato automaticamente.

Console

Per disattivare un flusso di attività

1. Apri la console Amazon RDS all'indirizzo https://console.aws.amazon.com/rds/.

2. Nel pannello di navigazione, scegliere Databases (Database).
3. Scegliere un cluster database per il quale si desidera interrompere il flusso di attività di database.
4. In Actions (Operazioni), scegliere Stop activity stream (Interrompi flusso di attività). Viene visualizzata la finestra Database Activity Stream (Flusso di attività di database).

    a. Scegliere Immediatamente.

    Selezionando Subito, il cluster database viene riavviata immediatamente. Se si sceglie Durante la finestra di manutenzione successiva, il cluster database non si riavvia subito. In questo caso, il flusso di attività del database non viene arrestato fino alla finestra di manutenzione successiva.

    b. Scegli Continue (Continua).

AWS CLI

Per arrestare i flussi di attività di database per il cluster di database, configurare il cluster utilizzando il comando stop-activity-stream della AWS CLI. Identifica la regione AWS per il cluster database mediante il parametro `--region`. Il parametro `--apply-immediately` è facoltativo.

Per Linux, macOS o Unix:

```
aws rds --region MY_REGION \
    stop-activity-stream \
    --resource-arn MY_CLUSTER_ARN \
    --apply-immediately
```

Per Windows:

```
aws rds --region MY_REGION ^
    stop-activity-stream ^
    --resource-arn MY_CLUSTER_ARN ^
    --apply-immediately
```

API RDS

Per interrompere i flussi di attività di database per il cluster di database, configura il cluster utilizzando l'operazione StopActivityStream. Identifica la regione AWS per il cluster database mediante il parametro `Region`. Il parametro `ApplyImmediately` è facoltativo.

Monitoraggio di flussi di attività di database

I flussi di attività di database monitorano e segnalano le attività. Il flusso di attività viene raccolto e trasmesso a Amazon Kinesis. Da Kinesis, è possibile monitorare il flusso di attività oppure altri servizi e applicazioni possono utilizzare il flusso di attività per ulteriori analisi. È possibile trovare il nome del flusso Kinesis sottostante utilizzando il comando della AWS CLI `describe-db-clusters` o l'operazione `DescribeDBClusters` dell'API RDS.

Aurora gestisce il flusso Kinesis per tuo conto come segue:

- Aurora crea automaticamente il flusso Kinesis con un periodo di conservazione di 24 ore.
- Aurora dimensiona il flusso Kinesis, se necessario.
- Se si interrompe il flusso di attività del database o si elimina il cluster di database, Aurora elimina il flusso Kinesis.

Le seguenti categorie di attività vengono monitorate e inserite nel log di controllo del flusso di attività:

- Comandi SQL: tutti i comandi SQL sono controllati e anche le istruzioni preparate, le funzioni integrate e le funzioni in PL/SQL Le chiamate alle procedure archiviate vengono controllate. Vengono inoltre controllate tutte le istruzioni SQL rilasciate all'interno di procedure o funzioni archiviate.
- Altre informazioni di database – L'attività monitorata include l'istruzione SQL completa, il conteggio righe delle righe interessate da comandi DML, gli oggetti ai quali si accede e il nome del database univoco. Per Aurora PostgreSQL, i flussi di attività del database monitorano anche le variabili di bind e i parametri della stored procedure.

  > **Important**
  >
  > Il testo SQL completo di ogni istruzione è visibile nel registro di controllo del flusso di attività, inclusi eventuali dati sensibili. Tuttavia, le password degli utenti del database vengono omesse se Aurora può stabilirle dal contesto, come nell'istruzione SQL seguente.
  >
  > ```
  > ALTER ROLE role-name WITH password
  > ```

- Informazioni di connessione – L'attività monitorata include informazioni di sessione e di rete, l'ID di processo del server e i codici di uscita.

Se un flusso di attività restituisce un errore durante il monitoraggio dell'istanza database, riceverai una notifica mediante eventi RDS.

Nelle sezioni seguenti puoi accedere ai flussi di attività del database, eseguirne l'audit ed elaborarli.

Argomenti

- Accesso a un flusso di attività da Amazon Kinesis
- Contenuto ed esempi di log di audit per i flussi di attività del database
- Array JSON databaseActivityEventList per flussi di attività del database
- Elaborazione di un flusso di attività del database utilizzando l'SDK AWS

Accesso a un flusso di attività da Amazon Kinesis

Quando abiliti un flusso di attività per un cluster database, viene creato automaticamente un flusso Kinesis. Da Kinesis, puoi monitorare l'attività del database in tempo reale. Per analizzare ulteriormente l'attività del database, puoi connettere il flusso Kinesis ad applicazioni consumer. È inoltre possibile connettere il flusso alle applicazioni di gestione della conformità, ad esempio Security Guardium IBM o SecureSphere Database Audit and Protection Imperva.

Puoi accedere al tuo flusso Kinesis dalla console RDS o dalla console Kinesis.

Come accedere a un flusso di attività da Kinesis utilizzando la console RDS

1. Apri la console Amazon RDS all'indirizzo https://console.aws.amazon.com/rds/.
2. Nel pannello di navigazione, scegliere Databases (Database).
3. Seleziona il cluster di database in cui hai avviato un flusso di attività.
4. Scegliere Configuration (Configurazione).
5. In Database activity stream (Flusso di attività del database), scegli il collegamento sotto Kinesis stream (Flusso Kinesis).
6. Nella console Kinesis, scegli Monitoring (Monitoraggio) per iniziare l'osservazione dell'attività del database.

Per accedere a un flusso di attività da Kinesis utilizzando la console Kinesis

1. Aprire la console Kinesis all'indirizzo https://console.aws.amazon.com/kinesis.
2. Scegliere il flusso di attività dall'elenco di flussi Kinesis.

Il nome di un flusso di attività include il prefisso `aws-rds-das-cluster-` seguito dall'ID risorsa del cluster database. Di seguito è riportato un esempio.

```
aws-rds-das-cluster-NHVOV4PCLWHGF52NP
```

Per utilizzare la console Amazon RDS per trovare l'ID risorsa per il cluster di database, scegli il cluster database dall'elenco di database, quindi seleziona la scheda Configuration (Configurazione).

Per utilizzare la AWS CLI per trovare il nome del flusso Kinesis completo per un flusso di attività, utilizzare una richiesta CLI describe-db-clusters e prendere nota del valore di `ActivityStreamKinesisStreamName` nella risposta.

3. Scegliere Monitoring (Monitoraggio) per iniziare l'osservazione dell'attività di database.

Per ulteriori informazioni sull'utilizzo di Amazon Kinesis, consulta Che cos'è Amazon Kinesis Data Streams?.

Contenuto ed esempi di log di audit per i flussi di attività del database

Gli eventi monitorati sono rappresentati nel flusso di attività del database come stringhe JSON. La struttura è costituita da un oggetto JSON contenente un `DatabaseActivityMonitoringRecord`, che a sua volta contiene un array `databaseActivityEventList` di eventi attività.

> **Note**
>
> Per i flussi di attività del database, l'array JSON `paramList` non include valori null dalle applicazioni Hibernate.

Argomenti

- Esempi di log di verifica per un flusso di attività
- Oggetto JSON DatabaseActivityMonitoringRecords
- Oggetto JSON databaseActivityEvents

Esempi di log di verifica per un flusso di attività

Di seguito sono riportati registri di controllo JSON decrittografati di esempio di record di eventi attività.

Example Record di evento di attività di un'istruzione Aurora PostgreSQL CONNECT SQL

Il seguente record di evento di attività mostra un accesso con l'utilizzo di un'istruzione SQL `CONNECT` (`command`) mediante un client psql (`clientApplication`).


```
{
  "type":"DatabaseActivityMonitoringRecords",
  "version":"1.1",
  "databaseActivityEvents":
    {
      "type":"DatabaseActivityMonitoringRecord",
      "clusterId":"cluster-4HNY5V4RRNPKKYB7ICFKE5JBQQ",
      "instanceId":"db-FZJTMYKCXQBUUZ6VLU7NW3ITCM",
      "databaseActivityEventList":[
        {
          "startTime": "2019-10-30 00:39:49.940668+00",
          "logTime": "2019-10-30 00:39:49.990579+00",
          "statementId": 1,
          "substatementId": 1,
          "objectType": null,
          "command": "CONNECT",
          "objectName": null,
          "databaseName": "postgres",
          "dbUserName": "rdsadmin",
          "remoteHost": "172.31.3.195",
          "remotePort": "49804",
          "sessionId": "5ce5f7f0.474b",
          "rowCount": null,
          "commandText": null,
          "paramList": [],
          "pid": 18251,
          "clientApplication": "psql",
          "exitCode": null,
          "class": "MISC",
          "serverVersion": "2.3.1",
          "serverType": "PostgreSQL",
          "serviceName": "Amazon Aurora PostgreSQL-Compatible edition",
          "serverHost": "172.31.3.192",
          "netProtocol": "TCP",
          "dbProtocol": "Postgres 3.0",
```

```
          "type": "record",
          "errorMessage": null
        }
      ]
    },
   "key":"decryption-key"
}
```

Example Record di evento attività di un'istruzione SQL CONNECT Aurora MySQL

Il seguente record di evento di attività mostra un accesso con l'utilizzo di un'istruzione SQL `CONNECT` (`command`) mediante un client mysql (`clientApplication`).

```
{
  "type":"DatabaseActivityMonitoringRecord",
  "clusterId":"cluster-some_id",
  "instanceId":"db-some_id",
  "databaseActivityEventList":[
    {
      "logTime":"2020-05-22 18:07:13.267214+00",
      "type":"record",
      "clientApplication":null,
      "pid":2830,
      "dbUserName":"rdsadmin",
      "databaseName":"",
      "remoteHost":"localhost",
      "remotePort":"11053",
      "command":"CONNECT",
      "commandText":"",
      "paramList":null,
      "objectType":"TABLE",
      "objectName":"",
      "statementId":0,
      "substatementId":1,
      "exitCode":"0",
      "sessionId":"725121",
      "rowCount":0,
      "serverHost":"master",
      "serverType":"MySQL",
      "serviceName":"Amazon Aurora MySQL",
      "serverVersion":"MySQL 5.7.12",
      "startTime":"2020-05-22 18:07:13.267207+00",
      "endTime":"2020-05-22 18:07:13.267213+00",
```

```
      "transactionId":"0",
      "dbProtocol":"MySQL",
      "netProtocol":"TCP",
      "errorMessage":"",
      "class":"MAIN"
    }
  ]
}
```

Example Record di evento attività di Aurora PostgreSQL

Il seguente esempio mostra un evento `CREATE TABLE` per Aurora PostgreSQL.

```
{
  "type":"DatabaseActivityMonitoringRecords",
  "version":"1.1",
  "databaseActivityEvents":
    {
      "type":"DatabaseActivityMonitoringRecord",
      "clusterId":"cluster-4HNY5V4RRNPKKYB7ICFKE5JBQQ",
      "instanceId":"db-FZJTMYKCXQBUUZ6VLU7NW3ITCM",
      "databaseActivityEventList":[
        {
          "startTime": "2019-05-24 00:36:54.403455+00",
          "logTime": "2019-05-24 00:36:54.494235+00",
          "statementId": 2,
          "substatementId": 1,
          "objectType": null,
          "command": "CREATE TABLE",
          "objectName": null,
          "databaseName": "postgres",
          "dbUserName": "rdsadmin",
          "remoteHost": "172.31.3.195",
          "remotePort": "34534",
          "sessionId": "5ce73c6f.7e64",
          "rowCount": null,
          "commandText": "create table my_table (id serial primary key, name
 varchar(32));",
          "paramList": [],
          "pid": 32356,
          "clientApplication": "psql",
          "exitCode": null,
          "class": "DDL",
          "serverVersion": "2.3.1",
```

```
          "serverType": "PostgreSQL",
          "serviceName": "Amazon Aurora PostgreSQL-Compatible edition",
          "serverHost": "172.31.3.192",
          "netProtocol": "TCP",
          "dbProtocol": "Postgres 3.0",
          "type": "record",
          "errorMessage": null
        }
      ]
    },
   "key":"decryption-key"
}
```

Example Record di evento attività di un'istruzione CREATE TABLE Aurora MySQL

Il seguente esempio mostra un'istruzione `CREATE TABLE` per Aurora MySQL. L'operazione è rappresentata come due record di eventi separati. Un evento ha `"class":"MAIN"`. L'altro evento ha `"class":"AUX"`. I messaggi potrebbero arrivare in qualsiasi ordine. Il campo `logTime` dell'evento `MAIN` è sempre precedente ai campi `logTime` di qualsiasi evento `AUX` corrispondente.

Nell'esempio seguente viene illustrato l'evento con un valore `class` pari a `MAIN`.

```
{
  "type":"DatabaseActivityMonitoringRecord",
  "clusterId":"cluster-some_id",
  "instanceId":"db-some_id",
  "databaseActivityEventList":[
    {
      "logTime":"2020-05-22 18:07:12.250221+00",
      "type":"record",
      "clientApplication":null,
      "pid":2830,
      "dbUserName":"master",
      "databaseName":"test",
      "remoteHost":"localhost",
      "remotePort":"11054",
      "command":"QUERY",
      "commandText":"CREATE TABLE test1 (id INT)",
      "paramList":null,
      "objectType":"TABLE",
      "objectName":"test1",
      "statementId":65459278,
      "substatementId":1,
```

```
      "exitCode":"0",
      "sessionId":"725118",
      "rowCount":0,
      "serverHost":"master",
      "serverType":"MySQL",
      "serviceName":"Amazon Aurora MySQL",
      "serverVersion":"MySQL 5.7.12",
      "startTime":"2020-05-22 18:07:12.226384+00",
      "endTime":"2020-05-22 18:07:12.250222+00",
      "transactionId":"0",
      "dbProtocol":"MySQL",
      "netProtocol":"TCP",
      "errorMessage":"",
      "class":"MAIN"
    }
  ]
}
```

Nell'esempio seguente viene illustrato l'evento corrispondente con un valore `class` pari a AUX.

```
{
  "type":"DatabaseActivityMonitoringRecord",
  "clusterId":"cluster-some_id",
  "instanceId":"db-some_id",
  "databaseActivityEventList":[
    {
      "logTime":"2020-05-22 18:07:12.247182+00",
      "type":"record",
      "clientApplication":null,
      "pid":2830,
      "dbUserName":"master",
      "databaseName":"test",
      "remoteHost":"localhost",
      "remotePort":"11054",
      "command":"CREATE",
      "commandText":"test1",
      "paramList":null,
      "objectType":"TABLE",
      "objectName":"test1",
      "statementId":65459278,
      "substatementId":2,
      "exitCode":"",
      "sessionId":"725118",
```

```
      "rowCount":0,
      "serverHost":"master",
      "serverType":"MySQL",
      "serviceName":"Amazon Aurora MySQL",
      "serverVersion":"MySQL 5.7.12",
      "startTime":"2020-05-22 18:07:12.226384+00",
      "endTime":"2020-05-22 18:07:12.247182+00",
      "transactionId":"0",
      "dbProtocol":"MySQL",
      "netProtocol":"TCP",
      "errorMessage":"",
      "class":"AUX"
    }
  ]
}
```

Example Record di evento attività di un'istruzione Aurora PostgreSQL SELECT

Il seguente esempio mostra un evento `SELECT` .

```
{
  "type":"DatabaseActivityMonitoringRecords",
  "version":"1.1",
  "databaseActivityEvents":
    {
      "type":"DatabaseActivityMonitoringRecord",
      "clusterId":"cluster-4HNY5V4RRNPKKYB7ICFKE5JBQQ",
      "instanceId":"db-FZJTMYKCXQBUUZ6VLU7NW3ITCM",
      "databaseActivityEventList":[
        {
          "startTime": "2019-05-24 00:39:49.920564+00",
          "logTime": "2019-05-24 00:39:49.940668+00",
          "statementId": 6,
          "substatementId": 1,
          "objectType": "TABLE",
          "command": "SELECT",
          "objectName": "public.my_table",
          "databaseName": "postgres",
          "dbUserName": "rdsadmin",
          "remoteHost": "172.31.3.195",
          "remotePort": "34534",
          "sessionId": "5ce73c6f.7e64",
          "rowCount": 10,
          "commandText": "select * from my_table;",
```

```
          "paramList": [],
          "pid": 32356,
          "clientApplication": "psql",
          "exitCode": null,
          "class": "READ",
          "serverVersion": "2.3.1",
          "serverType": "PostgreSQL",
          "serviceName": "Amazon Aurora PostgreSQL-Compatible edition",
          "serverHost": "172.31.3.192",
          "netProtocol": "TCP",
          "dbProtocol": "Postgres 3.0",
          "type": "record",
          "errorMessage": null
        }
      ]
    },
   "key":"decryption-key"
}
```

```
{
    "type": "DatabaseActivityMonitoringRecord",
    "clusterId": "",
    "instanceId": "db-4JCWQLUZVFYP7DIWP6JVQ77O3Q",
    "databaseActivityEventList": [
        {
            "class": "TABLE",
            "clientApplication": "Microsoft SQL Server Management Studio - Query",
            "command": "SELECT",
            "commandText": "select * from [testDB].[dbo].[TestTable]",
            "databaseName": "testDB",
            "dbProtocol": "SQLSERVER",
            "dbUserName": "test",
            "endTime": null,
            "errorMessage": null,
            "exitCode": 1,
            "logTime": "2022-10-06 21:24:59.9422268+00",
            "netProtocol": null,
            "objectName": "TestTable",
            "objectType": "TABLE",
            "paramList": null,
            "pid": null,
            "remoteHost": "local machine",
            "remotePort": null,
```

```
            "rowCount": 0,
            "serverHost": "172.31.30.159",
            "serverType": "SQLSERVER",
            "serverVersion": "15.00.4073.23.v1.R1",
            "serviceName": "sqlserver-ee",
            "sessionId": 62,
            "startTime": null,
            "statementId": "0x03baed90412f564fad640ebe51f89b99",
            "substatementId": 1,
            "transactionId": "4532935",
            "type": "record",
            "engineNativeAuditFields": {
                "target_database_principal_id": 0,
                "target_server_principal_id": 0,
                "target_database_principal_name": "",
                "server_principal_id": 2,
                "user_defined_information": "",
                "response_rows": 0,
                "database_principal_name": "dbo",
                "target_server_principal_name": "",
                "schema_name": "dbo",
                "is_column_permission": true,
                "object_id": 581577110,
                "server_instance_name": "EC2AMAZ-NFUJJNO",
                "target_server_principal_sid": null,
                "additional_information": "",
                "duration_milliseconds": 0,
                "permission_bitmask": "0x00000000000000000000000000000001",
                "data_sensitivity_information": "",
                "session_server_principal_name": "test",
                "connection_id": "AD3A5084-FB83-45C1-8334-E923459A8109",
                "audit_schema_version": 1,
                "database_principal_id": 1,
                "server_principal_sid":
 "0x0105000000000005150000000bdc2795e2d0717901ba6998cf4010000",
                "user_defined_event_id": 0,
                "host_name": "EC2AMAZ-NFUJJNO"
            }
        }
    ]
}
```

Example Record di evento attività di un'istruzione SELECT Aurora MySQL

Il seguente esempio mostra un evento `SELECT`.

Nell'esempio seguente viene illustrato l'evento con un valore `class` pari a `MAIN`.


```
{
  "type":"DatabaseActivityMonitoringRecord",
  "clusterId":"cluster-some_id",
  "instanceId":"db-some_id",
  "databaseActivityEventList":[
    {
      "logTime":"2020-05-22 18:29:57.986467+00",
      "type":"record",
      "clientApplication":null,
      "pid":2830,
      "dbUserName":"master",
      "databaseName":"test",
      "remoteHost":"localhost",
      "remotePort":"11054",
      "command":"QUERY",
      "commandText":"SELECT * FROM test1 WHERE id < 28",
      "paramList":null,
      "objectType":"TABLE",
      "objectName":"test1",
      "statementId":65469218,
      "substatementId":1,
      "exitCode":"0",
      "sessionId":"726571",
      "rowCount":2,
      "serverHost":"master",
      "serverType":"MySQL",
      "serviceName":"Amazon Aurora MySQL",
      "serverVersion":"MySQL 5.7.12",
      "startTime":"2020-05-22 18:29:57.986364+00",
      "endTime":"2020-05-22 18:29:57.986467+00",
      "transactionId":"0",
      "dbProtocol":"MySQL",
      "netProtocol":"TCP",
      "errorMessage":"",
      "class":"MAIN"
    }
  ]
```

```
}
```

Nell'esempio seguente viene illustrato l'evento corrispondente con un valore `class` pari a AUX.

```
{
  "type":"DatabaseActivityMonitoringRecord",
  "instanceId":"db-some_id",
  "databaseActivityEventList":[
    {
      "logTime":"2020-05-22 18:29:57.986399+00",
      "type":"record",
      "clientApplication":null,
      "pid":2830,
      "dbUserName":"master",
      "databaseName":"test",
      "remoteHost":"localhost",
      "remotePort":"11054",
      "command":"READ",
      "commandText":"test1",
      "paramList":null,
      "objectType":"TABLE",
      "objectName":"test1",
      "statementId":65469218,
      "substatementId":2,
      "exitCode":"",
      "sessionId":"726571",
      "rowCount":0,
      "serverHost":"master",
      "serverType":"MySQL",
      "serviceName":"Amazon Aurora MySQL",
      "serverVersion":"MySQL 5.7.12",
      "startTime":"2020-05-22 18:29:57.986364+00",
      "endTime":"2020-05-22 18:29:57.986399+00",
      "transactionId":"0",
      "dbProtocol":"MySQL",
      "netProtocol":"TCP",
      "errorMessage":"",
      "class":"AUX"
    }
  ]
}
```

Oggetto JSON DatabaseActivityMonitoringRecords

I record di eventi dell'attività del database si trovano in un oggetto JSON che contiene le seguenti informazioni.

Campo JSON	Tipo di dati	Descrizione
`type`	stringa	Il tipo di record JSON. Il valore è `DatabaseActivityMonitoringRecords` .
`version`	stringa	La versione dei record di monitoraggio delle attività del database. La versione dei record di attività del database generati dipende dalla versione del motore del cluster database. • I record di attività del database versione 1.1 vengono generati per i cluster di database Aurora PostgreSQL che eseguono le versioni del motore 10.10 e versioni secondarie successive e versioni del motore 11.5 e successive. • I record di attività del database versione 1.0 vengono generati per i cluster di database Aurora PostgreSQL che eseguono le versioni del motore 10.7 e 11.4. Tutti i seguenti campi sono nella versione 1.0 e nella versione 1.1, tranne dove espressamente indicato.
databaseActivityEvents	stringa	Un oggetto JSON contenente gli eventi di attività.

Campo JSON	Tipo di dati	Descrizione
key	stringa	Una chiave di crittografia utilizzata per decrittare Array JSON databaseActivityEventList

Oggetto JSON databaseActivityEvents

L'oggetto JSON `databaseActivityEvents` contiene le seguenti informazioni.

Campi di primo livello nel record JSON

Ogni evento nel registro di controllo viene racchiuso in un record in formato JSON. Questo record contiene i seguenti campi.

type

Questo campo ha sempre il valore `DatabaseActivityMonitoringRecords`.

versione

Questo campo rappresenta la versione del protocollo o del contratto di dati del flusso di attività del database. Definisce quali campi sono disponibili.

La versione 1.0 rappresenta il supporto dei flussi di attività dati originali per le versioni Aurora PostgreSQL 10.7 e 11.4. La versione 1.1 rappresenta il supporto dei flussi di attività dati per le versioni Aurora PostgreSQL 10.10 e successive e Aurora PostgreSQL 11.5 e successive. La versione 1.1 include i campi aggiuntivi `errorMessage` e `startTime`. La versione 1.2 rappresenta il supporto dei flussi di attività dati per Aurora MySQL 2.08 e versioni successive. La versione 1.2 include i campi aggiuntivi `endTime` e `transactionId`.

databaseActivityEvents

Stringa crittografata che rappresenta uno o più eventi di attività. È rappresentato come un array di byte base64. Quando si decrittografa la stringa, il risultato è un record in formato JSON con campi come illustrato negli esempi di questa sezione.

key

Chiave dati crittografata utilizzata per crittografare la stringa `databaseActivityEvents`. Si tratta della stessa AWS KMS key fornita all'avvio del flusso di attività del database.

Nell'esempio seguente viene illustrato il formato di questo record.

```
{
  "type":"DatabaseActivityMonitoringRecords",
  "version":"1.1",
  "databaseActivityEvents":"encrypted audit records",
  "key":"encrypted key"
}
```

Per decrittografare il contenuto del campo `databaseActivityEvents`, procedere come segue:

1. Decrittare il valore nel campo `key` JSON utilizzando la chiave KMS fornita all'avvio del flusso di attività del database. In questo modo viene restituita la chiave di crittografia dei dati in testo non crittografato.
2. Decodificare il valore nel campo `databaseActivityEvents` JSON con base64 per ottenere il testo cifrato, in formato binario, del payload di controllo.
3. Decifrare il testo cifrato binario con la chiave di crittografia dei dati decodificata nel primo passaggio.
4. Decomprimere il payload decrittografato.
   - Il payload crittografato è nel campo `databaseActivityEvents`.
   - Il campo `databaseActivityEventList` contiene una matrice di record di controllo. I campi `type` nella matrice possono essere `record` o `heartbeat`.

Il record dell'evento attività registro di controllo è un oggetto JSON che contiene le seguenti informazioni.

Campo JSON	Tipo di dati	Descrizione
`type`	stringa	Il tipo di record JSON. Il valore è `DatabaseActivityMonitoringRecord` .
`clusterId`	stringa	Identificatore di risorsa cluster di database. Corrisponde all'attributo cluster database `DbClusterResourceId` .
`instanceId`	stringa	Identificatore della risorsa istanza database. Corrisponde all'attributo di istanza database `DbiResourceId` .

Campo JSON	Tipo di dati	Descrizione
Array JSON databaseActivityEventList	stringa	Matrice di record di controllo delle attività o messaggi heartbeat.

Array JSON databaseActivityEventList per flussi di attività del database

Il payload del registro di controllo è un array JSON `databaseActivityEventList` crittografato. Le tabelle riportano in ordine alfabetico i campi per ogni evento di attività nella matrice `DatabaseActivityEventList` decrittata di un log di verifica. I campi sono diversi a seconda che si utilizzi Aurora PostgreSQL o Aurora MySQL. Consultare la tabella che si applica al modulo di gestione di database.

> **Important**
>
> Tale struttura di eventi è soggetta a modifiche. Aurora potrebbe aggiungere nuovi campi agli eventi di attività in futuro. Nelle applicazioni che analizzano i dati JSON, assicurarsi che il codice possa ignorare o eseguire le azioni appropriate per i nomi di campo sconosciuti.

Campi databaseActivityEventList per Aurora PostgreSQL

Di seguito sono riportati i campi di `databaseActivityEventList` per Aurora PostgreSQL.

Campo	Tipo di dati	Descrizione
`class`	stringa	La classe dell'evento attività. I valori validi per Aurora PostgreSQL sono i seguenti. • `ALL` • `CONNECT` – Un evento di connessione o disconnessione. • `DDL` – Un'istruzione DDL che non è inclusa nell'elenco di istruzioni per la classe `ROLE`. • `FUNCTION` – Una chiamata di funzione o un blocco `DO`.

Campo	Tipo di dati	Descrizione
		• `MISC` – Un comando vario quale `DISCARD`, `FETCH`, `CHECKPOINT` o `VACUUM`. • `NONE` • `READ` – Un'istruzione `SELECT` o `COPY` quando l'origine è una relazione o una query. • `ROLE` – Un'istruzione correlata ai ruoli e privilegi inclusi `GRANT`, `REVOKE` e `CREATE/ALTER/DROP ROLE`. • `WRITE` – Un'istruzione `INSERT`, `UPDATE`, `DELETE`, `TRUNCATE` o `COPY` quando la destinazione è una relazione.
`clientApplication`	stringa	L'applicazione utilizzata dal client per eseguire la connessione come segnalato dal client. Il client non deve fornire queste informazioni, pertanto il valore può essere nullo.
`command`	stringa	Il nome del comando SQL senza alcun dettaglio comando.
`commandText`	stringa	L'istruzione SQL effettiva passata dall'utente. Per Aurora PostgreSQL, il valore è identico all'istruzione SQL originale. Questo campo viene utilizzato per tutti i tipi di record tranne che i record di connessione e disconnessione, nel quale caso il valore è nullo. **Important** Il testo SQL completo di ogni istruzione è visibile nel registro di controllo del flusso di attività, inclusi eventuali dati sensibili. Tuttavia, le password degli utenti del database vengono omesse se Aurora può stabilirle dal contesto, come nell'istruzione SQL seguente. `ALTER ROLE role-name WITH password`
`databaseName`	stringa	Il database a cui è connesso l'utente.

Campo	Tipo di dati	Descrizione
`dbProtocol`	stringa	Il protocollo del database, ad esempio `Postgres 3.0`.
`dbUserName`	stringa	L'utente del database con cui il client è autenticato.
`errorMessage` (solo record di attività del database versione 1.1)	stringa	Se si verifica una errore qualsiasi, questo campo viene compilato con il messaggio di errore che verrebbe generato dal server di database. Il valore `errorMessage` è nullo per le istruzioni normali che non hanno generato un errore. Un errore è definito come qualsiasi attività che produce un evento del log degli errori PostgreSQL visibile al client in corrispondenza di un livello di gravità pari a `ERROR` o superiore. Per ulteriori informazioni, consulta la tabella dei livelli di gravità dei messaggi di PostgreSQL. Ad esempio, errori di sintassi e annullamenti delle query generano un messaggio di errore. Gli errori del server PostgreSQL interno, come gli errori del processo checkpointer in background non generano un messaggio di errore. Tuttavia, i record per tali eventi vengono comunque generati a prescindere dall'impostazione del livello di gravità del registro. Ciò impedisce agli aggressori di disattivare la registrazione per tentare di evitare il rilevamento. Vedere anche il campo `exitCode`.

Campo	Tipo di dati	Descrizione
`exitCode`	int	Un valore usato per un record di chiusura sessione. In una clean exit, contiene il codice di chiusura. Un codice di chiusura può sempre essere ottenuto in alcuni scenari di errore. Esempi sono se PostgreSQL esegue un `exit()` o se un operatore esegue un comando quale `kill -9`. Se si verifica un errore qualsiasi, il campo `exitCode` mostra il codice di errore SQL, `SQLSTATE`, come elencato in PostgreSQL Error Codes. Vedere anche il campo `errorMessage` .
`logTime`	stringa	Una timestamp come registrato nel percorso del codice di controllo. Rappresenta l'ora di fine dell'esecuzione dell'istruzione SQL. Vedere anche il campo `startTime` .
`netProtocol`	stringa	Il protocollo di comunicazione di rete.
`objectName`	stringa	Il nome dell'oggetto di database se l'istruzione SQL viene eseguita su uno. Questo campo viene utilizzato solo dove l'istruzione SQL è eseguita su un oggetto di database. Se l'istruzione SQL non è in esecuzione su un oggetto, questo valore è nullo.

Campo	Tipo di dati	Descrizione
objectType	stringa	Il tipo di oggetto di database, ad esempio tabella, indice, vista e così via. Questo campo viene utilizzato solo dove l'istruzione SQL è eseguita su un oggetto di database. Se l'istruzione SQL non è in esecuzione su un oggetto, questo valore è nullo. I valori validi includono i seguenti: • COMPOSITE TYPE • FOREIGN TABLE • FUNCTION • INDEX • MATERIALIZED VIEW • SEQUENCE • TABLE • TOAST TABLE • VIEW • UNKNOWN
paramList	stringa	Un array di parametri separati da virgole passati all'istruzione SQL. Se l'istruzione SQL non dispone di parametri, questo valore è un array vuoto.
pid	int	L'ID di processo del processo back-end allocato per servire la connessione client.
remoteHost	stringa	L'indirizzo IP del client o il nome host. Per Aurora PostgreSQL, quale viene utilizzato dipende dall'impostazione dei parametri log_hostname del database. Il valore di remoteHost include anche [local] e localhost, che indicano l'attività dell'utente rdsadmin.
remotePort	stringa	Il numero di porta del client.

Campo	Tipo di dati	Descrizione
`rowCount`	int	Il numero di righe della tabella influenzate o recuperate dall'istruzione SQL. Questo campo viene utilizzato solo per istruzioni SQL che sono istruzioni DML (Data Manipulation Language). Se l'istruzione SQL non è un'istruzione DML, questo valore è nullo.
`serverHost`	stringa	L'indirizzo IP host del server di database. Il valore di `serverHost` include anche `[local]` e `localhost` , che indicano l'attività dell'utente `rdsadmin`.
`serverType`	stringa	Il tipo di server di database, ad esempio `PostgreSQL` .
`serverVersion`	stringa	La versione del server di database, ad esempio `2.3.1` per Aurora PostgreSQL.
`serviceName`	stringa	Il nome del servizio, ad esempi `Amazon Aurora PostgreSQL-Compatible edition` .
`sessionId`	int	Un identificatore di sessione pseudo-univoco.
`sessionId`	int	Un identificatore di sessione pseudo-univoco.
`startTime` (solo record di attività del database versione 1.1)	stringa	L'ora di inizio dell'esecuzione dell'istruzione SQL. Per calcolare il tempo di esecuzione approssimativo dell'istruzione SQL, utilizzar `logTime - startTime` . Vedere anche il campo `logTime`.
`statementId`	int	Identificatore per l'istruzione SQL del client. Il contatore è a livello di sessione e si incrementa con ogni istruzione SQL immessa dal client.
`substatementId`	int	Identificatore di una subistruzione SQL. Questo valore conteggia le istruzioni secondarie contenute per ogni istruzione SQL identificata dal campo `statementId` .

Campo	Tipo di dati	Descrizione
`type`	stringa	Il tipo di evento, I valori validi sono `record` e `heartbeat` .

Campi databaseActivityEventList per Aurora MySQL

Di seguito sono riportati i campi di `databaseActivityEventList` per Aurora MySQL.

Campo	Tipo di dati	Descrizione
`class`	stringa	La classe dell'evento attività. I valori validi per Aurora MySQL sono i seguenti: • `MAIN` – L' evento primario che rappresenta un'istruzione SQL. • `AUX` – Un evento supplementare contenente ulteriori dettagli. Ad esempio, un'istruzione che rinomina un oggetto potrebbe avere un evento con classe `AUX` che riflette il nuovo nome. Per trovare gli eventi corrispondenti `MAIN` e `AUX` alla stessa istruzione, verificare la presenza di eventi diversi con gli stessi valori per il campo `pid` e per il campo `statement Id` .
`clientApp lication`	stringa	L'applicazione utilizzata dal client per eseguire la connessio ne come segnalato dal client. Il client non deve fornire queste informazioni, pertanto il valore può essere nullo.
`command`	stringa	Categoria generale dell'istruzione SQL. I valori di questo campo dipendono dal valore di `class`. I valori quando `class` è `MAIN` includono quanto segue: • `CONNECT` – Quando una sessione client è connessa. • `QUERY` – Istruzione SQL Accompagnato da uno o più eventi con un valore di `class` pari a `AUX`.

Campo	Tipo di dati	Descrizione
		• DISCONNECT – Quando una sessione client viene disconnessa. • FAILED_CONNECT – Quando un client tenta di connettersi ma non è in grado di farlo. • CHANGEUSER – Un cambiamento di stato che fa parte del protocollo di rete MySQL, non da una dichiarazione rilasciata. I valori quando class è AUX includono quanto segue: • READ – Un'istruzione SELECT o COPY quando l'origine è una relazione o una query. • WRITE – Un'istruzione INSERT, UPDATE, DELETE, TRUNCATE o COPY quando la destinazione è una relazione. • DROP – Eliminazione di un oggetto • CREATE – Creazione di un oggetto. • RENAME – Ridenominazione di un oggetto • ALTER – Per cambiare le proprietà di un oggetto

Campo	Tipo di dati	Descrizione
`commandText`	stringa	Per gli eventi con un valore `class` pari a `MAIN`, questo campo rappresenta l'istruzione SQL effettiva passata dall'utente. Questo campo viene utilizzato per tutti i tipi di record tranne che i record di connessione e disconnessione, nel quale caso il valore è nullo. Per gli eventi con un valore `class` pari a `AUX`, questo campo contiene informazioni supplementari sugli oggetti coinvolti nell'evento. Per Aurora MySQL, i caratteri quali le virgolette sono preceduti da una barra rovesciata che rappresenta un carattere di escape. **Important** Il testo SQL completo di ogni istruzione è visibile nel registro di controllo, inclusi eventuali dati sensibili. Tuttavia, le password degli utenti del database vengono omesse se Aurora può stabilirle dal contesto, come nell'istruzione SQL seguente. `mysql> SET PASSWORD = 'my-password ';` **Note** Specifica una password diversa dal prompt mostrato qui come best practice per la sicurezza.
`databaseName`	stringa	Il database a cui è connesso l'utente.
`dbProtocol`	stringa	Il protocollo di database. Attualmente, questo valore è sempre `MySQL` per Aurora MySQL.

Campo	Tipo di dati	Descrizione
`dbUserName`	stringa	L'utente del database con cui il client è autenticato.
`endTime` (solo record di attività del database versione 1.2)	stringa	L'ora di fine dell'esecuzione dell'istruzione SQL. È rappresentato in formato UTC (Coordinated Universal Time). Per calcolare il tempo di esecuzione dell'istruzione SQL, utilizzare `endTime - startTime` . Vedere anche il campo `startTime` .
`errorMessage` (solo record di attività del database versione 1.1)	stringa	Se si verifica una errore qualsiasi, questo campo viene compilato con il messaggio di errore che verrebbe generato dal server di database. Il valore `errorMessage` è nullo per le istruzioni normali che non hanno generato un errore. Un errore è definito come qualsiasi attività che produce un evento del log degli errori MySQL visibile al client in corrispondenza di un livello di gravità pari a `ERROR` o superiore. Per ulteriori informazioni, consulta il Log degli errori nel Manuale di riferimento di MySQL. Ad esempio, errori di sintassi e annullamenti delle query generano un messaggio di errore. Gli errori del server MySQL interno, come gli errori del processo checkpointer in background non generano un messaggio di errore. Tuttavia, i record per tali eventi vengono comunque generati a prescindere dall'impostazione del livello di gravità del registro. Ciò impedisce agli aggressori di disattivare la registrazione per tentare di evitare il rilevamento. Vedere anche il campo `exitCode`.
`exitCode`	int	Un valore usato per un record di chiusura sessione. In una clean exit, contiene il codice di chiusura. Un codice di chiusura può sempre essere ottenuto in alcuni scenari di errore. In questi casi, questo valore potrebbe essere zero o potrebbe essere vuoto.

Campo	Tipo di dati	Descrizione
`logTime`	stringa	Una timestamp come registrato nel percorso del codice di controllo. È rappresentato in formato UTC (Coordinated Universal Time). Per il modo più accurato per calcolare la durata dell'istruzione, vedere i campi `startTime` e `endTime`.
`netProtocol`	stringa	Il protocollo di comunicazione di rete. Attualmente, questo valore è sempre `TCP` per Aurora MySQL.
`objectName`	stringa	Il nome dell'oggetto di database se l'istruzione SQL viene eseguita su uno. Questo campo viene utilizzato solo dove l'istruzione SQL è eseguita su un oggetto di database. Se l'istruzione SQL non è in esecuzione su un oggetto, questo valore è nullo. Per costruire il nome completo dell'oggetto, combinare `databaseName` e `objectName` . Se la query coinvolge più oggetti, questo campo può essere un elenco di nomi separati da virgole.
`objectType`	stringa	Il tipo di oggetto di database, ad esempio tabella, indice e così via. Questo campo viene utilizzato solo dove l'istruzione SQL è eseguita su un oggetto di database. Se l'istruzione SQL non è in esecuzione su un oggetto, questo valore è nullo. I valori validi per Aurora MySQL includono i seguenti: • `INDEX` • `TABLE` • `UNKNOWN`
`paramList`	stringa	Questo campo non viene utilizzato per Aurora MySQL ed è sempre nullo.

Campo	Tipo di dati	Descrizione
`pid`	int	L'ID di processo del processo back-end allocato per servire la connessione client. Quando il server di database viene riavviato, le modifiche `pid` e il contatore per il campo `statementId` ricomincia.
`remoteHost`	stringa	L'indirizzo IP o il nome host del client che ha emesso l'istruzione SQL. Per Aurora MySQL, quale viene utilizzato dipende dall'impostazione dei parametri `skip_name_resolve` del database. Il valore `localhost` indica l'attività dell'utente `rdsadmin` speciale.
`remotePort`	stringa	Il numero di porta del client.
`rowCount`	int	Numero di righe restituite dall'istruzione SQL. Ad esempio, se un'istruzione SELECT restituisce 10 righe, RowCount è 10. Per le istruzioni INSERT o UPDATE, RowCount è 0.
`serverHost`	stringa	Identificatore dell'istanza del server di database.
`serverType`	stringa	Il tipo di server di database, ad esempio `MySQL`.
`serverVersion`	stringa	La versione del server di database. Attualmente, questo valore è sempre `MySQL 5.7.12` per Aurora MySQL.
`serviceName`	stringa	Il nome del servizio Attualmente, questo valore è sempre `Amazon Aurora MySQL` per Aurora MySQL.
`sessionId`	int	Un identificatore di sessione pseudo-univoco.
`startTime` (solo record di attività del database versione 1.1)	stringa	L'ora di inizio dell'esecuzione dell'istruzione SQL. È rappresentato in formato UTC (Coordinated Universal Time). Per calcolare il tempo di esecuzione dell'istruzione SQL, utilizzare `endTime - startTime` . Vedere anche il campo `endTime`.

Campo	Tipo di dati	Descrizione
`statementId`	int	Identificatore per l'istruzione SQL del client. Il contatore aumenta con ogni istruzione SQL immessa dal client. Il contatore viene reimpostato quando viene riavviata l'istanza database.
`substatementId`	int	Identificatore di una subistruzione SQL. Questo valore è 1 per gli eventi con classe `MAIN` e 2 per gli eventi con classe `AUX`. Utilizzare il campo `statementId` per identificare tutti gli eventi generati dalla stessa istruzione.
`transactionId` (solo record di attività del database versione 1.2)	int	Identificatore di una transazione.
`type`	stringa	Il tipo di evento, I valori validi sono `record` e `heartbeat` .

Elaborazione di un flusso di attività del database utilizzando l'SDK AWS

Puoi elaborare un flusso di attività a livello di programmazione utilizzando l'SDK AWS. Di seguito sono riportati esempi Java e Python completamente funzionanti dell'elaborazione del flusso di dati Kinesis.

Java

```
import java.io.ByteArrayInputStream;
import java.io.ByteArrayOutputStream;
import java.io.IOException;
import java.net.InetAddress;
import java.nio.ByteBuffer;
import java.nio.charset.StandardCharsets;
import java.security.NoSuchAlgorithmException;
import java.security.NoSuchProviderException;
import java.security.Security;
```

```
import java.util.HashMap;
import java.util.List;
import java.util.Map;
import java.util.UUID;
import java.util.zip.GZIPInputStream;

import javax.crypto.Cipher;
import javax.crypto.NoSuchPaddingException;
import javax.crypto.spec.SecretKeySpec;

import com.amazonaws.auth.AWSStaticCredentialsProvider;
import com.amazonaws.auth.BasicAWSCredentials;
import com.amazonaws.encryptionsdk.AwsCrypto;
import com.amazonaws.encryptionsdk.CryptoInputStream;
import com.amazonaws.encryptionsdk.jce.JceMasterKey;
import
 com.amazonaws.services.kinesis.clientlibrary.exceptions.InvalidStateException;
import com.amazonaws.services.kinesis.clientlibrary.exceptions.ShutdownException;
import com.amazonaws.services.kinesis.clientlibrary.exceptions.ThrottlingException;
import com.amazonaws.services.kinesis.clientlibrary.interfaces.IRecordProcessor;
import
 com.amazonaws.services.kinesis.clientlibrary.interfaces.IRecordProcessorCheckpointer;
import
 com.amazonaws.services.kinesis.clientlibrary.interfaces.IRecordProcessorFactory;
import
 com.amazonaws.services.kinesis.clientlibrary.lib.worker.InitialPositionInStream;
import
 com.amazonaws.services.kinesis.clientlibrary.lib.worker.KinesisClientLibConfiguration;
import com.amazonaws.services.kinesis.clientlibrary.lib.worker.ShutdownReason;
import com.amazonaws.services.kinesis.clientlibrary.lib.worker.Worker;
import com.amazonaws.services.kinesis.clientlibrary.lib.worker.Worker.Builder;
import com.amazonaws.services.kinesis.model.Record;
import com.amazonaws.services.kms.AWSKMS;
import com.amazonaws.services.kms.AWSKMSClientBuilder;
import com.amazonaws.services.kms.model.DecryptRequest;
import com.amazonaws.services.kms.model.DecryptResult;
import com.amazonaws.util.Base64;
import com.amazonaws.util.IOUtils;
import com.google.gson.Gson;
import com.google.gson.GsonBuilder;
import com.google.gson.annotations.SerializedName;
import org.bouncycastle.jce.provider.BouncyCastleProvider;

public class DemoConsumer {
```

```
    private static final String STREAM_NAME = "aws-rds-das-[cluster-external-
resource-id]";
    private static final String APPLICATION_NAME = "AnyApplication"; //unique
 application name for dynamo table generation that holds kinesis shard tracking
    private static final String AWS_ACCESS_KEY =
 "[AWS_ACCESS_KEY_TO_ACCESS_KINESIS]";
    private static final String AWS_SECRET_KEY =
 "[AWS_SECRET_KEY_TO_ACCESS_KINESIS]";
    private static final String DBC_RESOURCE_ID = "[cluster-external-resource-id]";
    private static final String REGION_NAME = "[region-name]"; //us-east-1, us-
east-2...
    private static final BasicAWSCredentials CREDENTIALS = new
 BasicAWSCredentials(AWS_ACCESS_KEY, AWS_SECRET_KEY);
    private static final AWSStaticCredentialsProvider CREDENTIALS_PROVIDER = new
 AWSStaticCredentialsProvider(CREDENTIALS);

    private static final AwsCrypto CRYPTO = new AwsCrypto();
    private static final AWSKMS KMS = AWSKMSClientBuilder.standard()
            .withRegion(REGION_NAME)
            .withCredentials(CREDENTIALS_PROVIDER).build();

    class Activity {
        String type;
        String version;
        String databaseActivityEvents;
        String key;
    }

    class ActivityEvent {
        @SerializedName("class") String _class;
        String clientApplication;
        String command;
        String commandText;
        String databaseName;
        String dbProtocol;
        String dbUserName;
        String endTime;
        String errorMessage;
        String exitCode;
        String logTime;
        String netProtocol;
        String objectName;
        String objectType;
```

```
        List<String> paramList;
        String pid;
        String remoteHost;
        String remotePort;
        String rowCount;
        String serverHost;
        String serverType;
        String serverVersion;
        String serviceName;
        String sessionId;
        String startTime;
        String statementId;
        String substatementId;
        String transactionId;
        String type;
    }

    class ActivityRecords {
        String type;
        String clusterId;
        String instanceId;
        List<ActivityEvent> databaseActivityEventList;
    }

    static class RecordProcessorFactory implements IRecordProcessorFactory {
        @Override
        public IRecordProcessor createProcessor() {
            return new RecordProcessor();
        }
    }

    static class RecordProcessor implements IRecordProcessor {

        private static final long BACKOFF_TIME_IN_MILLIS = 3000L;
        private static final int PROCESSING_RETRIES_MAX = 10;
        private static final long CHECKPOINT_INTERVAL_MILLIS = 60000L;
        private static final Gson GSON = new
GsonBuilder().serializeNulls().create();

        private static final Cipher CIPHER;
        static {
            Security.insertProviderAt(new BouncyCastleProvider(), 1);
            try {
                CIPHER = Cipher.getInstance("AES/GCM/NoPadding", "BC");
```

```
            } catch (NoSuchAlgorithmException | NoSuchPaddingException |
NoSuchProviderException e) {
                throw new ExceptionInInitializerError(e);
            }
        }

        private long nextCheckpointTimeInMillis;

        @Override
        public void initialize(String shardId) {
        }

        @Override
        public void processRecords(final List<Record> records, final
IRecordProcessorCheckpointer checkpointer) {
            for (final Record record : records) {
                processSingleBlob(record.getData());
            }

            if (System.currentTimeMillis() > nextCheckpointTimeInMillis) {
                checkpoint(checkpointer);
                nextCheckpointTimeInMillis = System.currentTimeMillis() +
CHECKPOINT_INTERVAL_MILLIS;
            }
        }

        @Override
        public void shutdown(IRecordProcessorCheckpointer checkpointer,
ShutdownReason reason) {
            if (reason == ShutdownReason.TERMINATE) {
                checkpoint(checkpointer);
            }
        }

        private void processSingleBlob(final ByteBuffer bytes) {
            try {
                // JSON $Activity
                final Activity activity = GSON.fromJson(new String(bytes.array(),
StandardCharsets.UTF_8), Activity.class);

                // Base64.Decode
                final byte[] decoded =
Base64.decode(activity.databaseActivityEvents);
                final byte[] decodedDataKey = Base64.decode(activity.key);
```

```
                Map<String, String> context = new HashMap<>();
                context.put("aws:rds:dbc-id", DBC_RESOURCE_ID);

                // Decrypt
                final DecryptRequest decryptRequest = new DecryptRequest()

.withCiphertextBlob(ByteBuffer.wrap(decodedDataKey)).withEncryptionContext(context);
                final DecryptResult decryptResult = KMS.decrypt(decryptRequest);
                final byte[] decrypted = decrypt(decoded,
getByteArray(decryptResult.getPlaintext()));

                // GZip Decompress
                final byte[] decompressed = decompress(decrypted);
                // JSON $ActivityRecords
                final ActivityRecords activityRecords = GSON.fromJson(new
String(decompressed, StandardCharsets.UTF_8), ActivityRecords.class);

                // Iterate throught $ActivityEvents
                for (final ActivityEvent event :
activityRecords.databaseActivityEventList) {
                    System.out.println(GSON.toJson(event));
                }
            } catch (Exception e) {
                // Handle error.
                e.printStackTrace();
            }
        }

        private static byte[] decompress(final byte[] src) throws IOException {
            ByteArrayInputStream byteArrayInputStream = new
ByteArrayInputStream(src);
            GZIPInputStream gzipInputStream = new
GZIPInputStream(byteArrayInputStream);
            return IOUtils.toByteArray(gzipInputStream);
        }

        private void checkpoint(IRecordProcessorCheckpointer checkpointer) {
            for (int i = 0; i < PROCESSING_RETRIES_MAX; i++) {
                try {
                    checkpointer.checkpoint();
                    break;
                } catch (ShutdownException se) {
```

```
                    // Ignore checkpoint if the processor instance has been shutdown
(fail over).
                    System.out.println("Caught shutdown exception, skipping
checkpoint." + se);
                    break;
                } catch (ThrottlingException e) {
                    // Backoff and re-attempt checkpoint upon transient failures
                    if (i >= (PROCESSING_RETRIES_MAX - 1)) {
                        System.out.println("Checkpoint failed after " + (i + 1) +
"attempts." + e);
                        break;
                    } else {
                        System.out.println("Transient issue when checkpointing -
attempt " + (i + 1) + " of " + PROCESSING_RETRIES_MAX + e);
                    }
                } catch (InvalidStateException e) {
                    // This indicates an issue with the DynamoDB table (check for
table, provisioned IOPS).
                    System.out.println("Cannot save checkpoint to the DynamoDB table
used by the Amazon Kinesis Client Library." + e);
                    break;
                }
                try {
                    Thread.sleep(BACKOFF_TIME_IN_MILLIS);
                } catch (InterruptedException e) {
                    System.out.println("Interrupted sleep" + e);
                }
            }
        }
    }

    private static byte[] decrypt(final byte[] decoded, final byte[] decodedDataKey)
throws IOException {
        // Create a JCE master key provider using the random key and an AES-GCM
encryption algorithm
        final JceMasterKey masterKey = JceMasterKey.getInstance(new
SecretKeySpec(decodedDataKey, "AES"),
                "BC", "DataKey", "AES/GCM/NoPadding");
        try (final CryptoInputStream<JceMasterKey> decryptingStream =
CRYPTO.createDecryptingStream(masterKey, new ByteArrayInputStream(decoded));
             final ByteArrayOutputStream out = new ByteArrayOutputStream()) {
            IOUtils.copy(decryptingStream, out);
            return out.toByteArray();
        }
```

```
    }

    public static void main(String[] args) throws Exception {
        final String workerId = InetAddress.getLocalHost().getCanonicalHostName() +
 ":" + UUID.randomUUID();
        final KinesisClientLibConfiguration kinesisClientLibConfiguration =
                new KinesisClientLibConfiguration(APPLICATION_NAME, STREAM_NAME,
 CREDENTIALS_PROVIDER, workerId);

 kinesisClientLibConfiguration.withInitialPositionInStream(InitialPositionInStream.LATEST);
        kinesisClientLibConfiguration.withRegionName(REGION_NAME);
        final Worker worker = new Builder()
                .recordProcessorFactory(new RecordProcessorFactory())
                .config(kinesisClientLibConfiguration)
                .build();

        System.out.printf("Running %s to process stream %s as worker %s...\n",
 APPLICATION_NAME, STREAM_NAME, workerId);

        try {
            worker.run();
        } catch (Throwable t) {
            System.err.println("Caught throwable while processing data.");
            t.printStackTrace();
            System.exit(1);
        }
        System.exit(0);
    }

    private static byte[] getByteArray(final ByteBuffer b) {
        byte[] byteArray = new byte[b.remaining()];
        b.get(byteArray);
        return byteArray;
    }
}
```

Python

```
import base64
import json
import zlib
import aws_encryption_sdk
from aws_encryption_sdk import CommitmentPolicy
```

```
from aws_encryption_sdk.internal.crypto import WrappingKey
from aws_encryption_sdk.key_providers.raw import RawMasterKeyProvider
from aws_encryption_sdk.identifiers import WrappingAlgorithm, EncryptionKeyType
import boto3

REGION_NAME = '<region>'                    # us-east-1
RESOURCE_ID = '<external-resource-id>'      # cluster-ABCD123456
STREAM_NAME = 'aws-rds-das-' + RESOURCE_ID  # aws-rds-das-cluster-ABCD123456

enc_client =
 aws_encryption_sdk.EncryptionSDKClient(commitment_policy=CommitmentPolicy.FORBID_ENCRYPT_AL

class MyRawMasterKeyProvider(RawMasterKeyProvider):
    provider_id = "BC"

    def __new__(cls, *args, **kwargs):
        obj = super(RawMasterKeyProvider, cls).__new__(cls)
        return obj

    def __init__(self, plain_key):
        RawMasterKeyProvider.__init__(self)
        self.wrapping_key =
 WrappingKey(wrapping_algorithm=WrappingAlgorithm.AES_256_GCM_IV12_TAG16_NO_PADDING,
                                        wrapping_key=plain_key,
 wrapping_key_type=EncryptionKeyType.SYMMETRIC)

    def _get_raw_key(self, key_id):
        return self.wrapping_key


def decrypt_payload(payload, data_key):
    my_key_provider = MyRawMasterKeyProvider(data_key)
    my_key_provider.add_master_key("DataKey")
    decrypted_plaintext, header = enc_client.decrypt(
        source=payload,

 materials_manager=aws_encryption_sdk.materials_managers.default.DefaultCryptoMaterialsManag
    return decrypted_plaintext


def decrypt_decompress(payload, key):
    decrypted = decrypt_payload(payload, key)
    return zlib.decompress(decrypted, zlib.MAX_WBITS + 16)
```

```
def main():
    session = boto3.session.Session()
    kms = session.client('kms', region_name=REGION_NAME)
    kinesis = session.client('kinesis', region_name=REGION_NAME)

    response = kinesis.describe_stream(StreamName=STREAM_NAME)
    shard_iters = []
    for shard in response['StreamDescription']['Shards']:
        shard_iter_response = kinesis.get_shard_iterator(StreamName=STREAM_NAME,
 ShardId=shard['ShardId'],

 ShardIteratorType='LATEST')
        shard_iters.append(shard_iter_response['ShardIterator'])

    while len(shard_iters) > 0:
        next_shard_iters = []
        for shard_iter in shard_iters:
            response = kinesis.get_records(ShardIterator=shard_iter, Limit=10000)
            for record in response['Records']:
                record_data = record['Data']
                record_data = json.loads(record_data)
                payload_decoded =
 base64.b64decode(record_data['databaseActivityEvents'])
                data_key_decoded = base64.b64decode(record_data['key'])
                data_key_decrypt_result =
 kms.decrypt(CiphertextBlob=data_key_decoded,

 EncryptionContext={'aws:rds:dbc-id': RESOURCE_ID})
                print (decrypt_decompress(payload_decoded,
 data_key_decrypt_result['Plaintext']))
            if 'NextShardIterator' in response:
                next_shard_iters.append(response['NextShardIterator'])
        shard_iters = next_shard_iters


if __name__ == '__main__':
    main()
```

Esempi di policy IAM per flussi di attività del database

Qualsiasi utente con privilegi di ruolo AWS Identity and Access Management (IAM) appropriati per i flussi di attività del database può creare, avviare, interrompere e modificare le impostazioni del flusso di attività per un'istanza DB del cluster . Queste operazioni sono incluse nel registro di controllo del flusso. Per le migliori pratiche di conformità, ti consigliamo di non fornire questi privilegi a. DBAs

Imposta l'accesso ai flussi di attività di database utilizzando policy IAM. Per ulteriori informazioni sull'autenticazione di Aurora, consulta Gestione accessi e identità per Amazon Aurora. Per ulteriori informazioni sulla creazione di policy IAM, consulta Creazione e utilizzo di una policy IAM per l'accesso al database IAM.

Example Policy per consentire la configurazione dei flussi di attività di database

Per fornire agli utenti l'accesso fine-grained per modificare i flussi di attività, utilizza la chiave di contesto dell'operazione specifica del servizio `rds:StartActivityStream` e `rds:StopActivityStream` in una policy IAM. L'esempio di policy IAM seguente consente a un utente o ruolo di configurare i flussi di attività.

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Sid": "ConfigureActivityStreams",
            "Effect": "Allow",
            "Action": [
                "rds:StartActivityStream",
                "rds:StopActivityStream"
            ],
            "Resource": "*"
        }
    ]
}
```

Example Policy per consentire l'avvio di flussi di attività di database

L'esempio di policy IAM seguente consente a un utente o ruolo di avviare i flussi di attività.

JSON

```
{
    "Version":"2012-10-17",
    "Statement":[
        {
            "Sid":"AllowStartActivityStreams",
            "Effect":"Allow",
            "Action":"rds:StartActivityStream",
            "Resource":"*"
        }
    ]
}
```

Example Policy per consentire l'interruzione di flussi di attività di database

L'esempio di policy IAM seguente consente a un utente o ruolo di interrompere i flussi di attività.

JSON

```
{
    "Version":"2012-10-17",
    "Statement":[
        {
            "Sid":"AllowStopActivityStreams",
            "Effect":"Allow",
            "Action":"rds:StopActivityStream",
            "Resource":"*"
        }
     ]
}
```

Example Policy per rifiutare l'avvio di flussi di attività di database

L'esempio di policy IAM seguente consente a un utente o ruolo di rifiutare l'avvio di flussi di attività.

JSON

```
{
```

```
    "Version":"2012-10-17",
    "Statement":[
        {
            "Sid":"DenyStartActivityStreams",
            "Effect":"Deny",
            "Action":"rds:StartActivityStream",
            "Resource":"*"
        }
     ]
}
```

Example Policy per rifiutare l'interruzione di flussi di attività di database

L'esempio di policy IAM seguente consente a un utente o ruolo di rifiutare l'interruzione di flussi di attività.

JSON

```
{
    "Version":"2012-10-17",
    "Statement":[
        {
            "Sid":"DenyStopActivityStreams",
            "Effect":"Deny",
            "Action":"rds:StopActivityStream",
            "Resource":"*"
        }
    ]
}
```

Monitoraggio delle minacce con la protezione RDS Amazon GuardDuty per Amazon Aurora

Amazon GuardDuty è un servizio di rilevamento delle minacce che aiuta a proteggere gli account, i container, i carichi di lavoro e i dati nell'ambiente AWS. Grazie a modelli di machine learning (ML) e funzionalità di rilevamento di anomalie e minacce, GuardDuty monitora continuamente origini di log e attività di runtime diverse per identificare e dare priorità ai potenziali rischi per la sicurezza e alle attività dannose nell'ambiente.

La protezione RDS GuardDuty analizza e profila gli eventi di accesso per individuare potenziali minacce di accesso ai database Amazon Aurora. Quando attivi la protezione RDS, GuardDuty utilizza gli eventi di accesso RDS dei database Aurora. La protezione RDS monitora questi eventi e li profila per potenziali minacce interne o attori esterni.

Per maggiori informazioni sull'abilitazione della protezione RDS GuardDuty, consulta Protezione GuardDuty RDS nella Guida per l'utente di Amazon GuardDuty.

Quando la protezione RDS rileva una potenziale minaccia, ad esempio uno schema insolito di tentativi di accesso riusciti o meno, GuardDuty genera un nuovo risultato con i dettagli sul database potenzialmente compromesso. Puoi visualizzare i dettagli dell'esito nella sezione di riepilogo dell'esito nella console Amazon GuardDuty. I dettagli dell'esito variano in base al tipo di esito. I dettagli principali, il tipo di risorsa e il ruolo della risorsa determinano il tipo di informazioni disponibili per qualsiasi esito. Per ulteriori informazioni sui dettagli comunemente disponibili sugli esiti e sui tipi di esito, consulta rispettivamente Dettagli degli esiti e Tipi di esito di Protezione GuardDuty RDS nella Guida per l'utente di Amazon GuardDuty.

Puoi attivare o disattivare la funzionalità di protezione RDS per qualsiasi Account AWSin ogni Regione AWS sia disponibile la funzionalità. Quando la protezione RDS non è abilitata, GuardDuty non rileva i database Aurora potenzialmente compromessi né fornisce i dettagli della compromissione.

Un account GuardDuty esistente può abilitare la protezione RDS con un periodo di prova di 30 giorni. Per un nuovo account GuardDuty, la protezione RDS è già abilitata e inclusa nel periodo di prova gratuito di 30 giorni. Per maggiori informazioni, consulta Stima dei costi di GuardDuty nella Guida per l'utente di Amazon GuardDuty.

Per informazioni sulle Regioni AWS in cui GuardDuty non supporta ancora la protezione RDS, consulta Disponibilità di funzionalità specifiche per ogni Regione nella Guida per l'utente di Amazon GuardDuty.

Per informazioni sulle versioni di database Aurora supportate dalla protezione RDS GuardDuty, consulta Database Amazon Aurora, Amazon RDS e Aurora Limitless supportati nella Guida per l'utente di Amazon GuardDuty.

Utilizzo di Amazon Aurora MySQL

Amazon Aurora MySQL è un motore del database relazionale completamente gestito compatibile con MySQL che unisce la velocità e l'affidabilità dei database commerciali di fascia alta alla semplicità e al costo ridotto dei database open source. Aurora MySQL è un sostituto drop-in di MySQL e semplifica e rende più conveniente dal punto di vista dei costi la configurazione, l'utilizzo e il dimensionamento delle implementazioni MySQL nuove ed esistenti, consentendo di concentrarti sulle tue attività e applicazioni. Amazon RDS fornisce l'amministrazione di Aurora gestendo le attività di database di routine, come il provisioning, l'applicazione di patch, il backup, il ripristino, il rilevamento degli errori e la correzione. Amazon RDS fornisce strumenti di migrazione per convertire, con la semplice pressione di un pulsante, le applicazioni Amazon RDS for MySQL e Amazon RDS for PostgreSQL in Aurora MySQL.

Argomenti

- Panoramica di Amazon Aurora MySQL
- Sicurezza con Amazon Aurora MySQL
- Aggiornamento delle applicazioni per la connessione ai cluster database Aurora MySQL utilizzando nuovi certificati TLS
- Utilizzo dell'autenticazione Kerberos per Aurora MySQL
- Migrazione di dati a un cluster di database Amazon Aurora MySQL
- Gestione di Amazon Aurora MySQL
- Ottimizzazione di Aurora MySQL
- Query parallela per Amazon Aurora MySQL
- Utilizzo dell'audit avanzato con un cluster di database Amazon Aurora MySQL
- Replica con Amazon Aurora MySQL
- Utilizzo dell'inoltro di scrittura locale in un cluster database Amazon Aurora MySQL
- Integrazione di Amazon Aurora MySQL con altri servizi AWS
- Modalità di laboratorio per Amazon Aurora MySQL
- Best practice con Amazon Aurora MySQL
- Risoluzione dei problemi alle prestazioni del database Amazon Aurora MySQL
- Riferimento Amazon Aurora MySQL
- Aggiornamenti del motore del database per Amazon Aurora MySQL

Panoramica di Amazon Aurora MySQL

Nelle sezioni seguenti viene offerta una panoramica di Amazon Aurora MySQL.

Argomenti

- Miglioramenti alle prestazioni di Amazon Aurora MySQL
- Amazon Aurora MySQL e dati spaziali
- Aurora MySQL versione 3 compatibile con MySQL 8.0
- Aurora MySQL versione 2 compatibile con MySQL 5.7

Miglioramenti alle prestazioni di Amazon Aurora MySQL

Amazon Aurora include opzioni che migliorano le prestazioni per supportare le differenti necessità dei database commerciali di fascia alta.

Inserimento rapido

L'inserimento rapido accelera gli inserimenti paralleli ordinati in base alla chiave primaria e si applica specificamente alle istruzioni `LOAD DATA` e `INSERT INTO ... SELECT ...`. L'inserimento rapido memorizza nella cache la posizione del cursore in un indice traversale mentre esegue l'istruzione. Evita l'attraversamento inutile nuovamente dell'indice.

L’inserimento rapido è abilitato solo per le tabelle InnoDB standard in Aurora MySQL versione 3.03.2 e successive. Questa ottimizzazione non funziona per le tabelle temporanee di InnoDB. È disabilitata in Aurora MySQL versione 2 per tutte le versioni 2.11 e 2.12. L’ottimizzazione dell’inserimento rapido funziona solo se l’ottimizzazione di Adaptive Hash Index è disabilitata.

Puoi monitorare i seguenti parametri per determinare l'efficacia dell'inserimento rapido per il cluster di database:

- `aurora_fast_insert_cache_hits`: un contatore che viene aumentato quando il cursore memorizzato nella cache viene recuperato e verificato.
- `aurora_fast_insert_cache_misses`: un contatore che viene aumentato quando il cursore memorizzato non è più valido e Aurora esegue un attraversamento di indice normale.

Puoi recuperare il valore corrente dei parametri di inserimento rapido utilizzando il seguente comando:

```
mysql> show global status like 'Aurora_fast_insert%';
```

Otterrai un output simile al seguente:

```
+--------------------------------+-----------+
| Variable_name                  | Value     |
+--------------------------------+-----------+
| Aurora_fast_insert_cache_hits  | 3598300   |
| Aurora_fast_insert_cache_misses | 436401336 |
+--------------------------------+-----------+
```

Amazon Aurora MySQL e dati spaziali

Nell'elenco seguente vengono riepilogate le principali caratteristiche spaziali di Aurora MySQL e descritto come queste corrispondono alle caratteristiche spaziali in MySQL:

- Aurora MySQL versione 2 supporta gli stessi tipi di dati spaziali e funzioni di relazioni spaziali di MySQL 5.7. Per ulteriori informazioni su questi tipi di dati e funzioni, consulta Tipi di dati spaziali e Funzioni di relazione spaziale nella documentazione MySQL 5.7.
- Aurora MySQL versione 3 supporta gli stessi tipi di dati spaziali e funzioni di relazioni spaziali di MySQL 8.0. Per ulteriori informazioni su questi tipi di dati e funzioni, consulta Tipi di dati spaziali e Funzioni di relazione spaziale nella documentazione MySQL 8.0.
- Aurora MySQL supporta l'indicizzazione spaziale su tabelle InnoDB. L'indicizzazione spaziale consente di migliorare le prestazioni su set di dati di grandi dimensioni per le query sui dati spaziali. In MySQL, l'indicizzazione spaziale per le tabelle InnoDB è disponibile in MySQL 5.7 e 8.0.

  Aurora MySQL usa una diversa strategia di indicizzazione spaziale rispetto a MySQL per alte prestazioni con le query spaziali. L'implementazione dell'indice spaziale Aurora utilizza una curva di riempimento dello spazio in un albero B, finalizzata a fornire prestazioni più elevate per scansioni degli intervalli spaziali rispetto a un albero R.

  > **Note**
  >
  > In Aurora MySQL, una transazione su una tabella con un indice spaziale definito su una colonna con un identificatore di riferimento spaziale (SRID) non può essere inserita in un'area selezionata per l'aggiornamento da un'altra transazione.

Le seguenti istruzioni DDL sono supportate per la creazione di indici su colonne che utilizzano tipi di dati spaziali.

CREATE TABLE

Puoi utilizzare le parole chiave `SPATIAL INDEX` in un'istruzione `CREATE TABLE` per aggiungere un indice spaziale a una colonna in una nuova tabella. Di seguito è riportato un esempio.

```
CREATE TABLE test (shape POLYGON NOT NULL, SPATIAL INDEX(shape));
```

ALTER TABLE

Puoi utilizzare le parole chiave `SPATIAL INDEX` in un'istruzione `ALTER TABLE` per aggiungere un indice spaziale a una colonna in una tabella esistente. Di seguito è riportato un esempio.

```
ALTER TABLE test ADD SPATIAL INDEX(shape);
```

CREATE INDEX

Puoi utilizzare la parola chiave `SPATIAL` in un'istruzione `CREATE INDEX` per aggiungere un indice spaziale a una colonna in una tabella esistente. Di seguito è riportato un esempio.

```
CREATE SPATIAL INDEX shape_index ON test (shape);
```

Aurora MySQL versione 3 compatibile con MySQL 8.0

È possibile utilizzare Aurora MySQL versione 3 per ottenere le più recenti funzionalità compatibili con MySQL, miglioramenti delle prestazioni e correzioni di bug. Di seguito, è possibile conoscere Aurora MySQL versione 3, con compatibilità MySQL 8.0. Puoi imparare come aggiornare cluster e applicazioni ad Aurora MySQL versione 3.

Alcune caratteristiche di Aurora, come Aurora Serverless v2, richiedono Aurora MySQL versione 3.

Argomenti

- Funzionalità della community MySQL 8.0
- Aurora MySQL versione 3 prerequisito per Aurora MySQL Serverless v2
- Note di rilascio di Aurora MySQL versione 3
- Nuove ottimizzazioni delle query parallele
- Ottimizzazioni per ridurre i tempi di riavvio del database

- Nuovo comportamento della tabella temporanea in Aurora MySQL versione 3
- Confronto tra Aurora MySQL versione 2 e Aurora MySQL versione 3
- Confronto tra Aurora MySQL versione 3 e MySQL 8.0 Community Edition
- Aggiornamento ad Aurora MySQL versione 3

Funzionalità della community MySQL 8.0

La versione iniziale di Aurora MySQL versione 3 è compatibile con la community MySQL 8.0.23. MySQL 8.0 introduce diverse nuove funzionalità, tra cui:

- Supporto per Atomic DDL (Data Definition Language). Per ulteriori informazioni, consulta Supporto per Atomic DDL (Data Definition Language)..
- Funzioni JSON. Per informazioni sull'utilizzo, consulta Funzioni JSON nel Manuale di riferimento MySQL.
- Funzioni finestra Per informazioni sull'utilizzo, consulta Funzioni finestra nel Manuale di riferimento MySQL.
- Espressioni di tabella comuni (CTE), utilizzando la clausola `WITH`. Per informazioni sull'utilizzo, consulta WITH (espressioni di tabella comuni) nel Manuale di riferimento MySQL.
- Ottimizzazione `ADD COLUMN` e clausole `RENAME COLUMN` per l'istruzione `ALTER TABLE`. Queste ottimizzazioni sono chiamate «DDL istantaneo». Aurora MySQL versione 3 è compatibile con la funzionalità DDL istantanea MySQL della community. L'ex caratteristica DDL veloce Aurora non viene utilizzata. Per informazioni sull'utilizzo del DDL istantaneo, vedere DDL istantaneo (Aurora MySQL versione 3).
- Indici discendenti, funzionali e invisibili. Per informazioni sull'utilizzo, consulta Indici invisibili,Indici discendenti, e Istruzioni CREATE INDEX nel Manuale di riferimento MySQL.
- Privilegi basati sui ruoli controllati tramite istruzioni SQL. Per ulteriori informazioni sulle modifiche al modello di privilegi, consulta Privilegio basato sui ruoli.
- clausole `NOWAIT` e `SKIP LOCKED` con istruzioni `SELECT ... FOR SHARE`. Queste clausole evitano di attendere che altre transazioni rilascino blocchi di riga. Per informazioni sull'utilizzo, consultare Blocco letture, nel manuale di riferimento di MySQL.
- Miglioramenti alla replica dei log binari (binlog). Per i dettagli di Aurora MySQL, consultare Replica dei log binari. In particolare, è possibile eseguire la replica filtrata. Per informazioni sull'utilizzo sulla replica filtrata, vedere Come i server valutano le regole di filtro delle repliche nel Manuale di riferimento MySQL.

- Suggerimenti. Alcuni dei suggerimenti compatibili con MySQL 8.0 erano già stati sottoposti a backport su Aurora MySQL versione 2. Per ulteriori informazioni sulla sicurezza con Aurora MySQL, consultare Suggerimenti di Aurora MySQL. Per l'elenco completo dei suggerimenti nella community MySQL 8.0, consultaSuggerimenti di ottimizzazione nel Manuale di riferimento MySQL.

Per l'elenco completo delle funzionalità aggiunte all'edizione della community di MySQL 8.0, consulta il post del blog L'elenco completo delle nuove funzionalità di MySQL 8.0.

Aurora MySQL versione 3 include anche modifiche alle parole chiave per un linguaggio inclusivo, con backport dalla community MySQL 8.0.26. Per i dettagli su tali modifiche, consultare Cambiamenti linguistici inclusivi per Aurora MySQL versione 3.

Aurora MySQL versione 3 prerequisito per Aurora MySQL Serverless v2

Aurora MySQL versione 3 è un prerequisito per tutte le istanze database in un cluster Aurora MySQL Serverless v2. Aurora MySQL Serverless v2 include il supporto per le istanze di lettura in un cluster di database e altre caratteristiche di Aurora che non sono disponibili per Aurora MySQL Serverless v1. Dispone anche di un dimensionamento più rapido e granulare rispetto ad Aurora MySQL Serverless v1.

Note di rilascio di Aurora MySQL versione 3

Per le Note di rilascio di tutte le release di Aurora MySQL versione 3, consultare Aggiornamenti del motore del database per Amazon Aurora MySQL versione 3 nelle Note di rilascio di Aurora MySQL.

Nuove ottimizzazioni delle query parallele

L'ottimizzazione delle query parallele Aurora ora si applica a più operazioni SQL:

- La query parallela ora si applica alle tabelle contenenti i tipi di dati `TEXT`, `BLOB`, `JSON`, `GEOMETRY`, e `VARCHAR` e `CHAR` superiori a 768 byte.
- La query parallela può ottimizzare le query che coinvolgono tabelle partizionate.
- La query parallela può ottimizzare le query che coinvolgono chiamate di funzioni aggregate nell'elenco di selezione e nella clausola `HAVING`.

Per ulteriori informazioni su questi miglioramenti, consultare Aggiornare cluster di query paralleli a Aurora MySQL versione 3. Per informazioni generali sulle query parallele Aurora, consultare Query parallela per Amazon Aurora MySQL.

Ottimizzazioni per ridurre i tempi di riavvio del database

Il cluster di database Aurora MySQL deve essere a disponibilità elevata durante le interruzioni pianificate e non pianificate.

Gli amministratori di database devono eseguire la manutenzione occasionale del database. Attività incluse nella manutenzione sono l'applicazione di patch al database, gli aggiornamenti, la modifica dei parametri del database che richiedono un riavvio manuale, l'esecuzione di un failover per ridurre il tempo impiegato da un'istanza per modificare la classe. Queste operazioni pianificate comportano tempi di inattività.

Tuttavia, i tempi di inattività possono essere causati anche da operazioni non pianificate, come un failover imprevisto dovuto a un guasto hardware sottostante o a una limitazione delle risorse del database. Tutte queste operazioni pianificate e non pianificate comportano il riavvio del database.

In Aurora MySQL 3.05 e versioni successive, abbiamo introdotto ottimizzazioni che riducono il tempo di riavvio del database. Queste ottimizzazioni consentono di ridurre fino al 65% i tempi di inattività rispetto alle ottimizzazioni precedenti e di ridurre le interruzioni dei carichi di lavoro del database dopo un riavvio.

Durante l'avvio del database, vengono inizializzati molti componenti della memoria interna. Il più grande di questi è il pool di buffer InnoDB, che in Aurora MySQL costituisce per impostazione predefinita il 75% della dimensione della memoria dell'istanza. I nostri test hanno rilevato che il tempo di inizializzazione è proporzionale alla dimensione del pool di buffer InnoDB e che si dimensiona in base alla grandezza della classe dell'istanza del database. Durante questa fase di inizializzazione, il database non può accettare connessioni, il che causa tempi di inattività più lunghi durante i riavvii. Durante la prima fase del riavvio rapido, Aurora MySQL ottimizza l'inizializzazione del pool di buffer, riducendo i tempi di inizializzazione del database e quindi il tempo di riavvio complessivo.

Per maggiori dettagli, consulta il blog Reduce downtime with Amazon Aurora MySQL database restart time optimizations.

Nuovo comportamento della tabella temporanea in Aurora MySQL versione 3

Aurora MySQL versione 3 gestisce le tabelle temporanee in maniera diversa dalle versioni precedenti di Aurora MySQL. Questo nuovo comportamento è ereditato dalla edizione della comunità MySQL 8.0. Esistono due tipi di tabelle temporanee che possono essere create con Aurora MySQL versione 3:

- Tabelle temporanee interne (o implicite): create dal motore Aurora MySQL per gestire operazioni quali l'ordinamento dell'aggregazione, le tabelle derivate o le espressioni di tabella comuni (CTE).
- Tabelle temporanee create dall'utente (o esplicite): create dal motore Aurora MySQL quando si utilizza l'istruzione `CREATE TEMPORARY TABLE`.

Ci sono ulteriori considerazioni per le tabelle temporanee interne e create dall'utente sulle istanze database di lettura Aurora. Queste modifiche vengono illustrate nelle sezioni seguenti.

Argomenti

- Motore di storage per tabelle temporanee (implicite) interne
- Limitazione delle dimensioni delle tabelle temporanee interne in memoria
- Mitigazione dei problemi di pienezza per le tabelle temporanee interne su repliche Aurora
- Ottimizzazione del parametro temptable_max_mmap nelle istanze database di Aurora MySQL
- Tabelle temporanee create dall'utente (esplicite) su istanze database di lettura
- Errori di creazione tabelle temporanee e mitigazione

Motore di storage per tabelle temporanee (implicite) interne

Quando si generano set di risultati intermedi, Aurora MySQL tenta inizialmente di scrivere nelle tabelle temporanee in memoria. Questa operazione potrebbe non riuscire, a causa di tipi di dati incompatibili o limiti configurati. In tal caso, la tabella temporanea viene convertita in una tabella temporanea su disco anziché essere conservata in memoria. Ulteriori informazioni sono disponibili in Utilizzo della tabella temporanea interna in MySQL nella documentazione MySQL.

In Aurora MySQL versione 3, il modo in cui funzionano le tabelle temporanee interne è diverso dalle versioni precedenti di Aurora MySQL. Invece di scegliere tra i motori di archiviazione InnoDB e MyISAM per tali tabelle temporanee, ora è possibile scegliere tra i motori di archiviazione `TempTable` e `MEMORY`.

Con il motore di archiviazione `TempTable`, è possibile fare una scelta aggiuntiva su come gestire determinati dati. I dati interessati fanno traboccare il pool di memoria che contiene tutte le tabelle temporanee interne per l'istanza database.

Tali scelte possono influenzare le prestazioni delle query che generano volumi elevati di dati temporanei, ad esempio durante l'esecuzione di aggregazioni come `GROUP BY` su tabelle grandi.

> **Tip**
>
> Se il carico di lavoro include query che generano tabelle temporanee interne, verificare le prestazioni dell'applicazione con questa modifica eseguendo benchmark e monitorando le metriche relative alle prestazioni.
> In alcuni casi, la quantità di dati temporanei si inserisce nella pool di memoria `TempTable` o fa solo traboccare il pool di memoria in piccola misura. In questi casi, si consiglia di utilizzare l'impostazione `TempTable` per tabelle temporanee interne e file mappati in memoria per contenere i dati di overflow. Questa è l'impostazione di default.

Il motore di archiviazione `TempTable` è il valore di default. `TempTable` utilizza un pool di memoria comune per tutte le tabelle temporanee che utilizzano questo motore, anziché un limite massimo di memoria per tabella. La dimensione di questo pool di memoria è specificata dal parametro temptable_max_ram. Il valore predefinito è 1 GiB su istanze database con 16 o più GiB di memoria e 16 MB su istanze database con meno di 16 GiB di memoria. La dimensione del pool di memoria influisce sul consumo di memoria a livello di sessione.

In alcuni casi, quando si utilizza il motore di archiviazione `TempTable` i dati temporanei potrebbero superare le dimensioni del pool di memoria. In tal caso, Aurora MySQL archivia i dati di overflow utilizzando un meccanismo secondario.

È possibile impostare il parametro temptable_max_mmap per scegliere se i dati eseguono l'overflow in file temporanei mappati in memoria o in tabelle temporanee interne di InnoDB su disco. I diversi formati di dati e i criteri di overflow di questi meccanismi di overflow possono influire sulle prestazioni delle query. Lo fanno influenzando la quantità di dati scritti su disco e la domanda sulla velocità effettiva di archiviazione su disco.

Aurora MySQL versione 3 archivia i dati di overflow nel modo seguente:

- Nell'istanza database di scrittura, i dati che eseguono l'overflow nelle tabelle temporanee interne di InnoDB o nei file temporanei mappati in memoria risiedono nella memoria locale dell'istanza.
- Nelle istanze database di lettura, i dati di overflow risiedono sempre in file temporanei mappati in memoria nella memoria locale.

  Le istanze di sola lettura non possono memorizzare dati sul volume del cluster Aurora.

I parametri di configurazione relativi alle tabelle temporanee interne si applicano in modo diverso alle istanze di scrittura e lettura nel cluster:

- Nelle istanze di lettura, Aurora MySQL utilizza sempre il motore di archiviazione `TempTable`.
- La dimensione per il valore di `temptable_max_mmap` predefinito è 1 GiB, per entrambe le istanze di scrittura e lettura, indipendentemente dalle dimensioni della memoria dell'istanza database. È possibile modificare questo valore sia nelle istanze di scrittura che in quelle di lettura.
- L'impostazione di `temptable_max_mmap` su `0` disattiva l'uso dei file temporanei mappati in memoria nelle istanze di scrittura.
- Non puoi impostare `temptable_max_mmap` su `0` nelle istanze di lettura.

> **Note**
>
> Ti consigliamo di non utilizzare il parametro temptable_use_mmap. È deprecato e il relativo supporto verrà rimosso in una versione futura di MySQL.

Limitazione delle dimensioni delle tabelle temporanee interne in memoria

Come discusso in Motore di storage per tabelle temporanee (implicite) interne, è possibile controllare le risorse temporanee delle tabelle a livello globale utilizzando le impostazioni temptable max_ram e temptable_max_mmap.

Inoltre, è possibile limitare le dimensioni di ogni singola tabella temporanea interna in memoria utilizzando il parametro database tmp_table_size. Questo limite serve a evitare che le singole query consumino una quantità eccessiva di risorse globali delle tabelle temporanee, che può influire sulle prestazioni delle query simultanee che richiedono tali risorse.

Il parametro `tmp_table_size` definisce le dimensioni massime delle tabelle temporanee create dal motore di storage `MEMORY` in Aurora MySQL versione 3.

In Aurora MySQL versione 3.04 e successive, il parametro `tmp_table_size` definisce anche le dimensioni massime delle tabelle temporanee create dal motore di storage `TempTable` quando il parametro database `aurora_tmptable_enable_per_table_limit` è impostato su `ON`. Questo comportamento è disabilitato per impostazione predefinita (`OFF`), che è identico al comportamento in Aurora MySQL versione 3.03 e precedenti.

- Quando `aurora_tmptable_enable_per_table_limit` è `OFF`, `tmp_table_size` non è considerato per le tabelle temporanee interne in memoria create dal motore di storage `TempTable`.

  Tuttavia, il limite di risorse `TempTable` globali rimane comunque in essere. Aurora MySQL presenta il seguente comportamento quando viene raggiunto il limite di risorse `TempTable` globale:

  - Istanze database di scrittura: Aurora MySQL converte automaticamente la tabella temporanea in memoria in una tabella temporanea su disco InnoDB.
  - Istanze database di lettura: la query termina con un errore.

  ```
  ERROR 1114 (HY000): The table '/rdsdbdata/tmp/#sqlxx_xxx' is full
  ```

- Quando `aurora_tmptable_enable_per_table_limit` è `ON`, Aurora MySQL presenta il seguente comportamento quando viene raggiunto il limite `tmp_table_size`:

  - Istanze database di scrittura: Aurora MySQL converte automaticamente la tabella temporanea in memoria in una tabella temporanea su disco InnoDB.
  - Istanze database di lettura: la query termina con un errore.

  ```
  ERROR 1114 (HY000): The table '/rdsdbdata/tmp/#sqlxx_xxx' is full
  ```

  Il limite di risorse `TempTable` globale e il limite per tabella si applicano entrambi in questo caso.

> **Note**
>
> Il parametro `aurora_tmptable_enable_per_table_limit` non ha effetto quando internal_tmp_mem_storage_engine è impostato su `MEMORY`. In questo caso, le dimensioni massime di una tabella temporanea in memoria sono definite dal valore tmp_table_size o dimensione max_heap_table_size, a seconda di quale sia il più piccolo.

Gli esempi seguenti mostrano il comportamento del parametro `aurora_tmptable_enable_per_table_limit`per le istanze database di scrittura e lettura.

Example dell'istanza database di scrittura con **aurora_tmptable_enable_per_table_limit** impostato su **OFF**

La tabella temporanea in memoria non viene convertita in una tabella temporanea su disco InnoDB.

```
mysql> set aurora_tmptable_enable_per_table_limit=0;
Query OK, 0 rows affected (0.00 sec)

mysql> select
 @@innodb_read_only,@@aurora_version,@@aurora_tmptable_enable_per_table_limit,@@temptable_max_r
+-------------------+-----------------+-----------------------------------------
+---------------------+----------------------+
| @@innodb_read_only | @@aurora_version | @@aurora_tmptable_enable_per_table_limit |
 @@temptable_max_ram | @@temptable_max_mmap |
+-------------------+-----------------+-----------------------------------------
+---------------------+----------------------+
|                  0 | 3.04.0          |                                        0 |
       1073741824 |          1073741824 |
+-------------------+-----------------+-----------------------------------------
+---------------------+----------------------+
1 row in set (0.00 sec)

mysql> show status like '%created_tmp_disk%';
+-------------------------+-------+
| Variable_name           | Value |
+-------------------------+-------+
| Created_tmp_disk_tables | 0     |
+-------------------------+-------+
1 row in set (0.00 sec)

mysql> set cte_max_recursion_depth=4294967295;
Query OK, 0 rows affected (0.00 sec)

mysql> WITH RECURSIVE cte (n) AS (SELECT 1 UNION ALL SELECT n + 1 FROM cte WHERE n <
 60000000) SELECT max(n) FROM cte;
+----------+
| max(n)   |
+----------+
| 60000000 |
+----------+
1 row in set (13.99 sec)

mysql> show status like '%created_tmp_disk%';
+-------------------------+-------+
```

```
| Variable_name           | Value |
+-------------------------+-------+
| Created_tmp_disk_tables | 0     |
+-------------------------+-------+
1 row in set (0.00 sec)
```

Example dell'istanza database di scrittura
con **aurora_tmptable_enable_per_table_limit** impostato su **ON**

La tabella temporanea in memoria viene convertita in una tabella temporanea su disco InnoDB.

```
mysql> set aurora_tmptable_enable_per_table_limit=1;
Query OK, 0 rows affected (0.00 sec)

mysql> select
 @@innodb_read_only,@@aurora_version,@@aurora_tmptable_enable_per_table_limit,@@tmp_table_size;
+--------------------+-----------------+-----------------------------------------
+------------------+
| @@innodb_read_only | @@aurora_version | @@aurora_tmptable_enable_per_table_limit |
 @@tmp_table_size |
+--------------------+-----------------+-----------------------------------------
+------------------+
|                  0 | 3.04.0          |                                        1 |
      16777216 |
+--------------------+-----------------+-----------------------------------------
+------------------+
1 row in set (0.00 sec)

mysql> set cte_max_recursion_depth=4294967295;
Query OK, 0 rows affected (0.00 sec)

mysql> show status like '%created_tmp_disk%';
+-------------------------+-------+
| Variable_name           | Value |
+-------------------------+-------+
| Created_tmp_disk_tables | 0     |
+-------------------------+-------+
1 row in set (0.00 sec)

mysql> WITH RECURSIVE cte (n) AS (SELECT 1 UNION ALL SELECT n + 1 FROM cte WHERE n <
 6000000) SELECT max(n) FROM cte;
+---------+
| max(n)  |
```

```
+---------+
| 6000000 |
+---------+
1 row in set (4.10 sec)

mysql> show status like '%created_tmp_disk%';
+-------------------------+-------+
| Variable_name           | Value |
+-------------------------+-------+
| Created_tmp_disk_tables | 1     |
+-------------------------+-------+
1 row in set (0.00 sec)
```

Example dell'istanza database di lettura con **aurora_tmptable_enable_per_table_limit** impostato su **OFF**

La query termina senza errori perché `tmp_table_size` non è applicabile e il limite di risorse `TempTable` globale non è stato raggiunto.

```
mysql> set aurora_tmptable_enable_per_table_limit=0;
Query OK, 0 rows affected (0.00 sec)

mysql> select
 @@innodb_read_only,@@aurora_version,@@aurora_tmptable_enable_per_table_limit,@@temptable_max_r
+--------------------+------------------+-----------------------------------------
+---------------------+----------------------+
| @@innodb_read_only | @@aurora_version | @@aurora_tmptable_enable_per_table_limit |
 @@temptable_max_ram | @@temptable_max_mmap |
+--------------------+------------------+-----------------------------------------
+---------------------+----------------------+
|                  1 | 3.04.0           |                                        0 |
       1073741824 |           1073741824 |
+--------------------+------------------+-----------------------------------------
+---------------------+----------------------+
1 row in set (0.00 sec)

mysql> set cte_max_recursion_depth=4294967295;
Query OK, 0 rows affected (0.00 sec)

mysql> WITH RECURSIVE cte (n) AS (SELECT 1 UNION ALL SELECT n + 1 FROM cte WHERE n <
 60000000) SELECT max(n) FROM cte;
+----------+
| max(n)   |
```

```
+----------+
| 60000000 |
+----------+
1 row in set (14.05 sec)
```

Example dell'istanza database di lettura
con **aurora_tmptable_enable_per_table_limit** impostato su **OFF**

Questa query raggiunge il limite di risorse TempTable globale
con `aurora_tmptable_enable_per_table_limit` impostato su OFF. La query termina con un errore su istanze di lettura.

```
mysql> set aurora_tmptable_enable_per_table_limit=0;
Query OK, 0 rows affected (0.00 sec)

mysql> select
 @@innodb_read_only,@@aurora_version,@@aurora_tmptable_enable_per_table_limit,@@temptable_max_r
+--------------------+------------------+-----------------------------------------
+----------------------+----------------------+
| @@innodb_read_only | @@aurora_version | @@aurora_tmptable_enable_per_table_limit |
 @@temptable_max_ram | @@temptable_max_mmap |
+--------------------+------------------+-----------------------------------------
+----------------------+----------------------+
|                  1 | 3.04.0           |                                        0 |
       1073741824 |           1073741824 |
+--------------------+------------------+-----------------------------------------
+----------------------+----------------------+
1 row in set (0.00 sec)

mysql> set cte_max_recursion_depth=4294967295;
Query OK, 0 rows affected (0.01 sec)

mysql> WITH RECURSIVE cte (n) AS (SELECT 1 UNION ALL SELECT n + 1 FROM cte WHERE n <
 120000000) SELECT max(n) FROM cte;
ERROR 1114 (HY000): The table '/rdsdbdata/tmp/#sqlfd_1586_2' is full
```

Example dell'istanza database di lettura
con **aurora_tmptable_enable_per_table_limit** impostato su **ON**

La query termina con un errore quando viene raggiunto il limite `tmp_table_size`.

```
mysql> set aurora_tmptable_enable_per_table_limit=1;
```

```
Query OK, 0 rows affected (0.00 sec)

mysql> select
 @@innodb_read_only,@@aurora_version,@@aurora_tmptable_enable_per_table_limit,@@tmp_table_size;
+--------------------+------------------+----------------------------------------
+------------------+
| @@innodb_read_only | @@aurora_version | @@aurora_tmptable_enable_per_table_limit |
 @@tmp_table_size |
+--------------------+------------------+----------------------------------------
+------------------+
|                  1 | 3.04.0           |                                        1 |
      16777216 |
+--------------------+------------------+----------------------------------------
+------------------+
1 row in set (0.00 sec)

mysql> set cte_max_recursion_depth=4294967295;
Query OK, 0 rows affected (0.00 sec)

mysql> WITH RECURSIVE cte (n) AS (SELECT 1 UNION ALL SELECT n + 1 FROM cte WHERE n <
 6000000) SELECT max(n) FROM cte;
ERROR 1114 (HY000): The table '/rdsdbdata/tmp/#sqlfd_8_2' is full
```

Mitigazione dei problemi di pienezza per le tabelle temporanee interne su repliche Aurora

Per evitare problemi di limitazione delle dimensioni per le tabelle temporanee, impostare i parametri `temptable_max_ram` e `temptable_max_mmap` su un valore combinato in grado di soddisfare i requisiti del carico di lavoro.

Presta attenzione durante l'impostazione del valore del parametro `temptable_max_ram`. Se si imposta un valore troppo alto, la memoria disponibile sull'istanza database viene ridotta e si può verificare una condizione di memoria insufficiente. Monitora la memoria liberabile media sull'istanza database. Quindi determina un valore appropriato per `temptable_max_ram` in modo da avere ancora una quantità ragionevole di memoria libera sull'istanza. Per ulteriori informazioni, consulta Problemi di memoria liberabile in Amazon Aurora.

È inoltre importante monitorare le dimensioni dello storage locale e il consumo temporaneo di spazio della tabella. È possibile monitorare l'archiviazione temporanea disponibile per un'istanza database specifica con il parametro `FreeLocalStorage` Amazon CloudWatch, come descritto in CloudWatch Parametri Amazon per Amazon Aurora.

> **Note**
>
> Questa procedura non funziona quando il parametro `aurora_tmptable_enable_per_table_limit` è impostato su `0N`. Per ulteriori informazioni, consulta Limitazione delle dimensioni delle tabelle temporanee interne in memoria.

Example 1

È noto che le tabelle temporanee raggiungono una dimensione cumulativa di 20 GiB. Desideri impostare tabelle temporanee in memoria su 2 GiB e aumentare fino a un massimo di 20 GiB su disco.

Imposta `temptable_max_ram` su **`2,147,483,648`** e `temptable_max_mmap` su **`21,474,836,480`**. Questi valori sono espressi in byte.

Queste impostazioni dei parametri garantiscono che le dimensioni delle tabelle temporanee possano aumentare fino a un totale cumulativo di 22 GiB.

Example 2

La dimensione istanza corrente è 16xlarge o superiore. Non conosci le dimensioni totali delle tabelle temporanee che potrebbero essere richieste. Vuoi poter utilizzare fino a 4 GiB in memoria e fino alla dimensione di storage massima disponibile su disco.

Imposta `temptable_max_ram` su **`4,294,967,296`** e `temptable_max_mmap` su **`1,099,511,627,776`**. Questi valori sono espressi in byte.

Qui stai impostando `temptable_max_mmap` su 1 TiB, che è inferiore allo storage locale massimo di 1,2 TiB su un'istanza database di Aurora 16xlarge.

Su una dimensione istanza più piccola, regola il valore di `temptable_max_mmap` in modo che non riempia lo storage locale disponibile. Ad esempio, un'istanza 2xlarge dispone solo di 160 GiB di storage locale disponibile. Pertanto, si consiglia di impostare su un valore inferiore a 160 GiB. Per ulteriori informazioni sullo storage locale disponibile per le dimensioni delle istanze database, consulta Limiti di storage temporaneo per Aurora MySQL.

Ottimizzazione del parametro temptable_max_mmap nelle istanze database di Aurora MySQL

Il parametro `temptable_max_mmap` in Aurora MySQL controlla la quantità massima di spazio su disco locale che può essere utilizzata dai file mappati in memoria prima di eseguire l'overflow nelle tabelle temporanee InnoDB su disco (nelle istanze database di scrittura) o causare un errore (nelle istanze database di lettura). L'impostazione corretta di questo parametro dell'istanza database può aiutare a ottimizzare le prestazioni delle istanze database.

Prerequisiti

1. Assicurarsi che lo schema delle prestazioni sia abilitato. È possibile verificarlo eseguendo il comando SQL seguente:

   ```
   SELECT @@performance_schema;
   ```

   Un valore di output di 1 indica che è abilitato.

2. Verificare che la strumentazione della memoria delle tabelle temporanee sia abilitata. È possibile verificarlo eseguendo il comando SQL seguente:

   ```
   SELECT name, enabled FROM performance_schema.setup_instruments WHERE name LIKE
    '%memory%temptable%';
   ```

   La colonna `enabled` mostra `YES` per le voci relative alla strumentazione della memoria delle tabelle temporanee.

Monitoraggio dell'utilizzo delle tabelle temporanee

Quando si imposta il valore iniziale per `temptable_max_mmap`, si consiglia di iniziare con l'80% della dimensione di archiviazione locale per la classe di istanze database che si sta utilizzando. Ciò garantisce che le tabelle temporanee dispongano di spazio su disco sufficiente per funzionare in modo efficiente, lasciando spazio per altri utilizzi del disco nell'istanza.

Per trovare la dimensione di archiviazione locale per la classe di istanze database, consulta Limiti di storage temporaneo per Aurora MySQL.

Ad esempio, se si utilizza la classe di istanze database db.r5.large, la dimensione di archiviazione locale è di 32 GiB. In questo caso, inizialmente si imposterà il parametro `temptable_max_mmap` sull'80% di 32 GiB, ovvero 25,6 GiB.

Dopo aver impostato il valore `temptable_max_mmap` iniziale, eseguire il carico di lavoro di picco sulle istanze Aurora MySQL. Monitorare l'utilizzo corrente e massimo del disco delle tabelle temporanee utilizzando la seguente query SQL:

```
SELECT event_name, current_count, current_alloc, current_avg_alloc, high_count,
 high_alloc, high_avg_alloc
FROM sys.memory_global_by_current_bytes WHERE event_name LIKE 'memory/temptable/%';
```

Questa query recupera le seguenti informazioni:

- `event_name`: nome dell'evento relativo all'utilizzo della memoria o del disco delle tabelle temporanee.
- `current_count`: numero corrente di blocchi di memoria o di disco allocati per le tabelle temporanee.
- `current_alloc`: quantità corrente di memoria o disco allocata per le tabelle temporanee.
- `current_avg_alloc`: dimensione media corrente dei blocchi di memoria o di disco per le tabelle temporanee.
- `high_count`: numero massimo di blocchi di memoria o di disco allocati per le tabelle temporanee.
- `high_alloc`: quantità massima di memoria o disco allocata per le tabelle temporanee.
- `high_avg_alloc`: dimensione media massima dei blocchi di memoria o di disco per le tabelle temporanee.

Se le query hanno esito negativo con un errore di tabella piena utilizzando questa impostazione, significa che il carico di lavoro richiede più spazio su disco per le operazioni sulle tabelle temporanee. In questo caso, valutare la possibilità di aumentare la dimensione dell'istanza database scegliendo un'istanza con più spazio di archiviazione locale.

Impostazione del valore **`temptable_max_mmap`** ottimale

Utilizzare la seguente procedura per monitorare e impostare la dimensione corretta per il parametro `temptable_max_mmap`.

1. Esaminare l'output della query precedente e individuare il picco di utilizzo del disco delle tabelle temporanee, come indicato dalla colonna `high_alloc`.
2. In base al picco di utilizzo del disco delle tabelle temporanee, modificare il parametro `temptable_max_mmap` nel gruppo di parametri di database per le istanze database di Aurora MySQL.

Impostare il valore in modo che sia leggermente superiore al picco di utilizzo del disco delle tabelle temporanee per far fronte a incrementi futuri.

3. Applicare le modifiche al gruppo di parametri alle istanze database.
4. Monitorare nuovamente l'utilizzo del disco delle tabelle temporanee durante il picco di carico di lavoro per assicurarsi che il nuovo valore `temptable_max_mmap` sia appropriato.
5. Ripetere i passaggi precedenti se necessario per eseguire il fine-tuning del parametro `temptable_max_mmap`.

Tabelle temporanee create dall'utente (esplicite) su istanze database di lettura

Puoi creare tabelle temporanee esplicite utilizzando la parola chiave `TEMPORARY` nell'istruzione `CREATE TABLE`. Le tabelle temporanee esplicite sono supportate nell'istanza database di scrittura in un cluster di database Aurora. Puoi inoltre utilizzare tabelle temporanee esplicite sulle istanze database di lettura, ma le tabelle non possono applicare l'uso del motore di storage InnoDB.

Per evitare errori durante la creazione di tabelle temporanee esplicite su istanze database di lettura di Aurora MySQL, assicurati che tutte le istruzioni `CREATE TEMPORARY TABLE` vengano eseguite in uno o entrambi i seguenti modi:

- Non specificare la clausola `ENGINE=InnoDB`.
- Non impostare la modalità SQL su `NO_ENGINE_SUBSTITUTION`.

Errori di creazione tabelle temporanee e mitigazione

L'errore che si riceve è diverso a seconda che si utilizzi una semplice istruzione `CREATE TEMPORARY TABLE` o la variazione `CREATE TEMPORARY TABLE AS SELECT`. Nell'esempio seguente vengono illustrati tipi diversi di errori.

Questo comportamento temporaneo della tabella si applica solo alle istanze di sola lettura. Questo primo esempio conferma che è il tipo di istanza a cui è connessa la sessione.

```
mysql> select @@innodb_read_only;
+--------------------+
| @@innodb_read_only |
+--------------------+
|                  1 |
+--------------------+
```

Per istruzioni semplici `CREATE TEMPORARY TABLE`, l'istruzione fallisce quando la modalità SQL `NO_ENGINE_SUBSTITUTION` è attivata. Quando `NO_ENGINE_SUBSTITUTION` è disattivato (impostazione predefinita), viene effettuata la sostituzione motore appropriata e la creazione della tabella temporanea ha esito positivo.

```
mysql> set sql_mode = 'NO_ENGINE_SUBSTITUTION';

mysql>  CREATE TEMPORARY TABLE tt2 (id int) ENGINE=InnoDB;
ERROR 3161 (HY000): Storage engine InnoDB is disabled (Table creation is disallowed).

mysql> SET sql_mode = '';

mysql> CREATE TEMPORARY TABLE tt4 (id int) ENGINE=InnoDB;

mysql> SHOW CREATE TABLE tt4\G
*************************** 1. row ***************************
       Table: tt4
Create Table: CREATE TEMPORARY TABLE `tt4` (
  `id` int DEFAULT NULL
) ENGINE=MyISAM DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_0900_ai_ci
```

Per le istruzioni `CREATE TEMPORARY TABLE AS SELECT`, l'istruzione ha esito negativo quando la modalità SQL `NO_ENGINE_SUBSTITUTION` è attivata. Quando `NO_ENGINE_SUBSTITUTION` è disattivato (impostazione predefinita), viene effettuata la sostituzione motore appropriata e la creazione della tabella temporanea ha esito positivo.

```
mysql> set sql_mode = 'NO_ENGINE_SUBSTITUTION';

mysql> CREATE TEMPORARY TABLE tt1 ENGINE=InnoDB AS SELECT * FROM t1;
ERROR 3161 (HY000): Storage engine InnoDB is disabled (Table creation is disallowed).

mysql> SET sql_mode = '';

mysql> show create table tt3;
+-------+-----------------------------------------------------------+
| Table | Create Table                                              |
+-------+-----------------------------------------------------------+
| tt3   | CREATE TEMPORARY TABLE `tt3` (
  `id` int DEFAULT NULL
) ENGINE=MyISAM DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_0900_ai_ci |
+-------+-----------------------------------------------------------+
```

```
1 row in set (0.00 sec)
```

Per ulteriori informazioni sugli aspetti di archiviazione e sulle implicazioni sulle prestazioni delle tabelle temporanee in Aurora MySQL versione 3, consulta il post di blog Use the TempTable storage engine on Amazon RDS for MySQL and Amazon Aurora MySQL.

Confronto tra Aurora MySQL versione 2 e Aurora MySQL versione 3

Utilizzare quanto segue per scoprire le modifiche di cui essere a conoscenza quando si aggiorna il cluster Aurora MySQL versione 2 alla versione 3.

Argomenti

- Supporto per Atomic DDL (Data Definition Language).
- Differenze di funzionalità tra Aurora MySQL versione 2 e 3
- Supporto delle classi di istanza
- Modifiche ai parametri per Aurora MySQL versione 3
- Variabili di stato
- Cambiamenti linguistici inclusivi per Aurora MySQL versione 3
- Valori AUTO_INCREMENT
- Replica dei log binari

Supporto per Atomic DDL (Data Definition Language).

Una delle modifiche più importanti da MySQL 5.7 a 8.0 è l'introduzione dell'Atomic Data Dictionary. Prima di MySQL 8.0, il dizionario dei dati MySQL utilizzava un approccio basato su file per archiviare metadati come definizioni di tabelle (.frm), trigger (.trg) e funzioni separatamente dai metadati del motore di archiviazione (come quelli di InnoDB). Ciò presentava alcuni problemi, tra cui il rischio che le tabelle diventassero orfane in caso di imprevisti durante un'operazione DDL, il che causava la mancata sincronizzazione tra metadati basati su file e metadati del motore di archiviazione.

Per risolvere questo problema, MySQL 8.0 ha introdotto l'Atomic Data Dictionary, che archivia tutti i metadati in un set di tabelle InnoDB interne nello schema `mysql`. Questa nuova architettura offre un modo transazionale e conforme agli standard ACID per gestire i metadati del database, risolvendo il problema di "Atomic DDL" derivante dal precedente approccio basato su file. Per ulteriori informazioni sull'Atomic Data Dictionary, consulta Removal of file-based metadata storage e Atomic data definition statement support nel manuale di riferimento di MySQL.

A causa di questa modifica all'architettura, è necessario considerare quanto segue durante l'aggiornamento da Aurora MySQL versione 2 alla versione 3:

- È necessario eseguire la migrazione dei metadati basati su file della versione 2 nelle nuove tabelle dei dizionari di dati durante il processo di aggiornamento alla versione 3. A seconda del numero di oggetti del database di cui viene eseguita la migrazione, questa operazione potrebbe richiedere tempo.
- Le modifiche hanno inoltre introdotto alcune nuove incompatibilità che potrebbero dover essere risolte prima di poter eseguire l'aggiornamento da MySQL 5.7 a 8.0. Ad esempio, 8.0 include alcune nuove parole chiave riservate che potrebbero entrare in conflitto con i nomi degli oggetti del database esistenti.

Per aiutarti a identificare queste incompatibilità prima di aggiornare il motore, Aurora MySQL esegue una serie di controlli di compatibilità degli aggiornamenti (controlli preliminari) per determinare se ci sono oggetti incompatibili nel dizionario del database, prima di eseguire l'aggiornamento del dizionario dei dati. Per ulteriori informazioni sui controlli preliminari, consulta Controlli preliminari per gli aggiornamenti a versioni principali per Aurora MySQL.

Differenze di funzionalità tra Aurora MySQL versione 2 e 3

Le seguenti caratteristiche di Amazon Aurora MySQL sono supportate in Aurora MySQL 5.7, ma attualmente non in Aurora MySQL 8.0:

- Non puoi utilizzare Aurora MySQL versione 3 per cluster Aurora Serverless v1. Aurora MySQL versione 3 funziona con Aurora Serverless v2.
- La modalità Lab non si applica ad Aurora MySQL versione 3. Non ci sono funzionalità in modalità laboratorio in Aurora MySQL versione 3. Instant DDL sostituisce la veloce funzionalità DDL online precedentemente disponibile in modalità laboratorio. Per vedere un esempio, consulta DDL istantaneo (Aurora MySQL versione 3).
- La cache delle query viene rimossa dalla community MySQL 8.0 e anche da Aurora MySQL versione 3.
- Aurora MySQL versione 3 è compatibile con la funzionalità hash join MySQL della community. L'implementazione specifica di Aurora di hash join in Aurora MySQL versione 2 non viene utilizzata. Per informazioni sull'utilizzo di hash join con la query parallela Aurora, consultare Abilitazione dell'hash join per cluster di query parallele e Suggerimenti di Aurora MySQL. Per informazioni generali sull'utilizzo su hash join, vedere Ottimizzazione hash join nel Manuale di riferimento MySQL.

- La procedura archiviata `mysql.lambda_async` resa obsoleta in Aurora MySQL versione 2 viene rimossa nella versione 3. Per la versione 3, utilizzare invece la funzione asincrona `lambda_async`.
- Il set di caratteri predefinito in Aurora MySQL versione 3 è `utf8mb4`. In Aurora MySQL versione 2, il set di caratteri predefinito era `latin1`. Per informazioni su questo set di caratteri, consultare Il set di caratteri utf8mb4 (4-Byte UTF-8 Unicode Encoding) nel Manuale di riferimento MySQL.

Alcune funzionalità di Aurora MySQL sono disponibili per alcune combinazioni di regione AWS e versione del motore del database. Per dettagli, consultare Funzionalità supportate in Amazon Aurora by e Regione AWS Aurora DB engine.

Supporto delle classi di istanza

Aurora MySQL versione 3 supporta un set di classi di istanza diverso da quello di Aurora MySQL versione 2:

- Per istanze più grandi, è possibile utilizzare le classi di istanza moderne come`db.r5`,`db.r6g`, `db.x2g`.
- Per istanze più piccole, è possibile utilizzare le classi di istanza moderne come `db.t3` e `db.t4g`.

> **Note**
>
> Consigliamo di utilizzare le classi di istanza database T solo per i server di sviluppo e test o altri server non di produzione. Per ulteriori informazioni sulle classi di istanza T, consulta Utilizzo delle classi di istanza T per lo sviluppo e i test.

Le seguenti classi di istanza di Aurora MySQL versione 2 non sono disponibili per Aurora MySQL versione 3:

- `db.r4`
- `db.r3`
- `db.t3.small`
- `db.t2`

Controllare gli script di amministrazione per eventuali istruzioni CLI che creano istanze database Aurora MySQL. Codificare i nomi delle classi di istanza che non sono disponibili per Aurora MySQL

versione 3. Se necessario, modificare i nomi delle classi di istanza con quelli supportati da Aurora MySQL versione 3.

> **Tip**
>
> Per controllare le classi di istanza che è possibile utilizzare per una combinazione specifica della versione di Aurora MySQL e regione AWS, utilizza il comando `describe-orderable-db-instance-options` AWS CLI.

Per i dettagli sulle classi di istanza Aurora, consultare Classi di istanze database Amazon Aurora.

Modifiche ai parametri per Aurora MySQL versione 3

Aurora MySQL versione 3 include nuovi parametri di configurazione a livello di cluster e a livello di istanza. Aurora MySQL versione 3 rimuove anche alcuni parametri presenti in Aurora MySQL versione 2. Alcuni nomi dei parametri vengono modificati a seguito dell'iniziativa per un linguaggio inclusivo. Per la compatibilità con le versioni precedenti, è comunque possibile recuperare i valori dei parametri utilizzando i vecchi nomi o i nuovi nomi. Tuttavia, è necessario utilizzare i nuovi nomi per specificare i valori dei parametri in un gruppo di parametri personalizzato.

In Aurora MySQL versione 3, il valore del parametro `lower_case_table_names` viene impostato in modo permanente al momento della creazione del cluster. Se si utilizza un valore non predefinito per questa opzione, configurare il gruppo di parametri personalizzati Aurora MySQL versione 3 prima dell'aggiornamento. Quindi specificare il gruppo di parametri durante l'operazione di creazione del cluster o del ripristino dello snapshot.

> **Note**
>
> Con un database globale Aurora basato su Aurora MySQL, non puoi eseguire un aggiornamento locale da Aurora MySQL versione 2 alla versione 3 se il parametro `lower_case_table_names` è attivato. Utilizzare invece il metodo di ripristino snapshot.

In Aurora MySQL versione 3, i parametri `init_connect` e `read_only` non si applicano agli utenti che dispongono del privilegio `CONNECTION_ADMIN`, incluso l'utente master Aurora. Per ulteriori informazioni, consulta Privilegio basato sui ruoli.

Per un elenco di tutti i parametri del cluster Aurora MySQL, consultare Parametri a livello di cluster. La tabella copre tutti i parametri di Aurora MySQL versione 2 e 3. La tabella include note che mostrano quali parametri sono nuovi in Aurora MySQL versione 3 o sono stati rimossi da Aurora MySQL versione 3.

Per un elenco di tutti i parametri dell'istanza Aurora MySQL, consultare Parametri a livello di istanza. La tabella copre tutti i parametri di Aurora MySQL versione 2 e 3. La tabella include note che mostrano quali parametri sono nuovi in Aurora MySQL versione 3 e quali parametri sono stati rimossi da Aurora MySQL versione 3. Include anche note che mostrano quali parametri erano modificabili nelle versioni precedenti ma non Aurora MySQL versione 3.

Per informazioni sui nomi dei parametri modificati, consultare Cambiamenti linguistici inclusivi per Aurora MySQL versione 3.

Variabili di stato

Per informazioni sulle variabili di stato che non sono applicabili a Aurora MySQL, vedere Le variabili di stato MySQL seguenti non si applicano ad Aurora MySQL.

Cambiamenti linguistici inclusivi per Aurora MySQL versione 3

La versione iniziale di Aurora MySQL versione 3 è compatibile con la community MySQL 8.0.23. Aurora MySQL versione 3 include anche i cambiamenti da MySQL 8.0.26 relativi alle parole chiave e agli schemi di sistema per il linguaggio inclusivo. Ad esempio, il comando `SHOW REPLICA STATUS` è ora preferito invece di `SHOW SLAVE STATUS`.

Le seguenti metriche Amazon CloudWatch hanno nuovi nomi in Aurora MySQL versione 3.

In Aurora MySQL versione 3, sono disponibili solo i nuovi nomi delle metriche. Assicurati di aggiornare qualsiasi allarme o altra automazione basata sui nomi delle metriche quando esegui l'aggiornamento a Aurora MySQL versione 3.

Vecchio nome	Nuovo nome	
`ForwardingMasterDMLLatency`	`ForwardingWriterDMLLatency`	
`ForwardingMasterOpenSessions`	`ForwardingWriterOpenSessions`	

Vecchio nome	Nuovo nome
AuroraDMLRejectedMasterFull	AuroraDMLRejectedWriterFull
ForwardingMasterDMLThroughput	ForwardingWriterDMLThroughput

Le seguenti variabili di stato hanno nuovi nomi in Aurora MySQL versione 3.

Per compatibilità, è possibile utilizzare entrambi i nomi nella versione iniziale di Aurora MySQL versione 3. I nomi delle variabili di stato precedenti devono essere rimossi in una versione futura.

Nome da rimuovere	Nome nuovo o preferito
Aurora_fwd_master_dml_stmt_duration	Aurora_fwd_writer_dml_stmt_duration
Aurora_fwd_master_dml_stmt_count	Aurora_fwd_writer_dml_stmt_count
Aurora_fwd_master_select_stmt_duration	Aurora_fwd_writer_select_stmt_duration
Aurora_fwd_master_select_stmt_count	Aurora_fwd_writer_select_stmt_count
Aurora_fwd_master_errors_session_timeout	Aurora_fwd_writer_errors_session_timeout
Aurora_fwd_master_open_sessions	Aurora_fwd_writer_open_sessions
Aurora_fwd_master_errors_session_limit	Aurora_fwd_writer_errors_session_limit

Nome da rimuovere	Nome nuovo o preferito
`Aurora_fwd_master_errors_rpc_timeout`	`Aurora_fwd_writer_errors_rpc_timeout`

I seguenti parametri di configurazione hanno nuovi nomi in Aurora MySQL versione 3.

Per la compatibilità, è possibile controllare i valori dei parametri nel client di `mysql` utilizzando entrambi i nomi nella versione iniziale di Aurora MySQL versione 3. Puoi utilizzare i nuovi nomi solo quando modifichi i valori in un gruppo di parametri personalizzati. I nomi dei parametri precedenti devono essere rimossi in una versione futura.

Nome da rimuovere	Nome nuovo o preferito
`aurora_fwd_master_idle_timeout`	`aurora_fwd_writer_idle_timeout`
`aurora_fwd_master_max_connections_pct`	`aurora_fwd_writer_max_connections_pct`
`master_verify_checksum`	`source_verify_checksum`
`sync_master_info`	`sync_source_info`
`init_slave`	`init_replica`
`rpl_stop_slave_timeout`	`rpl_stop_replica_timeout`
`log_slow_slave_statements`	`log_slow_replica_statements`
`slave_max_allowed_packet`	`replica_max_allowed_packet`
`slave_compressed_protocol`	`replica_compressed_protocol`

Nome da rimuovere	Nome nuovo o preferito
slave_exec_mode	replica_exec_mode
slave_type_conversions	replica_type_conversions
slave_sql_verify_checksum	replica_sql_verify_checksum
slave_parallel_type	replica_parallel_type
slave_preserve_commit_order	replica_preserve_commit_order
log_slave_updates	log_replica_updates
slave_allow_batching	replica_allow_batching
slave_load_tmpdir	replica_load_tmpdir
slave_net_timeout	replica_net_timeout
sql_slave_skip_counter	sql_replica_skip_counter
slave_skip_errors	replica_skip_errors
slave_checkpoint_period	replica_checkpoint_period
slave_checkpoint_group	replica_checkpoint_group
slave_transaction_retries	replica_transaction_retries

Nome da rimuovere	Nome nuovo o preferito
`slave_parallel_workers`	`replica_parallel_workers`
`slave_pending_jobs_size_max`	`replica_pending_jobs_size_max`
`pseudo_slave_mode`	`pseudo_replica_mode`

Le seguenti stored procedure hanno nuovi nomi in Aurora MySQL versione 3.

Per compatibilità, è possibile utilizzare entrambi i nomi nella versione iniziale di Aurora MySQL versione 3. I nomi delle procedure precedenti devono essere rimossi in una versione futura.

Nome da rimuovere	Nome nuovo o preferito
`mysql.rds_set_master_auto_position`	`mysql.rds_set_source_auto_position`
`mysql.rds_set_external_master`	`mysql.rds_set_external_source`
`mysql.rds_set_external_master_with_auto_position`	`mysql.rds_set_external_source_with_auto_position`
`mysql.rds_reset_external_master`	`mysql.rds_reset_external_source`
`mysql.rds_next_master_log`	`mysql.rds_next_source_log`

Valori AUTO_INCREMENT

In Aurora MySQL versione 3, Aurora conserva il valore `AUTO_INCREMENT` per ogni tabella quando riavvia ogni istanza database. In Aurora MySQL versione 2, il valore `AUTO_INCREMENT` non è stato conservato dopo un riavvio.

Il valore `AUTO_INCREMENT` non viene mantenuto quando si configura un nuovo cluster ripristinando da uno snapshot, eseguendo un ripristino point-in-time (PITR) e clonando un cluster. In questi casi, il valore `AUTO_INCREMENT` viene inizializzato sul valore in base al valore di colonna più grande nella tabella al momento della creazione dello snapshot. Questo comportamento è diverso da quello in RDS per MySQL 8.0, dove il valore `AUTO_INCREMENT` viene mantenuto durante queste operazioni.

Replica dei log binari

Nell'edizione della community MySQL 8.0, la replica del log binario è attivata per impostazione predefinita. In Aurora MySQL versione 3, la replica del log binario è disattivata per impostazione predefinita.

> **Tip**
>
> Se i requisiti di elevata disponibilità sono soddisfatti dalle funzionalità di replica incorporate di Aurora, è possibile lasciare disattivata la replica del log binario. In questo modo, è possibile evitare il sovraccarico delle prestazioni della replica del log binario. È inoltre possibile evitare il monitoraggio e la risoluzione dei problemi associati necessari per gestire la replica dei log binari.

Aurora supporta la replica dei log binari da una fonte compatibile con MySQL 5.7 a Aurora MySQL versione 3. Il sistema fonte può essere un cluster database Aurora MySQL, un'istanza database RDS for MySQL o un'istanza MySQL locale.

Come fa MySQL della community, Aurora MySQL supporta la replica da una fonte che esegue una versione specifica a una destinazione che esegue la stessa versione principale o una versione principale superiore. Ad esempio, la replica da un sistema compatibile con MySQL 5.6 ad Aurora MySQL versione 3 non è supportata. La replica da Aurora MySQL versione 3 a un sistema compatibile con MySQL 5.7 o compatibile con MySQL 5.6 non è supportata. Per i dettagli sull'utilizzo della replica dei log binari, consultare Replica tra Aurora e MySQL o tra Aurora e un altro cluster di database Aurora (replica dei log binari).

Aurora MySQL versione 3 include miglioramenti alla replica dei log binari nella community MySQL 8.0, come la replica filtrata. Per informazioni dettagliate sui miglioramenti di MySQL 8.0 della community, consultare Come i server valutano le regole di filtro delle repliche nel Manuale di riferimento MySQL.

Compressione delle transazioni per la replica dei registri binari

Per informazioni sull'utilizzo sulla compressione del log binario, vedere Compressione delle transazioni dei registri nel Manuale di riferimento di MySQL.

Le seguenti limitazioni si applicano alla compressione dei log binari in Aurora MySQL versione 3:

- Le transazioni i cui dati di registro binario sono superiori alla dimensione massima consentita del pacchetto non vengono compresse. Ciò è vero a prescindere che l'impostazione di compressione del registro binario di Aurora MySQL sia attivata. Tali transazioni vengono replicate senza essere compresse.
- Se si utilizza un connettore per l'acquisizione dati di modifica (CDC) che non supporta ancora MySQL 8.0, non è possibile utilizzare questa funzione. Ti consigliamo di testare accuratamente tutti i connettori di terze parti con la compressione del registro binario. Inoltre, ti consigliamo di eseguire questa operazione prima di attivare la compressione binlog su sistemi che utilizzano la replica del binlog per CDC.

Confronto tra Aurora MySQL versione 3 e MySQL 8.0 Community Edition

È possibile utilizzare le seguenti informazioni per conoscere le modifiche di cui essere a conoscenza quando si converte da un sistema diverso compatibile con MySQL 8.0 a Aurora MySQL versione 3.

In generale, Aurora MySQL versione 3 supporta il set di funzionalità della community MySQL 8.0.23. Alcune nuove funzionalità dell'edizione della community di MySQL 8.0 non si applicano ad Aurora MySQL. Alcune di queste funzionalità non sono compatibili con alcuni aspetti di Aurora, come l'architettura di archiviazione Aurora. Non sono necessarie altre funzionalità perché il servizio di gestione Amazon RDS offre funzionalità equivalenti. Le seguenti funzionalità della community MySQL 8.0 non sono supportate o funzionano in modo diverso in Aurora MySQL versione 3.

Per le note di rilascio di tutte le release di Aurora MySQL versione 3, consultare Aggiornamenti del motore del database per Amazon Aurora MySQL versione 3 nelle Note di rilascio di Aurora MySQL.

Argomenti

- Funzionalità di MySQL 8.0 non disponibili in Aurora MySQL versione 3
- Privilegio basato sui ruoli
- Individuazione dell'ID server del database
- Autenticazione

Funzionalità di MySQL 8.0 non disponibili in Aurora MySQL versione 3

Le seguenti funzionalità della community MySQL 8.0 non sono disponibili o funzionano in modo diverso in Aurora MySQL versione 3.

- I Resource groups e le istruzioni SQL associate non sono supportati in Aurora MySQL.
- Aurora MySQL non supporta i tablespace di undo definiti dall'utente e le istruzioni SQL associate, come `CREATE UNDO TABLESPACE`, `ALTER UNDO TABLESPACE ... SET INACTIVE` e `DROP UNDO TABLESPACE`.
- Aurora MySQL non supporta il troncamento dei tablespace di undo per le versioni di Aurora MySQL precedenti alla 3.06. In Aurora MySQL versioni 3.06 e successive, è supportato il troncamento automatico dei tablespace di undo.
- È supportato il plugin di convalida della password.
- Non è possibile modificare le impostazioni dei plugin MySQL, incluso il plugin di convalida della password.
- Il plugin X non è supportato.
- La replica da più origini non è supportata.

Privilegio basato sui ruoli

Con Aurora MySQL versione 3, non è possibile modificare le tabelle direttamente nel database `mysql`. In particolare, non è possibile configurare gli utenti inserendoli nella tabella `mysql.user`. Invece, si utilizzano istruzioni SQL per concedere privilegi basati sui ruoli. Inoltre, nel database `mysql`, non è possibile creare altri tipi di oggetti come le stored procedure. È comunque possibile interrogare le tabelle di `mysql`. Se si utilizza la replica del registro binario, le modifiche apportate direttamente alle tabelle di `mysql` sul cluster fonte non vengono replicate nel cluster di destinazione.

In alcuni casi, l'applicazione potrebbe utilizzare scorciatoie per creare utenti o altri oggetti inserendoli nelle tabelle di `mysql`. In tal caso, modifica il codice dell'applicazione per utilizzare le istruzioni corrispondenti come `CREATE USER`. Se l'applicazione crea stored procedure o altri oggetti nel database `mysql`, utilizzare un database di tipo diverso.

Per esportare i metadati per gli utenti del database durante la migrazione da un database MySQL esterno, è possibile utilizzare il comando MySQL Shell anziché `mysqldump`. Per ulteriori informazioni, consulta Instance Dump Utility, Schema Dump Utility, and Table Dump Utility.

Per semplificare la gestione delle autorizzazioni per molti utenti o applicazioni, è possibile utilizzare l'istruzione `CREATE ROLE` per creare un ruolo con una serie di autorizzazioni. Puoi quindi utilizzare

le istruzioni `GRANT` e `SET ROLE` e la funzione `current_role` per assegnare ruoli a utenti o applicazioni, cambiare il ruolo corrente e verificare quali ruoli sono in vigore. Per ulteriori informazioni sul sistema di autorizzazione basato sui ruoli in MySQL 8.0, consultare Utilizzo di ruoli nel Manuale di riferimento di MySQL.

> **Important**
>
> Si consiglia di non utilizzare l'utente master direttamente nelle applicazioni. Rispetta piuttosto la best practice di utilizzare un utente del database creato con i privilegi minimi richiesti per l'applicazione.

Argomenti

- rds_superuser_role
- Utente con ruolo di controllo dei privilegi per la replica di log binari
- Ruoli per l'accesso ad altri servizi AWS

rds_superuser_role

Aurora MySQL versione 3 include un ruolo speciale che ha tutti i seguenti privilegi. Il ruolo è denominato `rds_superuser_role`. L'utente amministrativo principale per ciascun cluster ha già concesso questo ruolo. Il ruolo `rds_superuser_role` include i seguenti privilegi per tutti gli oggetti del database:

- `ALTER`
- `APPLICATION_PASSWORD_ADMIN`
- `ALTER ROUTINE`
- `CONNECTION_ADMIN`
- `CREATE`
- `CREATE ROLE`
- `CREATE ROUTINE`
- `CREATE TEMPORARY TABLES`
- `CREATE USER`
- `CREATE VIEW`

- DELETE
- DROP
- DROP ROLE
- EVENT
- EXECUTE
- FLUSH_OPTIMIZER_COSTS(Aurora MySQL versione 3.09 e successive)
- FLUSH_STATUS(Aurora MySQL versione 3.09 e successive)
- FLUSH_TABLES(Aurora MySQL versione 3.09 e successive)
- FLUSH_USER_RESOURCES(Aurora MySQL versione 3.09 e successive)
- INDEX
- INSERT
- LOCK TABLES
- PROCESS
- REFERENCES
- RELOAD
- REPLICATION CLIENT
- REPLICATION SLAVE
- ROLE_ADMIN
- SET_USER_ID
- SELECT
- SHOW DATABASES
- SHOW_ROUTINE (Aurora MySQL versione 3.04 e successive)
- SHOW VIEW
- TRIGGER
- UPDATE
- XA_RECOVER_ADMIN

La definizione del ruolo include anche la WITH GRANT OPTION in modo che un utente amministrativo possa concedere tale ruolo ad altri utenti. In particolare, l'amministratore deve

concedere tutti i privilegi necessari per eseguire la replica del log binario con il cluster Aurora MySQL come destinazione.

> **Tip**
>
> Per visualizzare i dettagli completi delle autorizzazioni, inserire le seguenti istruzioni.
>
> ```
> SHOW GRANTS FOR rds_superuser_role@'%';
> SHOW GRANTS FOR name_of_administrative_user_for_your_cluster@'%';
> ```

Utente con ruolo di controllo dei privilegi per la replica di log binari

Aurora MySQL versione 3 include un utente con ruolo di controllo dei privilegi per la replica dei log binari (binlog), ossia `rdsrepladmin_priv_checks_user`. Oltre ai privilegi di `rds_superuser_role`, questo utente dispone del privilegio `replication_applier`.

Quando si attiva la replica binlog chiamando la stored procedure `mysql.rds_start_replication`, viene creato `rdsrepladmin_priv_checks_user`.

L'utente `rdsrepladmin_priv_checks_user@localhost` è un utente riservato. Non modificarlo.

Ruoli per l'accesso ad altri servizi AWS

Aurora MySQL versione 3 include ruoli che è possibile utilizzare per accedere ad altri servizi. AWS È possibile impostare molti di questi ruoli come alternativa alla concessione di privilegi. Ad esempio, specificare `GRANT AWS_LAMBDA_ACCESS TO user` anzichéGRANT `INVOKE LAMBDA ON *.* TO user`. Per le procedure di accesso ad altri AWS servizi, consulta. Integrazione di Amazon Aurora MySQL con altri servizi AWS Aurora MySQL versione 3 include i seguenti ruoli relativi all'accesso ad altri servizi: AWS

- `AWS_LAMBDA_ACCESS`: alternativa al privilegio `INVOKE LAMBDA`. Per informazioni sull'utilizzo, consulta Chiamare una funzione Lambda da un cluster DB Amazon Aurora MySQL.
- `AWS_LOAD_S3_ACCESS`: alternativa al privilegio `LOAD FROM S3`. Per informazioni sull'utilizzo, consulta Caricamento dei dati in un cluster DB Amazon Aurora MySQL da file di testo in un bucket Amazon S3.
- `AWS_SELECT_S3_ACCESS`: alternativa al privilegio `SELECT INTO S3`. Per informazioni sull'utilizzo, consulta Salvataggio dei dati da un cluster DB Amazon Aurora MySQL nei file di testo in un bucket Amazon S3.

- `AWS_COMPREHEND_ACCESS`: alternativa al privilegio `INVOKE COMPREHEND`. Per informazioni sull'utilizzo, consulta Concessione agli utenti del database dell'accesso a machine learning di Aurora.
- `AWS_SAGEMAKER_ACCESS`: alternativa al privilegio `INVOKE SAGEMAKER`. Per informazioni sull'utilizzo, consulta Concessione agli utenti del database dell'accesso a machine learning di Aurora.
- `AWS_BEDROCK_ACCESS`: non esiste un privilegio `INVOKE` analogo per Amazon Bedrock. Per informazioni sull'utilizzo, consulta Concessione agli utenti del database dell'accesso a machine learning di Aurora.

Quando si concede l'accesso utilizzando i ruoli in Aurora MySQL versione 3, si attiva anche il ruolo utilizzando l'istruzione `SET ROLE role_name` o `SET ROLE ALL`. L'esempio seguente mostra come. Sostituire il nome del ruolo appropriato per `AWS_SELECT_S3_ACCESS`.

```
# Grant role to user.

mysql> GRANT AWS_SELECT_S3_ACCESS TO 'user'@'domain-or-ip-address'

# Check the current roles for your user. In this case, the AWS_SELECT_S3_ACCESS role
 has not been activated.
# Only the rds_superuser_role is currently in effect.
mysql> SELECT CURRENT_ROLE();
+--------------------------+
| CURRENT_ROLE()           |
+--------------------------+
| `rds_superuser_role`@`%` |
+--------------------------+
1 row in set (0.00 sec)

# Activate all roles associated with this user using SET ROLE.
# You can activate specific roles or all roles.
# In this case, the user only has 2 roles, so we specify ALL.
mysql> SET ROLE ALL;
Query OK, 0 rows affected (0.00 sec)

# Verify role is now active
mysql> SELECT CURRENT_ROLE();
+-------------------------------------------------------+
| CURRENT_ROLE()                                        |
+-------------------------------------------------------+
```

```
| `AWS_SELECT_S3_ACCESS`@`%`,`rds_superuser_role`@`%` |
+-----------------------------------------------------+
```

Individuazione dell'ID server del database

L'ID del server del database (`server_id`) è necessario per la replica di log binari (binlog). Il metodo per l'individuazione dell'ID server è diverso in Aurora MySQL e in Community MySQL.

In Community MySQL, l'ID server è un numero, che si ottiene utilizzando la seguente sintassi durante la connessione al server:

```
mysql> select @@server_id;

+-------------+
| @@server_id |
+-------------+
| 2           |
+-------------+
1 row in set (0.00 sec)
```

In Aurora MySQL, l'ID server è l'ID dell'istanza database, che si ottiene utilizzando la seguente sintassi durante la connessione all'istanza database:

```
mysql> select @@aurora_server_id;

+------------------------+
| @@aurora_server_id     |
+------------------------+
| mydbcluster-instance-2 |
+------------------------+
1 row in set (0.00 sec)
```

Per ulteriori informazioni sulla replica binlog, consulta Replica tra Aurora e MySQL o tra Aurora e un altro cluster di database Aurora (replica dei log binari).

Autenticazione

Nella community MySQL 8.0, il plug-in di autenticazione di default è `caching_sha2_password`. Aurora MySQL versione 3 utilizza ancora il plug-in `mysql_native_password`. Non è possibile modificare l'impostazione `default_authentication_plugin`. Tuttavia, è possibile creare nuovi

utenti e modificare gli utenti correnti; le password individuali degli utenti utilizzano il nuovo plugin di autenticazione. Di seguito è riportato un esempio.

```
mysql> CREATE USER 'testnewsha'@'%' IDENTIFIED WITH caching_sha2_password BY
 'aNewShaPassword';
Query OK, 0 rows affected (0.74 sec)
```

Aggiornamento ad Aurora MySQL versione 3

Per informazioni sull'aggiornamento del database da Aurora MySQL versione 2 alla versione 3, consulta Aggiornamento della versione principale di un cluster di database Amazon Aurora MySQL.

Aurora MySQL versione 2 compatibile con MySQL 5.7

Questo argomento descrive le differenze tra Aurora MySQL versione 2 e MySQL 5.7 Community Edition.

> **Important**
>
> Il supporto standard per Aurora MySQL versione 2 è terminato il 31 ottobre 2024. Per ulteriori informazioni, consulta Preparazione per la fine del supporto standard per Amazon Aurora edizione compatibile con MySQL versione 2.

Funzionalità non supportate in Aurora MySQL versione 2

Le seguenti funzioni sono supportate in MySQL 5.7, ma al momento non sono incluse in Aurora MySQL versione 2:

- Istruzione SQL `CREATE TABLESPACE`
- Plugin replica gruppi
- Maggiori dimensioni pagina
- Caricamento buffer pool InnoDB all'avvio
- Plugin parser full-text InnoDB
- Replica multi-source
- Ridimensionamento buffer pool online

- Plugin di convalida della password: è possibile installare il plugin, ma non è supportato. Non è possibile personalizzare il plugin.
- Plugin riscrittura query
- Filtri replica
- Protocollo X

Per ulteriori informazioni sulle funzioni, consulta la documentazione di MySQL 5.7.

Comportamento del tablespace temporaneo in Aurora MySQL versione 2

In MySQL 5.7, il tablespace temporaneo si estende automaticamente e le sue dimensioni aumentano in base alle necessità per ospitare le tabelle temporanee su disco. Quando le tabelle temporanee vengono eliminate, lo spazio liberato può essere riutilizzato per nuove tabelle temporanee, ma il tablespace temporaneo rimane nella dimensione estesa e non si riduce. Il tablespace temporaneo viene eliminato e ricreato al riavvio del motore.

In Aurora MySQL versione 2, si applica il seguente comportamento:

- Per i nuovi cluster database Aurora MySQL creati con la versione 2.10 e successive, il tablespace temporaneo viene rimosso e ricreato al riavvio del database. La funzione di ridimensionamento dinamico può in tal modo recuperare lo spazio di archiviazione.
- Per i cluster database Aurora MySQL aggiornati alla:
  - Versione 2.10 o successive: il tablespace temporaneo viene rimosso e ricreato al riavvio del database. La funzione di ridimensionamento dinamico può in tal modo recuperare lo spazio di archiviazione.
  - Versione 2.09: il tablespace temporaneo non viene rimosso al riavvio del database.

Puoi controllare la dimensione del tablespace temporaneo sul cluster database Aurora MySQL versione 2 utilizzando la seguente query:

```
SELECT
    FILE_NAME,
    TABLESPACE_NAME,
    ROUND((TOTAL_EXTENTS * EXTENT_SIZE) / 1024 / 1024 / 1024, 4) AS SIZE
FROM
    INFORMATION_SCHEMA.FILES
WHERE
```

```
    TABLESPACE_NAME = 'innodb_temporary';
```

Per ulteriori informazioni, consulta The Temporary Tablespace (Il tablespace temporaneo) nella documentazione di MySQL.

Motore di archiviazione per le tabelle temporanee su disco

Aurora MySQL versione 2 utilizza diversi motori di archiviazione per le tabelle temporanee interne su disco a seconda del ruolo dell'istanza.

- Nell'istanza di scrittura, le tabelle temporanee su disco utilizzano il motore di archiviazione InnoDB per impostazione predefinita. Sono archiviate nel tablespace temporaneo del volume del cluster Aurora.

  È possibile modificare questo comportamento nell'istanza di scrittura modificando il valore del parametro database `internal_tmp_disk_storage_engine`. Per ulteriori informazioni, consulta Parametri a livello di istanza.
- Nelle istanze di lettura, le tabelle temporanee su disco utilizzano il motore di archiviazione MyISAM, che usa l'archiviazione locale. Questo perché le istanze di sola lettura non possono memorizzare dati sul volume del cluster Aurora.

Sicurezza con Amazon Aurora MySQL

La sicurezza di Amazon Aurora MySQL viene gestita su tre livelli:

- Per controllare chi è in grado di eseguire le operazioni di gestione di Amazon RDS nei cluster database e nelle istanze database Aurora MySQL, viene utilizzato AWS Identity and Access Management (IAM). Quando esegui la connessione ad AWS usando le credenziali IAM, il tuo account AWS deve disporre di policy IAM per la concessione delle autorizzazioni richieste per eseguire le operazioni di gestione di Amazon RDS. Per ulteriori informazioni, consulta Gestione accessi e identità per Amazon Aurora

  Se utilizzi IAM per accedere alla console Amazon RDS, innanzitutto assicurati di accedere alla Console di gestione AWS con le credenziali dell'utente IAM. Quindi, passa alla console Amazon RDS all'indirizzo https://console.aws.amazon.com/rds/.
- I cluster di database Aurora MySQL devono essere creati in un cloud privato virtuale (VPC) utilizzando il servizio Amazon VPC. Per controllare i dispositivi e le istanze Amazon EC2 che possono aprire le connessioni all'endpoint e alla porta dell'istanza database per i cluster di database Aurora MySQL in un VPC, puoi utilizzare un gruppo di sicurezza VPC. È possibile creare queste connessioni di endpoint e porta mediante Transport Layer Security (TLS). Le regole del firewall aziendale possono inoltre determinare se i dispositivi in esecuzione nell'azienda possono aprire connessioni a un'istanza database. Per ulteriori informazioni sui VPC, consulta VPC Amazon e Amazon Aurora.

  La tenancy VPC supportata dipende dalla classe di istanze database usata dai cluster DB di Aurora MySQL. Con la tenancy VPC `default`, VPC viene eseguito nell'hardware condiviso. Con la tenancy VPC `dedicated`, il VPC viene eseguito in un'istanza hardware dedicata. Le classi di istanze database espandibili supportano solo la locazione VPC di default. Le classi di istanza database con prestazioni espandibili includono le classi di istanza db.t2, db.t3 e db.t4g DB. Tutte le altre classi di istanza database di Aurora MySQL DB supportano sia il tenancy VPC di default che dedicato.

> **Note**
>
> Consigliamo di utilizzare le classi di istanza database T solo per i server di sviluppo e test o altri server non di produzione. Per ulteriori informazioni sulle classi di istanza T, consulta Utilizzo delle classi di istanza T per lo sviluppo e i test.

Per ulteriori informazioni sulle classi di istanza, consulta Classi di istanze database Amazon Aurora. Per ulteriori informazioni sulle tenancy VPC `default` e `dedicated`, consulta Istanze dedicate nella Guida per l'utente di Amazon Elastic Compute Cloud.

- Per autenticare l'accesso e le autorizzazioni per un cluster DB Amazon Aurora MySQL puoi seguire uno degli approcci qui riportati oppure utilizzare una loro combinazione:
  - Puoi adottare lo stesso approccio utilizzato per un'istanza standalone di MySQL.

    I comandi come `CREATE USER`, `RENAME USER`, `GRANT`, `REVOKE` e `SET PASSWORD` funzionano esattamente come nei database in locale, modificando direttamente le tabelle dello schema del database. Per ulteriori informazioni, consulta Access Control and Account Management nella documentazione MySQL.
  - Puoi anche utilizzare l'autenticazione database IAM.

    Questo metodo prevede l'autenticazione del database IAM nel cluster DB tramite un utente IAM oppure con un ruolo IAM e un token di autenticazione. Il token di autenticazione è un valore univoco, generato tramite il processo di firma Signature Version 4. Quando utilizzi l'autenticazione database IAM, puoi utilizzare le stesse credenziali per controllare l'accesso alle risorse AWS e ai database. Per ulteriori informazioni, consulta Autenticazione del database IAM .

> **Note**
>
> Per ulteriori informazioni, consulta Sicurezza in Amazon Aurora.

Nelle sezioni seguenti, consulta le informazioni sulle autorizzazioni utente per le connessioni Aurora MySQL e TLS con i cluster di database Aurora MySQL.

Argomenti

- Privilegi dell'utente master con Amazon Aurora MySQL.
- Connessioni TLS a cluster di database Aurora MySQL

Privilegi dell'utente master con Amazon Aurora MySQL.

Quando crei un'istanza database Amazon Aurora MySQL, l'utente master ha i seguenti privilegi predefiniti indicati in Privilegi dell'account utente master.

Per fornire i servizi di gestione per ogni cluster di database, vengono creati gli utenti `admin` e `rdsadmin` al momento della creazione del cluster di database. I tentativi di rimuovere, rinominare, cambiare la password o modificare i privilegi dell'account `rdsadmin` produrranno errori.

Nei cluster di database Aurora MySQL versione 2, gli utenti `admin` e `rdsadmin` vengono creati al momento della creazione del cluster di database. Nei cluster di database Aurora MySQL versione 3, vengono creati gli utenti `admin`, `rdsadmin` e `rds_superuser_role`.

> **Important**
>
> Si consiglia di non utilizzare l'utente master direttamente nelle applicazioni. Rispetta piuttosto la best practice di utilizzare un utente del database creato con i privilegi minimi richiesti per l'applicazione.

Per la gestione del cluster di database di Aurora MySQL, i comandi standard `kill` e `kill_query` sono stati limitati. Al loro posto, utilizza i comandi di Amazon RDS `rds_kill` e `rds_kill_query` per terminare le sessioni utente e le query nelle istanze database di Aurora MySQL.

> **Note**
>
> La crittografia di un'istanza database e delle snapshot non è supportata per la regione Cina (Ningxia).

Connessioni TLS a cluster di database Aurora MySQL

I cluster database Amazon Aurora MySQL supportano le connessioni Transport Layer Security (TLS) da applicazioni che utilizzano lo stesso processo e la stessa chiave pubblica delle istanze database RDS per MySQL.

Amazon RDS crea un certificato TLS e installa il certificato nell'istanza database quando Amazon RDS effettua il provisioning dell'istanza. Questi certificati sono firmati da un'autorità di certificazione. Il certificato TLS include l'endpoint dell'istanza database come il nome comune (CN) per il certificato TLS per la protezione contro attacchi di spoofing. Di conseguenza, è possibile utilizzare l'endpoint del cluster DB per la connessione a un cluster DB tramite TLS solo se il client supporta i nomi alternativi dell'oggetto (Subject Alternative Names, SAN). In caso contrario, dovrai usare l'endpoint dell'istanza di un'istanza di scrittura.

Per ulteriori informazioni sul download dei certificati, consultare Utilizzo SSL/TLS per crittografare una connessione a un'.

Consigliamo il driver JDBC di AWS come client che supporta SAN con TLS. Per ulteriori informazioni sul driver JDBC di AWS e le istruzioni complete per utilizzarlo, consulta il repository GitHub del driver JDBC per Amazon Web Services (AWS).

Argomenti

- Richiesta di una connessione TLS a un cluster DB Aurora MySQL
- Versioni TLS per Aurora MySQL
- Configurazione di suite di cifratura per connessioni ai cluster di database Aurora MySQL
- Crittografia delle connessioni a un cluster DB Aurora MySQL

Richiesta di una connessione TLS a un cluster DB Aurora MySQL

È possibile richiedere che tutte le connessioni utente al cluster database Aurora MySQL utilizzino TLS utilizzando il parametro cluster database `require_secure_transport`. Per impostazione predefinita, il parametro `require_secure_transport` è impostato su `OFF`. È possibile impostare il parametro `require_secure_transport` su `ON` per richiedere la crittografia TLS per le connessioni al cluster database.

Puoi impostare il valore del parametro `require_secure_transport` aggiornando il gruppo di parametri per il tuo cluster DB. Non è necessario riavviare il cluster DB affinché la modifica abbia effetto. Per ulteriori informazioni sui gruppi di parametri, consulta Gruppi di parametri per Amazon Aurora.

> **Note**
>
> Il parametro `require_secure_transport` è disponibile per Aurora MySQL versione 2 e 3. È possibile impostare questo parametro in un gruppo di parametri del cluster database personalizzato. Il parametro non è disponibile nei gruppi di parametri di istanza database.

Quando il parametro `require_secure_transport` è impostato su `ON` per un cluster DB, un client di database può connettersi ad esso se è in grado di stabilire una connessione crittografata. In caso contrario, viene restituito al client un messaggio di errore simile al seguente:

```
MySQL Error 3159 (HY000): Connections using insecure transport are prohibited while --
require_secure_transport=ON.
```

Versioni TLS per Aurora MySQL

Aurora MySQL supporta Transport Layer Security (TLS) versione 1.0, 1.1, 1.2 e 1.3. A partire da Aurora MySQL versione 3.04.0 e successive, è possibile utilizzare il protocollo TLS 1.3 per proteggere le connessioni. La tabella seguente mostra il supporto TLS per le versioni di Aurora MySQL.

Versione Aurora MySQL	TLS 1.0	TLS 1.1	TLS 1.2	TLS 1.3	Predefinito
Aurora MySQL versione 2	Deprecated	Deprecated	Supportato	Non supportato	Tutte le versioni TLS supportate
Aurora MySQL versione 3 (versioni precedenti a 3.04.0)	Deprecated	Deprecated	Supportato	Non supportato	Tutte le versioni TLS supportate
Aurora MySQL versione 3 (3.04.0 e versioni successive)	Non supportato	Non supportato	Supportato	Supportato	Tutte le versioni TLS supportate

Important

Se si utilizzano gruppi di parametri personalizzati per i cluster Aurora MySQL con versione 2 e versioni precedenti alla 3.04.0, si consiglia di utilizzare TLS 1.2 perché TLS 1.0 e 1.1

sono meno sicuri. L'edizione community di MySQL 8.0.26 e Aurora MySQL 3.03 e le relative versioni minori hanno reso obsoleto il supporto per le versioni TLS 1.1 e 1.0.
L'edizione community di MySQL 8.0.28 e le versioni compatibili di Aurora MySQL 3.04.0 e successive non supportano TLS 1.1 e TLS 1.0. Se si utilizza Aurora MySQL versione 3.04.0 o successive, non impostare il protocollo TLS su 1.0 e 1.1 nel gruppo di parametri personalizzati.
Per Aurora MySQL versione 3.04.0 e successive, l'impostazione predefinita è TLS 1.3 e TLS 1.2.

È possibile utilizzare il parametro del cluster database `tls_version` per indicare le versioni del protocollo consentite. Parametri client simili esistono per la maggior parte degli strumenti client o dei driver di database. Alcuni client meno recenti potrebbero non supportare versioni TLS più recenti. Per impostazione predefinita, il cluster DB tenta di utilizzare la versione del protocollo TLS più alta consentita dalla configurazione del server e del client.

Impostare il parametro del cluster `tls_version` DB su uno dei seguenti valori:

- `TLSv1.3`
- `TLSv1.2`
- `TLSv1.1`
- `TLSv1`

È anche possibile impostare il parametro `tls_version` come un stringa di elenco separato da virgole. Se si desidera utilizzare entrambi i protocolli TLS 1.2 e TLS 1.3, il parametro `tls_version` deve includere tutti i protocolli da quello più basso a quello più alto. In questo caso, `tls_version` è impostato come:

```
tls_version=TLSv1.2,TLSv1.3
```

Per informazioni sulla modifica di parametri in un gruppo di parametri del cluster database, consulta Modifica dei parametri in un gruppo di parametri del cluster di database in Amazon Aurora. Se utilizzi il plugin AWS CLI per modificare il parametro del cluster database `tls_version`, `ApplyMethod` deve essere impostato su `pending-reboot`. Quando il metodo di applicazione è `pending-reboot`, le modifiche ai parametri vengono applicate dopo l'arresto e il riavvio dei cluster database associati al gruppo di parametri.

Configurazione di suite di cifratura per connessioni ai cluster di database Aurora MySQL

Utilizzando suite di cifratura configurabili, è possibile avere maggiore controllo sulla sicurezza delle connessioni al database. È possibile specificare un elenco di suite di crittografia che si desidera abilitare per proteggere le connessioni TLS client al database. Con le suite di cifratura, è possibile controllare la crittografia di connessione accettata dal server di database. In tal modo si previene l'uso di crittografie obsolete o non sicure.

Le suite di cifratura configurabili sono supportate in Aurora MySQL versione 3 e Aurora MySQL versione 2. Per specificare l'elenco di cifrature TLS 1.2, TLS 1.1, TLS 1.0 consentite per la crittografia delle connessioni, modifica il parametro del cluster `ssl_cipher`. Imposta il parametro `ssl_cipher` in un gruppo di parametri cluster utilizzando la Console di gestione AWS, la AWS CLIo l'API RDS.

Imposta il parametro `ssl_cipher` su una stringa di valori di cifratura separati da virgole per la versione TLS. Per l'applicazione client, puoi specificare le cifrature da utilizzare per le connessioni crittografate utilizzando l'opzione `--ssl-cipher` durante la connessione al database. Per ulteriori informazioni sulla connessione al database, consulta Connessione a un cluster di database Amazon Aurora MySQL.

A partire da Aurora MySQL versione 3.04.0 e successive, è possibile specificare le suite di crittografia TLS 1.3. Per specificare le suite di crittografia TLS 1.3 consentite, modifica il parametro `tls_ciphersuites` nel gruppo di parametri. TLS 1.3 ha ridotto il numero di suite di crittografia disponibili a causa delle modifiche alla convenzione di denominazione che rimuove il meccanismo di scambio delle chiavi e il certificato utilizzati. Imposta il parametro `tls_ciphersuites` su una stringa di valori di cifratura separati da virgole per TLS 1.3.

La tabella seguente mostra le cifrature supportate insieme al protocollo di crittografia TLS e alle versioni valide del motore Aurora MySQL per ogni cifratura.

Crittografia	Protocollo di crittografia	Versioni di Aurora MySQL supportate
`ECDHE-RSA-AES128-SHA`	TLS 1.0	3.04.0 e versioni successive, 2.11.0 e versioni successive
`ECDHE-RSA-AES128-SHA256`	TLS 1.2	3.04.0 e versioni successive, 2.11.0 e versioni successive

Crittografia	Protocollo di crittografia	Versioni di Aurora MySQL supportate
ECDHE-RSA-AES128-GCM-SHA256	TLS 1.2	3.04.0 e versioni successive, 2.11.0 e versioni successive
ECDHE-RSA-AES256-SHA	TLS 1.0	3.04.0 e versioni successive, 2.11.0 e versioni successive
ECDHE-RSA-AES256-GCM-SHA384	TLS 1.2	3.04.0 e versioni successive, 2.11.0 e versioni successive
ECDHE-RSA-CHACHA20-POLY1305	TLS 1.2	3.04.0 e versioni successive, 2.11.0 e versioni successive
ECDHE-ECDSA-AES128-SHA	TLS 1.0	3.04.0 e versioni successive, 2.11.0 e versioni successive
ECDHE-ECDSA-AES256-SHA	TLS 1.0	3.04.0 e versioni successive, 2.11.0 e versioni successive
ECDHE-ECDSA-AES128-GCM-SHA256	TLS 1.2	3.04.0 e versioni successive, 2.11.0 e versioni successive
ECDHE-ECDSA-AES256-GCM-SHA384	TLS 1.2	3.04.0 e versioni successive, 2.11.0 e versioni successive
ECDHE-ECDSA-CHACHA20-POLY1305	TLS 1.2	3.04.0 e versioni successive, 2.11.0 e versioni successive
TLS_AES_128_GCM_SHA256	TLS 1.3	3.04.0 e versioni successive
TLS_AES_256_GCM_SHA384	TLS 1.3	3.04.0 e versioni successive
TLS_CHACHA20_POLY1305_SHA256	TLS 1.3	3.04.0 e versioni successive

Per informazioni sulla modifica di parametri in un gruppo di parametri del cluster database, consulta Modifica dei parametri in un gruppo di parametri del cluster di database in Amazon Aurora. Se usi l'interfaccia a riga di comando per modificare il parametro cluster di database `ssl_cipher`, assicurati di impostare `ApplyMethod` su `pending-reboot`. Quando il metodo di applicazione è `pending-reboot`, le modifiche ai parametri vengono applicate dopo l'arresto e il riavvio dei cluster database associati al gruppo di parametri.

È possibile utilizzare anche il comando della CLI describe-engine-default-cluster-parameters per determinare quali suite di cifratura sono attualmente supportate per una famiglia del gruppo di parametri specifici. L'esempio seguente mostra come ottenere i valori consentiti per il parametro del cluster `ssl_cipher` per Aurora MySQL versione 2.

```
aws rds describe-engine-default-cluster-parameters --db-parameter-group-family aurora-
mysql5.7

        ...some output truncated...
 {
        "ParameterName": "ssl_cipher",
        "ParameterValue": "ECDHE-RSA-AES128-SHA,ECDHE-RSA-AES128-SHA256,ECDHE-RSA-
AES128-GCM-SHA256,ECDHE-RSA-AES256-SHA,ECDHE-RSA-AES256-GCM-SHA384,ECDHE-RSA-CHACHA20-
POLY1305,ECDHE-ECDSA-AES256-SHA,ECDHE-ECDSA-CHACHA20-POLY1305,ECDHE-ECDSA-AES256-GCM-
SHA384,ECDHE-ECDSA-AES128-GCM-SHA256,ECDHE-ECDSA-AES128-SHA",
        "Description": "The list of permissible ciphers for connection encryption.",
        "Source": "system",
        "ApplyType": "static",
        "DataType": "list",
        "AllowedValues": "ECDHE-RSA-AES128-SHA,ECDHE-RSA-AES128-SHA256,ECDHE-RSA-
AES128-GCM-SHA256,ECDHE-RSA-AES256-SHA,ECDHE-RSA-AES256-GCM-SHA384,ECDHE-RSA-CHACHA20-
POLY1305,ECDHE-ECDSA-AES256-SHA,ECDHE-ECDSA-CHACHA20-POLY1305,ECDHE-ECDSA-AES256-GCM-
SHA384,ECDHE-ECDSA-AES128-GCM-SHA256,ECDHE-ECDSA-AES128-SHA",
        "IsModifiable": true,
        "SupportedEngineModes": [
            "provisioned"
        ]
    },
       ...some output truncated...
```

Per ulteriori informazioni sulle cifrature, consulta la variabile ssl_ciphers nella documentazione di MySQL. Per ulteriori informazioni sui formati di suite di cifratura, consulta la documentazione per il formato elenco openssl-ciphers e il formato stringa openssl-ciphers sul sito Web OpenSSL.

Crittografia delle connessioni a un cluster DB Aurora MySQL

Per crittografare le connessioni con il client `mysql` di default, avvia il client mysql utilizzando il parametro `--ssl-ca` per fare riferimento alla chiave pubblica, ad esempio:

Per MySQL 5.7 e 8.0:

```
mysql -h myinstance.123456789012.rds-us-east-1.amazonaws.com
--ssl-ca=full_path_to_CA_certificate --ssl-mode=VERIFY_IDENTITY
```

Per MySQL 5.6:

```
mysql -h myinstance.123456789012.rds-us-east-1.amazonaws.com
--ssl-ca=full_path_to_CA_certificate --ssl-verify-server-cert
```

Sostituire *`full_path_to_CA_certificate`* con il percorso completo al certificato CA (Certificate Authority). Per ulteriori informazioni sul download di un certificato, consultare Utilizzo SSL/TLS per crittografare una connessione a un'.

È possibile richiedere connessioni TLS per account utente specifici. Ad esempio, in base alla versione di MySQL, è possibile utilizzare una delle seguenti istruzioni per richiedere connessioni TLS per l'account utente `encrypted_user`.

Per MySQL 5.7 e 8.0:

```
ALTER USER 'encrypted_user'@'%' REQUIRE SSL;
```

Per MySQL 5.6:

```
GRANT USAGE ON *.* TO 'encrypted_user'@'%' REQUIRE SSL;
```

Quando utilizzi un proxy RDS, esegui la connessione all'endpoint del proxy anziché al normale endpoint del cluster. Puoi rendere SSL/TLS obbligatorio o facoltativo per le connessioni al proxy, allo stesso modo delle connessioni dirette al cluster DB Aurora. Per informazioni sull'utilizzo del proxy RDS, consulta Server proxy per Amazon RDS per Aurora.

> **Note**
>
> Per ulteriori informazioni sulle connessioni TLS con MySQL, consulta la documentazione di MySQL.

Aggiornamento delle applicazioni per la connessione ai cluster database Aurora MySQL utilizzando nuovi certificati TLS

A partire dal 13 gennaio 2023, sono stati pubblicati da Amazon RDS nuovi certificati dell'autorità di certificazione (CA) per la connessione ai cluster database Aurora utilizzando Transport Layer Security (TLS). Di seguito sono disponibili le informazioni sull'aggiornamento delle applicazioni per utilizzare i nuovi certificati.

Le informazioni in questo argomento possono aiutarti a stabilire se le applicazioni client utilizzano TLS per connettersi ai cluster database. In caso affermativo, puoi determinare anche se le applicazioni richiedono la verifica del certificato per la connessione.

> **Note**
>
> Alcune applicazioni sono configurate per connettersi ai cluster DB Aurora MySQL solo se sono in grado di verificare il certificato del server.
> Per queste applicazioni, è necessario aggiornare gli archivi di trust delle applicazioni client per includere i nuovi certificati CA.

Dopo aver aggiornato i certificati CA negli archivi attendibilità delle applicazioni client, puoi ruotare i certificati nei cluster DB. Consigliamo vivamente di testare queste procedure in un ambiente di sviluppo o di gestione temporanea prima di implementarle negli ambienti di produzione.

Per ulteriori informazioni sulla rotazione dei certificati, consulta Rotazione del certificato SSL/TLS. Per ulteriori informazioni sul download, consulta Utilizzo SSL/TLS per crittografare una connessione a un'. Per informazioni sull'utilizzo di TLS con i cluster database Aurora MySQL, consulta Connessioni TLS a cluster di database Aurora MySQL.

Argomenti

- Determinazione se un'applicazione si connette al cluster databse Aurora MySQL utilizzando SSL
- Determinare se un client richiede la verifica del certificato per la connessione
- Aggiornare l'archivio di trust delle applicazioni
- Codice Java di esempio per stabilire connessioni TLS

Determinazione se un'applicazione si connette al cluster databse Aurora MySQL utilizzando SSL

Se utilizzi Aurora MySQL versione 2 (compatibile con MySQL 5.7) e lo schema delle prestazioni è abilitato, esegui la query indicata di seguito per verificare se le connessioni utilizzano TLS. Per informazioni sull'abilitazione dello schema delle prestazioni, consulta l'argomento relativo alla guida rapida per lo schema delle prestazioni nella documentazione di MySQL.

```
mysql> SELECT id, user, host, connection_type
       FROM performance_schema.threads pst
       INNER JOIN information_schema.processlist isp
       ON pst.processlist_id = isp.id;
```

In questo output di esempio, puoi vedere che la tua sessione (`admin`) e un'applicazione collegata come `webapp1` stanno entrambe usando TLS.

```
+----+-----------------+------------------+-----------------+
| id | user            | host             | connection_type |
+----+-----------------+------------------+-----------------+
|  8 | admin           | 10.0.4.249:42590 | SSL/TLS         |
|  4 | event_scheduler | localhost        | NULL            |
| 10 | webapp1         | 159.28.1.1:42189 | SSL/TLS         |
+----+-----------------+------------------+-----------------+
3 rows in set (0.00 sec)
```

Determinare se un client richiede la verifica del certificato per la connessione

Puoi verificare se i client JDBC e MySQL richiedono la verifica del certificato per la connessione.

JDBC

L'esempio seguente con MySQL Connector/J 8.0 mostra un modo per verificare le proprietà della connessione JDBC di un'applicazione per determinare se le connessioni riuscite richiedono un certificato valido. Per ulteriori informazioni su tutte le opzioni di connessione JDBC per MySQL, consulta l'argomento relativo alle proprietà di configurazione nella documentazione di MySQL.

Quando utilizzi MySQL Connector/J 8.0, la connessione TLS richiede la verifica del certificato CA del server se nelle proprietà di connessione `sslMode` è impostato su `VERIFY_CA` o `VERIFY_IDENTITY`, come nell'esempio seguente.

```
Properties properties = new Properties();
properties.setProperty("sslMode", "VERIFY_IDENTITY");
properties.put("user", DB_USER);
properties.put("password", DB_PASSWORD);
```

> **Note**
>
> Se si utilizza MySQL Java Connector v5.1.38 o versione successiva o MySQL Java Connector v8.0.9 o versione successiva per connettersi ai database, anche se non sono state configurate esplicitamente le applicazioni per l'utilizzo di TLS durante la connessione ai database, questi driver client utilizzano automaticamente TLS. Inoltre, quando utilizzano TLS, eseguono la verifica parziale del certificato e non riescono a connettersi se il certificato del server di database è scaduto.

MySQL

I seguenti esempi con il client MySQL mostrano due modi per verificare la connessione MySQL di uno script per determinare se le connessioni riuscite richiedono un certificato valido. Per ulteriori informazioni su tutte le opzioni di connessione con il client MySQL, consulta l'argomento relativo alla configurazione lato client delle connessioni crittografate nella documentazione di MySQL.

Quando utilizzi il client MySQL 5.7 o MySQL 8.0, la connessione TLS richiede la verifica del certificato CA del server se per l'opzione `--ssl-mode` viene specificato `VERIFY_CA` o `VERIFY_IDENTITY`, come nell'esempio seguente.

```
mysql -h mysql-database.rds.amazonaws.com -uadmin -ppassword --ssl-ca=/tmp/ssl-cert.pem
 --ssl-mode=VERIFY_CA
```

Quando utilizzi il client MySQL 5.6, la connessione SSL richiede la verifica del certificato CA del server se viene specificata l'opzione `--ssl-verify-server-cert`, come nell'esempio seguente.

```
mysql -h mysql-database.rds.amazonaws.com -uadmin -ppassword --ssl-ca=/tmp/ssl-cert.pem
 --ssl-verify-server-cert
```

Aggiornare l'archivio di trust delle applicazioni

Per informazioni sull'aggiornamento dell'archivio attendibilità per le applicazioni MySQL, consulta l'argomento relativo all'installazione dei certificati SSL nella documentazione di MySQL.

> **Note**
>
> Quando aggiorni l'archivio di trust puoi conservare i certificati meno recenti oltre ad aggiungere i nuovi certificati.

Aggiornare l'archivio di trust delle applicazioni per JDBC

Puoi aggiornare l'archivio di trust delle applicazioni che utilizzano JDBC per le connessioni TLS.

Per ulteriori informazioni sul download del certificato root, consulta Utilizzo SSL/TLS per crittografare una connessione a un'.

Per gli script di esempio che importano i certificati, consulta Script di esempio per l'importazione di certificati nel tuo archivio di trust.

Se utilizzi il driver mysql JDBC in un'applicazione, imposta le seguenti proprietà nell'applicazione.

```
System.setProperty("javax.net.ssl.trustStore", certs);
System.setProperty("javax.net.ssl.trustStorePassword", "password");
```

> **Note**
>
> Specifica una password diversa dal prompt mostrato qui come best practice per la sicurezza.

Quando avvii l'applicazione, imposta le seguenti proprietà.

```
java -Djavax.net.ssl.trustStore=/path_to_truststore/MyTruststore.jks -
Djavax.net.ssl.trustStorePassword=my_truststore_password com.companyName.MyApplication
```

Codice Java di esempio per stabilire connessioni TLS

L'esempio di codice seguente mostra come configurare la connessione SSL che convalida il certificato del server utilizzando JDBC.

```
public class MySQLSSLTest {

        private static final String DB_USER = "user name";
        private static final String DB_PASSWORD = "password";
        // This key store has only the prod root ca.
        private static final String KEY_STORE_FILE_PATH = "file-path-to-keystore";
        private static final String KEY_STORE_PASS = "keystore-password";

    public static void test(String[] args) throws Exception {
        Class.forName("com.mysql.jdbc.Driver");


        System.setProperty("javax.net.ssl.trustStore", KEY_STORE_FILE_PATH);
        System.setProperty("javax.net.ssl.trustStorePassword", KEY_STORE_PASS);

        Properties properties = new Properties();
        properties.setProperty("sslMode", "VERIFY_IDENTITY");
        properties.put("user", DB_USER);
        properties.put("password", DB_PASSWORD);

        Connection connection = DriverManager.getConnection("jdbc:mysql://jagdeeps-ssl-
test.cni62e2e7kwh.us-east-1.rds.amazonaws.com:3306",properties);
        Statement stmt=connection.createStatement();

        ResultSet rs=stmt.executeQuery("SELECT 1 from dual");

        return;
    }
}
```

Important

Dopo aver determinato che le connessioni al database utilizzano TLS e aver aggiornato l'archivio di trust delle applicazioni, puoi aggiornare il database per utilizzare i certificati rds-ca-rsa2048-g1. Per istruzioni, consulta la fase 3 in Aggiornamento del certificato CA modificando l'istanza database .

Utilizzo dell'autenticazione Kerberos per Aurora MySQL

Puoi utilizzare l'autenticazione Kerberos per autenticare gli utenti quando si connettono al cluster di database Aurora MySQL. A tale scopo, devi configurare il cluster di database per utilizzare AWS Directory Service for Microsoft Active Directory per l'autenticazione Kerberos. AWS Directory Service for Microsoft Active Directory è anche chiamato AWS Managed Microsoft AD. È una funzionalità disponibile con Directory Service. Per ulteriori informazioni, consultare Che cos'è Directory Service?nella Guida di amministrazione di AWS Directory Service.

Per iniziare, crea una directory AWS Managed Microsoft AD in cui archiviare le credenziali utente. Quindi fornisci al cluster di database Aurora MySQL il dominio di Active Directory e altre informazioni. Quando gli utenti eseguono l'autenticazione con il cluster di database Aurora MySQL, le richieste di autenticazione vengono inoltrate alla directory AWS Managed Microsoft AD.

Mantenere tutte le credenziali nella stessa directory consente di ridurre il tempo e l'impegno. Con questo approccio, hai una posizione centralizzata per archiviare e gestire le credenziali per più cluster di database. L'uso di una directory può inoltre migliorare il profilo di sicurezza complessivo.

Puoi inoltre accedere alle credenziali da Microsoft Active Directory on-premise. A tale scopo, crea una relazione di dominio trusting in modo che la directory AWS Managed Microsoft AD consideri attendibile Microsoft Active Directory on-premise. In questo modo, gli utenti possono accedere ai cluster di database Aurora MySQL con la stessa esperienza di autenticazione unica (SSO) di Windows dei carichi di lavoro nella rete on-premise.

Un database può usare Kerberos, AWS Identity and Access Management (IAM) o autenticazione Kerberos e IAM insieme. Tuttavia, poiché l'autenticazione Kerberos e IAM forniscono metodi di autenticazione diversi, un utente specifico può accedere a un database utilizzando solo uno o l'altro metodo di autenticazione, ma non entrambi. Per ulteriori informazioni sull'autenticazione IAM, consulta Autenticazione del database IAM .

Indice

- Panoramica dell'autenticazione Kerberos per cluster di database Aurora MySQL
- Limitazioni dell'autenticazione Kerberos per Aurora MySQL
- Configurazione dell'autenticazione Kerberos per cluster di database Aurora MySQL
  - Passaggio 1: creare una directory utilizzando AWS Managed Microsoft AD
  - Fase 2: (facoltativa) creazione di un trust per una Active Directory on-premise
  - Fase 3: creazione di un ruolo IAM per l'utilizzo da parte di Amazon Aurora

- Fase 4: creazione e configurazione di utenti
- Fase 5: creazione o modifica di un cluster di database Aurora MySQL
- Fase 6: creazione di utenti Aurora MySQL che utilizzano l'autenticazione Kerberos
  - Modifica di un accesso Aurora MySQL esistente
- Fase 7: configurazione di un client MySQL
- Fase 8: (facoltativo) configurazione il confronto dei nomi utente senza distinzione tra maiuscole e minuscole
- Connessione ad Aurora MySQL con l'autenticazione Kerberos
  - Utilizzo dell'accesso Kerberos di Aurora MySQL per la connessione al cluster di database
  - Autenticazione Kerberos con i database globali Aurora
  - Migrazione da RDS per MySQL ad Aurora MySQL
  - Memorizzazione del ticket nella cache impedita
  - Registrazione per l'autenticazione Kerberos
- Gestione di un cluster di database in un dominio
  - Appartenenza al dominio

Panoramica dell'autenticazione Kerberos per cluster di database Aurora MySQL

Per configurare l'autenticazione Kerberos per un cluster di database Aurora MySQL, completa le fasi generali riportate di seguito. che vengono descritte dettagliatamente più avanti.

1. Utilizza AWS Managed Microsoft AD per creare una directory AWS Managed Microsoft AD. Per creare la directory puoi utilizzare la Console di gestione AWS, AWS CLI o la Directory Service. Per ulteriori informazioni, consulta Creazione della directory AWS Managed Microsoft AD nella Guida per l'amministrazione di AWS Directory Service.
2. Creare un ruolo AWS Identity and Access Management (IAM) che utilizza la policy IAM gestita `AmazonRDSDirectoryServiceAccess`. Il ruolo consente ad Amazon Aurora di effettuare chiamate alla tua directory.

   Affinché il ruolo consenta l'accesso, l'endpoint AWS Security Token Service (AWS STS) deve essere attivato nella Regione AWS per l'account AWS. Gli endpoint AWS STS sono attivi per impostazione predefinita in tutte le Regioni AWS e puoi utilizzarli senza ulteriori interventi. Per

ulteriori informazioni, consulta Attivazione e disattivazione di AWS STS in una Regione AWS nella Guida per l'utente di IAM.

3. Crea e configura utenti nella directory AWS Managed Microsoft AD utilizzando gli strumenti di Microsoft Active Directory. Per ulteriori informazioni sulla creazione di utenti in Active Directory, consulta Gestione di utenti e gruppi in AWS Managed Microsoft AD nella Guida all'amministrazione di AWS Directory Service.
4. Crea o modifica un cluster di database Aurora MySQL. Se si utilizza l'interfaccia a riga di comando (CLI) o l'API RDS nella richiesta di creazione, specificare un identificatore di dominio con il parametro `Domain`. Utilizza l'identificatore `d-*` generato al momento della creazione della directory e il nome del ruolo IAM creato.

   Se modifichi un cluster di database Aurora MySQL esistente per utilizzare l'autenticazione Kerberos, imposta i parametri di dominio e ruolo IAM per il cluster di database. Individua il cluster di database nello stesso VPC della directory di dominio.
5. Utilizza le credenziali dell'utente principale Amazon RDS per connetterti al cluster di database Aurora MySQL. Crea l'utente del database in Aurora MySQL utilizzando le istruzioni riportate in Fase 6: creazione di utenti Aurora MySQL che utilizzano l'autenticazione Kerberos.

   Gli utenti creati in questo modo possono accedere al cluster di database Aurora MySQL utilizzando l'autenticazione Kerberos. Per ulteriori informazioni, consulta Connessione ad Aurora MySQL con l'autenticazione Kerberos.

Per usare l'autenticazione Kerberos con una Microsoft Active Directory on-premise o auto ospitato, crea un trust tra foreste. Un trust tra foreste è una relazione di trust tra due gruppi di domini. La fiducia può essere a senso unico o bidirezionale. Per ulteriori informazioni sulla configurazione di trust tra foreste tramite Directory Service, consulta Quando creare una relazione di trust nella Guida di amministrazione di AWS Directory Service.

Limitazioni dell'autenticazione Kerberos per Aurora MySQL

Le seguenti limitazioni si applicano all'autenticazione Kerberos per Aurora MySQL:

- L'autenticazione Kerberos è supportata per Aurora MySQL versione 3.03 e successive.

  Per ulteriori informazioni sul supporto Regione AWS, consulta Autenticazione Kerberos con Aurora MySQL.

- Per utilizzare l'autenticazione Kerberos con Aurora MySQL, il client o il connettore MySQL deve utilizzare la versione 8.0.26 o successiva su piattaforme Unix e 8.0.27 o successiva su Windows. In caso contrario, il plugin `authentication_kerberos_client` lato client non è disponibile e non puoi autenticarti.
- Solo AWS Managed Microsoft AD è supportato su Aurora MySQL. Tuttavia, puoi aggiungere i cluster di database Aurora MySQL a domini Microsoft AD gestita condivisi di proprietà di account diversi nella stessa Regione AWS.

  Inoltre puoi utilizzare la tua Active Directory on-premise. Per ulteriori informazioni, consulta Fase 2: (facoltativa) creazione di un trust per una Active Directory on-premise.
- Quando utilizzi Kerberos per autenticare un utente che si connette al cluster Aurora MySQL dai client MySQL o dai driver nel sistema operativo Windows, per impostazione predefinita i caratteri maiuscoli e minuscoli del nome utente del database devono corrispondere a quelli dell'utente in Active Directory. Ad esempio, se l'utente in Active Directory è `Admin`, il nome utente del database deve essere `Admin`.

  Tuttavia, ora puoi utilizzare il confronto dei nomi utente senza distinzione tra maiuscole e minuscole con il plugin `authentication_kerberos`. Per ulteriori informazioni, consulta Fase 8: (facoltativo) configurazione il confronto dei nomi utente senza distinzione tra maiuscole e minuscole.
- È necessario riavviare le istanze database di lettura dopo aver attivato la funzionalità per installare il plugin `authentication_kerberos`.
- La replica su istanze database che non supportano il plugin `authentication_kerberos` può causare un errore di replica.
- Affinché i database globali Aurora utilizzino l'autenticazione Kerberos, è necessario configurarla per ogni cluster del database globale.
- Il nome di dominio non deve contenere più di 62 caratteri.
- Non modificare la porta del cluster di database dopo aver abilitato l'autenticazione Kerberos. Se si modifica la porta, l'autenticazione Kerberos non funzionerà.

Configurazione dell'autenticazione Kerberos per cluster di database Aurora MySQL

AWS Managed Microsoft ADDa utilizzare per configurare l'autenticazione Kerberos per un cluster Aurora MySQL DB. Per configurare l'autenticazione Kerberos, completa la procedura seguente.

Argomenti

- Passaggio 1: creare una directory utilizzando AWS Managed Microsoft AD
- Fase 2: (facoltativa) creazione di un trust per una Active Directory on-premise
- Fase 3: creazione di un ruolo IAM per l'utilizzo da parte di Amazon Aurora
- Fase 4: creazione e configurazione di utenti
- Fase 5: creazione o modifica di un cluster di database Aurora MySQL
- Fase 6: creazione di utenti Aurora MySQL che utilizzano l'autenticazione Kerberos
- Fase 7: configurazione di un client MySQL
- Fase 8: (facoltativo) configurazione il confronto dei nomi utente senza distinzione tra maiuscole e minuscole

Passaggio 1: creare una directory utilizzando AWS Managed Microsoft AD

Directory Servicecrea una Active Directory completamente gestita nel AWS cloud. Quando crei una AWS Managed Microsoft AD directory, Directory Service crea due controller di dominio e server DNS (Domain Name System) per tuo conto. I server di directory vengono creati in sottoreti diverse in un VPC. Questa ridondanza assicura che la directory rimanga accessibile anche se si verifica un errore.

Quando crei una AWS Managed Microsoft AD directory, Directory Service esegue le seguenti attività per tuo conto:

- Configura una Active Directory all'interno del VPC.
- Crea un account amministratore della directory con il nome utente `Admin` e la password specificata. Puoi utilizzare questo account per gestire le directory.

  > **Note**
  >
  > Assicurati di salvare questa password. Directory Servicenon la memorizza. È possibile reimpostarla ma non recuperarla.

- Crea un gruppo di sicurezza per i controller della directory.

Quando si avvia unAWS Managed Microsoft AD, AWS crea un'unità organizzativa (OU) che contiene tutti gli oggetti della directory. Questa unità organizzativa ha lo stesso nome NetBIOS che hai immesso al momento della creazione della directory e Si trova nella radice del dominio, che è di proprietà e gestita daAWS.

L'`Admin`account creato con la AWS Managed Microsoft AD directory dispone delle autorizzazioni per le attività amministrative più comuni dell'unità organizzativa, tra cui:

- Creazione, aggiornamento o eliminazione di utenti
- Aggiunta di risorse al tuo dominio, come file o server di stampa, quindi assegnazione delle autorizzazioni per tali risorse a utenti e dell'UO
- Crea contenitori aggiuntivi OUs
- Delega dell'autorità
- Ripristino degli oggetti eliminati dal cestino di Active Directory
- Esegui i PowerShell moduli Windows AD e DNS sul servizio Web Active Directory

L'account `Admin` dispone anche dei diritti per eseguire queste attività in tutto il dominio:

- gestione delle configurazioni DNS (aggiunta, eliminazione o aggiornamento di record, zone e server d'inoltro);
- visualizzazione di log di eventi DNS;
- visualizzazione di log di eventi di sicurezza.

Per creare una directory con AWS Managed Microsoft AD

1. Accedi a Console di gestione AWS e apri la Directory Service console all'indirizzo https://console.aws.amazon.com/directoryservicev2/.
2. Nel riquadro di navigazione, seleziona Directories (Directory) e quindi Set up directory (Configura la directory).
3. Scegli AWS Managed Microsoft AD. AWS Managed Microsoft ADè l'unica opzione attualmente utilizzabile con Amazon RDS.
4. Immettere le seguenti informazioni:

   Nome DNS directory

   Il nome completo della directory, ad esempio **`corp.example.com`**.

   Nome NetBIOS della directory

   Nome breve per la directory, ad esempio **`CORP`**.

Descrizione della directory

(Opzionale) Una descrizione della directory.

Password amministratore

La password dell'amministratore della directory. Con il processo di creazione della directory viene generato un account amministratore con il nome utente Admin e questa password.

La password dell'amministratore della directory e non può includere il termine "admin". La password distingue tra maiuscole e minuscole e la lunghezza deve essere compresa tra 8 e 64 caratteri. Deve anche contenere un carattere di almeno tre delle seguenti quattro categorie:

- Lettere minuscole (a–z)
- Lettere maiuscole (A–Z)
- Numeri (0–9)
- Caratteri non alfanumerici (~!@#$%^&*_-+=`|\(){}[]:;"'<>,.?/)

Confirm password (Conferma password)

La password dell'amministratore immessa nuovamente.

5. Scegli Next (Successivo).
6. Immettere le seguenti informazioni nella sezione Networking (Rete) e quindi scegliere Next (Avanti):

   VPC

   VPC per la directory. Crea il cluster di database Aurora MySQL in questo stesso VPC.

   Sottoreti

   Sottoreti per i server di directory. Le due sottoreti devono trovarsi in diverse zone di disponibilità.
7. Esaminare le informazioni relative alla directory e apportare eventuali modifiche. Quando le informazioni sono corrette, scegli Create Directory (Crea directory).
   Pagina dei dettagli della directory durante la creazione

Per creare la directory sono necessari alcuni minuti. Una volta creata correttamente la directory, il valore Status (Stato) viene modificato in Active (Attivo).

Per consultare le informazioni sulla directory, selezionare il nome della directory nell'elenco di directory. Prendi nota del valore di ID directory perché è necessario quando crei o modifichi il cluster di database Aurora MySQL.
ID directory nella pagina Dettagli della directory

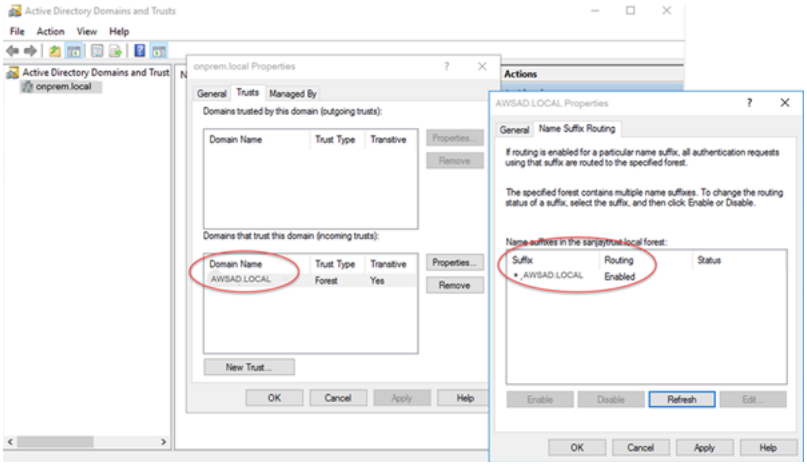
Fase 2: (facoltativa) creazione di un trust per una Active Directory on-premise

Se non prevedi di utilizzare Microsoft Active Directory locale, passa a Fase 3: creazione di un ruolo IAM per l'utilizzo da parte di Amazon Aurora.

Per utilizzare l'autenticazione Kerberos con l'Active Directory locale, è necessario creare una relazione di dominio affidabile utilizzando un trust di foresta tra Microsoft Active Directory locale e la AWS Managed Microsoft AD directory (creata in). Passaggio 1: creare una directory utilizzando AWS Managed Microsoft AD Il trust può essere unidirezionale, in cui la directory AWS Managed Microsoft AD considera attendibile Microsoft Active Directory locale. Il trust può anche essere bidirezionale, in cui entrambe le Active Directory si considerano reciprocamente attendibili. Per ulteriori informazioni sulla configurazione dei trust utilizzandoDirectory Service, consulta When to create a trust relationship nella Administration Guide. AWS Directory Service

> **Note**
>
> Se utilizzi una Microsoft Active Directory locale:
>
> - I client Windows non possono connettersi con endpoint personalizzati Aurora. Per ulteriori informazioni, consulta Connessioni degli endpoint Amazon Aurora.
> - Per i database globali:
>   - I client Windows possono connettersi solo con endpoint di istanza o cluster nella Regione AWS primaria del database globale.
>   - I client Windows non possono connettersi utilizzando gli endpoint del cluster in modalità secondaria. Regioni AWS

Assicurati che il nome di dominio di Microsoft Active Directory locale includa un routing del suffisso DNS che corrisponde alla nuova relazione di trust creata. Il risultato è mostrato nella screenshot seguente.



Fase 3: creazione di un ruolo IAM per l'utilizzo da parte di Amazon Aurora

Affinché Amazon Aurora possa Directory Service chiamarti, hai bisogno di un ruolo AWS Identity and Access Management (IAM) che utilizzi la policy IAM gestita. `AmazonRDSDirectoryServiceAccess` Questo ruolo permette ad Aurora di effettuare chiamate ad Directory Service.

Quando crei un cluster DB utilizzando e disponi Console di gestione AWS dell'`iam:CreateRole`autorizzazione, la console crea questo ruolo automaticamente. In questo caso, il nome del ruolo è `rds-directoryservice-kerberos-access-role`. In caso contrario, è necessario creare manualmente il ruolo IAM. Quando crei questo ruolo IAM`Directory Service`, scegli e allega la policy AWS gestita `AmazonRDSDirectoryServiceAccess` ad esso.

Per ulteriori informazioni sulla creazione di ruoli IAM per un servizio, consulta Creating a role to delegate permissions to an AWS service nella IAM User Guide.

Facoltativamente, puoi creare policy con le autorizzazioni richieste anziché utilizzare la policy IAM gestita `AmazonRDSDirectoryServiceAccess`. In questo caso, il ruolo IAM deve avere la seguente policy di attendibilità IAM.

JSON

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Sid": "",
      "Effect": "Allow",
```

```
      "Principal": {
        "Service": [
          "directoryservice.rds.amazonaws.com",
          "rds.amazonaws.com"
        ]
      },
      "Action": "sts:AssumeRole"
    }
  ]
}
```

Il ruolo deve anche disporre della seguente policy del ruolo IAM.

JSON

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Action": [
        "ds:DescribeDirectories",
        "ds:AuthorizeApplication",
        "ds:UnauthorizeApplication",
        "ds:GetAuthorizedApplicationDetails"
      ],
    "Effect": "Allow",
    "Resource": "*"
    }
  ]
}
```

Fase 4: creazione e configurazione di utenti

È possibile creare utenti con lo strumento Utenti Active Directory e computer. Questo strumento fa parte degli strumenti Active Directory Domain Services e Active Directory Lightweight Directory Services. Gli utenti possono essere individui singoli o entità che hanno accesso alla tua directory.

Per creare utenti in una Directory Service directory, usi un' EC2 istanza locale o Amazon basata su Microsoft Windows che viene aggiunta alla tua Directory Service directory. Devi anche essere connesso all'istanza come utente che dispone dei privilegi per creare utenti. Per ulteriori informazioni,

consulta Gestione di utenti e gruppi in AWS Managed Microsoft AD nella AWS Directory Service - Guida di amministrazione.

Fase 5: creazione o modifica di un cluster di database Aurora MySQL

Crea o modifica un cluster di database Aurora MySQL da usare con la directory. Puoi utilizzare la console o l'AWS CLIAPI RDS per associare un cluster DB a una directory. Questa operazione può essere eseguita in uno dei seguenti modi:

- Crea un nuovo cluster Aurora MySQL DB utilizzando la console, il comando _create-db-clusterCLI o l'operazione Create RDS API. DBCluster

  Per istruzioni, consulta Creazione di un cluster database Amazon Aurora.
- Modifica un cluster Aurora MySQL DB esistente utilizzando la console, il comando CLI o l'operazione Modify modify-db-clusterRDS API. DBCluster

  Per istruzioni, consulta Modifica di un cluster database Amazon Aurora.
- Ripristina un cluster Aurora MySQL DB da un'istantanea DB utilizzando la console, il comando restore-db-cluster-fromCLI -snapshot o l'operazione API Restore RDS. DBCluster FromSnapshot

  Per istruzioni, consulta Ripristino da uno snapshot cluster database.
- Ripristina un cluster Aurora MySQL DB point-in-time utilizzando la console, il comando - _ restore-db-cluster-topoint-in-timeCLI o l'operazione Restore RDS API. DBCluster ToPointInTime

  Per istruzioni, consulta Ripristino di un cluster di database a un determinato momento.

L'autenticazione Kerberos è supportata solo per i cluster di database Aurora MySQL in un VPC. Il cluster di database può trovarsi nello stesso VPC della directory o in un VPC diverso. Il VPC del cluster di database deve disporre di un gruppo di sicurezza VPC che consenta la comunicazione in uscita verso la directory.

Console

Quando utilizzi la console per creare, modificare o ripristinare un cluster di database, scegli Autenticazione Kerberos nella sezione Autenticazione database. Scegli Browse Directory (Sfoglia directory) quindi seleziona la directory oppure scegli Create a new directory (Crea una nuova directory).
Impostazione dell'autenticazione Kerberos durante la creazione di un cluster di database

AWS CLI

Quando utilizzi l'API AWS CLI o RDS, associa un cluster DB a una directory. Per consentire al cluster di database di utilizzare la directory del dominio che hai creato sono necessari i seguenti parametri:

- Per il parametro `--domain`, utilizza l'identificatore di dominio (identificatore "d-*") generato durante la creazione della directory.
- Per il parametro `--domain-iam-role-name`, utilizza il ruolo creato che utilizza la policy IAM gestita `AmazonRDSDirectoryServiceAccess`.

Ad esempio, il comando CLI seguente modifica un cluster di database per utilizzare una directory.

Per Linux, macOS o Unix:

```
aws rds modify-db-cluster \
    --db-cluster-identifier mydbcluster \
    --domain d-ID \
    --domain-iam-role-name role-name
```

Per Windows:

```
aws rds modify-db-cluster ^
    --db-cluster-identifier mydbcluster ^
    --domain d-ID ^
    --domain-iam-role-name role-name
```

> **Important**
>
> Se modifichi un cluster di database per abilitare l'autenticazione Kerberos, riavvia le istanze database di lettura dopo aver apportato la modifica.

Fase 6: creazione di utenti Aurora MySQL che utilizzano l'autenticazione Kerberos

Il cluster DB viene aggiunto al AWS Managed Microsoft AD dominio. Pertanto, puoi creare utenti Aurora MySQL dagli utenti di Active Directory nel tuo dominio. Le autorizzazioni del database vengono gestite tramite autorizzazioni Aurora MySQL standard concesse e revocate da questi utenti.

È possibile consentire a un utente di Active Directory di autenticarsi con Aurora MySQL. Per fare ciò, utilizza innanzitutto le credenziali utente principale Amazon RDS per connetterti al cluster di database Aurora MySQL come con qualsiasi altro cluster di database. Dopo aver eseguito l'accesso, crea un utente autenticato esternamente con l'autenticazione Kerberos in Aurora MySQL come illustrato di seguito:

```
CREATE USER user_name@'host_name' IDENTIFIED WITH 'authentication_kerberos' BY
 'realm_name';
```

- Sostituisci *user_name* con il nome utente. Gli utenti (persone e applicazioni) del tuo dominio ora possono connettersi al cluster di database da un computer client associato al dominio utilizzando l'autenticazione Kerberos.
- Sostituisci *host_name* con il nome host. È possibile utilizzare `%` come carattere jolly. È inoltre possibile usare indirizzi IP specifici per il nome host.
- Sostituisci *realm_name* con il nome dell'area di directory del dominio. Il nome del realm è in genere uguale al nome di dominio DNS in lettere maiuscole, ad esempio `CORP.EXAMPLE.COM`. Un realm è un gruppo di sistemi che utilizzano lo stesso Kerberos Key Distribution Center.

L'esempio seguente crea un utente del database con il nome `Admin` che esegue l'autenticazione in Active Directory con il nome del realm `MYSQL.LOCAL`.

```
CREATE USER Admin@'%' IDENTIFIED WITH 'authentication_kerberos' BY 'MYSQL.LOCAL';
```

Modifica di un accesso Aurora MySQL esistente

È possibile modificare un accesso Aurora MySQL esistente per utilizzare l'autenticazione Kerberos usando la seguente sintassi:

```
ALTER USER user_name IDENTIFIED WITH 'authentication_kerberos' BY 'realm_name';
```

Fase 7: configurazione di un client MySQL

Per configurare un client MySQL, procedi come indicato di seguito:

1. Crea un file `krb5.conf` (o equivalente) che punti al dominio.
2. Verifica che il traffico possa fluire tra l'host del client eDirectory Service. Utilizza un'utilità di rete come Netcat per le operazioni seguenti:

- Verifica il traffico su DNS per la porta 53.
- Verifica il traffico verso TCP/UDP la porta 53 e per Kerberos, che include le porte 88 e 464 per. Directory Service

3. Verifica che il traffico scorra senza problemi tra l'host client e l'istanza database sulla porta del database. Ad esempio, utilizza `mysql` per connetterti e accedere al database.

Di seguito è riportato un esempio `krb5.conf` di contenuto per. AWS Managed Microsoft AD

```
[libdefaults]
 default_realm = EXAMPLE.COM
[realms]
 EXAMPLE.COM = {
  kdc = example.com
  admin_server = example.com
 }
[domain_realm]
 .example.com = EXAMPLE.COM
 example.com = EXAMPLE.COM
```

Di seguito è riportato un esempio di contenuto `krb5.conf` per una Microsoft Active Directory on-premise.

```
[libdefaults]
 default_realm = EXAMPLE.COM
[realms]
 EXAMPLE.COM = {
  kdc = example.com
  admin_server = example.com
 }
 ONPREM.COM = {
  kdc = onprem.com
  admin_server = onprem.com
 }
[domain_realm]
 .example.com = EXAMPLE.COM
 example.com = EXAMPLE.COM
 .onprem.com = ONPREM.COM
 onprem.com = ONPREM.COM
 .rds.amazonaws.com = EXAMPLE.COM
 .amazonaws.com.cn = EXAMPLE.COM
```

```
.amazon.com = EXAMPLE.COM
```

Fase 8: (facoltativo) configurazione il confronto dei nomi utente senza distinzione tra maiuscole e minuscole

Per impostazione predefinita, a livello di distinzione tra maiuscole e minuscole, il nome utente del database MySQL deve corrispondere a quello dell'accesso di Active Directory. Tuttavia, ora puoi utilizzare il confronto dei nomi utente senza distinzione tra maiuscole e minuscole con il plugin `authentication_kerberos`. A tale scopo, devi impostare il parametro `authentication_kerberos_caseins_cmp` del cluster database su `true`.

Utilizzo del confronto dei nomi utente senza distinzione tra maiuscole e minuscole

1. Crea un gruppo di parametri per il cluster database personalizzato. Seguire le procedure indicate in Creazione di un gruppo di parametri del cluster di database in Amazon Aurora.
2. Modifica il nuovo gruppo di parametri per impostare il valore di `authentication_kerberos_caseins_cmp` su `true`. Seguire le procedure indicate in Modifica dei parametri in un gruppo di parametri del cluster di database in Amazon Aurora.
3. Associa il gruppo di parametri del cluster di database al cluster database Aurora MySQL. Seguire le procedure indicate in Associazione di un gruppo di parametri del cluster di database a un cluster di database in Amazon Aurora.
4. Riavviare il cluster database.

Connessione ad Aurora MySQL con l'autenticazione Kerberos

Per evitare errori, usa un client MySQL versione 8.0.26 o successiva su piattaforme Unix e 8.0.27 o successiva su Windows.

Utilizzo dell'accesso Kerberos di Aurora MySQL per la connessione al cluster di database

Per connetterti ad Aurora MySQL con l'autenticazione Kerberos, accedi come utente del database creato utilizzando le istruzioni riportate in Fase 6: creazione di utenti Aurora MySQL che utilizzano l'autenticazione Kerberos.

Al prompt dei comandi, connettiti a uno degli endpoint associati al cluster di database Aurora MySQL. Quando viene richiesta la password, immetti la password Kerberos associata al nome utente.

Quando ti autentichi con Kerberos, viene generato un ticket di concessione (TGT) se non ne esiste già uno. Il plugin `authentication_kerberos` utilizza il TGT per ottenere un ticket di servizio, che viene poi inviato al server di database Aurora MySQL.

Puoi utilizzare il client MySQL per connetterti ad Aurora MySQL con l'autenticazione Kerberos usando Windows o Unix.

Unix

Puoi connetterti usando i seguenti metodi:

- Ottieni il TGT manualmente. In questo caso, non è necessario fornire la password al client MySQL.
- Fornisci la password per l'accesso ad Active Directory direttamente al client MySQL.

Il plugin lato client è supportato nelle piattaforme Unix per i client MySQL versione 8.0.26 e successive.

Per connettersi ottenendo il TGT manualmente

1. Nell'interfaccia della linea di comando, utilizza il seguente comando per ottenere il TGT.

   ```
   kinit user_name
   ```

2. Usa il comando `mysql` seguente per accedere all'endpoint dell'istanza database del cluster di database.

   ```
   mysql -h DB_instance_endpoint -P 3306 -u user_name -p
   ```

   > **Note**
   >
   > L'autenticazione non riesce se il keytab viene ruotato sull'istanza database. In questo caso, ottieni un nuovo TGT eseguendo nuovamente `kinit`.

Per connettersi direttamente

1. Nell'interfaccia della linea di comando usa il comando `mysql` seguente per accedere all'endpoint dell'istanza database del cluster di database.

```
mysql -h DB_instance_endpoint -P 3306 -u user_name -p
```

2. Immetti la password per l'utente di Active Directory.

Windows

In Windows, l'autenticazione viene solitamente eseguita al momento dell'accesso, quindi non è necessario ottenere il TGT manualmente per connettersi al cluster di database Aurora MySQL. Le maiuscole e minuscole del nome utente del database devono corrispondere alle maiuscole e minuscole dell'utente in Active Directory. Ad esempio, se l'utente in Active Directory è `Admin`, il nome utente del database deve essere `Admin`.

Il plug-in lato client è supportato su Windows per i client MySQL versione 8.0.27 e successive.

Per connettersi direttamente

- Nell'interfaccia della linea di comando usa il comando `mysql` seguente per accedere all'endpoint dell'istanza database del cluster di database.

```
mysql -h DB_instance_endpoint -P 3306 -u user_name
```

Autenticazione Kerberos con i database globali Aurora

L'autenticazione Kerberos per Aurora MySQL è supportata per i database globali Aurora. Per autenticare gli utenti sul cluster di database secondario utilizzando l'Active Directory del cluster di database primario, replica l'Active Directory sulla Regione AWS secondaria. L'autenticazione Kerberos viene attivata sul cluster secondario utilizzando lo stesso ID di dominio del cluster primario. La replica AWS Managed Microsoft AD è supportata solo con la versione Enterprise di Active Directory. Per ulteriori informazioni, consulta Replica multiregionale nella Guida per l'amministrazione di AWS Directory Service.

Migrazione da RDS per MySQL ad Aurora MySQL

Dopo aver eseguito la migrazione da RDS per MySQL ad Aurora MySQL con l'autenticazione Kerberos abilitata, modifica gli utenti creati con il plugin `auth_pam` per utilizzare il plugin `authentication_kerberos`. Per esempio:

```
ALTER USER user_name IDENTIFIED WITH 'authentication_kerberos' BY 'realm_name';
```

Memorizzazione del ticket nella cache impedita

Se non esiste un TGT valido all'avvio, l'applicazione client MySQL può ottenerlo e memorizzarlo nella cache. Se vuoi evitare che il TGT venga memorizzato nella cache, imposta un parametro di configurazione nel file `/etc/krb5.conf`.

> **Note**
>
> Questa configurazione si applica solo agli host client che eseguono Unix, non Windows.

Per impedire la memorizzazione del TGT nella cache

- Aggiungi una sezione `[appdefaults]` a `/etc/krb5.conf` come segue:

```
[appdefaults]
  mysql = {
    destroy_tickets = true
  }
```

Registrazione per l'autenticazione Kerberos

La variabile di ambiente `AUTHENTICATION_KERBEROS_CLIENT_LOG` imposta il livello di registrazione per l'autenticazione Kerberos. È possibile utilizzare i log per il debug sul lato client.

I valori consentiti sono compresi tra 1 e 5. I messaggi del log vengono scritti nell'output di errore standard. La seguente tabella descrive ciascun livello di registrazione.

Livello di logging	Descrizione
1 o non impostato	Nessuna registrazione
2	Messaggi di errore
3	Messaggi di errore e avviso
4	Messaggi di errore, avviso e informazioni

Livello di logging	Descrizione
5	Messaggi di errore, avviso, informazioni e debug

Gestione di un cluster di database in un dominio

Puoi utilizzare la AWS CLI o l'API RDS per gestire il cluster di database e la sua relazione con la tua Active Directory gestita. Ad esempio, è possibile associare un'autenticazione di Active Directory per Kerberos e annullare l'associazione di una Active Directory per disattivare l'autenticazione Kerberos. Puoi anche spostare un cluster di database affinché venga autenticato esternamente da una Microsoft Active Directory a un'altra.

Ad esempio, puoi utilizzare l'API Amazon RDS per effettuare quanto segue:

- Per tentare di attivare nuovamente l'autenticazione Kerberos per un'appartenenza non riuscita, utilizza l'operazione API `ModifyDBInstance` e specifica l'ID della directory dell'appartenenza corrente.
- Per aggiornare il nome del ruolo IAM dell'appartenenza, utilizza l'operazione API `ModifyDBInstance` e specifica l'ID della directory dell'appartenenza corrente e il nuovo ruolo IAM.
- Per disattivare l'autenticazione Kerberos in un cluster di database, utilizza l'operazione API `ModifyDBInstance` e specifica `none` come parametro di dominio.
- Per spostare un cluster di database da un dominio a un altro, utilizza l'operazione API `ModifyDBInstance` e specifica l'identificatore del nuovo dominio come parametro di dominio.
- Per elencare l'appartenenza per ogni cluster di database, utilizza l'operazione API `DescribeDBInstances`.

Appartenenza al dominio

Quando il cluster di database viene creato o modificato diventa membro del dominio. Puoi visualizzare lo stato dell'appartenenza al dominio per il cluster di database eseguendo il comando CLI describe-db-clusters. Lo stato del cluster di database può essere uno dei seguenti:

- `kerberos-enabled`: il cluster di database ha l'autenticazione Kerberos attivata.
- `enabling-kerberos`: AWS sta attivando l'autenticazione Kerberos sul cluster di database.

- `pending-enable-kerberos`: l'attivazione dell'autenticazione Kerberos è in sospeso su questo cluster di database.
- `pending-maintenance-enable-kerberos`: AWS proverà ad attivare l'autenticazione Kerberos sul cluster di database durante la prossima finestra di manutenzione pianificata.
- `pending-disable-kerberos`: la disattivazione dell'autenticazione Kerberos è in sospeso su questo cluster di database.
- `pending-maintenance-disable-kerberos`: AWS proverà a disattivare l'autenticazione Kerberos sul cluster di database durante la prossima finestra di manutenzione pianificata.
- `enable-kerberos-failed`: un problema di configurazione ha impedito ad AWS di attivare l'autenticazione Kerberos sul cluster di database. Verifica e correggi la configurazione prima di eseguire nuovamente il comando di modifica del cluster di database.
- `disabling-kerberos`: AWS sta disattivando l'autenticazione Kerberos sul cluster di database.

Una richiesta per attivare l'autenticazione Kerberos potrebbe non andare a buon fine a causa di un problema di connettività di rete o un ruolo IAM non corretto. Ad esempio, supponi di creare un cluster di database o di modificare un cluster di database esistente e il tentativo di attivare l'autenticazione Kerberos non riesce. In questo caso, esegui nuovamente il comando di modifica o modifica il cluster di database appena creato per l'aggiunta al dominio.

Migrazione di dati a un cluster di database Amazon Aurora MySQL

Hai diverse opzioni per migrare i dati da un database esistente a un cluster database Amazon Aurora MySQL. Le opzioni di migrazione dipendono anche dal database da cui esegui la migrazione e dalle dimensioni dei dati sottoposti a migrazione.

Ci sono due diversi tipi di migrazione: fisica e logica. La migrazione fisica significa che le copie fisiche dei file di database vengono utilizzate per migrare il database. La migrazione logica significa che la migrazione viene effettuata applicando modifiche di database logiche, come inserimenti, aggiornamenti ed eliminazioni.

La migrazione fisica ha i vantaggi seguenti:

- La migrazione fisica è più veloce della migrazione logica, specialmente per database di grandi dimensioni.
- Le prestazioni del database non ne risentono quando un backup viene utilizzato per la migrazione fisica.
- La migrazione fisica può migrare tutto nel database origine, comprese componenti di database complesse.

La migrazione fisica ha i limiti seguenti:

- Il parametro `innodb_page_size` deve essere impostato al valore predefinito (`16KB`).
- Il parametro `innodb_data_file_path` deve essere configurato con un solo file di dati che utilizza il nome di file di dati predefinito `"ibdata1:12M:autoextend"`. I database con due file di dati o con un file di dati con un nome diverso non possono essere migrati utilizzando questo metodo.

  Di seguito sono riportati esempi di nomi di file che non sono permessi: `"innodb_data_file_path=ibdata1:50M; ibdata2:50M:autoextend"` e `"innodb_data_file_path=ibdata01:50M:autoextend"`.
- Il parametro `innodb_log_files_in_group` deve essere impostato al valore predefinito (`2`).

La migrazione logica ha i vantaggi seguenti:

- Puoi migrare i sottoinsiemi del database, come tabelle specifiche o parti di una tabella.
- I dati possono essere migrati indipendentemente dalla struttura fisica dello storage.

La migrazione logica ha i limiti seguenti:

- La migrazione logica solitamente è più lenta della migrazione fisica.
- I componenti di database complessi possono rallentare il processo di migrazione logica. In alcuni casi, i componenti di database complessi possono anche bloccare la migrazione logica.

La tabella seguente descrive le opzioni e il tipo di migrazione per ogni opzione.

Migrazione da	Tipo di migrazione	Soluzione
Un'istanza database RDS for MySQL	Fisica	Puoi eseguire la migrazione da un'istanza database RDS for MySQL creando prima di tutto una replica di lettura Aurora MySQL di un'istanza database MySQL. Quando il ritardo di replica tra un'istanza database MySQL e la replica di lettura Aurora MySQL è 0, puoi indirizzare le applicazioni del client a leggere dalla replica di lettura Aurora e poi interrompere la replica per rendere la replica di lettura Aurora MySQL un cluster di database standalone Aurora MySQL per lettura e scrittura. Per informazioni dettagliate, consulta Migrazione di dati da un'istanza database RDS per MySQL a un cluster database Amazon Aurora MySQL utilizzando una replica di lettura Aurora.
Una snapshot DB di RDS for MySQL	Fisica	Puoi migrare i dati direttamente da un RDS for MySQL snapshot DB a un cluster database Amazon Aurora MySQL. Per informazioni dettagliate, consultare Migrazione di una snapshot RDS for MySQL a Aurora.
Un database MySQL esterno a Amazon RDS	Logica	Puoi creare un dump dei dati utilizzando l'utilità `mysqldump` e importare i dati in un cluster di database Amazon Aurora MySQL esistente. Per informazioni dettagliate, consultare Migrazione logica da MySQL ad Amazon Aurora MySQL mediante mysqldump.

Migrazione da	Tipo di migrazione	Soluzione
		Per esportare i metadati per gli utenti del database durante la migrazione da un database MySQL esterno, è possibile utilizzare anche il comando MySQL Shell anziché `mysqldump` . Per ulteriori informazioni, consulta Instance Dump Utility, Schema Dump Utility, and Table Dump Utility. **Note** L'utility mysqlpump è obsoleta a partire da MySQL 8.0.34.
Un database MySQL esterno a Amazon RDS	Fisica	Puoi copiare i file di backup dal database a un bucket Amazon Simple Storage Service (Amazon S3) e quindi ripristinare un cluster di database Amazon Aurora MySQL da tali file. Questa opzione può essere molto più rapida rispetto alla migrazione dei dati con `mysqldump` . Per informazioni dettagliate, consulta Migrazione fisica da MySQL utilizzando XtraBackup Percona e Amazon S3.

Migrazione da	Tipo di migrazione	Soluzione
Un database MySQL esterno a Amazon RDS	Logica	Puoi salvare i dati dal database come file di testo e copiare tali file in un bucket Amazon S3. Puoi quindi caricare i dati in un cluster di database Aurora MySQL esistente utilizzando il comando MySQL `LOAD DATA FROM S3`. Per ulteriori informazioni, consulta Caricamento dei dati in un cluster DB Amazon Aurora MySQL da file di testo in un bucket Amazon S3.
Un database non compatibile con MySQL	Logica	Puoi utilizzare AWS Database Migration Service (AWS DMS) per eseguire la migrazione dei dati da un database non compatibile con MySQL. Per ulteriori informazioni su AWS DMS, consulta Che cos'è AWS Database Migration Service?

Note

Se stai migrando un database MySQL esterno ad Amazon RDS, le opzioni di migrazione descritte nella tabella sono applicabili solo se il database supporta gli spazi tabelle InnoDB o MyISAM.

Se il database MySQL che stai migrando ad Aurora MySQL utilizza `memcached`, rimuovi `memcached` prima di eseguire la migrazione.

Non è possibile migrare ad Aurora MySQL 3.05 e versioni successive da alcune versioni precedenti di MySQL 8.0, tra cui 8.0.11, 8.0.13 e 8.0.15. Si consiglia di eseguire l'aggiornamento a MySQL 8.0.28 prima della migrazione.

La migrazione di dati da un database MySQL esterno a un cluster database Amazon Aurora MySQL

Se il database supporta gli spazi tabelle InnoDB o MyISAM, hai queste opzioni per migrare i dati a un cluster database Amazon Aurora MySQL:

- Puoi creare un dump dei dati utilizzando l'utilità `mysqldump` e importare i dati in un cluster di database Amazon Aurora MySQL esistente. Per ulteriori informazioni, consulta Migrazione logica da MySQL ad Amazon Aurora MySQL mediante mysqldump.
- Puoi copiare i file di backup completi e incrementali dal database a un bucket Amazon S3 ed eseguire il ripristino su un cluster database Amazon Aurora MySQL da tali file. Questa opzione può essere molto più rapida rispetto alla migrazione dei dati con `mysqldump`. Per ulteriori informazioni, consulta Migrazione fisica da MySQL utilizzando XtraBackup Percona e Amazon S3.

Argomenti

- Migrazione fisica da MySQL utilizzando XtraBackup Percona e Amazon S3
- Migrazione logica da MySQL ad Amazon Aurora MySQL mediante mysqldump

Migrazione fisica da MySQL utilizzando XtraBackup Percona e Amazon S3

Puoi copiare i file di backup completi e incrementali dal database MySQL versione 5.7 o 8.0 di origine a un bucket S3. Puoi quindi eseguire il ripristino in un cluster database Amazon Aurora MySQL con la stessa versione principale del motore d9 datahase da tali file.

Questa opzione può essere molto più veloce rispetto alla migrazione dei dati utilizzando `mysqldump`, in quanto l'utilizzo di `mysqldump` riproduce tutti i comandi per ricreare lo schema e i dati dal database di origine nel cluster di database Aurora MySQL. Copiando i file di dati MySQL origine, Aurora MySQL può immediatamente utilizzare quei file come dati per un cluster di database Aurora MySQL.

Puoi anche ridurre al minimo i tempi di inattività utilizzando la replica dei log binari durante il processo di migrazione. Se utilizzi la replica dei log binari, il database MySQL esterno rimane aperto per transazioni mentre i dati sono in fase di migrazione verso il cluster database Aurora MySQL. Dopo la creazione del cluster di database Aurora MySQL utilizza la replica dei log binari per sincronizzare il cluster di database Aurora MySQL con le transazioni avvenute dopo il backup. Quando il cluster di database Aurora MySQL è aggiornato con il database MySQL, la migrazione viene terminata passando completamente al cluster di database Aurora MySQL per le nuove transazioni. Per ulteriori

informazioni, consulta Sincronizzazione del cluster database Amazon Aurora MySQL con il database MySQL utilizzando la replica.

Indice

- Considerazioni e limitazioni
- Prima di iniziare
  - Installazione di Percona XtraBackup
  - Autorizzazioni richieste
  - Creazione del ruolo del servizio IAM
- Esecuzione del backup di file da ripristinare come un cluster database Amazon Aurora MySQL
  - Creazione di un backup completo con Percona XtraBackup
  - Utilizzo di backup incrementali con Percona XtraBackup
  - Considerazioni sul backup
- Ripristino di un cluster di database Amazon Aurora MySQL da un bucket Amazon S3
- Sincronizzazione del cluster database Amazon Aurora MySQL con il database MySQL utilizzando la replica
  - Configurazione del database MySQL esterno e del cluster database Aurora MySQL per la replica crittografata
  - Sincronizza il cluster database Amazon Aurora MySQL con il database MySQL esterno
- Riduzione dei tempi di migrazione fisica ad Amazon Aurora MySQL
  - Tipi di tabella non supportati
  - Account utente con privilegi non supportati
  - Privilegi dinamici in Aurora MySQL versione 3
  - Oggetti archiviati con 'rdsadmin'@'localhost' come definer

Considerazioni e limitazioni

Le seguenti limitazioni e considerazioni si applicano al ripristino su un cluster database Amazon Aurora MySQL da un bucket Amazon S3:

- Puoi eseguire la migrazione dei tuoi dati solo a un nuovo cluster database e non a un cluster database esistente.

- È necessario utilizzare Percona per eseguire il backup dei dati XtraBackup su S3. Per ulteriori informazioni, consulta Installazione di Percona XtraBackup.
- Il bucket Amazon S3 e il cluster Aurora MySQL DB devono trovarsi nella stessa regione. AWS
- Non puoi eseguire il ripristino dai seguenti elementi:
  - Esportazione di uno snapshot cluster database in Amazon S3. Non è possibile eseguire la migrazione dei dati da un'esportazione snapshot del cluster database al bucket S3.
  - Un database di origine crittografato, ma puoi crittografare i dati migrati. Durante il processo di migrazione puoi anche lasciare i dati non crittografati.
  - Un database MySQL 5.5 o 5.6
- Percona Server per MySQL non è supportato come database di origine, perché può contenere tabelle `compression_dictionary*` nello schema `mysql`.
- Non è possibile eseguire il ripristino da un cluster database Aurora Serverless.
- La migrazione alle versioni precedenti non è supportata per le versioni principali e secondarie. Ad esempio, non puoi eseguire la migrazione da MySQL versione 8.0 ad Aurora MySQL versione 2 (compatibile con MySQL 5.7), né da MySQL versione 8.0.32 ad Aurora MySQL versione 3.03, che è compatibile con la versione 8.0.26 della community MySQL.
- Non è possibile migrare ad Aurora MySQL 3.05 e versioni successive da alcune versioni precedenti di MySQL 8.0, tra cui 8.0.11, 8.0.13 e 8.0.15. Si consiglia di eseguire l'aggiornamento a MySQL 8.0.28 prima della migrazione.
- L'importazione da Amazon S3 non è supportata sulla classe di istanza database db.t2.micro. Tuttavia, puoi eseguire il ripristino in una classe istanza database diversa e modificare la classe di istanza in seguito. Per maggiori informazioni sulle classi di istanza database, consulta Classi di istanze database Amazon Aurora.
- Amazon S3 limita la dimensione del file caricato in un bucket S3 a 5 TB. Se un file di backup supera i 5 TB, devi dividerlo in file più piccoli.
- Amazon RDS limita il numero di file caricati in un bucket S3 a 1 milione. Se i dati di backup del database, inclusi tutti i backup completi e incrementali, superano 1 milione di file, utilizza un file Gzip (.gz), tar (.tar.gz) o Percona xbstream (.xbstream) per archiviare i file dei backup completi e incrementali nel bucket S3. Percona XtraBackup 8.0 supporta solo Percona xbstream per la compressione.
- Per fornire servizi di gestione per ogni cluster database, viene creato l'utente `rdsadmin` al momento della creazione del cluster database. Poiché si tratta di un utente riservato in RDS, si applicano le seguenti limitazioni:

  - Le funzioni, le procedure, le viste, gli eventi e i trigger con il definer `'rdsadmin'@'localhost'` non vengono importati. Per ulteriori informazioni, consultare Oggetti archiviati con 'rdsadmin'@'localhost' come definer e Privilegi dell'utente master con Amazon Aurora MySQL..
  - Quando viene creato il cluster database Aurora MySQL, viene creato un utente master con i privilegi massimi supportati. Durante il ripristino dal backup, tutti i privilegi non supportati assegnati agli utenti importati vengono rimossi automaticamente durante l'importazione.

    Per identificare gli utenti che potrebbero essere interessati da questo problema, consulta Account utente con privilegi non supportati. Per ulteriori informazioni sui privilegi supportati in Aurora MySQL, consulta Privilegio basato sui ruoli.
- Per la versione 3 di Aurora MySQL, i privilegi dinamici non vengono importati. I privilegi dinamici supportati da Aurora possono essere importati dopo la migrazione. Per ulteriori informazioni, consulta Privilegi dinamici in Aurora MySQL versione 3.
- Le tabelle create dall'utente nello schema `mysql` non viene migrato.
- Il parametro `innodb_data_file_path` deve essere configurato con un solo file di dati che utilizza il nome di file di dati predefinito `ibdata1:12M:autoextend`. I database con due file di dati o con un file di dati con un nome diverso non possono essere migrati utilizzando questo metodo.

  Di seguito sono riportati esempi di nomi di file non consentiti: `innodb_data_file_path=ibdata1:50M`, `ibdata2:50M:autoextend` e `innodb_data_file_path=ibdata01:50M:autoextend`.
- Non puoi eseguire la migrazione da un database di origine con tabelle definite all'esterno della directory dei dati MySQL predefinita.
- La dimensione massima supportata per i backup non compressi che utilizzano questo metodo è attualmente limitata a 64 TiB. Per i backup compressi, questo limite viene ridotto per tenere conto dei requisiti dello spazio di decompressione. In questi casi, la dimensione massima di backup supportata sarebbe è (`64 TiB - compressed backup size`).
- Aurora MySQL non supporta l'importazione di MySQL e di altri componenti e plugin esterni.
- Aurora MySQL non ripristina tutto dal database. È consigliabile salvare lo schema e i valori del database per i seguenti elementi dal database MySQL di origine e quindi aggiungerli al cluster database Aurora MySQL ripristinato dopo che è stato creato:
  - Account utenti
  - Funzioni
  - Procedure archiviate

- Informazioni fuso orario. Le informazioni sul fuso orario vengono caricate dal sistema operativo locale per il cluster database Aurora MySQL. Per ulteriori informazioni, consulta Fuso orario locale per i cluster DB Amazon Aurora.

Prima di iniziare

Prima di poter copiare i dati in un bucket Amazon S3 ed eseguire il ripristino in un cluster database da tali file, devi eseguire quanto segue:

- Installa Percona sul tuo server locale XtraBackup .
- Permetti a Aurora MySQL di accedere al bucket Amazon S3 per tuo conto.

Installazione di Percona XtraBackup

Amazon Aurora può ripristinare un cluster DB da file creati utilizzando Percona. XtraBackup Puoi installare Percona XtraBackup da Software Downloads - Percona.

Per la migrazione a MySQL 5.7, usa Percona 2.4. XtraBackup

Per la migrazione a MySQL 8.0, usa Percona 8.0. XtraBackup Assicurati che la versione di Percona sia compatibile con la XtraBackup versione del motore del tuo database di origine.

Autorizzazioni richieste

Per migrare i dati MySQL in un cluster database Amazon Aurora MySQL, sono necessarie molte autorizzazioni:

- L'utente che richiede ad Aurora di creare un nuovo cluster da un bucket Amazon S3 deve avere l'autorizzazione per elencare i bucket per il proprio account. AWS Concedi all'utente questa autorizzazione utilizzando una AWS Identity and Access Management policy (IAM).
- Aurora richiede l'autorizzazione di agire per tuo conto per accedere al bucket Amazon S3 dove archivi i file utilizzati per creare il cluster di database Amazon Aurora MySQL. Concedi ad Aurora le autorizzazioni necessarie utilizzando un ruolo di servizio IAM.
- L'utente che effettua la richiesta deve avere anche l'autorizzazione di elencare i ruoli IAM per l'account AWS.
- Se l'utente che effettua la richiesta deve creare un ruolo di servizio IAM o richiedere che Aurora crei un ruolo di servizio IAM (utilizzando la console), l'utente deve avere l'autorizzazione di creare un ruolo IAM per l'account AWS.

- Se prevedi di crittografare i dati durante il processo di migrazione, aggiorna la policy IAM dell'utente che eseguirà la migrazione per concedere l'accesso RDS all'utente AWS KMS keys utilizzato per la crittografia dei backup. Per istruzioni, consulta Creazione di una policy IAM per accedere alle AWS KMS risorse.

Ad esempio, la seguente policy IAM concede all'utente le autorizzazioni minime necessarie per utilizzare la console per elencare ruoli IAM, creare un ruolo IAM, elencare i bucket Amazon S3 per l'account ed elencare le chiavi KMS.

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "iam:ListRoles",
                "iam:CreateRole",
                "iam:CreatePolicy",
                "iam:AttachRolePolicy",
                "s3:ListBucket",
                "kms:ListKeys"
            ],
            "Resource": "*"
        }
    ]
}
```

Inoltre, affinché un utente associ un ruolo IAM a un bucket Amazon S3, l'utente IAM deve avere l'autorizzazione `iam:PassRole` per quel ruolo IAM. Quest'autorizzazione permette a un amministratore di limitare quali ruoli IAM un utente può associare a un bucket Amazon S3.

Ad esempio, la policy IAM permette a un utente di associare il ruolo denominato `S3Access` a un bucket Amazon S3.

JSON

```
{
    "Version":"2012-10-17",
    "Statement":[
        {
            "Sid":"AllowS3AccessRole",
            "Effect":"Allow",
            "Action":"iam:PassRole",
            "Resource":"arn:aws:iam::123456789012:role/S3Access"
        }
    ]
}
```

Per ulteriori informazioni sulle autorizzazioni utente IAM, consulta Gestione dell'accesso tramite policy.

Creazione del ruolo del servizio IAM

Puoi Console di gestione AWS creare un ruolo per te scegliendo l'opzione Crea un nuovo ruolo (illustrata più avanti in questo argomento). Se selezioni questa opzione e specifichi un nome per il nuovo ruolo, Aurora crea il ruolo del servizio IAM necessario affinché Aurora acceda al bucket Amazon S3 con il nome che fornisci.

In alternativa, puoi creare manualmente il ruolo utilizzando la procedura seguente.

Per creare un ruolo IAM affinché Aurora possa accedere a Amazon S3

1. Completa le fasi descritte in Creazione di una policy IAM per l'accesso alle risorse Amazon S3.
2. Completa le fasi descritte in Creazione di un ruolo IAM per consentire ad Amazon Aurora di accedere ai servizi AWS.
3. Completa le fasi descritte in Associazione di un ruolo IAM a un cluster DB Amazon Aurora MySQL.

Esecuzione del backup di file da ripristinare come un cluster database Amazon Aurora MySQL

Puoi creare un backup completo dei file del tuo database MySQL utilizzando XtraBackup Percona e caricare i file di backup su un bucket Amazon S3. In alternativa, se utilizzi già Percona XtraBackup

per eseguire il backup dei file del tuo database MySQL, puoi caricare le directory e i file di backup completi e incrementali esistenti su un bucket Amazon S3.

Argomenti

- Creazione di un backup completo con Percona XtraBackup
- Utilizzo di backup incrementali con Percona XtraBackup
- Considerazioni sul backup

Creazione di un backup completo con Percona XtraBackup

Per creare un backup completo dei file del database MySQL che possono essere ripristinati da Amazon S3 per creare un cluster Aurora MySQL DB, utilizza l'utilità Percona () per eseguire il backup del database. XtraBackup `xtrabackup`

Ad esempio, il comando seguente consente di creare un backup di un database MySQL e memorizzare i file nella cartella `/on-premises/s3-restore/backup`.

```
xtrabackup --backup --user=<myuser> --password=<password> --target-dir=</on-premises/
s3-restore/backup>
```

Se desideri comprimere il backup in un singolo file (che può essere diviso, se necessario), puoi utilizzare l'opzione `--stream` per salvare il backup in uno dei seguenti formati:

- Gzip (.gz)
- tar (.tar)
- Percona xbstream (.xbstream)

Il comando seguente consente di creare un backup del database MySQL diviso in più file Gzip.

```
xtrabackup --backup --user=<myuser> --password=<password> --stream=tar \
   --target-dir=</on-premises/s3-restore/backup> | gzip - | split -d --bytes=500MB \
   - </on-premises/s3-restore/backup/backup>.tar.gz
```

Il comando seguente consente di creare un backup del database MySQL diviso in più file tar.

```
xtrabackup --backup --user=<myuser> --password=<password> --stream=tar \
```

```
   --target-dir=</on-premises/s3-restore/backup> | split -d --bytes=500MB \
   - </on-premises/s3-restore/backup/backup>.tar
```

Il comando seguente consente di creare un backup del database MySQL diviso in più file xbstream.

```
xtrabackup --backup --user=<myuser> --password=<password> --stream=xbstream \
   --target-dir=</on-premises/s3-restore/backup> | split -d --bytes=500MB \
   - </on-premises/s3-restore/backup/backup>.xbstream
```

> **Note**
>
> Se viene visualizzato il seguente errore, potrebbe essere causato dalla combinazione di formati di file nel comando:
>
> ```
> ERROR:/bin/tar: This does not look like a tar archive
> ```

Dopo aver eseguito il backup del database MySQL utilizzando l'utilità XtraBackup Percona, puoi copiare le directory e i file di backup in un bucket Amazon S3.

Per informazioni sulla creazione e il caricamento di un file in un bucket Amazon S3, consulta Nozioni di base su Amazon Simple Storage Service nella Guida alle operazioni di base di Amazon S3.

Utilizzo di backup incrementali con Percona XtraBackup

Amazon Aurora MySQL supporta backup completi e incrementali creati con Percona. XtraBackup Se utilizzi già Percona XtraBackup per eseguire backup completi e incrementali dei file del tuo database MySQL, non è necessario creare un backup completo e caricare i file di backup su Amazon S3. Puoi, invece, risparmiare tempo copiando le directory e i file di backup esistenti per i tuoi backup completi e incrementali in un bucket Amazon S3. Per ulteriori informazioni, consulta Create an incremental backup (Creazione di un backup incrementale) sul sito Web di Percona.

Durante la copia dei file del backup completo e incrementale in un bucket Amazon S3, devi copiare in modo ricorsivo i contenuti della directory di base. Questi contenuti includono il backup completo e anche tutte le directory e i file del backup incrementale. Questa copia deve mantenere la struttura di directory nel bucket Amazon S3. Aurora esegue l'iterazione di tutti i file e le directory. Aurora usa il file `xtrabackup-checkpoints` incluso con ogni backup incrementale per identificare la directory di base e ordinare i backup incrementali in base all'intervallo dei numeri di sequenza log (LSN).

Per informazioni sulla creazione e il caricamento di un file in un bucket Amazon S3, consulta Nozioni di base su Amazon Simple Storage Service nella Guida alle operazioni di base di Amazon S3.

Considerazioni sul backup

Aurora non supporta i backup parziali creati utilizzando Percona. XtraBackup Non puoi usare le opzioni seguenti per creare un backup parziale quando esegui il backup dei file di origine per il database: `--tables`, `--tables-exclude`, `--tables-file`, `--databases`, `--databases-exclude` o `--databases-file`.

Per ulteriori informazioni sul backup del database con Percona XtraBackup, consulta Percona XtraBackup - Documentation and Work with binary logs sul sito Web di Percona.

Aurora supporta i backup incrementali creati utilizzando Percona. XtraBackup Per ulteriori informazioni, consulta Create an incremental backup (Creazione di un backup incrementale) sul sito Web di Percona.

Aurora consuma i file di backup in base al nome del file. Assicurati di assegnare un nome ai file di backup con l'estensione file appropriata in base al formato file,— ad esempio, `.xbstream` per i file archiviati utilizzando il formato Percona xbstream.

Aurora consuma i file di backup in ordine alfabetico e anche in ordine numerico naturale. Utilizza sempre l'opzione `split` quando invii il comando `xtrabackup` per assicurarti che i file di backup vengano scritti e denominati nell'ordine corretto.

Amazon S3 limita la dimensione del file caricato in un bucket Amazon S3 a 5 TB. Se i dati di backup per il database eccedono 5 TB, utilizza il comando `split` per suddividere i file di backup in file multipli che hanno meno di 5 TB ciascuno.

Aurora limita il numero di file di origine caricati in un bucket Amazon S3 a 1 milione di file. In alcuni casi, i dati di backup per il database, compresi i backup completi e incrementali, possono ammontare a un numero alto di file. In questi casi, utilizza un file tarball (.tar.gz) per archiviare i file di backup completi e incrementali in un bucket Amazon S3.

Quando carichi un file in un bucket Amazon S3, puoi utilizzare la crittografia lato server per crittografare i dati. Puoi ripristinare un cluster database Amazon Aurora MySQL da quei file crittografati. Amazon Aurora MySQL può ripristinare un cluster di database con file crittografati utilizzando i seguenti tipi di crittografia lato server:

- Crittografia lato server con chiavi gestite da Amazon S3–, (SSE-S3): –ogni oggetto viene crittografato con una chiave unica che utilizza una crittografia avanzata a più fattori.

- Crittografia lato server con chiavi AWS KMS gestite (SSE-KMS): simile a SSE-S3, ma è possibile creare e gestire autonomamente le chiavi di crittografia e altre differenze.

Per informazioni sull'utilizzo della crittografia lato server durante il caricamento di file in un bucket Amazon S3, consulta Protezione dei dati con la crittografia lato server nella Guida per sviluppatori Amazon S3.

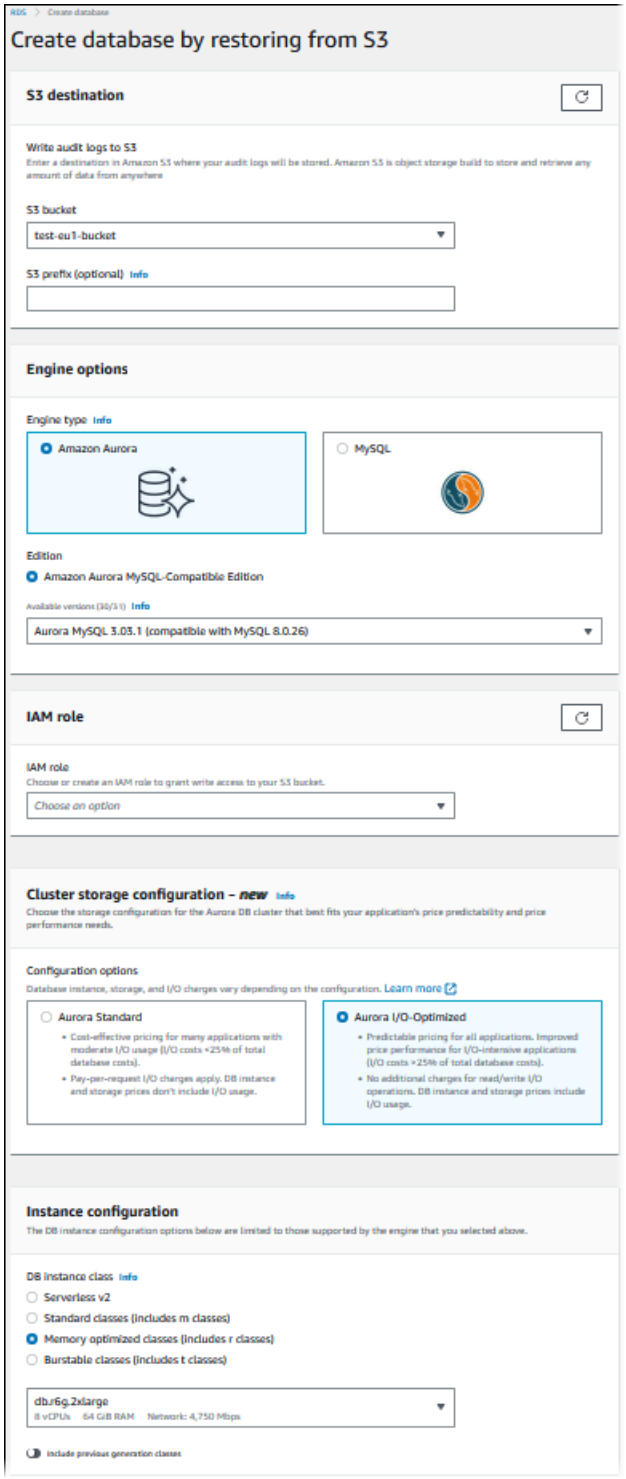
Ripristino di un cluster di database Amazon Aurora MySQL da un bucket Amazon S3

Puoi ripristinare i file di backup dal bucket Amazon S3 per creare un nuovo cluster di database Amazon Aurora MySQL utilizzando la console Amazon RDS.

Per ripristinare un cluster di database Amazon Aurora MySQL da file in un bucket Amazon S3

1. Accedi a Console di gestione AWS e apri la console Amazon RDS all'indirizzo https://console.aws.amazon.com/rds/.
2. Nell'angolo in alto a destra della console Amazon RDS, scegli la AWS regione in cui creare il tuo cluster DB. Scegli la stessa AWS regione del bucket Amazon S3 che contiene il backup del database.
3. Nel riquadro di navigazione, scegliere Databases (Database) e Restore from S3 (Ripristina da S3).
4. Seleziona Ripristina da S3.

   Sarà visualizzata la pagina Crea database ripristinando da S3 .



5. In Destinazione S3:

   a. Selezionare il bucket S3 che contiene i file di backup.

   b. (Facoltativo) Per S3 folder path prefix (Prefisso percorso cartella S3) inserire un prefisso del percorso per i file archiviati nel bucket Amazon S3.

Se non si specifica un prefisso, RDS crea l'istanza database utilizzando tutti i file e le cartelle nella cartella root del bucket S3. Se si specifica un prefisso, RDS crea l'istanza database utilizzando tutti i file e le cartelle nel bucket S3 in cui il percorso del file inizia con il prefisso specificato.

Ad esempio, si supponga di archiviare i file di backup su S3 in una sottocartella denominata backups e di disporre di più set di file di backup, ciascuno nella sua directory (gzip_backup1, gzip_backup2 e così via). In questo caso, specificare un prefisso di backups/gzip_backup1 per eseguire il ripristino dai file nella cartella gzip_backup1.

6. In Opzioni motore:

   a. Per Engine type (Tipo di motore), seleziona Amazon Aurora.

   b. Per Version (Versione), scegliere la versione del motore Aurora MySQL per l'istanza database ripristinata.

7. Per Ruolo IAM, puoi scegliere un ruolo IAM esistente.

8. (Facoltativo) Puoi anche creare un nuovo ruolo IAM selezionando Crea un nuovo ruolo. In tal caso:

   a. Specifica il nome del ruolo IAM.

   b. Specifica se consentire l'accesso alla chiave KMS:

      - Se non hai crittografato i file di backup, seleziona No.
      - Se hai crittografato i file di backup con AES-256 (SSE-S3) quando li hai caricati su Amazon S3, seleziona No. In questo caso, i dati vengono decrittati automaticamente.
      - Se hai crittografato i file di backup con la crittografia lato server AWS KMS (SSE-KMS) quando li hai caricati su Amazon S3, scegli Sì. In seguito, scegli la chiave corretta per AWS KMS key.

        Console di gestione AWSCrea una policy IAM che consente ad Aurora di decrittografare i dati.

   Per ulteriori informazioni, consulta Protezione dei dati con la crittografia lato server nella Guida per sviluppatori Amazon S3.

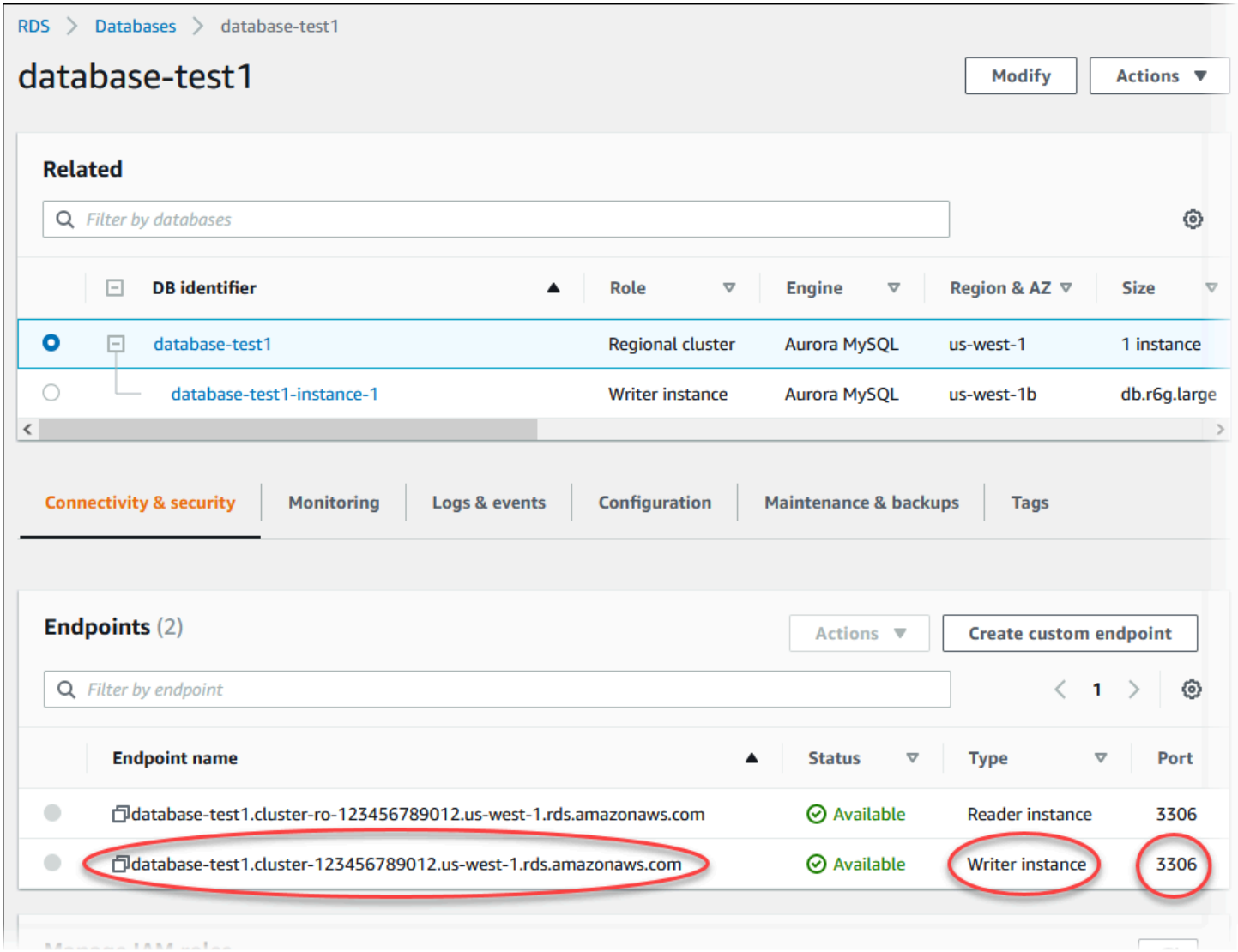
9. Scegliere le impostazioni per il cluster database, ad esempio la configurazione dell'archiviazione del cluster database, la classe di istanza database, l'identificatore del cluster database e le

credenziali di accesso. Per informazioni su ciascuna impostazione, consulta Impostazioni per cluster di database Aurora.

10. Personalizzare le impostazioni aggiuntive per il cluster database Aurora MySQL in base alle esigenze.
11. Scegliere Create database (Crea database) per avviare l'istanza database Aurora.

Nella console Amazon RDS la nuova istanza database viene visualizzata nell'elenco delle istanze database. L'istanza database rimane nello stato creating (creazione in corso) fino a quando non è stata creata e non è pronta per l'uso. Quando lo stato passa su available (disponibile), puoi connetterti all'istanza principale per il cluster database. A seconda della classe di istanza database e dello store allocato, potrebbero trascorrere diversi minuti prima che la nuova istanza sia disponibile.

Per visualizzare il cluster appena creato, seleziona la visualizzazione Databases (Database) nella console Amazon RDS quindi seleziona il cluster di database. Per ulteriori informazioni, consulta Visualizzazione di un cluster di database Amazon Aurora.



Prendi nota della porta e dell'endpoint di scrittura del cluster database. Utilizza la porta e l'endpoint di scrittura del cluster database nelle stringhe di connessione JDBC e ODBC per un'applicazione che esegue operazioni di scrittura o lettura.

Sincronizzazione del cluster database Amazon Aurora MySQL con il database MySQL utilizzando la replica

Per fare in modo che ci sia poco o nessun tempo di inattività durante la migrazione, puoi replicare le transazioni eseguite sul database MySQL nel cluster database Aurora MySQL. La replica permette al cluster database di essere aggiornato con le transazioni nel database MySQL avvenute durante la migrazione. Quando il cluster database è completamente aggiornato, puoi fermare la replica e terminare la migrazione su Aurora MySQL.

Argomenti

- Configurazione del database MySQL esterno e del cluster database Aurora MySQL per la replica crittografata
- Sincronizza il cluster database Amazon Aurora MySQL con il database MySQL esterno

Configurazione del database MySQL esterno e del cluster database Aurora MySQL per la replica crittografata

Per replicare i dati in maniera sicura, puoi utilizzare la replica crittografata.

> **Note**
>
> Se non hai bisogno di utilizzare la replica crittografata, puoi saltare queste fasi e passare alle istruzioni in Sincronizza il cluster database Amazon Aurora MySQL con il database MySQL esterno.

Seguono i prerequisiti per l'utilizzo della replica crittografata:

- Secure Sockets Layer (SSL) deve essere abilitato su un database primario MySQL esterno.
- Una chiave e un certificato client devono essere preparati per il cluster database Aurora MySQL.

Durante la replica crittografata, il cluster di database Aurora MySQL agisce come un client per il server di database MySQL. I certificati e le chiavi per il client Aurora MySQL sono in file in formato .pem.

Per configurare il database MySQL esterno e il cluster database Aurora MySQL per la replica crittografata

1. Assicurati di essere preparato per la replica crittografata:
   - Se SSL non è abilitato sul database MySQL primario esterno e non disponi di una chiave client e di un certificato client preparato, abilita SSL sul server di database MySQL e genera la chiave client e il certificato client necessari.
   - Se SSL è abilitato sul primario esterno, fornisci un certificato e una chiave client per il cluster database Aurora MySQL. Se non disponi di questi elementi, genera una nuova chiave e un nuovo certificato per il cluster di database Aurora MySQL. Per firmare il certificato client, devi avere la chiave autorità certificato che hai utilizzato per configurare SSL nel database primario esterno MySQL.

Per ulteriori informazioni, consulta Creating SSL Certificates and Keys Using openssl nella documentazione MySQL.

Hai bisogno del certificato autorità certificato, della chiave client e del certificato client.

2. Esegui la connessione al cluster database Aurora MySQL come primario utilizzando SSL.

   Per informazioni sulla connessione a un cluster di database Aurora MySQL con SSL, consulta Connessioni TLS a cluster di database Aurora MySQL.

3. Esegui la procedura archiviata mysql.rds_import_binlog_ssl_material per importare le informazioni SSL nel cluster di database Aurora MySQL.

   Per il parametro `ssl_material_value` inserisci le informazioni dai file in formato .pem per il cluster di database Aurora MySQL nel payload JSON corretto.

   L'esempio seguente importa le informazioni SSL in un cluster di database Aurora MySQL. Nei file in formato .pem, il codice del corpo è in genere più lungo del codice del corpo riportato nell'esempio.

```
call mysql.rds_import_binlog_ssl_material(
'{"ssl_ca":"-----BEGIN CERTIFICATE-----
AAAAB3NzaC1yc2EAAAADAQABAAABAQClKsfkNkuSevGj3eYhCe53pcjqP3maAhDFcvBS7O6V
hz2ItxCih+PnDSUaw+WNQn/mZphTk/a/gU8jEzoOWbkM4yxyb/wB96xbiFveSFJuOp/d6RJhJOI0iBXr
lsLnBItntckiJ7FbtxJMXLvvwJryDUilBMTjYtwB+QhYXUMOzce5Pjz5/i8SeJtjnV3iAoG/cQk+0FzZ
qaeJAAHco+CY/5WrUBkrHmFJr6HcXkvJdWPkYQS3xqC0+FmUZofz221CBt5IMucxXPkX4rWi+z7wB3Rb
BQoQzd8v7yeb7OzlPnWOyN0qFU0XA246RA8QFYiCNYwI3f05p6KLxEXAMPLE
-----END CERTIFICATE-----\n","ssl_cert":"-----BEGIN CERTIFICATE-----
AAAAB3NzaC1yc2EAAAADAQABAAABAQClKsfkNkuSevGj3eYhCe53pcjqP3maAhDFcvBS7O6V
hz2ItxCih+PnDSUaw+WNQn/mZphTk/a/gU8jEzoOWbkM4yxyb/wB96xbiFveSFJuOp/d6RJhJOI0iBXr
lsLnBItntckiJ7FbtxJMXLvvwJryDUilBMTjYtwB+QhYXUMOzce5Pjz5/i8SeJtjnV3iAoG/cQk+0FzZ
qaeJAAHco+CY/5WrUBkrHmFJr6HcXkvJdWPkYQS3xqC0+FmUZofz221CBt5IMucxXPkX4rWi+z7wB3Rb
BQoQzd8v7yeb7OzlPnWOyN0qFU0XA246RA8QFYiCNYwI3f05p6KLxEXAMPLE
-----END CERTIFICATE-----\n","ssl_key":"-----BEGIN RSA PRIVATE KEY-----
AAAAB3NzaC1yc2EAAAADAQABAAABAQClKsfkNkuSevGj3eYhCe53pcjqP3maAhDFcvBS7O6V
hz2ItxCih+PnDSUaw+WNQn/mZphTk/a/gU8jEzoOWbkM4yxyb/wB96xbiFveSFJuOp/d6RJhJOI0iBXr
lsLnBItntckiJ7FbtxJMXLvvwJryDUilBMTjYtwB+QhYXUMOzce5Pjz5/i8SeJtjnV3iAoG/cQk+0FzZ
qaeJAAHco+CY/5WrUBkrHmFJr6HcXkvJdWPkYQS3xqC0+FmUZofz221CBt5IMucxXPkX4rWi+z7wB3Rb
BQoQzd8v7yeb7OzlPnWOyN0qFU0XA246RA8QFYiCNYwI3f05p6KLxEXAMPLE
-----END RSA PRIVATE KEY-----\n"}');
```

Per ulteriori informazioni, consultare mysql.rds_import_binlog_ssl_material e Connessioni TLS a cluster di database Aurora MySQL.

> **Note**
>
> Dopo aver eseguito la procedura, i segreti vengono archiviati in file. Per eliminare i file in un secondo momento, puoi eseguire la stored procedure mysql.rds_remove_binlog_ssl_material.

Sincronizza il cluster database Amazon Aurora MySQL con il database MySQL esterno

Puoi sincronizzare i cluster database Amazon Aurora MySQL con il database MySQL utilizzando la replica.

Per sincronizzare il cluster database Aurora MySQL con il database MySQL utilizzando la replica

1. Assicurati che il file /etc/my.cnf per il database MySQL esterno abbia le voci rilevanti.

   Se la replica crittografata non è necessaria, assicurati che il database MySQL esterno venga avviato con i bin binari (binlog) abilitati e SSL disabilitato. Seguono le voci rilevanti nel file /etc/my.cnf per dati crittografati.

   ```
   log-bin=mysql-bin
   server-id=2133421
   innodb_flush_log_at_trx_commit=1
   sync_binlog=1
   ```

   Se la replica crittografata è necessaria, assicurati che il database MySQL esterno venga avviato con SSL e i binlog abilitati. Le voci rilevanti nel file /etc/my.cnf includono le posizioni del file .pem per il server di database MySQL.

   ```
   log-bin=mysql-bin
   server-id=2133421
   innodb_flush_log_at_trx_commit=1
   sync_binlog=1
   ```

```
# Setup SSL.
ssl-ca=/home/sslcerts/ca.pem
ssl-cert=/home/sslcerts/server-cert.pem
ssl-key=/home/sslcerts/server-key.pem
```

Puoi verificare l'abilitazione di SSL utilizzando il seguente comando.

```
mysql> show variables like 'have_ssl';
```

L'output deve essere simile a quanto segue.

```
+~-~-~-~-~-~-~-~-~-~-~-~-~-~--+~-~-~-~-~-~--+
| Variable_name | Value |
+~-~-~-~-~-~-~-~-~-~-~-~-~-~--+~-~-~-~-~-~--+
| have_ssl      | YES   |
+~-~-~-~-~-~-~-~-~-~-~-~-~-~--+~-~-~-~-~-~--+
1 row in set (0.00 sec)
```

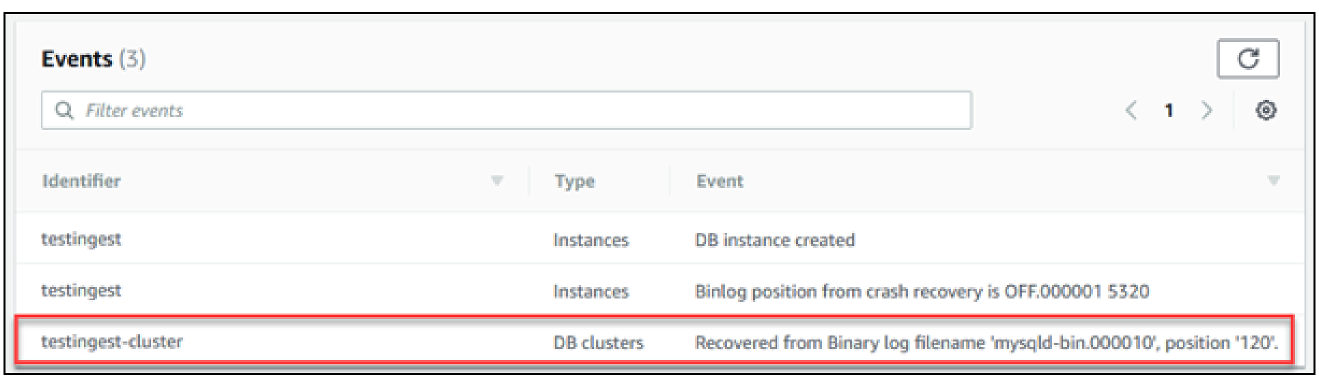
2. Determina la posizione iniziale del log binario per la replica: Specifichi la posizione per avviare la replica in una fase successiva.

   Utilizzando il Console di gestione AWS

   a. Accedi a Console di gestione AWS e apri la console Amazon RDS all'indirizzo https://console.aws.amazon.com/rds/.

   b. Nel riquadro di navigazione selezionare Events (Eventi).

   c. Nell'elenco Events (Eventi), annota la posizione nell'evento Recovered from Binary log filename (Recuperato dal nome di file del log binario).



Utilizzando il AWS CLI

È inoltre possibile ottenere il nome e la posizione del file binlog utilizzando il comando AWS CLIdescribe-events. Di seguito viene illustrato un esempio del comando `describe-events`.

```
PROMPT> aws rds describe-events
```

Nell'output, identifica l'evento che mostra la posizione del binlog.

3. Durante la connessione al database esterno MySQL, crea un utente da utilizzare per la replica. Questo account viene utilizzato unicamente per la replica e deve essere limitato al dominio personale per aumentare la sicurezza. Di seguito è riportato un esempio.

```
mysql> CREATE USER '<user_name>'@'<domain_name>' IDENTIFIED BY '<password>';
```

L'utente richiede i privilegi `REPLICATION CLIENT` e `REPLICATION SLAVE`. Concedi questi privilegi all'utente.

```
GRANT REPLICATION CLIENT, REPLICATION SLAVE ON *.* TO
 '<user_name>'@'<domain_name>';
```

Se non hai bisogno di utilizzare la replica crittografata, richiedi le connessioni SSL per l'utente replica. Ad esempio, puoi usare la seguente istruzione per richiedere connessioni SSL per l'account utente *`<user_name>`*.

```
GRANT USAGE ON *.* TO '<user_name>'@'<domain_name>' REQUIRE SSL;
```

> **Note**
>
> Se `REQUIRE SSL` non è incluso, la connessione di replica potrebbe ridiventare una connessione non crittografata.

4. Nella console Amazon RDS, aggiungi l'indirizzo IP del server che ospita il database MySQL esterno al gruppo di sicurezza VPC per il cluster di database Aurora MySQL. Per ulteriori

informazioni sulla modifica di un gruppo di sicurezza VPC, consulta Gruppi di sicurezza per il VPC nella Guida per l'utente di Amazon Virtual Private Cloud.

Potrebbe anche essere necessario configurare la rete locale per consentire le connessioni dall'indirizzo IP del cluster database Aurora MySQL, affinché possa comunicare con il database MySQL esterno. Per individuare l'indirizzo IP del cluster di database Aurora MySQL, utilizzare il comando `host`.

```
host <db_cluster_endpoint>
```

Il nome host è il nome DNS dell'endpoint del cluster di database Aurora MySQL.

5. Abilita la replica del log binario eseguendo la stored procedure mysql.rds_reset_external_master (Aurora MySQL versione 2) o mysql.rds_reset_external_source (Aurora MySQL versione 3). Questa procedura archiviata ha la seguente sintassi.

```
CALL mysql.rds_set_external_master (
  host_name
  , host_port
  , replication_user_name
  , replication_user_password
  , mysql_binary_log_file_name
  , mysql_binary_log_file_location
  , ssl_encryption
);

CALL mysql.rds_set_external_source (
  host_name
  , host_port
  , replication_user_name
  , replication_user_password
  , mysql_binary_log_file_name
  , mysql_binary_log_file_location
  , ssl_encryption
);
```

Per informazioni sui parametri, consulta mysql.rds_reset_external_master (Aurora MySQL versione 2) e mysql.rds_reset_external_source (Aurora MySQL versione 3).

Per `mysql_binary_log_file_name` e `mysql_binary_log_file_location`, utilizza la posizione nell'evento Recovered from Binary log filename (Recuperato dal nome di file del log binario) che hai annotato in precedenza.

Se i dati nel cluster di database Aurora MySQL non sono crittografati, il parametro `ssl_encryption` deve essere impostato su `0`. Se i dati sono crittografati, il parametro `ssl_encryption`, deve essere impostato su `1`.

L'esempio seguente esegue la procedura per un cluster database Aurora MySQL che ha dati crittografati.

```
CALL mysql.rds_set_external_master(
  'Externaldb.some.com',
  3306,
  'repl_user'@'mydomain.com',
  'password',
  'mysql-bin.000010',
  120,
  1);

CALL mysql.rds_set_external_source(
  'Externaldb.some.com',
  3306,
  'repl_user'@'mydomain.com',
  'password',
  'mysql-bin.000010',
  120,
  1);
```

Questa procedura archiviata imposta i parametri che il cluster database Aurora MySQL utilizza per la connessione al database MySQL esterno e per leggere il log binario. Se i dati sono crittografati, scarica anche il certificato autorità certificato SSL, il certificato client e la chiave client nel disco locale.

6. Avvia la replica del log binario eseguendo la procedura archiviata mysql.rds_start_replication.

```
CALL mysql.rds_start_replication;
```

7. Monitora quanto è indietro il cluster database Aurora MySQL rispetto al database primario MySQL di replica. A questo scopo, esegui la connessione al cluster database Aurora MySQL ed esegui il comando seguente.

```
Aurora MySQL version 2:
SHOW SLAVE STATUS;

Aurora MySQL version 3:
SHOW REPLICA STATUS;
```

   Nell'output del comando, il campo `Seconds Behind Master` indica quanto è indietro il cluster database Aurora MySQL rispetto al primario MySQL. Quando questo valore corrisponde a `0` (zero), il cluster database Aurora MySQL è aggiornato rispetto al primario e puoi passare alla fase successiva per interrompere la replica.

8. Esegui la connessione al database primario MySQL di replica e interrompi la replica. Per eseguire questa operazione, chiama la stored procedure mysql.rds_stop_replication.

```
CALL mysql.rds_stop_replication;
```

Riduzione dei tempi di migrazione fisica ad Amazon Aurora MySQL

Puoi apportare le seguenti modifiche al database per velocizzare il processo di migrazione di un database ad Amazon Aurora MySQL.

> **Important**
>
> Assicurati di eseguire questi aggiornamenti su una copia di un database di produzione, anziché sul database di produzione stesso. Puoi eseguire il backup della copia e quindi ripristinarlo nel cluster database Aurora MySQL per evitare interruzioni del servizio nel database di produzione.

Tipi di tabella non supportati

Aurora MySQL supporta solo il motore InnoDB per le tabelle di database. Se il database include tabelle MyISAM, queste devono essere convertite prima della loro migrazione ad Aurora MySQL. Il

processo di conversione richiede ulteriore spazio per la conversione da MyISAM a InnoDB durante la procedura di migrazione.

Per ridurre la possibilità di rimanere senza spazio o per velocizzare il processo di migrazione, converti tutte le tabelle MyISAM in tabelle InnoDB prima di migrarle. Le dimensioni della tabella InnoDB risultante è equivalente alle dimensioni necessarie da Aurora MySQL per quella tabella. Per convertire una tabella MyISAM in InnoDB, esegui il comando seguente:

```
ALTER TABLE schema.table_name engine=innodb, algorithm=copy;
```

Aurora MySQL non supporta le pagine o le tabelle compresse, ovvero le tabelle create con `ROW_FORMAT=COMPRESSED` o `COMPRESSION = {"zlib"|"lz4"}`.

Per ridurre la possibilità di rimanere senza spazio o per velocizzare il processo di migrazione, espandi le tabelle compresse impostando `ROW_FORMAT` su `DEFAULT`, `COMPACT`, `DYNAMIC`, o `REDUNDANT`. Per le pagine compresse, imposta `COMPRESSION="none"`.

Per ulteriori informazioni, consulta InnoDB row formats e InnoDB table and page compression nella documentazione di MySQL.

Puoi utilizzare il seguente script SQL nell'istanza database MySQL esistente per elencare le tabelle nel database che sono tabelle MyISAM o compresse.

```
-- This script examines a MySQL database for conditions that block
-- migrating the database into Aurora MySQL.
-- It must be run from an account that has read permission for the
-- INFORMATION_SCHEMA database.

-- Verify that this is a supported version of MySQL.

select msg as `==> Checking current version of MySQL.`
from
  (
  select
    'This script should be run on MySQL version 5.6 or higher. ' +
    'Earlier versions are not supported.' as msg,
    cast(substring_index(version(), '.', 1) as unsigned) * 100 +
      cast(substring_index(substring_index(version(), '.', 2), '.', -1)
      as unsigned)
    as major_minor
  ) as T
```

```
where major_minor <> 506;


-- List MyISAM and compressed tables. Include the table size.

select concat(TABLE_SCHEMA, '.', TABLE_NAME) as `==> MyISAM or Compressed Tables`,
round(((data_length + index_length) / 1024 / 1024), 2) "Approx size (MB)"
from INFORMATION_SCHEMA.TABLES
where
  ENGINE <> 'InnoDB'
  and
  (
    -- User tables
    TABLE_SCHEMA not in ('mysql', 'performance_schema',
                         'information_schema')
    or
    -- Non-standard system tables
    (
      TABLE_SCHEMA = 'mysql' and TABLE_NAME not in
        (
          'columns_priv', 'db', 'event', 'func', 'general_log',
          'help_category', 'help_keyword', 'help_relation',
          'help_topic', 'host', 'ndb_binlog_index', 'plugin',
          'proc', 'procs_priv', 'proxies_priv', 'servers', 'slow_log',
          'tables_priv', 'time_zone', 'time_zone_leap_second',
          'time_zone_name', 'time_zone_transition',
          'time_zone_transition_type', 'user'
        )
    )
  )
  or
  (
    -- Compressed tables
       ROW_FORMAT = 'Compressed'
  );
```

Account utente con privilegi non supportati

Gli account utente con privilegi non supportati da Aurora MySQL vengono importati senza i privilegi non supportati. Per l'elenco dei privilegi supportati, consulta Privilegio basato sui ruoli.

Puoi eseguire la seguente query SQL sul database di origine per elencare gli account utente con privilegi non supportati.

```
SELECT
    user,
    host
FROM
    mysql.user
WHERE
    Shutdown_priv = 'y'
    OR File_priv = 'y'
    OR Super_priv = 'y'
    OR Create_tablespace_priv = 'y';
```

Privilegi dinamici in Aurora MySQL versione 3

I privilegi dinamici non vengono importati. Aurora MySQL versione 3 supporta i privilegi dinamici seguenti.

```
'APPLICATION_PASSWORD_ADMIN',
'CONNECTION_ADMIN',
'REPLICATION_APPLIER',
'ROLE_ADMIN',
'SESSION_VARIABLES_ADMIN',
'SET_USER_ID',
'XA_RECOVER_ADMIN'
```

Lo script di esempio seguente concede i privilegi dinamici supportati agli account utente nel cluster database Aurora MySQL.

```
-- This script finds the user accounts that have Aurora MySQL supported dynamic
 privileges
-- and grants them to corresponding user accounts in the Aurora MySQL DB cluster.

/home/ec2-user/opt/mysql/8.0.26/bin/mysql -uusername -pxxxxx -P8026 -h127.0.0.1 -BNe
 "SELECT
  CONCAT('GRANT ', GRANTS, ' ON *.* TO ', GRANTEE ,';') AS grant_statement
  FROM (select GRANTEE, group_concat(privilege_type) AS GRANTS FROM
 information_schema.user_privileges
      WHERE privilege_type IN (
        'APPLICATION_PASSWORD_ADMIN',
        'CONNECTION_ADMIN',
        'REPLICATION_APPLIER',
        'ROLE_ADMIN',
        'SESSION_VARIABLES_ADMIN',
```

```
        'SET_USER_ID',
        'XA_RECOVER_ADMIN')
      AND GRANTEE NOT IN (\"'mysql.session'@'localhost'\",
\"'mysql.infoschema'@'localhost'\",\"'mysql.sys'@'localhost'\") GROUP BY GRANTEE)
      AS PRIVGRANTS; " | /home/ec2-user/opt/mysql/8.0.26/bin/mysql -u master_username -
p master_password -h DB_cluster_endpoint
```

Oggetti archiviati con 'rdsadmin'@'localhost' come definer

Le funzioni, le procedure, le viste, gli eventi e i trigger con `'rdsadmin'@'localhost'` come definer non vengono importati.

Puoi usare il seguente script SQL nel database MySQL di origine per elencare gli oggetti archiviati con il definire non supportato.

```
-- This SQL query lists routines with `rdsadmin`@`localhost` as the definer.

SELECT
    ROUTINE_SCHEMA,
    ROUTINE_NAME
FROM
    information_schema.routines
WHERE
    definer = 'rdsadmin@localhost';

-- This SQL query lists triggers with `rdsadmin`@`localhost` as the definer.

SELECT
    TRIGGER_SCHEMA,
    TRIGGER_NAME,
    DEFINER
FROM
    information_schema.triggers
WHERE
    DEFINER = 'rdsadmin@localhost';

-- This SQL query lists events with `rdsadmin`@`localhost` as the definer.

SELECT
    EVENT_SCHEMA,
    EVENT_NAME
FROM
    information_schema.events
```

```
WHERE
    DEFINER = 'rdsadmin@localhost';

-- This SQL query lists views with `rdsadmin`@`localhost` as the definer.
SELECT
    TABLE_SCHEMA,
    TABLE_NAME
FROM
    information_schema.views
WHERE
    DEFINER = 'rdsadmin@localhost';
```

Migrazione logica da MySQL ad Amazon Aurora MySQL mediante mysqldump

Poiché Amazon Aurora MySQL è un database compatibile con MySQL, puoi utilizzare l'utilità `mysqldump` per copiare i dati dal database MySQL oppure l'utilità `mariadb-dump` per copiare i dati dal database MariaDB database in un cluster di database Aurora MySQL esistente.

Per informazioni su come eseguire questa procedura con i database MySQL o MariaDB di dimensioni molto grandi, consulta i seguenti argomenti nella Guida per l'utente di Amazon Relational Database Service:

- MySQL: Importazione dei dati in un database Amazon RDS per MySQL con tempo di inattività ridotto
- MariaDB: Importazione dei dati in un database Amazon RDS per MariaDB con tempo di inattività ridotto

Per i database MySQL o MariaDB che contengono una quantità inferiore di dati, consulta i seguenti argomenti nella Guida per l'utente di Amazon Relational Database Service:

- MySQL: Importazione di dati da un database MySQL esterno in un'istanza database Amazon RDS per MySQL
- MariaDB: Importazione di dati da un database MariaDB esterno in un'istanza database Amazon RDS per MariaDB

Migrazione dei dati da un'istanza database RDS per MySQL a un cluster di database Amazon Aurora MySQL

Puoi migrare (copiare) i dati a un cluster di database Amazon Aurora MySQL direttamente da un'istanza database RDS per MySQL.

Argomenti

- Migrazione di una snapshot RDS for MySQL a Aurora
- Migrazione di dati da un'istanza database RDS per MySQL a un cluster database Amazon Aurora MySQL utilizzando una replica di lettura Aurora

> **Note**
>
> Poiché Amazon Aurora MySQL è compatibile con MySQL, puoi migrare i dati dal database MySQL impostando la replica tra il database MySQL e un cluster di database Amazon Aurora MySQL. Per ulteriori informazioni, consulta Replica con Amazon Aurora.

Migrazione di una snapshot RDS for MySQL a Aurora

Puoi effettuare la migrazione di uno snapshot DB di un'istanza database RDS for MySQL per creare un cluster database Aurora MySQL. Il nuovo cluster DB Aurora MySQL viene popolato con i dati dall'istanza database RDS for MySQL originale. Lo snapshot di database deve essere realizzato da un'istanza database Amazon RDS che esegue MySQL compatibile con Aurora MySQL.

Puoi effettuare la migrazione di una snapshot DB manuale o automatizzata. Dopo aver creato il cluster database, puoi le creare le repliche Aurora facoltative.

> **Note**
>
> Puoi anche migrare un'istanza database RDS per MySQL a un cluster di database Aurora MySQL creando una replica di lettura Aurora dell'istanza database RDS per MySQL di origine. Per ulteriori informazioni, consulta Migrazione di dati da un'istanza database RDS per MySQL a un cluster database Amazon Aurora MySQL utilizzando una replica di lettura Aurora.

Non è possibile migrare ad Aurora MySQL 3.05 e versioni successive da alcune versioni precedenti di MySQL 8.0, tra cui 8.0.11, 8.0.13 e 8.0.15. Si consiglia di eseguire l'aggiornamento a MySQL 8.0.28 prima della migrazione.

La procedura generale da seguire è riportata di seguito:

1. Determina la quantità di spazio da predisporre per il cluster database Aurora MySQL. Per ulteriori informazioni, consulta Di quanto spazio ho bisogno?
2. Utilizza la console per creare lo snapshot nella regione AWS in cui si trova l'istanza Amazon RDS MySQL. Per informazioni sulla creazione di uno snapshot DB, consulta Creazione di uno snapshot DB.
3. Se lo snapshot DB non si trova nella stessa regione AWS del cluster database, utilizza la console Amazon RDS per copiare lo snapshot DB in quella regione AWS. Per informazioni sulla copia di una snapshot DB, consulta Copia di una snapshot DB.
4. Utilizza la console per migrare la snapshot DB e creare un cluster database Aurora MySQL con gli stessi database dell'istanza database MySQL originale.

⚠ Warning

Amazon RDS limita ogni account AWS a una copia di snapshot in ogni regione AWS alla volta.

Di quanto spazio ho bisogno?

Quando effettui la migrazione di una snapshot di un'istanza database MySQL in un cluster di database Aurora MySQL, Aurora utilizza un volume Amazon Elastic Block Store (Amazon EBS) per formattare i dati dalla snapshot prima della migrazione. In alcuni casi, è necessario ulteriore spazio per formattare i dati per la migrazione.

Le tabelle che non sono tabelle MyISAM e non sono compresse possono avere 16 TB di dimensioni. Se hai tabelle MyISAM, Aurora deve utilizzare ulteriori spazio nel volume per convertire le tabelle in modo che siano compatibili con Aurora MySQL. Se hai tabelle compresse, Aurora deve utilizzare ulteriore spazio nel volume per espandere queste tabelle prima di archiviarle nel volume cluster Aurora. A causa dei requisiti di spazio aggiuntivo, devi accertarti che nessuna delle tabelle MyISAM e compresse migrate dall'istanza database MySQL abbia una dimensione superiore a 8 TB.

Riduzione della quantità di spazio necessaria per migrare i dati in Amazon Aurora MySQL

Potrebbe essere necessario modificare lo schema del database prima di migrarlo in Amazon Aurora. Tale modifica può essere utile nei casi seguenti:

- Desideri accelerare il processo di migrazione.
- Non sai con precisione di quanto spazio necessiti.
- Hai provato a effettuare la migrazione dei dati ma la migrazione non è andata a buon fine per insufficienza di spazio.

Puoi applicare le seguenti modifiche per migliorare il processo di migrazione di un database in Amazon Aurora.

> **Important**
>
> Accertati di eseguire questi aggiornamenti in una nuova istanza database recuperata da una snapshot di un database di produzione, piuttosto che su un'istanza di produzione. Puoi migrare i dati dalla snapshot della nuova istanza database nel cluster database Aurora per evitare interruzioni di servizio nel database di produzione.

Tipo tabella	Limite o linea guida
Tabelle MyISAM	Aurora MySQL supporta solo tabelle InnoDB. Se hai tabelle MyISAM nel database, devono essere convertite prima di essere migrate in Aurora MySQL. Il processo di conversione richiede ulteriore spazio per la conversione da MyISAM a InnoDB durante la procedura di migrazione. Per ridurre la possibilità di rimanere senza spazio o per velocizza re il processo di migrazione, converti tutte le tabelle MyISAM in tabelle InnoDB prima di migrarle. Le dimensioni della tabella InnoDB risultante è equivalente alle dimensioni necessarie da Aurora MySQL per quella tabella. Per convertire una tabella MyISAM in InnoDB, esegui il comando seguente:

Tipo tabella	Limite o linea guida
	`alter table <schema>.<table_name> engine=innodb, algorithm=copy;`
Tabelle compresse	Aurora MySQL non supporta tabelle compresse (ovvero tabelle create con `ROW_FORMAT=COMPRESSED`). Per ridurre la possibilità di rimanere senza spazio o per velocizzare il processo di migrazione, espandi le tabelle compresse impostando `ROW_FORMAT` su `DEFAULT`, `COMPACT`, `DYNAMIC`, o `REDUNDANT`. Per ulteriori informazioni, consulta Formati di riga in InnoDB nella documentazione di MySQL.

Puoi utilizzare il seguente script SQL nell'istanza database MySQL esistente per elencare le tabelle nel database che sono tabelle MyISAM o compresse.

```
-- This script examines a MySQL database for conditions that block
-- migrating the database into Amazon Aurora.
-- It needs to be run from an account that has read permission for the
-- INFORMATION_SCHEMA database.

-- Verify that this is a supported version of MySQL.

select msg as `==> Checking current version of MySQL.`
from
  (
  select
    'This script should be run on MySQL version 5.6 or higher. ' +
    'Earlier versions are not supported.' as msg,
    cast(substring_index(version(), '.', 1) as unsigned) * 100 +
      cast(substring_index(substring_index(version(), '.', 2), '.', -1)
      as unsigned)
    as major_minor
  ) as T
where major_minor <> 506;


-- List MyISAM and compressed tables. Include the table size.

select concat(TABLE_SCHEMA, '.', TABLE_NAME) as `==> MyISAM or Compressed Tables`,
```

```
round(((data_length + index_length) / 1024 / 1024), 2) "Approx size (MB)"
from INFORMATION_SCHEMA.TABLES
where
  ENGINE <> 'InnoDB'
  and
  (
    -- User tables
    TABLE_SCHEMA not in ('mysql', 'performance_schema',
                         'information_schema')
    or
    -- Non-standard system tables
    (
      TABLE_SCHEMA = 'mysql' and TABLE_NAME not in
        (
          'columns_priv', 'db', 'event', 'func', 'general_log',
          'help_category', 'help_keyword', 'help_relation',
          'help_topic', 'host', 'ndb_binlog_index', 'plugin',
          'proc', 'procs_priv', 'proxies_priv', 'servers', 'slow_log',
          'tables_priv', 'time_zone', 'time_zone_leap_second',
          'time_zone_name', 'time_zone_transition',
          'time_zone_transition_type', 'user'
        )
    )
  )
  or
  (
    -- Compressed tables
       ROW_FORMAT = 'Compressed'
  );
```

Lo script produce output simile all'output nell'esempio seguente. L'esempio visualizza due tabelle che devono essere convertite da MyISAM a InnoDB. L'output include anche le dimensioni approssimative di ogni tabella in megabyte (MB).

```
+---------------------------------+------------------+
| ==> MyISAM or Compressed Tables | Approx size (MB) |
+---------------------------------+------------------+
| test.name_table                 |          2102.25 |
| test.my_table                   |            65.25 |
+---------------------------------+------------------+
2 rows in set (0.01 sec)
```

Migrazione di uno snapshot DB di RDS per MySQL in un cluster database Aurora MySQL

Puoi effettuare la migrazione di uno snapshot DB di un'istanza database di RDS per MySQL per creare un cluster database Aurora MySQL utilizzando la Console di gestione AWS o la AWS CLI. Il nuovo cluster DB Aurora MySQL viene popolato con i dati dall'istanza database RDS for MySQL originale. Per informazioni sulla creazione di uno snapshot DB, consulta Creazione di uno snapshot DB.

Se lo snapshot DB non si trova nella regione AWS in cui desideri posizionare i dati, copia lo snapshot DB nella regione AWS. Per informazioni sulla copia di una snapshot DB, consulta Copia di una snapshot DB.

Console

Quando esegui la migrazione dello snapshot di database utilizzando la Console di gestione AWS, vengono eseguite le operazioni necessarie per creare solo il cluster di database.

Puoi decidere anche che il nuovo cluster DB Aurora MySQL venga crittografato a riposo utilizzando una AWS KMS key.

Per effettuare una snapshot DB MySQL utilizzando la Console di gestione AWS

1. Accedi alla Console di gestione AWS e apri la console Amazon RDS all'indirizzo https://console.aws.amazon.com/rds/.
2. Inizia la migrazione dall'istanza database MySQL o dalla snapshot:

   Per iniziare la migrazione dall'istanza database:

   1. Nel riquadro di navigazione, seleziona Databases (Database), quindi seleziona l'istanza database MySQL.
   2. In Actions (Operazioni) scegliere Migrate latest snapshot (Migrazione snapshot più recente).

   Per iniziare la migrazione dalla snapshot:

   1. SelezionareSnapshots (Snapshot).
   2. Nella pagina Snapshots (Snapshot) seleziona la snapshot che desideri migrare in un cluster di database Aurora MySQL.
   3. Scegliere Snapshot Actions (Operazioni snapshot), quindi scegliere Migrate Snapshot (Migrazione snapshot).

Viene visualizzata la pagina Migrate Database (Migrazione database).

3. Impostare i valori seguenti nella pagina Migrate Database (Migra database):

   - Migrate to DB Engine (Migrazione al motore DB): seleziona `aurora`.
   - DB Engine Version (Versione motore database): seleziona la versione di motore database per il cluster di database Aurora MySQL.
   - Classe di istanza database: scegliere una classe dell'istanza database che abbia lo spazio di archiviazione e la capacità richiesti per il database, ad esempio `db.r3.large`. I volumi dei cluster Aurora aumentano automaticamente quando aumenta la quantità di dati nel database. Un volume del cluster Aurora può raggiungere una dimensione massima di 128 tebibytes (TiB). È quindi sufficiente selezionare una classe di istanza database che soddisfi i requisiti di storage correnti. Per ulteriori informazioni, consulta Panoramica dell'archiviazione di Amazon Aurora.
   - Identificatore istanze DB: digitare un nome per il cluster database univoco per l'account nella regione AWS selezionata. Questo identificatore viene utilizzato negli indirizzi degli endpoint per le istanze nel cluster database. Puoi scegliere di aggiungere informazioni utili al nome, ad esempio includendo la regione AWS e il motore di database selezionato, come **`aurora-cluster1`**.

     L'identificatore istanze database presenta i seguenti vincoli:

     - Deve contenere da 1 a 63 caratteri alfanumerici o trattini.
     - Il primo carattere deve essere una lettera.
     - Non può terminare con un trattino o contenere due trattini consecutivi.
     - Deve essere univoco per tutte le istanze database, per ogni account AWS e in ogni regione AWS.
   - Virtual Private Cloud (VPC): se hai già un VPC esistente, puoi utilizzarlo con il cluster di database Aurora MySQL selezionando l'identificatore VPC, ad esempio `vpc-a464d1c1`. Per ulteriori informazioni sulla creazione di un VPC, consulta Tutorial: Creazione di un Amazon VPC da utilizzare con un cluster database (solo IPv4).

     Diversamente, puoi stabilire che Aurora crei un VPC per te selezionando Create a new VPC (Crea un nuovo VPC).

- DB subnet group (Gruppo di sottoreti DB): un gruppo di sottoreti esistente può essere utilizzato con il cluster database Aurora MySQL selezionando l'identificatore del gruppo di sottoreti, ad esempio `gs-subnet-group1`.

  Diversamente, puoi stabilire che Aurora crei un gruppo di sottoreti selezionando Create a new subnet group (Crea nuovo gruppo di sottoreti).

- Public accessibility (Accesso pubblico): seleziona No (No) per specificare che le istanze nel cluster database si possono accedere solamente da risorse all'interno del proprio VPC. Selezionare Yes (Si) per specificare che le istanze nel cluster database sono accessibili dalle risorse nella rete pubblica. Il valore predefinito è Yes (Sì).

  > **Note**
  >
  > Il cluster DB di produzione potrebbe non trovarsi in una sottorete pubblica, poiché solo i server applicativi richiedono l'accesso al cluster DB. Se non è necessario che il cluster database si trovi in una sottorete pubblica, impostare Publicly Accessible (Accessibile pubblicamente) su No.

- Availability Zone (Zona di disponibilità): seleziona la zona di disponibilità per ospitare l'istanza primaria per il cluster di database Aurora MySQL. Per permettere a Aurora di selezionare automaticamente una zona di disponibilità, seleziona No Preference (Nessuna preferenza).

- Database Port (Porta database): digita la porta predefinita da usare quando si effettua la connessione a istanze nel cluster di database Aurora MySQL. Il valore di default è `3306`.

  > **Note**
  >
  > Potrebbe essere attivo un firewall aziendale che non permette l'accesso alle porte predefinite come la porta predefinita di MySQL 3306. In questo caso, fornire un valore di porta permesso dal firewall aziendale. Ricorda il valore di porta per utilizzarlo successivamente, al momento della connessione al cluster di database Aurora MySQL.

- Encryption (Crittografia): seleziona Enable Encryption (Abilita crittografia) perché il nuovo cluster di database Aurora MySQL venga crittografato mentre è inattivo. Se scegli Enable Encryption (Abilita crittografia), devi scegliere una chiave KMS come valore AWS KMS key.

Se la snapshot DB non è crittografata, specifica una chiave di crittografia per fare in modo che il cluster database sia crittografato mentre è inattivo.

Se la snapshot DB è crittografata, specifica una chiave di crittografia per fare in modo che il cluster database sia crittografato mentre è inattivo utilizzando la chiave di crittografia specificata. Puoi specificare la chiave di crittografia utilizzata dalla snapshot DB o una chiave diversa. Non puoi creare un cluster database non crittografato da una snapshot DB crittografata.

- Aggiornamento automatico della versione secondaria: questa impostazione non si applica ai cluster di database Aurora MySQL.

  Per ulteriori informazioni sugli aggiornamenti del motore per Aurora MySQL, consultare Aggiornamenti del motore del database per Amazon Aurora MySQL.

4. Selezionare Migrate (Migra) per effettuare la migrazione della snapshot DB.
5. Selezionare Instances (Istanze) e quindi l'icona con la freccia per visualizzare i dettagli del cluster DB e monitorare l'avanzamento della migrazione. Nella pagina dei dettagli, puoi trovare l'endpoint del cluster utilizzato per la connessione all'istanza principale del cluster DB. Per ulteriori informazioni sulla connessione a un cluster di database Aurora MySQL, consulta Connessione a un cluster database Amazon Aurora.

AWS CLI

Puoi creare un cluster database Aurora da uno snapshot DB di una istanza database RDS for MySQL utilizzando il comando `restore-db-cluster-from-snapshot` con i seguenti parametri:

- `--db-cluster-identifier`: nome del cluster di database da creare.
- `--engine aurora-mysql`: per un cluster di database compatibile con MySQL 5.7 o 8.0.
- `--kms-key-id`: la AWS KMS key con la quale crittografare opzionalmente il cluster di database, in base al fatto se lo snapshot di database è stato crittografato o meno.
  - Se la snapshot DB non è crittografata, specifica una chiave di crittografia per fare in modo che il cluster database sia crittografato mentre è inattivo. Altrimenti, il cluster database non è crittografato.
  - Se la snapshot DB è crittografata, specifica una chiave di crittografia per fare in modo che il cluster database sia crittografato mentre è inattivo utilizzando la chiave di crittografia specificata. Altrimenti, il cluster database è crittografato mentre è inattivo utilizzando la chiave di crittografia per la snapshot DB.

> **Note**
>
> Non puoi creare un cluster database non crittografato da una snapshot DB crittografata.

- `--snapshot-identifier`: l'Amazon Resource Name (ARN) dello snapshot DB da migrare. Per ulteriori informazioni sugli ARN di Amazon RDS, consulta Amazon Relational Database Service (Amazon RDS).

Quando migri la snapshot DB utilizzando il comando `RestoreDBClusterFromSnapshot`, il comando crea sia il cluster database sia l'istanza primaria.

In questo esempio, crei un cluster di database compatibile con MySQL 5.7–, denominato *`mydbcluster`* da una snapshot di database con un nome ARN impostato su *`mydbsnapshotARN`*.

Per Linux, macOS o Unix:

```
aws rds restore-db-cluster-from-snapshot \
    --db-cluster-identifier mydbcluster \
    --snapshot-identifier mydbsnapshotARN \
    --engine aurora-mysql
```

Per Windows:

```
aws rds restore-db-cluster-from-snapshot ^
    --db-cluster-identifier mydbcluster ^
    --snapshot-identifier mydbsnapshotARN ^
    --engine aurora-mysql
```

In questo esempio, crei un cluster di database compatibile con MySQL 5.7–, denominato *`mydbcluster`* da una snapshot di database con un nome ARN impostato su *`mydbsnapshotARN`*.

Per Linux, macOS o Unix:

```
aws rds restore-db-cluster-from-snapshot \
    --db-cluster-identifier mydbcluster \
    --snapshot-identifier mydbsnapshotARN \
    --engine aurora-mysql
```

Per Windows:

```
aws rds restore-db-cluster-from-snapshot ^
    --db-cluster-identifier mydbcluster ^
    --snapshot-identifier mydbsnapshotARN ^
    --engine aurora-mysql
```

Migrazione di dati da un'istanza database RDS per MySQL a un cluster database Amazon Aurora MySQL utilizzando una replica di lettura Aurora

Aurora utilizza la funzionalità di replica dei log binari del motore di database MySQL per creare un tipo speciale di database cluster chiamato replica di lettura Aurora per un'istanza database RDS per MySQL di origine. Gli aggiornamenti effettuati sull'istanza database RDS per MySQL di origine vengono replicati in modo asincrono nella replica di lettura Aurora.

Consigliamo di utilizzare questa funzionalità per eseguire la migrazione da un'istanza database RDS per MySQL a un cluster database Aurora MySQL creando una replica di lettura Aurora dell'istanza database MySQL di origine. Quando il ritardo di replica tra un'istanza database RDS per MySQL e la replica di lettura Aurora è 0, puoi indirizzare le applicazioni client alla replica di lettura Aurora e poi arrestare la replica per rendere la replica di lettura Aurora un cluster database Aurora MySQL standalone. La migrazione potrebbe richiedere del tempo, circa diverse ore per tebibyte (TiB) di dati.

Per l'elenco delle regioni in cui è disponibile Aurora, consulta Amazon Aurora in Riferimenti generali di AWS.

Quando crei una replica di lettura Aurora di un'istanza database RDS per MySQL, Amazon RDS crea uno snapshot database dell'istanza database RDS per MySQL di origine (privata per Amazon RDS e senza addebiti). Amazon RDS in seguito migra i dati dallo snapshot DB alla replica di lettura Aurora. Dopo che i dati dello snapshot database sono stati migrati al nuovo cluster database Aurora MySQL, Amazon RDS avvia la replica tra l'istanza database RDS per MySQL e il cluster database Aurora MySQL. Se l'istanza database RDS per MySQL contiene tabelle che utilizzano motori di archiviazione diversi da InnoDB o che utilizzano il formato di riga compresso, puoi velocizzare il processo di creazione di una replica di lettura Aurora modificando le tabelle in modo che vengano utilizzati il motore di archiviazione InnoDB e il formato di riga dinamico prima di creare la replica di lettura Aurora. Per ulteriori informazioni sul processo di copia di una snapshot di database MySQL in un cluster di database Aurora MySQL, consulta Migrazione dei dati da un'istanza database RDS per MySQL a un cluster di database Amazon Aurora MySQL.

Puoi avere una sola replica di lettura Aurora per un'istanza database RDS per MySQL.

> **Note**
>
> Possono verificarsi problemi di replica a causa delle differenze di funzionalità tra Aurora MySQL e la versione del motore di database MySQL dell'istanza database RDS per MySQL, ossia l'istanza di replica primaria. Se si verifica un errore, puoi trovare supporto nel forum della community Amazon RDS oppure contattando Supporto AWS.

Non puoi creare una replica di lettura Aurora se l'istanza database RDS per MySQL è già l'origine di una replica di lettura tra regioni.
Non è possibile migrare ad Aurora MySQL 3.05 e versioni successive da alcune versioni precedenti di RDS per MySQL 8.0, tra cui 8.0.11, 8.0.13 e 8.0.15. Si consiglia di eseguire l'aggiornamento a RDS per MySQL versione 8.0.28 prima della migrazione.

Per ulteriori informazioni sulle repliche di lettura MySQL, consulta la sezione relativa all'uso di repliche di lettura con istanze database di MariaDB, MySQL e PostgreSQL.

Creazione di una replica di lettura Aurora

Puoi creare una replica di lettura Aurora per un'istanza database RDS per MySQL utilizzando la console, la AWS CLI o l'API RDS.

Console

Creazione di una replica di lettura Aurora da un'istanza database RDS per MySQL di origine

1. Accedi alla Console di gestione AWS e apri la console Amazon RDS all'indirizzo https://console.aws.amazon.com/rds/.
2. Nel riquadro di navigazione, scegliere Databases (Database).
3. Scegliere l'istanza database MySQL da usare come origine per la replica di lettura Aurora.
4. Per Actions (Operazioni), scegliere Create Aurora read replica (Crea replica di lettura Aurora).
5. Scegliere le specifiche del cluster di database da utilizzare per la replica di lettura Aurora, come descritto nella tabella seguente.

Opzione	Descrizione
DB instance class (Classe istanza database)	Scegliere una classe di istanza database che definisca i requisiti di elaborazione e di memoria per l'istanza primaria nel cluster DB. Per ulteriori informazioni sulle opzioni di classe di istanza database, consulta Classi di istanze database Amazon Aurora.
Multi-AZ deployment (Implementazione Multi-AZ)	Seleziona Crea replica in una zona diversa per creare una replica di standby del nuovo cluster di database in un'altra zona di disponibilità nella regione AWS

Opzione	Descrizione
	di destinazione per il supporto per failover. Per altre informazioni sulle zone di disponibilità multiple, consulta Regioni e zone di disponibilità.
DB instance identifier (Identificatore istanze DB)	Digitare un nome per l'istanza primaria nel cluster di database della replica di lettura Aurora. Questo identificatore viene utilizzato nell'indirizzo dell'endpoint per l'istanza primaria del nuovo cluster di database. L'identificatore istanze database presenta i seguenti vincoli: • Deve contenere da 1 a 63 caratteri alfanumerici o trattini. • Il primo carattere deve essere una lettera. • Non può terminare con un trattino o contenere due trattini consecutivi. • Deve essere univoco per tutte le istanze database, per ogni account AWS e in ogni regione AWS Poiché il cluster di database della replica di lettura Aurora viene creato da uno snapshot dell'istanza database di origine, il nome utente master e la password master per la replica di lettura Aurora corrispondono al nome utente master e alla password master per l'istanza database di origine.
Virtual Private Cloud (VPC)	Selezionare il VPC che ospita il cluster di database. Selezionare Create new VPC (Crea nuovo VPC) per fare in modo che Aurora crei automaticamente un VPC. Per ulteriori informazioni, consulta Prerequisiti per i cluster di database.

Opzione	Descrizione
Gruppo di sottoreti DB	Scegliere il gruppo di sottoreti di database da utilizzare per il cluster di database. Selezionare Create new DB subnet group (Crea nuovo gruppo di sottoreti DB) per fare in modo che Aurora crei automaticamente un gruppo di sottoreti di database. Per ulteriori informazioni, consulta Prerequisiti per i cluster di database.
Public accessibility (Accesso pubblico)	Selezionare `Yes` (Sì) per assegnare al cluster di database un indirizzo IP pubblico. Altrimenti, selezionare `No`. Le istanze nel cluster di database possono essere una combinazione di istanze database pubbliche e private. Per ulteriori informazioni su come non consentire l'accesso pubblico per le istanze, consulta Nascondere cluster database in un VPC da Internet.
Availability zone (Zona di disponibilità)	Stabilire se si desidera specificare una zona di disponibilità particolare. Per ulteriori informazioni sulle zone di disponibilità, consultare Regioni e zone di disponibilità.
VPC security group (firewall) (Gruppo di sicurezza VPC (firewall))	Selezionare Create a new VPC security group (Crea nuovo gruppo di sicurezza VPC) per fare in modo che Aurora crei automaticamente un gruppo di sicurezza VPC. Selezionare Select existing VPC security groups (Seleziona gruppi di sicurezza VPC esistenti) per specificare uno o più gruppi di sicurezza VPC per proteggere l'accesso di rete al cluster di database. Per ulteriori informazioni, consulta Prerequisiti per i cluster di database.

Opzione	Descrizione
Database port (Porta del database)	Specifica la porta per applicazioni e utilità da utilizzare per accedere al database. I cluster di database Aurora MySQL hanno come porta predefinita MySQL, 3306. I firewall presso alcune aziende bloccano le connessio ni a questa porta. Se il firewall dell'azienda blocca la porta predefinita, scegliere un'altra porta per il nuovo cluster di database.
DB parameter group (Gruppo di parametri database)	Scegliere un gruppo parametri del database per il cluster di database Aurora MySQL. Puoi utilizzar e il gruppo parametri del database predefinito di Aurora oppure puoi creare il tuo gruppo parametri del database personalizzato. Per ulteriori informazioni sui gruppi di parametri database, consulta Gruppi di parametri per Amazon Aurora.
DB cluster parameter group (Gruppo di parametri del cluster database)	Scegliere un gruppo parametri del database per il cluster di database Aurora MySQL. Puoi utilizzare il gruppo di parametri del cluster di database predefinito di Aurora oppure puoi creare il tuo gruppo di parametri . Per ulteriori informazioni sui gruppi di parametri del cluster di database, consulta Gruppi di parametri per Amazon Aurora.

Opzione	Descrizione
Encryption (Crittografia)	Selezionare Disable encryption (Disabilita crittografia) se non desideri che il nuovo cluster di database sia crittografato. Selezionare Enable Encryption (Abilita crittografia) affinché il nuovo cluster di database Aurora sia crittografato mentre è inattivo. Se scegli Enable encryption (Abilita crittografia), devi scegliere una chiave KMS come valore AWS KMS key. Se l'istanza database MySQL non è crittografata, specifica una chiave di crittografia per fare in modo che il cluster di database sia crittografato mentre è inattivo. Se l'istanza database MySQL è crittografata, specifica una chiave di crittografia per fare in modo che il cluster di database sia crittografato mentre è inattivo utilizzan do la chiave di crittografia specificata. Puoi specificare la chiave di crittografia utilizzata dall'istanza database MySQL o una chiave diversa. Non puoi creare un cluster di database non crittografato da un'istanza database MySQL crittografata.
Priorità	Scegliere una priorità di failover per il cluster di database. Se non si specifica alcun valore, l'imposta zione predefinita è tier-1 (livello 1). Questa priorità determina l'ordine di promozione delle repliche di Aurora durante il recupero da un errore dell'istanza principale. Per ulteriori informazioni, consulta Tolleranz a ai guasti di un cluster DB Aurora.
Backup retention period (Periodo di retention dei backup)	Selezionare l'intervallo di tempo, da 1 a 35 giorni, per il quale Aurora conserva le copie di backup del database. Le copie di backup possono essere utilizzat e per operazioni di ripristino point-in-time (PITR) del database al secondo.

Opzione	Descrizione
Monitoraggio avanzato	Scegliere Enable enhanced monitoring (Abilita monitoraggio avanzato) per abilitare la raccolta di parametri in tempo reale per il sistema operativo su cui viene eseguito il cluster DB. Per ulteriori informazioni, consulta Monitoraggio dei parametri del sistema operativo con il monitoraggio avanzato.
Monitoring Role (Ruolo monitoraggio)	Disponibile solo se Enhanced Monitoring (Monitoraggio avanzato) è impostato su Enable enhanced monitoring (Abilita monitoraggio avanzato). Selezionare il ruolo IAM creato per consentire ad Aurora di comunicare con Amazon CloudWatch Logs oppure scegliere Default (Predefinito) per fare in modo che Aurora crei automaticamente un ruolo denominato `rds-monitoring-role` . Per ulteriori informazioni, consulta Monitoraggio dei parametri del sistema operativo con il monitoraggio avanzato.
Granularity (Granularità)	Disponibile solo se Enhanced Monitoring (Monitoraggio avanzato) è impostato su Enable enhanced monitoring (Abilita monitoraggio avanzato). Impostare l'intervallo, in secondi, tra le operazioni di raccolta dei parametri per il cluster DB.
Auto minor version upgrade (Aggiornamento automatico della versione secondaria)	Queste impostazioni non si applicano ai cluster di database Aurora MySQL. Per ulteriori informazioni sugli aggiornamenti del motore per Aurora MySQL, consultare Aggiornamenti del motore del database per Amazon Aurora MySQL.

Opzione	Descrizione
Maintenance window (Finestra di manutenzione)	Selezionare Select window (Seleziona finestra) e specifica l'intervallo temporale settimanale durante il quale può avvenire la manutenzione del sistema. Oppure, seleziona No preference (Nessuna preferenz a) per consentire ad Aurora di assegnare un periodo casuale.

6. Scegliere Create read replica (Crea replica di lettura).

AWS CLI

Per creare una replica di lettura Aurora da un'istanza database RDS per MySQL di origine, utilizza i comandi AWS CLI `create-db-cluster` e `create-db-instance` per creare un nuovo cluster database Aurora MySQL. Quando chiami il comando `create-db-cluster`, includi il parametro `--replication-source-identifier` per identificare l'ARN (Amazon Resource Name) per l'istanza database MySQL di origine. Per ulteriori informazioni sugli ARN di Amazon RDS, consulta Amazon Relational Database Service (Amazon RDS).

Non specificare il nome utente master, la password master o il nome del database, in quanto la replica di lettura Aurora utilizza lo stesso nome utente master, la password master e il nome del database dell'istanza database MySQL di origine.

Per Linux, macOS o Unix:

```
aws rds create-db-cluster --db-cluster-identifier sample-replica-cluster --engine
 aurora \
    --db-subnet-group-name mysubnetgroup --vpc-security-group-ids sg-c7e5b0d2 \
    --replication-source-identifier arn:aws:rds:us-west-2:123456789012:db:primary-
mysql-instance
```

Per Windows:

```
aws rds create-db-cluster --db-cluster-identifier sample-replica-cluster --engine
 aurora ^
    --db-subnet-group-name mysubnetgroup --vpc-security-group-ids sg-c7e5b0d2 ^
    --replication-source-identifier arn:aws:rds:us-west-2:123456789012:db:primary-
mysql-instance
```

Se utilizzi la console per creare una replica di lettura Aurora, Aurora crea automaticamente l'istanza primaria per la replica di lettura Aurora del cluster di database. Se utilizzi la AWS CLI per creare una replica di lettura Aurora, è necessario creare in modo esplicito l'istanza primaria per il cluster di database. L'istanza primaria è la prima istanza creata in un cluster di database.

Puoi creare un'istanza primaria per il cluster di database utilizzando il comando `create-db-instance` dell'AWS CLI con i seguenti parametri.

- `--db-cluster-identifier`

  Nome del cluster DB.
- `--db-instance-class`

  Nome della classe dell'istanza database da utilizzare per l'istanza primaria.
- `--db-instance-identifier`

  Nome dell'istanza primaria.
- `--engine aurora`

In questo esempio, crei un'istanza primaria denominata *`myreadreplicainstance`* per il cluster di database denominato *`myreadreplicacluster`*, utilizzando la classe dell'istanza database specificata in *`myinstanceclass`*.

Example

Per Linux, macOS o Unix:

```
aws rds create-db-instance \
    --db-cluster-identifier myreadreplicacluster \
    --db-instance-class myinstanceclass \
    --db-instance-identifier myreadreplicainstance \
    --engine aurora
```

Per Windows:

```
aws rds create-db-instance ^
    --db-cluster-identifier myreadreplicacluster ^
    --db-instance-class myinstanceclass ^
    --db-instance-identifier myreadreplicainstance ^
    --engine aurora
```

API RDS

Per creare una replica di lettura Aurora da un'istanza database RDS per MySQL di origine, utilizza i comandi API `CreateDBCluster` e `CreateDBInstance` di Amazon RDS per creare una nuova istanza primaria e un nuovo cluster database Aurora. Non specificare il nome utente master, la password master o il nome del database, in quanto la replica di lettura Aurora utilizza lo stesso nome utente master, la password master e il nome del database dell'istanza database RDS per MySQL di origine.

Puoi creare un nuovo cluster database Aurora per una replica di lettura Aurora da un'istanza database RDS per MySQL di origine utilizzando il comando `CreateDBCluster` dell'API Amazon RDS con i seguenti parametri:

- `DBClusterIdentifier`

  Nome del cluster di database da creare.
- `DBSubnetGroupName`

  Nome del gruppo di sottoreti database da associare al cluster DB.
- `Engine=aurora`
- `KmsKeyId`

  La AWS KMS key con la quale crittografare opzionalmente il cluster DB, in base al fatto se l'istanza database MySQL è stata crittografata o meno.
  - Se l'istanza database MySQL non è crittografata, specifica una chiave di crittografia per fare in modo che il cluster di database sia crittografato mentre è inattivo. Altrimenti, il cluster di database è crittografato mentre è inattivo utilizzando la chiave di crittografia predefinita per l'account.
  - Se l'istanza database MySQL è crittografata, specifica una chiave di crittografia per fare in modo che il cluster di database sia crittografato mentre è inattivo utilizzando la chiave di crittografia specificata. Altrimenti, il cluster di database è crittografato mentre è inattivo utilizzando la chiave di crittografia per l'istanza database MySQL.

  > **Note**
  >
  > Non puoi creare un cluster di database non crittografato da un'istanza database MySQL crittografata.
- `ReplicationSourceIdentifier`

L'Amazon Resource Name (ARN) per l'istanza database MySQL origine. Per ulteriori informazioni sugli ARN di Amazon RDS, consulta Amazon Relational Database Service (Amazon RDS).

- `VpcSecurityGroupIds`

  Elenco dei gruppi di sicurezza VPC EC2 da associare a questo cluster di database.

In questo esempio, crei un cluster di database denominato *`myreadreplicacluster`* da un'istanza database MySQL con un nome ARN impostato su *`mysqlprimaryARN`*, associato a un gruppo di sottoreti database denominato *`mysubnetgroup`* e a un gruppo di sicurezza VPC denominato *`mysecuritygroup`*.

Example

```
https://rds.us-east-1.amazonaws.com/
    ?Action=CreateDBCluster
    &DBClusterIdentifier=myreadreplicacluster
    &DBSubnetGroupName=mysubnetgroup
    &Engine=aurora
    &ReplicationSourceIdentifier=mysqlprimaryARN
    &SignatureMethod=HmacSHA256
    &SignatureVersion=4
    &Version=2014-10-31
    &VpcSecurityGroupIds=mysecuritygroup
    &X-Amz-Algorithm=AWS4-HMAC-SHA256
    &X-Amz-Credential=AKIADQKE4SARGYLE/20150927/us-east-1/rds/aws4_request
    &X-Amz-Date=20150927T164851Z
    &X-Amz-SignedHeaders=content-type;host;user-agent;x-amz-content-sha256;x-amz-date
    &X-Amz-Signature=6a8f4bd6a98f649c75ea04a6b3929ecc75ac09739588391cd7250f5280e716db
```

Se utilizzi la console per creare una replica di lettura Aurora, Aurora crea automaticamente l'istanza primaria per la replica di lettura Aurora del cluster di database. Se utilizzi la AWS CLI per creare una replica di lettura Aurora, è necessario creare in modo esplicito l'istanza primaria per il cluster di database. L'istanza primaria è la prima istanza creata in un cluster di database.

Puoi creare un'istanza primaria per il cluster di database utilizzando il comando `CreateDBInstance` dell'API Amazon RDS con i seguenti parametri:

- `DBClusterIdentifier`

  Nome del cluster DB.

- `DBInstanceClass`

  Nome della classe dell'istanza database da utilizzare per l'istanza primaria.

- `DBInstanceIdentifier`

  Nome dell'istanza primaria.

- `Engine=aurora`

In questo esempio, crei un'istanza primaria denominata *`myreadreplicainstance`* per il cluster di database denominato *`myreadreplicacluster`*, utilizzando la classe dell'istanza database specificata in *`myinstanceclass`*.

Example

```
https://rds.us-east-1.amazonaws.com/
    ?Action=CreateDBInstance
    &DBClusterIdentifier=myreadreplicacluster
    &DBInstanceClass=myinstanceclass
    &DBInstanceIdentifier=myreadreplicainstance
    &Engine=aurora
    &SignatureMethod=HmacSHA256
    &SignatureVersion=4
    &Version=2014-09-01
    &X-Amz-Algorithm=AWS4-HMAC-SHA256
    &X-Amz-Credential=AKIADQKE4SARGYLE/20140424/us-east-1/rds/aws4_request
    &X-Amz-Date=20140424T194844Z
    &X-Amz-SignedHeaders=content-type;host;user-agent;x-amz-content-sha256;x-amz-date
    &X-Amz-Signature=bee4aabc750bf7dad0cd9e22b952bd6089d91e2a16592c2293e532eeaab8bc77
```

Visualizzazione di una replica di lettura Aurora

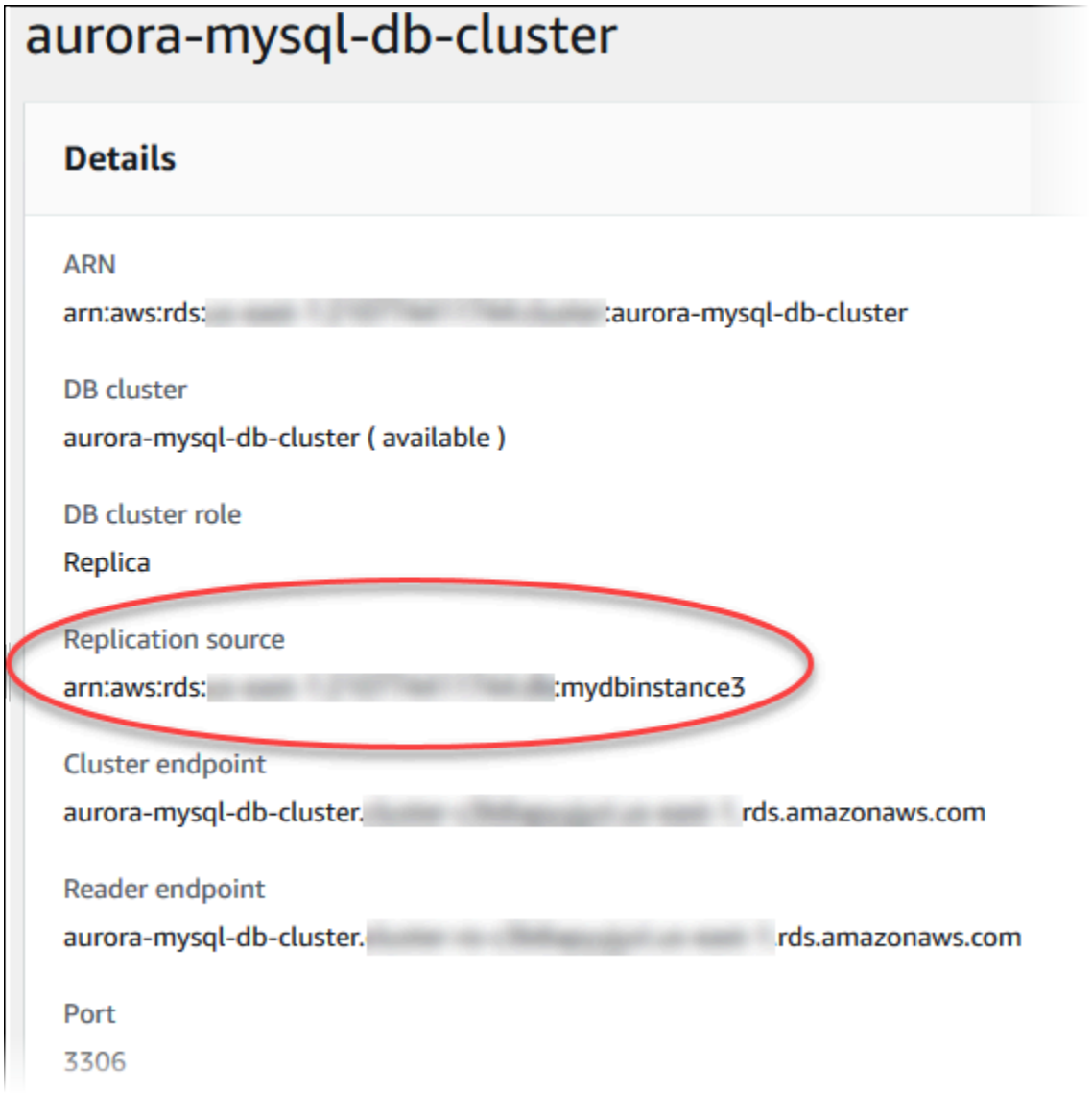
Puoi visualizzare le relazioni di replica da MySQL a Aurora MySQL per i cluster di database Aurora MySQL utilizzando la Console di gestione AWS o la AWS CLI.

Console

Per visualizzare l'istanza database MySQL primaria per una replica di lettura Aurora

1. Accedi alla Console di gestione AWS e apri la console Amazon RDS all'indirizzo https://console.aws.amazon.com/rds/.

2. Nel riquadro di navigazione, scegliere Databases (Database).
3. Selezionare il cluster di database affinché la replica di lettura Aurora mostri i propri dettagli. Le informazioni sull'istanza database MySQL primaria si trovano nel campo Replication source (Origine replica).



AWS CLI

Per visualizzare le relazioni di replica tra MySQL e Aurora MySQL per i cluster di database Aurora MySQL tramite la AWS CLI, utilizza i comandi `describe-db-clusters` e `describe-db-instances`.

Per determinare quale istanza database MySQL è quella primaria, utilizzare `describe-db-clusters` e specificare l'identificatore cluster della replica di lettura Aurora per l'opzione `--db-`

`cluster-identifier`. Fai riferimento all'elemento `ReplicationSourceIdentifier` nell'output per l'ARN dell'istanza database che costituisce il principale di replica.

Per determinare quale cluster di database è la replica di lettura Aurora, utilizzare `describe-db-instances` e specificare l'identificatore istanze dell'istanza database MySQL per l'opzione `--db-instance-identifier`. Fai riferimento all'elemento `ReadReplicaDBClusterIdentifiers` nell'output per l'identificatore cluster di database della replica di lettura Aurora.

Example

Per Linux, macOS o Unix:

```
aws rds describe-db-clusters \
    --db-cluster-identifier myreadreplicacluster
```

```
aws rds describe-db-instances \
    --db-instance-identifier mysqlprimary
```

Per Windows:

```
aws rds describe-db-clusters ^
    --db-cluster-identifier myreadreplicacluster
```

```
aws rds describe-db-instances ^
    --db-instance-identifier mysqlprimary
```

Promozione di una replica di lettura Aurora

Al termine della migrazione, puoi promuovere la replica di lettura Aurora a un cluster di database autonomo utilizzando la Console di gestione AWS o la AWS CLI.

Quindi, puoi indirizzare le applicazioni client all'endpoint per la replica di lettura Aurora. Per ulteriori informazioni sugli endpoint Aurora, consulta Connessioni degli endpoint Amazon Aurora. La promozione dovrebbe completarsi abbastanza velocemente; nel corso della promozione, è possibile eseguire operazioni di lettura e scrittura sulla replica di lettura Aurora. Tuttavia, non è possibile eliminare l'istanza database MySQL master o scollegare l'istanza database e la replica di lettura Aurora durante questo periodo.

Prima di promuovere la replica di lettura Aurora, interrompi la scrittura di transazioni nell'istanza database MySQL di origine e aspetta che il ritardo di replica nella replica di lettura Aurora raggiunga

0. Puoi visualizzare il ritardo della replica per una replica di lettura Aurora chiamando il comando `SHOW SLAVE STATUS` (Aurora MySQL versione 2) o `SHOW REPLICA STATUS` (Aurora MySQL versione 3) sulla replica di lettura Aurora. Controlla il valore secondi di latenza dal principale.

Puoi iniziare a scrivere la replica di lettura Aurora dopo che le transazioni di scrittura al master sono iniziate e il ritardo della replica equivale a 0. Se scrivi la replica di lettura Aurora prima di ciò e modifichi le tabelle la cui modifica è in corso anche nel master MySQL, rischi di interrompere la replica su Aurora. Se ciò avviene, devi eliminare la replica di lettura Aurora e crearla di nuovo.

Console

Come promuovere una replica di lettura Aurora a cluster di database Aurora

1. Accedi alla Console di gestione AWS e apri la console Amazon RDS all'indirizzo https://console.aws.amazon.com/rds/.
2. Nel riquadro di navigazione, scegliere Databases (Database).
3. Seleziona il cluster di database per la replica di lettura Aurora.
4. In Actions (Operazioni), selezionare Promote (Promuovi).
5. Selezionare Promote read replica (Promuovi replica di lettura).

Dopo la promozione, verificare che la promozione sia stata completata utilizzando la procedura seguente.

Per confermare che la replica di lettura Aurora è stata promossa

1. Accedi alla Console di gestione AWS e apri la console Amazon RDS all'indirizzo https://console.aws.amazon.com/rds/.
2. Nel riquadro di navigazione selezionare Events (Eventi).
3. Nella pagina Events (Eventi), verificare che esista un evento `Promoted Read Replica cluster to a stand-alone database cluster` per il cluster promosso.

Al completamento della promozione, l'istanza database MySQL master e la replica di lettura Aurora non sono collegate e puoi eliminare in modo sicuro l'istanza database, se lo desideri.

AWS CLI

Per promuovere una replica di lettura Aurora a cluster di database autonomo, utilizza il comando `promote-read-replica-db-cluster` della AWS CLI.

Example

Per Linux, macOS o Unix:

```
aws rds promote-read-replica-db-cluster \
    --db-cluster-identifier myreadreplicacluster
```

Per Windows:

```
aws rds promote-read-replica-db-cluster ^
    --db-cluster-identifier myreadreplicacluster
```

Gestione di Amazon Aurora MySQL

Nelle sezioni seguenti viene descritta la gestione di un cluster database Amazon Aurora MySQL.

Argomenti

- Gestione delle prestazioni e del dimensionamento per Amazon Aurora MySQL
- Backtrack di un cluster database Aurora
- Test di Amazon Aurora MySQL mediante query Fault Injection
- Alterazione delle tabelle in Amazon Aurora mediante Fast DDL
- Visualizzazione dello stato del volume per un cluster DB Aurora MySQL

Gestione delle prestazioni e del dimensionamento per Amazon Aurora MySQL

Dimensionamento delle istanze database Aurora MySQL

È possibile dimensionare le istanze database Aurora MySQL in due diversi modi, ovvero tramite il dimensionamento delle istanze e il dimensionamento della lettura. Per ulteriori informazioni sul dimensionamento della lettura, consulta Dimensionamento della lettura.

Puoi dimensionare il cluster database Aurora MySQL in base alle necessità, modificando la classe di istanza database per ogni istanza database nel cluster database. Aurora MySQL supporta diverse classi di istanza database ottimizzate per Aurora. Non utilizzare classi di istanza db.t2 o db.t3 per cluster Aurora più grandi di dimensioni maggiori di 40 TB. Per le specifiche delle classi di istanza database supportate da Aurora MySQL, consulta Classi di istanze database Amazon Aurora.

> **Note**
>
> Consigliamo di utilizzare le classi di istanza database T solo per i server di sviluppo e test o altri server non di produzione. Per ulteriori informazioni sulle classi di istanza T, consulta Utilizzo delle classi di istanza T per lo sviluppo e i test.

Numero massimo di connessioni a un'istanza database Aurora MySQL

Il numero massimo di connessioni consentite a un'istanza database di Aurora MySQL è determinato dal parametro `max_connections` nel gruppo di parametri a livello di istanza per l'istanza database.

La tabella seguente elenca il valore predefinito risultante di `max_connections` per ciascuna classe di istanza database disponibile per Aurora MySQL. È possibile aumentare il numero massimo di connessioni all'istanza database Aurora MySQL eseguendo il dimensionamento dell'istanza a una classe di istanza database con più memoria oppure impostando un valore maggiore per il parametro `max_connections` nel gruppo di parametri DB per l'istanza, fino a 16.000.

> **Tip**
>
> Se le applicazioni aprono e chiudono frequentemente connessioni o mantengono un numero elevato di connessioni di lunga durata aperte, ti consigliamo di utilizzare Amazon RDS Proxy. Proxy RDS è un proxy di database completamente gestito e ad alta disponibilità che utilizza il pooling di connessioni per condividere connessioni di database in modo sicuro ed efficiente. Per ulteriori informazioni sul Proxy RDS, consulta Server proxy per Amazon RDS per Aurora.

Per informazioni dettagliate su come le istanze Aurora Serverless v2 gestiscono questo parametro, vedi Numero massimo connessioni per Aurora Serverless v2.

Classe di istanza	Valore predefinito di max_connections		
db.t2.small	45		
db.t2.medium	90		
db.t3.small	45		
db.t3.medium	90		
db.t3.large	135		
db.t4g.medium	90		
db.t4g.large	135		
db.r3.large	1000		

Classe di istanza	Valore predefinito di max_connections		
db.r3.xlarge	2000		
db.r3.2xlarge	3000		
db.r3.4xlarge	4000		
db.r3.8xlarge	5000		
db.r4.large	1000		
db.r4.xlarge	2000		
db.r4.2xlarge	3000		
db.r4.4xlarge	4000		
db.r4.8xlarge	5000		
db.r4.16xlarge	6000		
db.r5.large	1000		
db.r5.xlarge	2000		
db.r5.2xlarge	3000		
db.r5.4xlarge	4000		
db.r5.8xlarge	5000		
db.r5.12xlarge	6000		
db.r5.16xlarge	6000		
db.r5.24xlarge	7000		

Classe di istanza	Valore predefinito di max_connections		
db.r6g.large	1000		
db.r6g.xlarge	2000		
db.r6g.2xlarge	3000		
db.r6g.4xlarge	4000		
db.r6g.8xlarge	5000		
db.r6g.12xlarge	6000		
db.r6g.16xlarge	6000		
db.r6i.large	1000		
db.r6i.xlarge	2000		
db.r6i.2xlarge	3000		
db.r6i.4xlarge	4000		
db.r6i.8xlarge	5000		
db.r6i.12xlarge	6000		
db.r6i.16xlarge	6000		
db.r6i.24xlarge	7000		
db.r6i.32xlarge	7000		
db.r7g.large	1000		
db.r7g.xlarge	2000		

Classe di istanza	Valore predefinito di max_connections		
db.r7g.2xlarge	3000		
db.r7g.4xlarge	4000		
db.r7g.8xlarge	5000		
db.r7g.12xlarge	6000		
db.r7g.16xlarge	6000		
db.r7i.large	1000		
db.r7i.xlarge	2000		
db.r7i.2xlarge	3000		
db.r7i.4xlarge	4000		
db.r7i.8xlarge	5000		
db.r7i.12xlarge	6000		
db.r7i.16xlarge	6000		
db.r7i.24xlarge	7000		
db.r7i.48xlarge	8000		
db.r8g.large	1000		
db.r8g.xlarge	2000		
db.r8g.2xlarge	3000		
db.r8g.4xlarge	4000		

Classe di istanza	Valore predefinito di max_connections
db.r8g.8xlarge	5000
db.r8g.12xlarge	6000
db.r8g.16xlarge	6000
db.r8g.24xlarge	7000
db.r8g.48xlarge	8000
db.x2g.large	2000
db.x2g.xlarge	3000
db.x2g.2xlarge	4000
db.x2g.4xlarge	5000
db.x2g.8xlarge	6000
db.x2g.12xlarge	7000
db.x2g.16xlarge	7000

Tip

Il calcolo del parametro `max_connections` utilizza il logaritmo in base 2 (diverso dal logaritmo naturale) e il valore `DBInstanceClassMemory` in byte per la classe di istanza Aurora MySQL selezionata. Il parametro accetta solo valori interi, con parti decimali troncate dai calcoli. La formula implementa i limiti di connessione come segue:

- Incremento di connessione 1.000 per istanze R3, R4 e R5 di dimensioni superiori
- Incremento di connessione 45 per le varianti di memoria delle istanze T2 e T3

Esempio: per db.r6g.large, mentre la formula calcola 1.069,2, il sistema implementa 1.000 per mantenere modelli incrementali coerenti.

Se crei un nuovo gruppo di parametri per personalizzare il valore predefinito del limite di connessioni, verificherai che il limite di connessioni predefinito deriva da una formula basata sul valore `DBInstanceClassMemory`. Come illustrato nella precedente tabella, la formula produce limiti di connessione che aumentano di 1000 al raddoppio della memoria tra istanze R3, R4 e R5 sempre più grandi e di 45 per dimensioni di memoria diverse delle istanze T2 e T3.

Consultare Specifica dei parametri del database per ulteriori informazioni su come è calcolato `DBInstanceClassMemory`.

Le istanze database Aurora MySQL e RDS for MySQL hanno quantità di sovraccarico della memoria differenti. Pertanto, il valore `max_connections` può essere diverso per istanze database Aurora MySQL e RDS MySQL che utilizzano la stessa classe di istanza. I valori nella tabella si applicano solo alle istanze database Aurora MySQL.

Note

I limiti di connettività sensibilmente inferiori per le istanze T2 e T3 sono dovute al fatto che con Aurora queste classi di istanze sono destinate solo a scenari di test e sviluppo e non ai carichi di lavoro di produzione.

I limiti di connessione predefiniti si adattano ai sistemi che utilizzano i valori predefiniti per altre funzioni ad alto consumo di memori, come il pool di buffer e la cache delle query. Se modifichi queste altre impostazioni per il cluster, pensa a regolare il limite di connessione in base alla possibilità di aumento o diminuzione della memoria disponibile nelle istanze database.

Limiti di storage temporaneo per Aurora MySQL

Aurora MySQL memorizza tabelle e indici nel sottosistema di archiviazione Aurora. Aurora MySQL utilizza l’archiviazione locale o temporanea separata per i file temporanei non persistenti e le tabelle temporanee non InnoDB. Inoltre, sono inclusi i file utilizzati per scopi quali l'ordinamento di set di dati di grandi dimensioni durante l'elaborazione delle query o per le operazioni di creazione dell'indice. Non include le tabelle temporanee InnoDB.

Per ulteriori informazioni sulle tabelle temporanee in Aurora MySQL versione 3, consulta Nuovo comportamento della tabella temporanea in Aurora MySQL versione 3. Per ulteriori informazioni sulle tabelle temporanee nella versione 2, consulta Comportamento del tablespace temporaneo in Aurora MySQL versione 2.

I dati e i file temporanei su questi volumi vengono persi all'avvio e all'arresto dell'istanza database e durante la sostituzione dell'host.

Questi volumi di archiviazione locale sono supportati da Amazon Elastic Block Store (Amazon EBS) e possono essere estesi utilizzando una classe di istanza database più grande. Per ulteriori informazioni sullo storage, consultare Archiviazione Amazon Aurora.

L'archiviazione locale viene utilizzata anche per importare dati da Amazon S3 utilizzando `LOAD DATA FROM S3` o `LOAD XML FROM S3` e per esportare dati in S3 utilizzando SELECT INTO OUTFILE S3. Per ulteriori informazioni sull'importazione e l'esportazione in S3, consulta quanto segue:

- Caricamento dei dati in un cluster DB Amazon Aurora MySQL da file di testo in un bucket Amazon S3
- Salvataggio dei dati da un cluster DB Amazon Aurora MySQL nei file di testo in un bucket Amazon S3

Aurora MySQL utilizza un'archiviazione permanente separata per i log degli errori, i log generali, i log delle query lente e i log di audit per la maggior parte delle classi di istanza database Aurora MySQL (esclusi i tipi di classi di istanza a prestazioni espandibili come db.t2, db.t3 e db.t4g). I dati su questo volume vengono mantenuti all'avvio e all'arresto dell'istanza database e durante la sostituzione dell'host.

Questo volume di archiviazione permanente è supportato anche da Amazon EBS e ha dimensioni fisse in base alla classe di istanza database. Non può essere esteso utilizzando una classe di istanza database più grande.

La tabella seguente mostra la quantità massima di spazio di archiviazione temporanea e permanente disponibile per ogni classe di istanza database Aurora MySQL. Per ulteriori informazioni sul supporto della classe di istanza database per Aurora, consultare Classi di istanze database Amazon Aurora.

DB instance class (Classe istanza database)	Spazio di archiviazione locale/temporanea massimo disponibile (GiB)	Spazio di archiviazione massimo aggiuntivo disponibile per i file di log (GiB)
db.x2g.16xlarge	1280	500
db.x2g.12xlarge	960	500
db.x2g.8xlarge	640	500
db.x2g.4xlarge	320	500
db.x2g.2xlarge	160	60
db.x2g.xlarge	80	60
db.x2g.large	40	60
db.r8g.48xlarge	3840	500
db.r8g.24xlarge	1920	500
db.r8g.16xlarge	1280	500
db.r8g.12xlarge	960	500
db.r8g.8xlarge	640	500
db.r8g.4xlarge	320	500
db.r8g.2xlarge	160	60
db.r8g.xlarge	80	60
db.r8g.large	32	60
db.r7i.48xlarge	3840	500
db.r7i.24xlarge	1920	500
db.r7i.16xlarge	1280	500

DB instance class (Classe istanza database)	Spazio di archiviazione locale/temporanea massimo disponibile (GiB)	Spazio di archiviazione massimo aggiuntivo disponibile per i file di log (GiB)
db.r7i.12xlarge	960	500
db.r7i.8xlarge	640	500
db.r7i.4xlarge	320	500
db.r7i.2xlarge	160	60
db.r7i.xlarge	80	60
db.r7i.large	32	60
db.r7g.16xlarge	1280	500
db.r7g.12xlarge	960	500
db.r7g.8xlarge	640	500
db.r7g.4xlarge	320	500
db.r7g.2xlarge	160	60
db.r7g.xlarge	80	60
db.r7g.large	32	60
db.r6i.32xlarge	2560	500
db.r6i.24xlarge	1920	500
db.r6i.16xlarge	1280	500
db.r6i.12xlarge	960	500
db.r6i.8xlarge	640	500
db.r6i.4xlarge	320	500

DB instance class (Classe istanza database)	Spazio di archiviazione locale/temporanea massimo disponibile (GiB)	Spazio di archiviazione massimo aggiuntivo disponibile per i file di log (GiB)
db.r6i.2xlarge	160	60
db.r6i.xlarge	80	60
db.r6i.large	32	60
db.r6g.16xlarge	1280	500
db.r6g.12xlarge	960	500
db.r6g.8xlarge	640	500
db.r6g.4xlarge	320	500
db.r6g.2xlarge	160	60
db.r6g.xlarge	80	60
db.r6g.large	32	60
db.r5.24xlarge	1920	500
db.r5.16xlarge	1280	500
db.r5.12xlarge	960	500
db.r5.8xlarge	640	500
db.r5.4xlarge	320	500
db.r5.2xlarge	160	60
db.r5.xlarge	80	60
db.r5.large	32	60
db.r4.16xlarge	1280	500

DB instance class (Classe istanza database)	Spazio di archiviazione locale/temporanea massimo disponibile (GiB)	Spazio di archiviazione massimo aggiuntivo disponibile per i file di log (GiB)
db.r4.8xlarge	640	500
db.r4.4xlarge	320	500
db.r4.2xlarge	160	60
db.r4.xlarge	80	60
db.r4.large	32	60
db.t4g.large	32	–
db.t4g.medium	32	–
db.t3.large	32	–
db.t3.medium	32	–
db.t3.small	32	–
db.t2.medium	32	–
db.t2.small	32	–

Important

Questi valori rappresentano la quantità massima teorica di spazio di archiviazione libero su ogni istanza database. Lo spazio di archiviazione locale effettivo disponibile potrebbe essere inferiore. Aurora utilizza una parte dello spazio di archiviazione locale per i processi di gestione e l'istanza database utilizza una parte dello spazio di archiviazione locale anche prima che vengano caricati i dati. È possibile monitorare lo storage temporaneo disponibile per un'istanza database specifica con il parametro `FreeLocalStorage` CloudWatch, come descritto in CloudWatch Parametri Amazon per Amazon Aurora. Al momento è possibile verificare la quantità di spazio di archiviazione gratuito. È inoltre possibile tracciare la quantità di spazio di archiviazione gratuito nel tempo. Il monitoraggio dello spazio di archiviazione

gratuito nel tempo consente di determinare se il valore è in aumento o in diminuzione o di trovare i valori minimi, massimi o medi.
(Non valido per Aurora Serverless v2)

Backtrack di un cluster database Aurora

Con Amazon Aurora edizione compatibile con MySQL, puoi eseguire il backtrack di un cluster database a un orario specifico, senza ripristinare i dati da un backup.

Indice

- Panoramica del backtrack
  - Finestra di backtrack
  - Orario di backtrack
  - Limiti del backtrack
- Disponibilità di regioni e versioni
- Considerazioni sull'aggiornamento per i cluster abilitati per backtrack
- Configurazione del backtrack di un cluster di database Aurora MySQL
- Esecuzione di un backtrack per un cluster di database Aurora MySQL
- Monitoraggio del backtrack per un cluster di database Aurora MySQL
- Abbonamento a un evento di backtrack con la console
- Recupero di backtrack esistenti
- Disabilitazione del backtrack per un cluster di database Aurora MySQL

Panoramica del backtrack

Il backtrack "riavvolge" il cluster database all'orario che specifichi. Il backtrack non è un'operazione sostitutiva del backup del cluster database che consente di eseguire un ripristino point-in-time dello stesso. Fornisce tuttavia i seguenti vantaggi rispetto alle operazioni di backup e ripristino tradizionali:

- Puoi annullare facilmente eventuali errori. Se esegui accidentalmente un'operazione distruttiva, come DELETE senza una clausola WHERE, puoi eseguire il backtrack del cluster database a un orario antecedente all'operazione distruttiva con un'interruzione minima del servizio.
- Puoi eseguire il backtrack di un cluster database rapidamente. Il ripristino point-in-time di un cluster database avvia un nuovo cluster database e lo ripristina a partire dai dati di backup o da

uno snapshot e questa operazione può durare varie ore. Il backtrack di un cluster database non richiede un nuovo cluster database e "riavvolge" il cluster all'orario specificato in pochi minuti.

- Puoi esplorare le modifiche ai dati precedenti. Puoi eseguire ripetutamente il backtrack di un cluster database avanti e indietro nel tempo per determinare quando si è verificata una determinata modifica. Ad esempio, puoi eseguire il backtrack di un cluster database tre ore indietro e quindi un'ora avanti. In questo caso, l'orario di backtrack è due ore prima l'orario di origine.

> **Note**
>
> Per informazioni sul ripristino point-in-time di un cluster database, consulta Panoramica di backup e ripristino di un cluster di database Aurora.

Finestra di backtrack

Con il backtrack, si ha una finestra di backtrack target e una finestra di backtrack effettiva:

- La finestra di backtrack target è il periodo di tempo durante il quale desideri eseguire il backtrack del cluster di database. Quando abiliti il backtrack, specifichi una finestra di backtrack target. Ad esempio, potresti specificare una finestra di backtrack target di 24 ore se desideri eseguire un backtrack di un giorno per il cluster di database.
- La finestra di backtrack effettiva è il periodo di tempo effettivo durante il quale puoi eseguire il backtrack del cluster di database, che può essere inferiore alla finestra di backtrack target. La finestra di backtrack effettiva è basata sul carico di lavoro e sullo storage disponibile per archiviare le informazioni relative alle modifiche del database denominate record di modifica.

Quando aggiorni il cluster di database Aurora con il backtrack abilitato, generi record di modifica. Aurora conserva i record di modifica per la finestra di backtrack di destinazione e per l'archiviazione di questi dati si applica una tariffa oraria. La finestra di backtrack target e il carico di lavoro sul cluster di database determinano il numero di record di modifica che vengono archiviati. Il carico di lavoro è il numero di modifiche che apporti al cluster di database in un determinato periodo di tempo. Il numero di record di modifica archiviati nella finestra di backtrack quando il carico di lavoro è pesante è maggiore rispetto a quello con un carico di lavoro leggero.

Puoi considerare la finestra di backtrack target come l'obiettivo per il periodo di tempo massimo durante il quale desideri eseguire il backtrack del cluster di database. Nella maggior parte dei casi, puoi eseguire un backtrack corrispondente al periodo di tempo massimo specificato. Tuttavia, in

alcuni casi, il cluster di database non può archiviare un numero sufficiente di record di modifica per eseguire un backtrack corrispondente al periodo di tempo massimo e la finestra di backtrack effettiva è più piccola della finestra di backtrack target. In genere, la finestra di backtrack effettiva è più piccola di quella target quando il carico di lavoro è estremamente pesante sul cluster di database. Quando la finestra di backtrack effettiva è più piccola di quella target, ti inviamo una notifica.

Quando il backtrack è abilitato per un cluster di database ed elimini una tabella archiviata in quel cluster, Aurora conserva quella tabella nei record di modifica del backtrack. In questo modo, puoi ritornare a un orario antecedente a quello in cui hai eliminato la tabella. Se tuttavia non hai spazio sufficiente nella finestra di backtrack per archiviare la tabella, è possibile che questa venga rimossa dai record di modifica del backtrack.

Orario di backtrack

Aurora esegue sempre il backtrack a un orario coerente per il cluster di database. In questo modo, viene eliminata la possibilità di transazioni di cui non è stato eseguito il commit al completamento del backtrack. Quando specifichi un orario per un backtrack, Aurora sceglie automaticamente l'orario coerente più vicino possibile. Questo approccio significa che il backtrack completato potrebbe non corrispondere esattamente all'ora specificata, ma è possibile determinare l'ora esatta per un backtrack utilizzando il comando CLI describe-db-cluster-backtracks AWS . Per ulteriori informazioni, consulta Recupero di backtrack esistenti.

Limiti del backtrack

I seguenti limiti si applicano al backtrack:

- Il backtrack è disponibile solo per i cluster di database creati con la funzionalità di backtrack abilitata. Non puoi modificare un cluster di database per abilitare la funzionalità di backtrack. Puoi abilitare il backtrack quando crei un nuovo cluster di database o ripristini uno snapshot di un cluster di database.
- Il limite per una finestra di backtrack è 72 ore.
- Il backtrack influisce sull'intero cluster di database. Ad esempio, non puoi eseguire un backtrack selettivo di una singola tabella o di un singolo aggiornamento di dati.
- Non puoi creare repliche di lettura tra Regioni da un cluster abilitato al backtrack, ma puoi comunque abilitare la replica dei log binari (binlog) sul cluster. Se tenti di eseguire il backtrack di un cluster di database per il quale è abilitata la registrazione di log binari, in genere si verifica un errore a meno che tu non abbia scelto di forzare il backtrack. Qualsiasi tentativo di forzare un backtrack interromperà le repliche di lettura successive e interferirà con altre operazioni come le distribuzioni. blue/green

- Non puoi eseguire il backtrack di un clone di database a un orario antecedente a quello di creazione del clone. Puoi tuttavia utilizzare il database originale per eseguire il backtrack a un orario antecedente a quello di creazione del clone. Per ulteriori informazioni sulla creazione di cloni di database, consulta Clonazione di un volume per un cluster di database Amazon Aurora.
- Il backtrack comporta una breve interruzione dell'istanza database. Devi arrestare o interrompere le applicazioni prima di avviare un'operazione di backtrack per assicurarti che non vi siano nuove richieste di lettura o scrittura. Durante l'operazione di backtrack, Aurora interrompe il database, chiude tutte le connessioni aperte ed elimina le letture e scritture di cui non è stato eseguito il commit. Attende quindi il completamento dell'operazione di backtrack.
- Non è possibile ripristinare un'istantanea interregionale di un cluster abilitato al backtrack in una regione che non supporta il backtracking. AWS
- Se si esegue un aggiornamento locale di un cluster abilitato per il backtrack da Aurora MySQL versione 2 alla versione 3, non è possibile eseguire il backtrack a un punto temporale precedente all'esecuzione dell'aggiornamento.

Disponibilità di regioni e versioni

Backtrack non è disponibile per Aurora PostgreSQL.

Di seguito sono riportati i motori supportati e la disponibilità nelle regioni per backtrack con Aurora MySQL.

Region	Aurora MySQL versione 3	Aurora MySQL versione 2
Stati Uniti orientali (Virginia settentrionale)	Tutte le versioni	Tutte le versioni
Stati Uniti orientali (Ohio)	Tutte le versioni	Tutte le versioni
Stati Uniti occidentali (California settentrionale)	Tutte le versioni	Tutte le versioni

Region	Aurora MySQL versione 3	Aurora MySQL versione 2
Stati Uniti occidentali (Oregon)	Tutte le versioni	Tutte le versioni
Africa (Città del Capo)	–	–
Asia Pacifico (Hong Kong)	–	–
Asia Pacifico (Giacarta)	–	–
Asia Pacifico (Malesia)	–	–
Asia Pacifico (Melbourne)	–	–
Asia Pacifico (Mumbai)	Tutte le versioni	Tutte le versioni
Asia Pacifico (Nuova Zelanda)	–	–
Asia Pacifico (Osaka-Locale)	Tutte le versioni	Versione 2.07.3 e versioni successive
Asia Pacifico (Seul)	Tutte le versioni	Tutte le versioni
Asia Pacifico (Singapore)	Tutte le versioni	Tutte le versioni
Asia Pacifico (Sydney)	Tutte le versioni	Tutte le versioni

Region	Aurora MySQL versione 3	Aurora MySQL versione 2
Asia Pacifico (Taipei)	–	–
Asia Pacifico (Thailandia)	–	–
Asia Pacifico (Tokyo)	Tutte le versioni	Tutte le versioni
Canada (Centrale)	Tutte le versioni	Tutte le versioni
Canada occidentale (Calgary)	–	–
Cina (Pechino)	–	–
China (Ningxia)	–	–
Europa (Francoforte)	Tutte le versioni	Tutte le versioni
Europa (Irlanda)	Tutte le versioni	Tutte le versioni
Europa (Londra)	Tutte le versioni	Tutte le versioni
Europa (Milano)	–	–
Europa (Parigi)	Tutte le versioni	Tutte le versioni
Europa (Spagna)	–	–
Europa (Stoccolma)	–	–
Europa (Zurigo)	–	–
Israele (Tel Aviv)	–	–

Region	Aurora MySQL versione 3	Aurora MySQL versione 2
Messico (centrale)	–	–
Medio Oriente (Bahrein)	–	–
Medio Oriente (Emirati Arabi Uniti)	–	–
Sud America (San Paolo)	–	–
AWS GovCloud (Stati Uniti orientali)	–	–
AWS GovCloud (Stati Uniti occidentali)	–	–

Considerazioni sull'aggiornamento per i cluster abilitati per backtrack

Puoi aggiornare un cluster di database abilitato per il backtrack da Aurora MySQL versione 2 alla versione 3 perché tutte le versioni secondarie di Aurora MySQL versione 3 sono supportate per il backtrack.

Configurazione del backtrack di un cluster di database Aurora MySQL

Per utilizzare la funzionalità di backtrack, devi abilitarla e specificare una finestra di backtrack target. In caso contrario, il backtrack è disabilitato.

Nella finestra di backtrack target, specifica il periodo di tempo durante il quale desideri "riavvolgere" il database utilizzando la funzionalità di backtrack. Aurora prova conservare un numero di record di modifica sufficiente per supportare quella finestra temporale.

Console

Puoi utilizzare la console per configurare il backtrack quando crei un nuovo cluster di database. È inoltre possibile modificare un cluster DB per modificare la finestra di backtrack per un cluster abilitato per backtrack. Se si disattiva completamente il backtracking per un cluster impostando la finestra backtrack su 0, non sarà poi possibile attivare nuovamente il backtrack per quel cluster.

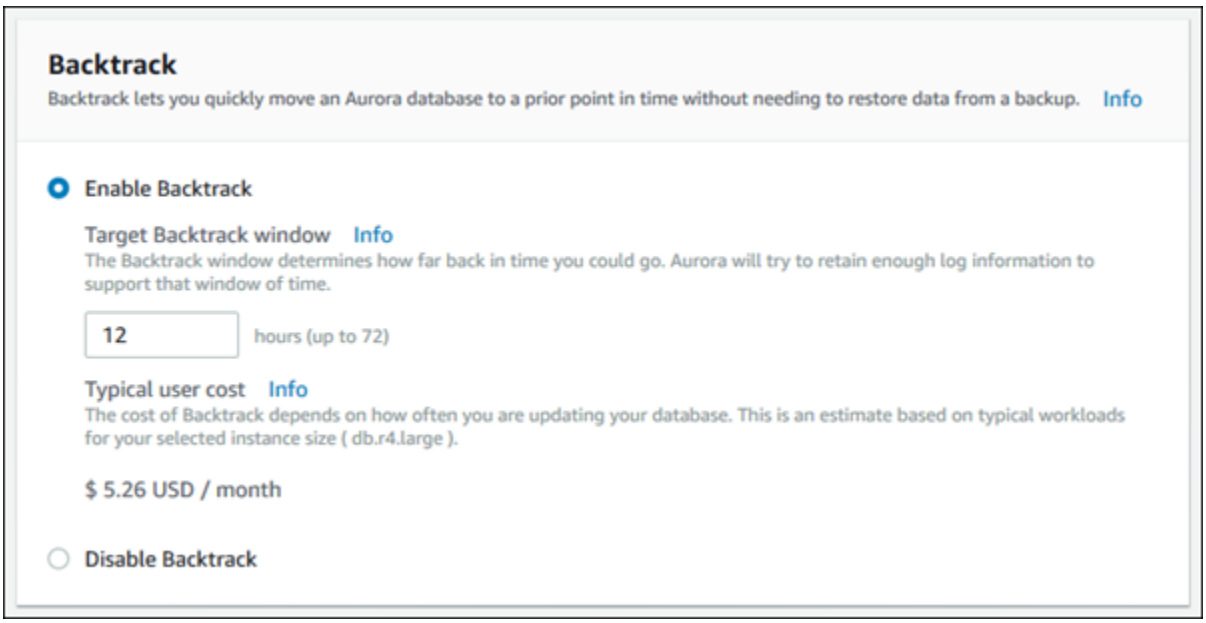
Argomenti

- Configurazione del backtrack mediante la console alla creazione di un cluster di database
- Configurazione del backtrack mediante la console alla modifica di un cluster database

Configurazione del backtrack mediante la console alla creazione di un cluster di database

Quando crei un nuovo cluster di database Aurora MySQL puoi configurare il backtrack scegliendo Enable Backtrack (Abilita backtrack) e specificando un valore per Target Backtrack window (Finestra di backtrack target) che sia maggiore di zero nella sezione Backtrack.

Per creare un cluster database, segui le istruzioni in Creazione di un cluster database Amazon Aurora. L'immagine seguente mostra la sezione Backtrack.



Quando crei un nuovo cluster database, Aurora non dispone dei dati relativi al carico di lavoro del cluster database. Di conseguenza, non può stimare un costo per il nuovo cluster database. La console indica tuttavia un costo utente standard per la finestra di backtrack target specificata basato su un carico di lavoro tipico. Questo costo standard è un riferimento generale per il costo della funzionalità di backtrack.

> **Important**
>
> Il costo effettivo potrebbe non corrispondere a quello standard in quanto è basato sul carico di lavoro del cluster database.

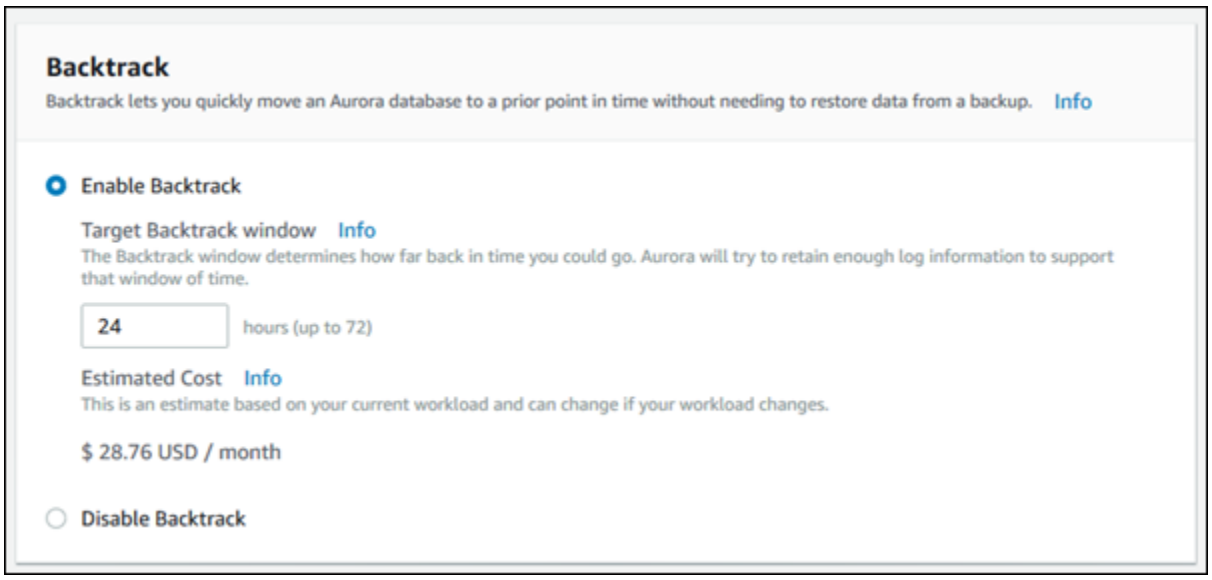
Configurazione del backtrack mediante la console alla modifica di un cluster database

Puoi modificare il backtrack per un cluster database mediante la console.

> **Note**
>
> Attualmente, è possibile modificare il backtracking solo per un cluster DB in cui è abilitata la funzionalità Backtrack. La sezione Backtrack non viene visualizzata per un cluster DB creato con la funzionalità Backtrack disattivata o se la funzionalità Backtrack è stata disattivata per il cluster DB.

Per modificare il backtrack per un cluster database mediante la console

1. Accedi alla Console di gestione AWS e apri la console Amazon RDS all'indirizzo https://console.aws.amazon.com/rds/.
2. Scegliere Databases (Database).
3. Scegliere il cluster da modificare, quindi scegliere Modify (Modifica).
4. In Target Backtrack window (Finestra di backtrack target), modificare il periodo di tempo durante il quale si desidera eseguire il backtrack. Il limite è 72 ore.



La console mostra il costo stimato per il periodo di tempo specificato in base al carico di lavoro precedente del cluster database:

- Se il backtrack è stato disabilitato sul cluster database, il costo stimato è basato sul parametro `VolumeWriteIOPS` per il cluster database in Amazon CloudWatch.
- Se il backtrack è stato abilitato in precedenza sul cluster database, il costo stimato è basato sul parametro `BacktrackChangeRecordsCreationRate` per il cluster database in Amazon CloudWatch.

5. Scegli Continue (Continua).
6. In Scheduling of Modifications (Pianificazione delle modifiche), scegliere una delle seguenti opzioni:
   - Apply during the next scheduled maintenance window (Applica durante la prossima finestra di manutenzione pianificata): –attendere la finestra di manutenzione successiva per applicare la modifica di Target Backtrack window (Finestra di backtrack target).
   - Apply immediately (Applica immediatamente): –applicare la modifica di Target Backtrack window (Finestra di backtrack target) appena possibile.
7. Scegliere Modify cluster (Modifica cluster).

AWS CLI

Quando crei un nuovo cluster database Aurora MySQL utilizzando il comando create-db-cluster di AWS CLI, puoi configurare il backtrack specificando un valore di `--backtrack-window` maggiore

di zero. Il valore `--backtrack-window` specifica la finestra di backtrack target. Per ulteriori informazioni, consulta Creazione di un cluster database Amazon Aurora.

Puoi anche specificare il valore `--backtrack-window` utilizzando i seguenti comandi della CLI AWS:

- modify-db-cluster
- restore-db-cluster-from-s3
- restore-db-cluster-from-snapshot
- restore-db-cluster-to-point-in-time

La procedura seguente descrive come modificare la finestra di backtrack target per un cluster database mediante l AWS CLI.

Per modificare la finestra di backtrack target per un cluster di database mediante l AWS CLI

- Chiamare il comando modify-db-cluster della CLI AWS e specificare i seguenti valori:
  - `--db-cluster-identifier`:– il nome del cluster database.
  - `--backtrack-window`:– il numero massimo di secondi durante i quali si desidera eseguire il backtrack del cluster database.

  L'esempio seguente imposta la finestra di backtrack target per `sample-cluster` su un giorno (86.400 secondi).

  Per Linux, macOS o Unix:

```
aws rds modify-db-cluster \
    --db-cluster-identifier sample-cluster \
    --backtrack-window 86400
```

  Per Windows:

```
aws rds modify-db-cluster ^
    --db-cluster-identifier sample-cluster ^
    --backtrack-window 86400
```

> **Note**
>
> Attualmente, è possibile abilitare il backtrack solo per un cluster database creato con la funzionalità di backtrack abilitata.

API RDS

Quando crei un nuovo cluster database Aurora MySQL utilizzando l'operazione API Amazon RDS CreateDBCluster, puoi configurare il backtrack specificando un valore `BacktrackWindow` maggiore di zero. Il valore `BacktrackWindow` specifica la finestra di backtrack target per il cluster database specificato nel valore `DBClusterIdentifier`. Per ulteriori informazioni, consulta Creazione di un cluster database Amazon Aurora.

Puoi anche specificare il valore `BacktrackWindow` utilizzando le seguenti operazioni API:

- ModifyDBCluster
- RestoreDBClusterFromS3
- RestoreDBClusterFromSnapshot
- RestoreDBClusterToPointInTime

> **Note**
>
> Attualmente, è possibile abilitare il backtrack solo per un cluster database creato con la funzionalità di backtrack abilitata.

Esecuzione di un backtrack per un cluster di database Aurora MySQL

Puoi eseguire il backtrack di un cluster database a un timestamp di backtrack specificato. Se il timestamp di backtrack non è anteriore al primo orario di backtrack possibile e non è nel futuro, il backtrack del cluster database viene eseguito a quel timestamp.

In caso contrario, si ha in genere un errore. Inoltre, se cerchi di eseguire il backtrack di un cluster database per il quale la registrazione binaria è abilitata, in genere si verifica un errore a meno che non hai scelto di forzare il backtrack. Forzare un backtrack può interferire con altre operazioni che utilizzano la registrazione binaria.

Important

Il backtrack non genera voci binlog per le modifiche che esegue. Se la registrazione binaria è abilitata per il cluster database, il backtrack potrebbe non essere compatibile con l'implementazione binlog.

Note

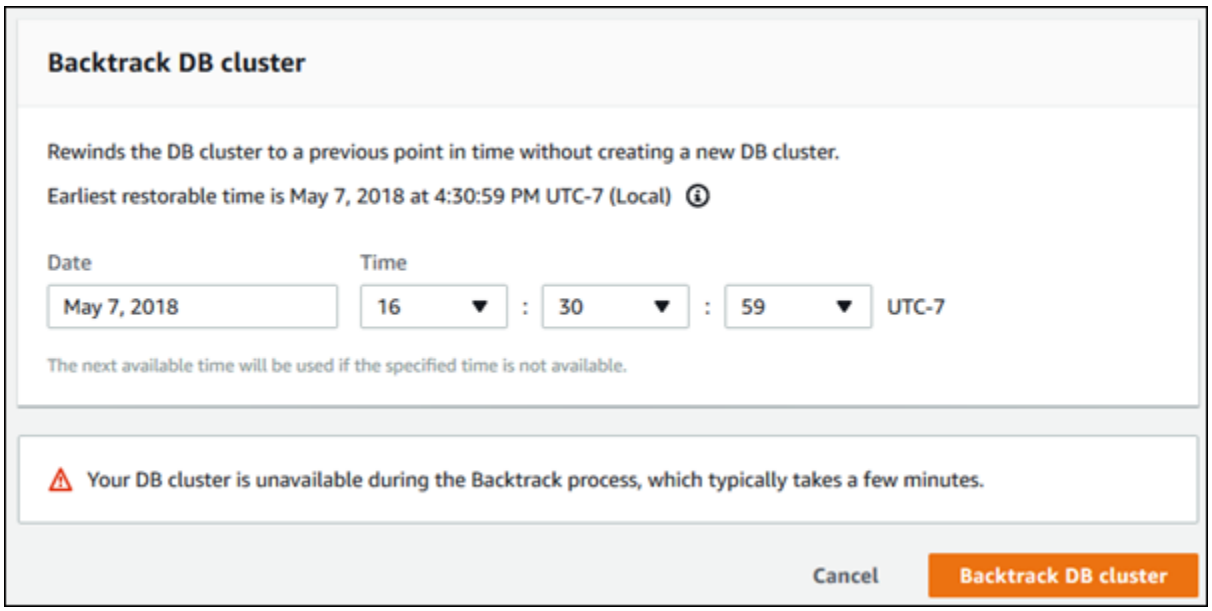
Per i cloni di database, non puoi eseguire il backtrack del cluster database a un orario anteriore a quello di creazione del clone. Per ulteriori informazioni sulla creazione di cloni di database, consulta Clonazione di un volume per un cluster di database Amazon Aurora.

Console

La procedura seguente descrive come eseguire un'operazione di backtrack per un cluster database mediante la console.

Per eseguire un'operazione di backtrack mediante la console

1. Accedi alla Console di gestione AWS e apri la console Amazon RDS all'indirizzo https://console.aws.amazon.com/rds/.
2. Nel riquadro di navigazione, scegliere Instances (Istanze).
3. Scegliere l'istanza principale del cluster database per il quale si intende eseguire il backtrack.
4. In Actions (Operazioni), scegliere Backtrack DB cluster (Esegui backtrack del cluster database).
5. Nella pagina Backtrack DB cluster (Esegui backtrack del cluster database), inserire il timestamp di backtrack a cui eseguire il backtrack del cluster database.



6. Scegliere Backtrack DB cluster (Esegui backtrack del cluster database).

AWS CLI

La procedura seguente descrive il modo in cui eseguire il backtrack di un cluster database mediante l AWS CLI.

Per eseguire il backtrack di un cluster database mediante l AWS CLI

- Chiamare il comando backtrack-db-cluster dell'CLI AWS e specificare i seguenti valori:
  - `--db-cluster-identifier`:– il nome del cluster database.
  - `--backtrack-to`:– il timestamp di backtrack a cui eseguire il backtrack del cluster database, specificato nel formato ISO 8601.

  L'esempio seguente esegue il backtrack del cluster database `sample-cluster` con il timestamp 19 marzo 2018, ore 10.

  Per Linux, macOS o Unix:

```
aws rds backtrack-db-cluster \
    --db-cluster-identifier sample-cluster \
    --backtrack-to 2018-03-19T10:00:00+00:00
```

Per Windows:

```
aws rds backtrack-db-cluster ^
    --db-cluster-identifier sample-cluster ^
    --backtrack-to 2018-03-19T10:00:00+00:00
```

API RDS

Per eseguire il backtrack di un cluster database mediante l'API Amazon RDS, utilizza l'operazione BacktrackDBCluster. Questa operazione esegue il backtrack del cluster database specificato nel valore `DBClusterIdentifier` all'orario specificato.

Monitoraggio del backtrack per un cluster di database Aurora MySQL

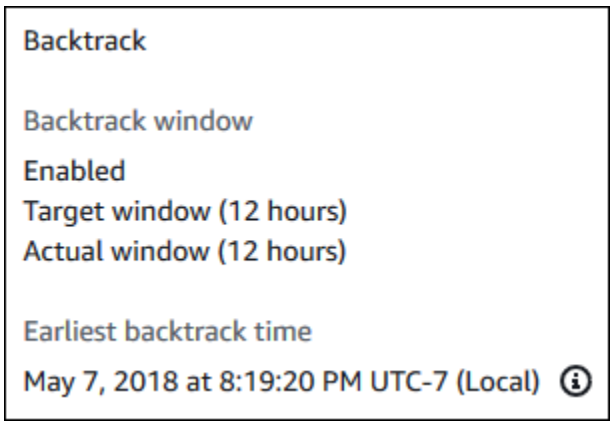
Puoi visualizzare le informazioni relative al backtrack e monitorare i parametri di backtrack per un cluster database.

Console

Per visualizzare le informazioni relative al backtrack e monitorare i parametri di backtrack mediante la console

1. Accedi alla Console di gestione AWS e apri la console Amazon RDS all'indirizzo https://console.aws.amazon.com/rds/.
2. Scegliere Databases (Database).
3. Scegliere il nome del cluster database di cui si intende visualizzare le informazioni.

   Le informazioni relative al backtrack sono visualizzate nella sezione Backtrack.



Quando il backtrack è abilitato, sono disponibili le seguenti informazioni:

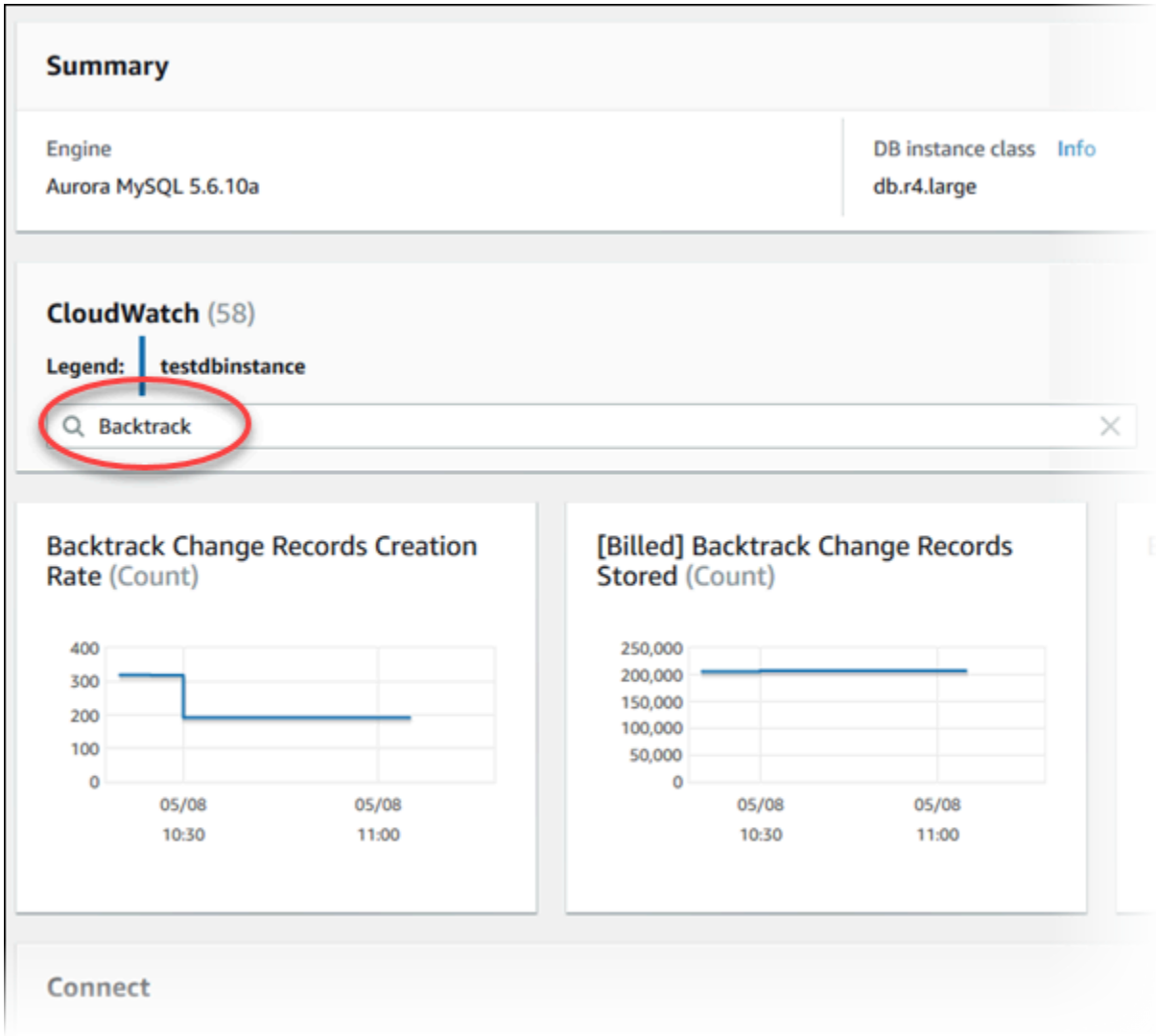
- Target window (Finestra target):– il periodo di tempo corrente specificato per la finestra di backtrack target. Il target è il periodo di tempo massimo durante il quale è possibile effettuare il backtrack se lo storage è sufficiente.
- Actual window (Finestra effettiva):– il periodo di tempo effettivo durante il quale è possibile eseguire il backtrack, che può essere inferiore alla finestra di backtrack target. La finestra di backtrack effettiva è basata sul carico di lavoro e sullo storage disponibile per conservare i record di modifica del backtrack.
- Earliest backtrack time (Primo orario di backtrack): –specifica il primo orario di backtrack possibile per il cluster database. Non è possibile eseguire il backtrack del cluster database a un orario antecedente a quello visualizzato.

4. Per visualizzare i parametri di backtrack per il cluster database, procedere come segue:

   a. Nel riquadro di navigazione, scegliere Instances (Istanze).

   b. Scegliere il nome dell'istanza principale per il cluster database di cui si intende visualizzarne i dettagli.

   c. Nella sezione CloudWatch, digitare **`Backtrack`** nella casella CloudWatch per visualizzare solo i parametri di backtrack.



Sono visualizzati i seguenti parametri:

- Backtrack Change Records Creation Rate (Count) (Frequenza di creazione record modifiche backtrack (numero)):– questo parametro visualizza il numero di record di modifica di backtrack creati in un intervallo di cinque minuti per il cluster database. È possibile utilizzare questo parametro per stimare il costo di backtrack per la finestra di backtrack target.
- [Billed] Backtrack Change Records Stored (Count) ([Fatturato] Record modifiche backtrack archiviati (numero)):– questo parametro visualizza il numero effettivo di record di modifica di backtrack utilizzati dal cluster di database.
- Backtrack Window Actual (Minutes) (Finestra di backtrack effettiva) (Minuti): –questo parametro indica se c'è una differenza tra la finestra di backtrack target e la finestra di backtrack effettiva. Ad esempio, se la finestra di backtrack target è 2 ore (120 minuti) e

questo parametro indica che la finestra di backtrack effettiva è 100 minuti, quest'ultima è più piccola della finestra target.

- Backtrack Window Alert (Count) (Avviso finestra di backtrack (numero)): –questo parametro indica quante volte la finestra di backtrack effettiva è inferiore alla finestra di backtrack target per un determinato periodo di tempo.

> **Note**
>
> I parametri seguenti potrebbero essere indietro rispetto all'orario corrente:
>
> - Backtrack Change Records Creation Rate (Count) (Frequenza di creazione record modifiche backtrack (numero))
> - [Billed] Backtrack Change Records Stored (Count) ([Fatturato] Record modifiche backtrack archiviati (numero))

AWS CLI

La procedura seguente descrive come visualizzare informazioni relative al backtrack per un cluster di database mediante l AWS CLI.

Per visualizzare informazioni relative al backtrack per un cluster di database mediante l AWS CLI

- Chiamare il comando describe-db-clusters dell'CLI AWS e specificare i seguenti valori:
  - `--db-cluster-identifier`:– il nome del cluster database.

  L'esempio seguente elenca le informazioni relative al backtrack per `sample-cluster`.

  Per Linux, macOS o Unix:

```
aws rds describe-db-clusters \
    --db-cluster-identifier sample-cluster
```

  Per Windows:

```
aws rds describe-db-clusters ^
    --db-cluster-identifier sample-cluster
```

API RDS

Per visualizzare le informazioni relative al backtrack per un cluster di database mediante l'API Amazon RDS, utilizzare l'operazione DescribeDBClusters. Questa operazione restituisce le informazioni relative al backtrack per il cluster database specificato nel valore `DBClusterIdentifier`.

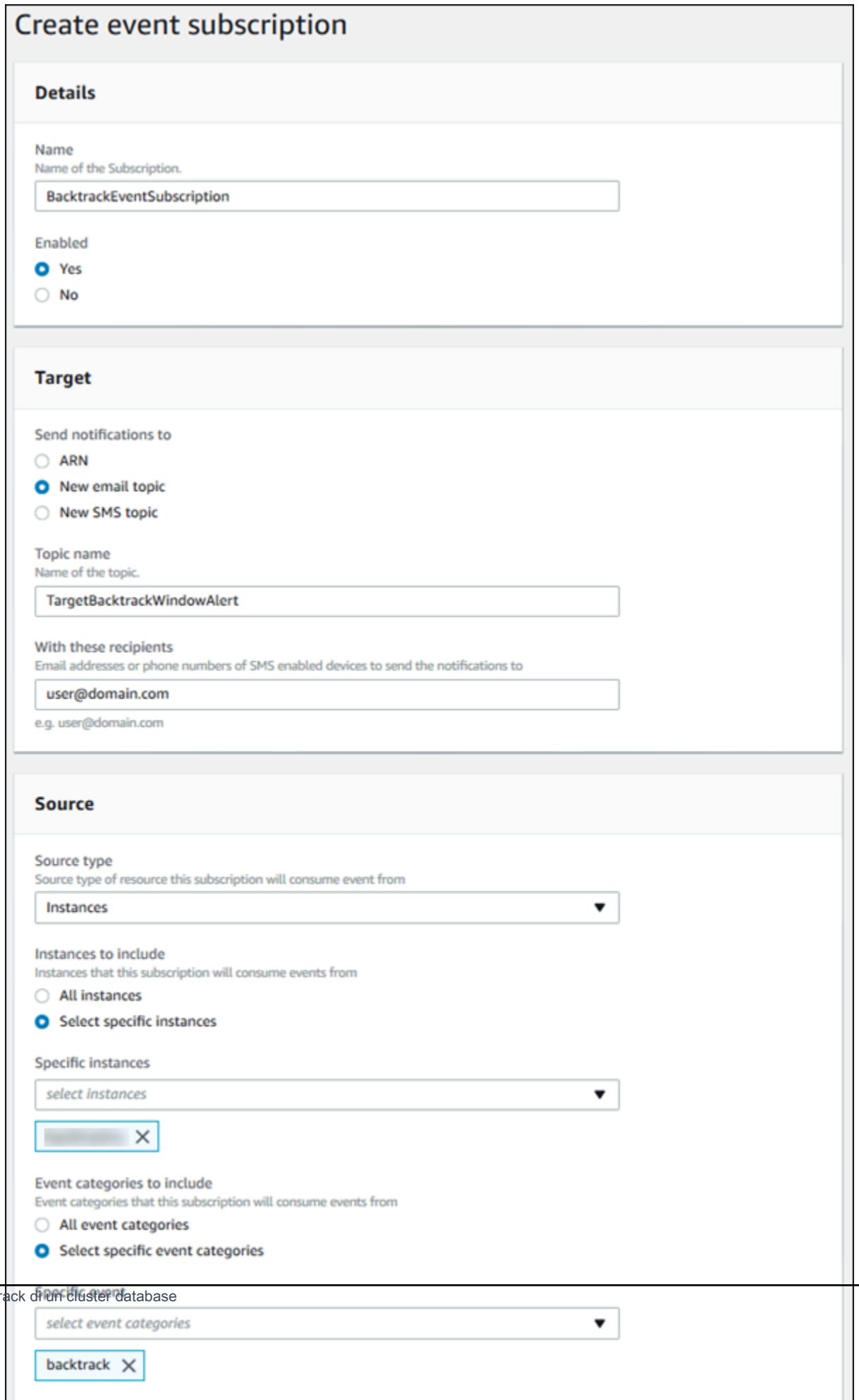
Abbonamento a un evento di backtrack con la console

La procedura seguente descrive come abbonarsi a un evento di backtrack mediante la console. L'evento ti invia un'email o una notifica di testo quando la finestra di backtrack effettiva è più piccola della finestra di backtrack target.

Per visualizzare le informazioni di backtrack mediante la console

1. Accedi a Console di gestione AWS e apri la console Amazon RDS all'indirizzo https://console.aws.amazon.com/rds/.
2. Scegliere Event subscriptions (Abbonamenti a eventi).
3. Scegliere Create event subscription (Crea abbonamento a eventi).
4. Nella casella Name (Nome), digitare un nome per l'abbonamento all'evento e assicurarsi che Yes (Sì) è selezionato per Enabled (Abilitato).
5. Nella sezione Target, scegliere New email topic (Nuovo argomento e-mail).
6. In Topic name (Nome argomento), digitare un nome per l'argomento e in With these recipients (Con questi destinatari), digitare gli indirizzi e-mail o i numeri di telefono per ricevere le notifiche.
7. Nella sezione Source (origine), scegliere Instances (Istanze) per Source type (Tipo origine).
8. Per Instances to include (Istanze da includere), scegliere Select specific instances (Seleziona specifiche istanze) e scegliere l'istanza database.
9. In Event categories to include (Categorie di eventi da includere), scegliere Select specific event categories (Seleziona specifiche categorie di eventi), quindi scegliere Backtrack.

   La pagina dovrebbe essere simile alla seguente.


Create event subscription
Details
Name
Name of the Subscription.
BacktrackEventSubscription
Enabled
Yes
No
Target
Send notifications to
ARN
New email topic
New SMS topic
Topic name
Name of the topic.
TargetBacktrackWindowAlert
With these recipients
Email addresses or phone numbers of SMS enabled devices to send the notifications to
user@domain.com
e.g. user@domain.com
Source
Source type
Source type of resource this subscription will consume event from
Instances
Instances to include
Instances that this subscription will consume events from
All instances
Select specific instances
Specific instances
select instances
Event categories to include
Event categories that this subscription will consume events from
All event categories
Select specific event categories
select event categories
backtrack

10. Scegliere Create (Crea).

Recupero di backtrack esistenti

Puoi recuperare informazioni relative a backtrack esistenti per un cluster di database. Queste informazioni includono l'identificatore univoco del backtrack, la data e l'ora di inizio e di fine del backtrack, la data e l'ora in cui il backtrack è stato richiesto e lo stato corrente del backtrack.

> **Note**
>
> Non è attualmente possibile recuperare backtrack esistenti mediante la console.

AWS CLI

La procedura seguente descrive come recuperare backtrack esistenti per un cluster di database mediante l AWS CLI.

Per recuperare i backtrack esistenti utilizzando il AWS CLI

- Chiamate il comando describe-db-cluster-backtracks AWS CLI e fornite i seguenti valori:
  - `--db-cluster-identifier`:– il nome del cluster database.

  L'esempio seguente recupera i backtrack esistenti per `sample-cluster`.

  Per Linux, macOS o Unix:

```
aws rds describe-db-cluster-backtracks \
    --db-cluster-identifier sample-cluster
```

  Per Windows:

```
aws rds describe-db-cluster-backtracks ^
    --db-cluster-identifier sample-cluster
```

API RDS

Per recuperare informazioni sui backtrack per un cluster DB utilizzando l'API Amazon RDS, utilizza l'operazione DBClusterDescrivi Backtracks. Questa operazione restituisce le informazioni relative ai backtrack per il cluster database specificato nel valore `DBClusterIdentifier`.

Disabilitazione del backtrack per un cluster di database Aurora MySQL

Puoi disabilitare la funzionalità di backtrack per un cluster di database.

Console

Puoi disabilitare il backtrack per un cluster di database mediante la console. Dopo aver disattivato completamente il backtracking per un cluster, non sarà possibile attivarlo di nuovo per tale cluster.

Per disabilitare la funzionalità di backtrack per un cluster di database mediante la console

1. Accedi alla Console di gestione AWS e apri la console Amazon RDS all'indirizzo https://console.aws.amazon.com/rds/.
2. Scegliere Databases (Database).
3. Scegliere il cluster da modificare, quindi scegliere Modify (Modifica).
4. Nella sezione Backtrack, scegliere Disable Backtrack (Disabilita backtrack).
5. Scegli Continue (Continua).
6. In Scheduling of Modifications (Pianificazione delle modifiche), scegliere una delle seguenti opzioni:
   - Apply during the next scheduled maintenance window (Applica durante la prossima finestra di manutenzione pianificata):– attendere la finestra di manutenzione successiva per applicare la modifica.
   - Apply immediately (Applica immediatamente):– applicare la modifica appena possibile.
7. Scegliere Modify Cluster (Modifica cluster).

AWS CLI

Puoi disabilitare la funzionalità di backtrack per un cluster di database mediante l'AWS CLI impostando la finestra di backtrack target su `0` (zero). Dopo aver disattivato completamente il backtracking per un cluster, non sarà possibile attivarlo di nuovo per tale cluster.

Per modificare la finestra di backtrack target per un cluster di database mediante l AWS CLI

- Chiamare il comando modify-db-cluster della CLI AWS e specificare i seguenti valori:
  - `--db-cluster-identifier`:– il nome del cluster database.
  - `--backtrack-window` – specifica `0` per disattivare il backtracking.

  L'esempio seguente disabilita la funzionalità di backtrack per `sample-cluster` impostando `--backtrack-window` su `0`.

  Per Linux, macOS o Unix:

```
aws rds modify-db-cluster \
    --db-cluster-identifier sample-cluster \
    --backtrack-window 0
```

  Per Windows:

```
aws rds modify-db-cluster ^
    --db-cluster-identifier sample-cluster ^
    --backtrack-window 0
```

API RDS

Per disabilitare la funzionalità di backtrack per un cluster di database mediante l'API Amazon RDS, utilizzare l'operazione ModifyDBCluster. Imposta il valore `BacktrackWindow` su `0` (zero) e specifica il cluster di database nel valore `DBClusterIdentifier`. Dopo aver disattivato completamente il backtracking per un cluster, non sarà possibile attivarlo di nuovo per tale cluster.

Test di Amazon Aurora MySQL mediante query Fault Injection

Puoi testare la tolleranza ai guasti del tuo cluster database Aurora MySQL utilizzando query Fault Injection. Le query di errore di iniezione vengono emesse come comandi SQL per un'istanza Amazon Aurora. Queste funzionalità ti consentono di pianificare l'occorrenza simulata di uno dei seguenti eventi:

- Un arresto anomalo di un'istanza database di lettura o di scrittura

- Un errore di una replica Aurora
- Un errore del disco
- Una congestione del disco

Quando una query Fault Injection specifica un arresto anomalo, impone un arresto anomalo dell'istanza database Aurora MySQL. Le altre query fault injection generano simulazioni di eventi di errore, ma non causano l'evento. Quando invii una query fault injection, specifichi anche la durata della simulazione dell'evento di errore.

Puoi inviare una query fault injection a una delle istanze di replica Aurora eseguendo la connessione all'endpoint per la replica Aurora. Per ulteriori informazioni, consulta Connessioni degli endpoint Amazon Aurora.

L'esecuzione di query fault injection richiede tutti i privilegi dell'utente master. Per ulteriori informazioni, consulta Privilegi dell'account utente master.

Test dell'arresto anomalo di un'istanza

Puoi forzare un arresto anomalo di un'istanza Amazon Aurora utilizzando la query fault injection `ALTER SYSTEM CRASH`.

Per questa query fault injection, non si verificherà un failover. Se desideri testare un failover, puoi scegliere l'operazione di istanza Failover per il cluster di database nella console di RDS, oppure utilizzare il comando failover-db-cluster della AWS CLI o l'operazione dell'API RDS FailoverDBCluster.

Sintassi

```
ALTER SYSTEM CRASH [ INSTANCE | DISPATCHER | NODE ];
```

Opzioni

Questa query fault injection accetta uno dei seguenti tipi di arresto anomalo:

- **`INSTANCE`** — Viene simulato un arresto anomalo del database compatibile con MySQL per l'istanza Amazon Aurora.
- **`DISPATCHER`** — Viene simulato un arresto anomalo del dispatcher sull'istanza di scrittura per il cluster database di Aurora. Il dispatcher scrive gli aggiornamenti sul volume del cluster per un cluster di database Amazon Aurora.

- **`NODE`** — Viene simulato un arresto anomalo del database compatibile con MySQL e del dispatcher per l'istanza Amazon Aurora. Per questa simulazione fault injection, viene inoltre eliminata la cache.

Il tipo di arresto anomalo predefinito è `INSTANCE`.

Test di un errore di replica Aurora

Puoi simulare l'errore di una replica Aurora mediante la query fault injection `ALTER SYSTEM SIMULATE READ REPLICA FAILURE`.

Un errore di replica Aurora blocca tutte le richieste da un'istanza di scrittura a una replica Aurora o tutte le repliche Aurora nel cluster di database per un periodo di tempo specificato. Al termine di tale periodo, le repliche Aurora interessate saranno automaticamente sincronizzate con l'istanza di scrittura.

Sintassi

```
ALTER SYSTEM SIMULATE percentage_of_failure PERCENT READ REPLICA FAILURE
    [ TO ALL | TO "replica name" ]
    FOR INTERVAL quantity { YEAR | QUARTER | MONTH | WEEK | DAY | HOUR | MINUTE |
 SECOND };
```

Opzioni

Questa query fault injection accetta i seguenti parametri:

- **`percentage_of_failure`** — La percentuale di richieste da bloccare durante l'evento di errore. Può essere un valore tra 0 e 100. Se specifichi 0, non viene bloccata alcuna richiesta. Se specifichi 100, vengono bloccate tutte le richieste.
- Failure type (Tipo di errore) — Il tipo di errore da simulare. Specifica `TO ALL` per simulare errori per tutte le repliche Aurora nel cluster di database. Specifica `TO` e il nome della replica Aurora per simulare un errore di un'unica replica Aurora. Il tipo di errore predefinito è `TO ALL`.
- **`quantity`** — L'intervallo di tempo durante il quale simulare l'errore di replica di Aurora. L'intervallo è un numero seguito da un'unità di tempo. Il valore nell'unità specificata indica la durata della simulazione. Ad esempio, con `20 MINUTE` la simulazione durerà 20 minuti.

> **Note**
>
> Fai attenzione quando specifichi l'intervallo di tempo per l'evento di errore della replica Aurora. Se specifichi un intervallo di tempo troppo lungo e l'istanza master scrive una grande quantità di dati durante l'evento di errore, il cluster di database Aurora potrebbe ritenere che si è verificato un arresto anomalo della replica Aurora e quindi sostituirla.

Test di un errore del disco

Puoi simulare un errore del disco per un cluster database Aurora mediante la query fault injection `ALTER SYSTEM SIMULATE DISK FAILURE`.

Durante la simulazione di un errore del disco, il cluster di database Aurora contrassegna in modo aleatorio i segmenti del disco come difettosi. Le richieste a tali segmenti saranno bloccate per la durata della simulazione.

Sintassi

```
ALTER SYSTEM SIMULATE percentage_of_failure PERCENT DISK FAILURE
    [ IN DISK index | NODE index ]
    FOR INTERVAL quantity { YEAR | QUARTER | MONTH | WEEK | DAY | HOUR | MINUTE |
 SECOND };
```

Opzioni

Questa query fault injection accetta i seguenti parametri:

- **`percentage_of_failure`** — la percentuale del disco da contrassegnare come difettosa durante l'evento di errore. Può essere un valore tra 0 e 100. Se specifichi 0, nessuna parte del disco è contrassegnata come difettosa. Se specifichi 100, tutto il disco è contrassegnato come difettoso.
- **`DISK index`** — uno specifico blocco di dati logico per il quale simulare l'evento di errore. Se viene superato l'intervallo di blocchi di dati logici disponibili, viene generato un errore indicante il valore di indice massimo che puoi specificare. Per ulteriori informazioni, consulta Visualizzazione dello stato del volume per un cluster DB Aurora MySQL.
- **`NODE index`** — uno specifico nodo di storage per il quale simulare l'evento di errore. Se viene superato l'intervallo di nodi di storage disponibili, viene generato un errore indicante il valore di

indice massimo che puoi specificare. Per ulteriori informazioni, consulta Visualizzazione dello stato del volume per un cluster DB Aurora MySQL.

- **`quantity`** — l'intervallo di tempo durante il quale simulare l'errore del disco. L'intervallo è un numero seguito da un'unità di tempo. Il valore nell'unità specificata indica la durata della simulazione. Ad esempio, con `20 MINUTE` la simulazione durerà 20 minuti.

Test di una congestione del disco

Puoi simulare un errore del disco per un cluster database Aurora mediante la query fault injection `ALTER SYSTEM SIMULATE DISK CONGESTION`.

Durante la simulazione della congestione del disco, il cluster di database Aurora contrassegna in modo aleatorio i segmenti del disco come congestionati. Le richieste a tali segmenti saranno ritardate di un periodo di tempo compreso tra il ritardo minimo e quello massimo specificati per la durata della simulazione.

Sintassi

```
ALTER SYSTEM SIMULATE percentage_of_failure PERCENT DISK CONGESTION
    BETWEEN minimum AND maximum MILLISECONDS
    [ IN DISK index | NODE index ]
    FOR INTERVAL quantity { YEAR | QUARTER | MONTH | WEEK | DAY | HOUR | MINUTE |
 SECOND };
```

Opzioni

Questa query fault injection accetta i seguenti parametri:

- **`percentage_of_failure`** — La percentuale del disco da contrassegnare come congestionata durante l'evento di errore. Può essere un valore tra 0 e 100. Se specifichi 0, nessuna parte del disco è contrassegnata come congestionata. Se specifichi 100, tutto il disco è contrassegnato come congestionato.
- **`DISK index`** o **`NODE index`** — Uno specifico disco o nodo per il quale simulare l'evento di errore. Se viene superato l'intervallo di indici per il disco o il nodo, viene generato un errore indicante il valore di indice massimo che puoi specificare.
- **`minimum`** e **`maximum`** — Il valore minimo e massimo del ritardo di congestione in millisecondi. I segmenti del disco contrassegnati come congestionati saranno ritardati per un periodo di tempo

aleatorio compreso tra il valore minimo e quello massimo in millisecondi per la durata della simulazione.

- **`quantity`** — L'intervallo di tempo durante il quale simulare la congestione del disco. L'intervallo è un numero seguito da un'unità di tempo. Il valore nell'unità specificata indica la durata della simulazione. Ad esempio, con `20 MINUTE` la simulazione durerà 20 minuti.

Alterazione delle tabelle in Amazon Aurora mediante Fast DDL

Amazon Aurora include ottimizzazioni per eseguire un'operazione `ALTER TABLE` in atto, quasi istantaneamente. L'operazione viene eseguita senza la copia della tabella e senza impatto materiale sulle altre istruzioni DML. Poiché l'operazione non consuma storage temporaneo per la copia della tabella, rende le istruzioni DDL pratiche anche nel caso di tabelle di grandi dimensioni su tipi di classi small.

Aurora MySQL versione 3 è compatibile con la funzionalità MySQL 8.0 chiamata Instant DDL. Aurora MySQL versione 2 utilizza un'implementazione diversa chiamata Fast DDL.

Argomenti

- DDL istantaneo (Aurora MySQL versione 3)
- Fast DDL (Aurora MySQL versione 2)

DDL istantaneo (Aurora MySQL versione 3)

L'ottimizzazione eseguita da Aurora MySQL versione 3 per migliorare l'efficienza di alcune operazioni DDL è chiamata DDL istantaneo.

Aurora MySQL versione 3 è compatibile con il DDL istantaneo della community MySQL 8.0. Si esegue un'operazione DDL istantanea utilizzando la clausola `ALGORITHM=INSTANT` con l'istruzione `ALTER TABLE`. Per informazioni sulla sintassi e sull'utilizzo del DDL istantaneo, vedere ALTER TABLE e Online DDL Operations (Operazioni DDL online) nella documentazione MySQL.

I seguenti esempi dimostrano la funzionalità DDL immediata. Le istruzioni `ALTER TABLE` aggiungono colonne e modificano i valori di default delle colonne. Gli esempi includono colonne regolari e virtuali e tabelle regolari e partizionate. In ogni fase, è possibile visualizzare i risultati emettendo `SHOW CREATE TABLE` e istruzioni `DESCRIBE`.

```
mysql> CREATE TABLE t1 (a INT, b INT, KEY(b)) PARTITION BY KEY(b) PARTITIONS 6;
Query OK, 0 rows affected (0.02 sec)
```

```
mysql> ALTER TABLE t1 RENAME TO t2, ALGORITHM = INSTANT;
Query OK, 0 rows affected (0.01 sec)

mysql> ALTER TABLE t2 ALTER COLUMN b SET DEFAULT 100, ALGORITHM = INSTANT;
Query OK, 0 rows affected (0.00 sec)

mysql> ALTER TABLE t2 ALTER COLUMN b DROP DEFAULT, ALGORITHM = INSTANT;
Query OK, 0 rows affected (0.01 sec)

mysql> ALTER TABLE t2 ADD COLUMN c ENUM('a', 'b', 'c'), ALGORITHM = INSTANT;
Query OK, 0 rows affected (0.01 sec)

mysql> ALTER TABLE t2 MODIFY COLUMN c ENUM('a', 'b', 'c', 'd', 'e'), ALGORITHM =
 INSTANT;
Query OK, 0 rows affected (0.01 sec)

mysql> ALTER TABLE t2 ADD COLUMN (d INT GENERATED ALWAYS AS (a + 1) VIRTUAL), ALGORITHM
 = INSTANT;
Query OK, 0 rows affected (0.02 sec)

mysql> ALTER TABLE t2 ALTER COLUMN a SET DEFAULT 20,
    ->   ALTER COLUMN b SET DEFAULT 200, ALGORITHM = INSTANT;
Query OK, 0 rows affected (0.01 sec)

mysql> CREATE TABLE t3 (a INT, b INT) PARTITION BY LIST(a)(
    ->   PARTITION mypart1 VALUES IN (1,3,5),
    ->   PARTITION MyPart2 VALUES IN (2,4,6)
    -> );
Query OK, 0 rows affected (0.03 sec)

mysql> ALTER TABLE t3 ALTER COLUMN a SET DEFAULT 20, ALTER COLUMN b SET DEFAULT 200,
 ALGORITHM = INSTANT;
Query OK, 0 rows affected (0.01 sec)

mysql> CREATE TABLE t4 (a INT, b INT) PARTITION BY RANGE(a)
    ->   (PARTITION p0 VALUES LESS THAN(100), PARTITION p1 VALUES LESS THAN(1000),
    ->   PARTITION p2 VALUES LESS THAN MAXVALUE);
Query OK, 0 rows affected (0.05 sec)

mysql> ALTER TABLE t4 ALTER COLUMN a SET DEFAULT 20,
    ->   ALTER COLUMN b SET DEFAULT 200, ALGORITHM = INSTANT;
Query OK, 0 rows affected (0.01 sec)
```

```
/* Sub-partitioning example */
mysql> CREATE TABLE ts (id INT, purchased DATE, a INT, b INT)
    ->   PARTITION BY RANGE( YEAR(purchased) )
    ->     SUBPARTITION BY HASH( TO_DAYS(purchased) )
    ->     SUBPARTITIONS 2 (
    ->       PARTITION p0 VALUES LESS THAN (1990),
    ->       PARTITION p1 VALUES LESS THAN (2000),
    ->       PARTITION p2 VALUES LESS THAN MAXVALUE
    ->    );
Query OK, 0 rows affected (0.10 sec)

mysql> ALTER TABLE ts ALTER COLUMN a SET DEFAULT 20,
    ->   ALTER COLUMN b SET DEFAULT 200, ALGORITHM = INSTANT;
Query OK, 0 rows affected (0.01 sec)
```

Fast DDL (Aurora MySQL versione 2)

Fast DDL di Aurora MySQL è un'ottimizzazione progettata per migliorare le prestazioni di determinate modifiche allo schema, come l'aggiunta o l'eliminazione di colonne, riducendo i tempi di inattività e l'utilizzo delle risorse. Consente di completare queste operazioni in modo più efficiente rispetto ai metodi DDL tradizionali.

> **Important**
>
> Attualmente, la modalità Lab di Aurora deve essere abilitata per utilizzare Fast DDL. Per ulteriori informazioni su come abilitare la modalità di laboratorio per , consulta Modalità di laboratorio per Amazon Aurora MySQL.
> L'ottimizzazione Fast DDL è stata inizialmente introdotta nella modalità Lab di Aurora MySQL versione 2 per migliorare l'efficienza di alcune operazioni DDL. In Aurora MySQL versione 3 la modalità Lab è stata rimossa e Fast DDL è stata sostituita dalla funzionalità Instant DDL di MySQL 8.0.

In MySQL, molte operazioni DDL (Data Definition Language) hanno un impatto considerevole sulle prestazioni.

Ad esempio, supponiamo che utilizzi un'operazione `ALTER TABLE` per aggiungere una colonna a una tabella. A seconda dell'algoritmo specificato per l'operazione, questa può comportare quanto segue:

- Creazione di una copia completa della tabella
- Creazione di una tabella temporanea per elaborare operazioni DML (Data Manipulation Language) simultanee
- Ricostruzione di tutti gli indici per la tabella
- Applicazione dei blocchi di tabelle durante l'applicazione di modifiche DML simultanee
- Rallentamento del throughput DML simultaneo

Questo impatto sulle prestazioni può essere particolarmente complesso in ambienti con tabelle di grandi dimensioni o volumi di transazioni elevati. Fast DDL aiuta a mitigare queste sfide ottimizzando le modifiche allo schema, il che consente operazioni più rapide e meno dispendiose in termini di risorse.

Limiti di Fast DDL

Fast DDL presenta attualmente le seguenti limitazioni:

- Supporta solo l'aggiunta di una colonna nullable, senza valori predefiniti, alla fine di una tabella esistente.
- La DDL veloce non funziona per le tabelle partizionate.
- DDL rapida non funziona per le tabella InnoDB che utilizzano il formato riga REDUNDANT.
- La DDL veloce non funziona per le tabelle con indici di ricerca full-text.
- Se la dimensione di record massima possibile per l'operazione DDL è troppo grande, Fast DDL non viene utilizzato. Una dimensione di record è troppo importante se è più grande della metà della dimensione della pagina. La dimensione massima di un record viene calcolata sommando la dimensione massima di ogni colonna. Per le colonne di dimensione variabile, conformemente agli standard InnoDB, i byte extern non sono inclusi nel calcolo.

Sintassi di Fast DDL

```
ALTER TABLE tbl_name ADD COLUMN col_name column_definition
```

Questa istruzione accetta le seguenti opzioni:

- **`tbl_name`** — il nome della tabella da modificare.
- **`col_name`** — il nome della colonna da aggiungere.

- **`col_definition`** — la definizione della colonna da aggiungere.

> **Note**
>
> Devi specificare una definizione di colonna nullable senza un valore predefinito. In caso contrario, Fast DDL non viene utilizzato.

Esempi di Fast DDL

Negli esempi seguenti viene illustrata la velocità delle operazioni Fast DDL. Il primo esempio SQL esegue istruzioni `ALTER TABLE` su una tabella di grandi dimensioni senza utilizzare Fast DDL. Questa operazione richiede molto tempo. Un esempio della CLI mostra come abilitare Fast DDL per il cluster. Quindi un altro esempio SQL esegue le stesse istruzioni `ALTER TABLE` su una tabella identica. Con Fast DDL abilitato, l'operazione è molto veloce.

In questo esempio viene utilizzata la tabella `ORDERS` del benchmark TPC-H, contenente 150 milioni di righe. Questo cluster utilizza intenzionalmente una classe di istanza relativamente piccola, per dimostrare quanto tempo possano richiedere le istruzioni `ALTER TABLE` quando non è possibile utilizzare Fast DDL. Nell'esempio viene creato un clone della tabella originale contenente dati identici. Il controllo dell'impostazione `aurora_lab_mode` conferma che il cluster non può utilizzare Fast DDL poiché la modalità lab non è abilitata. Quindi le istruzioni `ALTER TABLE ADD COLUMN` richiedono molto tempo per aggiungere nuove colonne alla fine della tabella.

```
mysql> create table orders_regular_ddl like orders;
Query OK, 0 rows affected (0.06 sec)

mysql> insert into orders_regular_ddl select * from orders;
Query OK, 150000000 rows affected (1 hour 1 min 25.46 sec)

mysql> select @@aurora_lab_mode;
+------------------+
| @@aurora_lab_mode |
+------------------+
|                0 |
+------------------+

mysql> ALTER TABLE orders_regular_ddl ADD COLUMN o_refunded boolean;
Query OK, 0 rows affected (40 min 31.41 sec)

mysql> ALTER TABLE orders_regular_ddl ADD COLUMN o_coverletter varchar(512);
```

```
Query OK, 0 rows affected (40 min 44.45 sec)
```

Questo esempio esegue la stessa preparazione di una tabella di grandi dimensioni dell'esempio precedente. Tuttavia, non è possibile abilitare semplicemente la modalità lab all'interno di una sessione SQL interattiva. Tale impostazione deve essere abilitata in un gruppo di parametri personalizzato. Ciò richiede l'uscita dalla sessione `mysql` e l'esecuzione di alcuni comandi della CLI AWS o l'utilizzo della Console di gestione AWS.

```
mysql> create table orders_fast_ddl like orders;
Query OK, 0 rows affected (0.02 sec)

mysql> insert into orders_fast_ddl select * from orders;
Query OK, 150000000 rows affected (58 min 3.25 sec)

mysql> set aurora_lab_mode=1;
ERROR 1238 (HY000): Variable 'aurora_lab_mode' is a read only variable
```

Per abilitare la modalità lab per il cluster è necessario lavorare con un gruppo di parametri. In questo esempio della CLI AWS viene utilizzato un gruppo di parametri del cluster per garantire che tutte le istanze database nel cluster utilizzino lo stesso valore per l'impostazione della modalità lab.

```
$ aws rds create-db-cluster-parameter-group \
  --db-parameter-group-family aurora5.7 \
    --db-cluster-parameter-group-name lab-mode-enabled-57 --description 'TBD'
$ aws rds describe-db-cluster-parameters \
  --db-cluster-parameter-group-name lab-mode-enabled-57 \
    --query '*[*].[ParameterName,ParameterValue]' \
      --output text | grep aurora_lab_mode
aurora_lab_mode 0
$ aws rds modify-db-cluster-parameter-group \
  --db-cluster-parameter-group-name lab-mode-enabled-57 \
    --parameters ParameterName=aurora_lab_mode,ParameterValue=1,ApplyMethod=pending-
reboot
{
    "DBClusterParameterGroupName": "lab-mode-enabled-57"
}

# Assign the custom parameter group to the cluster that's going to use Fast DDL.
$ aws rds modify-db-cluster --db-cluster-identifier tpch100g \
  --db-cluster-parameter-group-name lab-mode-enabled-57
{
  "DBClusterIdentifier": "tpch100g",
```

```
  "DBClusterParameterGroup": "lab-mode-enabled-57",
  "Engine": "aurora-mysql",
  "EngineVersion": "5.7.mysql_aurora.2.10.2",
  "Status": "available"
}

# Reboot the primary instance for the cluster tpch100g:
$ aws rds reboot-db-instance --db-instance-identifier instance-2020-12-22-5208
{
  "DBInstanceIdentifier": "instance-2020-12-22-5208",
  "DBInstanceStatus": "rebooting"
}

$ aws rds describe-db-clusters --db-cluster-identifier tpch100g \
  --query '*[].[DBClusterParameterGroup]' --output text
lab-mode-enabled-57

$ aws rds describe-db-cluster-parameters \
  --db-cluster-parameter-group-name lab-mode-enabled-57 \
    --query '*[*].{ParameterName:ParameterName,ParameterValue:ParameterValue}' \
      --output text | grep aurora_lab_mode
aurora_lab_mode 1
```

Nell'esempio seguente vengono illustrati i passaggi rimanenti dopo che la modifica del gruppo di parametri ha effetto. Si verifica l'impostazione `aurora_lab_mode` per assicurarsi che il cluster possa utilizzare Fast DDL. Quindi viene eseguita l'istruzione `ALTER TABLE` per aggiungere colonne alla fine di un'altra tabella di grandi dimensioni. Questa volta, le istruzioni vengono completate molto rapidamente.

```
mysql> select @@aurora_lab_mode;
+-------------------+
| @@aurora_lab_mode |
+-------------------+
|                 1 |
+-------------------+

mysql> ALTER TABLE orders_fast_ddl ADD COLUMN o_refunded boolean;
Query OK, 0 rows affected (1.51 sec)

mysql> ALTER TABLE orders_fast_ddl ADD COLUMN o_coverletter varchar(512);
Query OK, 0 rows affected (0.40 sec)
```

Visualizzazione dello stato del volume per un cluster DB Aurora MySQL

In Amazon Aurora, un volume di cluster di database è costituito da una raccolta di blocchi logici. Ognuno di questi rappresenta 10 gigabyte di storage allocato. Questi blocchi sono denominati gruppi di protezione.

I dati in ogni gruppo di protezione sono replicati su sei dispositivi di storage fisico denominati nodi di storage. Questi nodi sono allocati su tre zone di disponibilità (AZ) nella regione AWS in cui si trova il cluster DB. Ogni nodo di storage contiene a sua volta uno o più blocchi di dati logici per il volume di cluster di database. Per ulteriori informazioni sui gruppi di protezione e i nodi di storage, consulta Introduzione del motore di storage Aurora nel blog AWS Database.

Puoi simulare l'errore di un intero nodo di storage o di un singolo blocco di dati logico in un nodo di storage. A questo proposito, devi utilizzare l'istruzione fault injection `ALTER SYSTEM SIMULATE DISK FAILURE`. Per l'istruzione, specifichi il valore di indice di uno specifico blocco di dati logico o nodo di storage. Tuttavia, se specifichi un valore di indice superiore al numero di blocchi di dati logici o nodi di storage utilizzati dal volume di cluster di database, l'istruzione restituisce un errore. Per ulteriori informazioni sulle query fault injection, consulta Test di Amazon Aurora MySQL mediante query Fault Injection.

Puoi evitare tale errore utilizzando l'istruzione `SHOW VOLUME STATUS`. L'istruzione restituisce due variabili di stato del server, `Disks` e `Nodes`. Queste variabili rappresentano rispettivamente il numero totale di blocchi di dati logici e di nodi di storage per il volume di cluster di database.

Sintassi

```
SHOW VOLUME STATUS
```

Esempio

Il seguente esempio illustra un risultato SHOW VOLUME STATUS tipico.

```
mysql> SHOW VOLUME STATUS;
+---------------+-------+
| Variable_name | Value |
+---------------+-------+
| Disks         | 96    |
| Nodes         | 74    |
```

```
+----------------+--------+
```

Ottimizzazione di Aurora MySQL

Gli eventi di attesa e gli stati di thread sono un importante strumento di ottimizzazione per Aurora MySQL. Se si riesce a scoprire perché le sessioni stanno aspettando risorse e cosa stanno facendo, è possibile ridurre meglio i colli di bottiglia. È possibile utilizzare le informazioni contenute in questa sezione per trovare possibili cause e azioni correttive.

Amazon DevOps Guru per RDS è in grado di determinare in modo proattivo se i database Aurora MySQL presentano condizioni problematiche che potrebbero causare problemi più seri in seguito. Amazon DevOps Guru per RDS pubblica una spiegazione e i suggerimenti per le azioni correttive negli approfondimenti proattivi. Questa sezione contiene gli approfondimenti per i problemi più comuni.

> **Important**
>
> Gli eventi di attesa e gli stati del thread in questa sezione sono specifici di Aurora MySQL. Utilizza le informazioni contenute in questa sezione per regolare solo Amazon Aurora, non Amazon RDS for MySQL.
> Alcuni eventi di attesa in questa sezione non hanno analoghi nelle versioni open source di questi motori di database. Altri eventi di attesa hanno gli stessi nomi degli eventi nei motori open source, ma si comportano in modo diverso. Ad esempio, l'archiviazione di Amazon Aurora funziona in modo diverso dall'archiviazione open source, pertanto gli eventi di attesa correlati all'archiviazione indicano condizioni di risorse differenti.

Argomenti

- Concetti essenziali per la regolazione di Aurora MySQL
- Regolazione di Aurora MySQL con eventi di attesa
- Regolazione di Aurora MySQL con stati del thread
- Ottimizzazione di Aurora MySQL con approfondimenti proattivi di Amazon DevOps Guru

Concetti essenziali per la regolazione di Aurora MySQL

Prima di regolare il database di Aurora MySQL, assicurati di sapere quali sono gli eventi di attesa e gli stati del thread e perché si verificano. Esaminare anche l'architettura di base della memoria e del disco di Aurora MySQL quando si utilizza il motore di archiviazione InnoDB. Per un utile diagramma architettonico, consulta ilManuale di riferimento MySQL.

Argomenti

- Eventi di attesa di Aurora MySQL
- Stati del thread di Aurora MySQL
- Memoria di Aurora MySQL
- Procedure di Aurora MySQL

Eventi di attesa di Aurora MySQL

Un evento di attesa indica una risorsa per la quale è in attesa una sessione. Ad esempio, l'evento di attesa `io/socket/sql/client_connection` indica che un thread sta per gestire una nuova connessione. Le risorse tipiche che una sessione attende includono quanto segue:

- Accesso a thread singolo a un buffer, ad esempio, quando una sessione sta tentando di modificare un buffer
- Una riga attualmente bloccata da un'altra sessione
- Un file di dati letto
- Scrittura di un file log

Ad esempio, per soddisfare una query, la sessione potrebbe eseguire una scansione completa della tabella. Se i dati non sono già in memoria, la sessione attende il completamento dell'I/O del disco. Quando i buffer vengono letti in memoria, potrebbe essere necessario attendere perché altre sessioni accedono agli stessi buffer. Il database registra le attese utilizzando un evento di attesa predefinito. Questi eventi sono raggruppati in categorie.

Un evento di attesa non mostra di per sé un problema di prestazioni. Ad esempio, se i dati richiesti non sono in memoria, è necessaria la lettura dei dati dal disco. Se una sessione blocca una riga per un aggiornamento, un'altra sessione attende che la riga venga sbloccata in modo che possa aggiornarla. Un commit richiede di attendere il completamento della scrittura su un file di registro. Le attese sono parte integrante del normale funzionamento di un database.

Un gran numero di eventi di attesa in genere mostrano un problema di prestazioni. In questi casi, è possibile utilizzare i dati degli eventi di attesa per determinare dove stanno trascorrendo il tempo delle sessioni. Ad esempio, se un report che in genere viene eseguito in minuti ora viene eseguito per ore, è possibile identificare gli eventi di attesa che contribuiscono maggiormente al tempo di attesa totale. Se è possibile determinare le cause degli eventi di attesa principali, a volte è possibile apportare

modifiche che migliorano le prestazioni. Ad esempio, se la sessione è in attesa su una riga bloccata da un'altra sessione, è possibile terminare la sessione di blocco.

Stati del thread di Aurora MySQL

Uno stato generale del thread è un valore `State` associato all'elaborazione generale delle query. Ad esempio, lo stato del thread `sending data` indica che un thread sta leggendo e filtrando le righe per una query per determinare il set di risultati corretto.

Puoi usare gli stati del thread per regoalre Aurora MySQL in modo simile a come usi gli eventi di attesa. Ad esempio, frequenti occorrenze di `sending data` di solito indicano che una query non utilizza un indice. Per ulteriori informazioni sugli stati del thread, consulta Stati generali del thread nel Manuale di riferimento di MySQL.

Quando si utilizza Performance Insights, è vera una delle seguenti condizioni:

- Performance Schema è attivato: Aurora MySQL mostra gli eventi di attesa anziché lo stato del thread.
- Performance Schema non è attivato: Aurora MySQL mostra lo stato del thread.

Consigliamo di configurare il Performance Schema per la gestione automatica. Il Performance Schema fornisce ulteriori informazioni dettagliate e strumenti migliori per indagare sui potenziali problemi di prestazioni. Per ulteriori informazioni, consulta Panoramica dello schema di prestazioni per Approfondimenti sulle prestazioni su Aurora MySQL.

Memoria di Aurora MySQL

In Aurora MySQL, le aree di memoria più importanti sono il pool di buffer e il buffer di registro.

Argomenti

- Pool di buffer

Pool di buffer

Il pool di buffer è l'area di memoria condivisa in cui Aurora MySQL memorizza nella cache i dati di tabella e indice. Le query possono accedere ai dati utilizzati di frequente direttamente dalla memoria senza leggere dal disco.

Il pool di buffer è strutturato come un elenco di pagine collegato. Una pagina può contenere più righe. Aurora MySQL utilizza un algoritmo utilizzato meno di recente (LRU) per invecchiare le pagine fuori dal pool.

Per ulteriori informazioni, consulta Buffer Pool (Pool di buffer) nel Manuale di riferimento di MySQL.

Procedure di Aurora MySQL

Aurora MySQL utilizza un modello di procedure molto diverso da Aurora PostgreSQL.

Argomenti

- Server MySQL (mysqld)
- Thread
- Pool di thread

Server MySQL (mysqld)

Il server MySQL è una procedura singola del sistema operativo denominato mysqld. Il server MySQL non genera procedure aggiuntive. Pertanto, un database di Aurora MySQL utilizza mysqld per eseguire la maggior parte del suo lavoro.

Quando si avvia il server MySQL, ascolta le connessioni di rete dai client MySQL. Quando un client si connette al database, mysqld apre un thread.

Thread

I thread del gestore delle connessioni associano ogni connessione client a un thread dedicato. Questo thread gestisce l'autenticazione, esegue istruzioni e restituisce i risultati al client. Il gestore delle connessioni crea nuovi thread quando necessario.

La cache dei thread è il set di thread disponibili. Quando una connessione termina, MySQL restituisce il thread alla cache del thread se la cache non è piena. La variabile del sistema `thread_cache_size` determina la dimensione della cache del thread.

Pool di thread

Il pool di thread è costituito da un certo numero di gruppi di thread. Ogni gruppo gestisce una serie di connessioni client. Quando un client si connette al database, il pool di thread assegna le connessioni

ai gruppi di thread in modo round-robin. Il pool di thread separa connessioni e thread. Non esiste una relazione fissa tra le connessioni e i thread che eseguono istruzioni ricevute da tali connessioni.

Regolazione di Aurora MySQL con eventi di attesa

La tabella seguente riassume gli eventi di attesa di Aurora MySQL che più comunemente indicano problemi di prestazioni. I seguenti eventi di attesa sono un sottoinsieme dell'elenco in Eventi di attesa Aurora MySQL.

Evento di attesa	Description
cpu	Questo evento si verifica quando un thread è attivo nella CPU o è in attesa della CPU.
io/aurora_redo_log_flush	Questo evento si verifica quando una sessione sta scrivendo dati permanenti nell'arch iviazione di Aurora.
io/aurora_respond_to_client	Questo evento si verifica quando un thread è in attesa di restituire una serie di risultati a un client.
io/redo_log_flush	Questo evento si verifica quando una sessione sta scrivendo dati permanenti nell'arch iviazione di Aurora.
io/socket/sql/client_connection	Questo evento si verifica quando un thread sta per gestire una nuova connessione.
io/table/sql/handler	Questo evento si verifica quando il lavoro è stato delegato a un motore di archiviazione.
synch/cond/innodb/row_lock_wait	Questo evento si verifica quando una sessione ha bloccato una riga per un aggiornamento e un'altra sessione tenta di aggiornare la stessa riga.
synch/cond/innodb/row_lock_wait_cond	Questo evento si verifica quando una sessione ha bloccato una riga per un aggiornamento e

Evento di attesa	Description
	un'altra sessione tenta di aggiornare la stessa riga.
synch/cond/sql/MDL_context::COND_wait_status	Questo evento si verifica quando ci sono thread in attesa del blocco dei metadati di una tabella.
synch/mutex/innodb/aurora_lock_thread_slot_futex	Questo evento si verifica quando una sessione ha bloccato una riga per un aggiornamento e un'altra sessione tenta di aggiornare la stessa riga.
synch/mutex/innodb/buf_pool_mutex	Questo evento si verifica quando un thread ha acquisito un blocco nel pool di buffer InnoDB per accedere a una pagina in memoria.
synch/mutex/innodb/fil_system_mutex	Questo evento si verifica quando una sessione è in attesa di accedere alla cache di memoria del tablespace.
synch/mutex/innodb/trx_sys_mutex	Questo evento si verifica quando c'è un'elevata attività del database con un numero elevato di transazioni.
synch/sxlock/innodb/hash_table_locks	Questo evento si verifica quando le pagine non trovate nel pool del buffer devono essere lette da un file.
synch/mutex/innodb/temp_pool_manager_mutex	Questo evento si verifica quando una sessione è in attesa di acquisire un mutex per la gestione del pool di tablespace temporanee della sessione.

cpu

L'evento di attesa `cpu` si verifica quando un thread è attivo nella CPU o è in attesa della CPU.

Argomenti

- Versioni del motore supportate
- Context
- Probabili cause di aumento delle attese
- Azioni

Versioni del motore supportate

Queste informazioni sull'evento di attesa sono supportate per le seguenti versioni del motore:

- Aurora MySQL versioni 2 e 3

Context

Per ogni vCPU, una connessione può eseguire operazioni su questa CPU. In alcune situazioni, il numero di connessioni attive pronte per l'esecuzione è superiore al numero di vCPUs. Questo squilibrio provoca connessioni in attesa di risorse della CPU. Se il numero di connessioni attive rimane costantemente superiore al numero di vCPU, l'istanza subisce un conflitto della CPU. La contesa provoca il verificarsi dell'evento di attesa `cpu`.

> **Note**
>
> Il parametro di Performance Insights per la CPU è `DBLoadCPU`. Il valore per `DBLoadCPU` può differire dal valore del parametro CloudWatch `CPUUtilization`. Quest'ultimo parametro viene raccolta dall'HyperVisor per un'istanza del database.

I parametri del sistema operativo del Performance Insights forniscono informazioni dettagliate sull'utilizzo della CPU. Ad esempio, puoi visualizzare i seguenti parametri:

- `os.cpuUtilization.nice.avg`
- `os.cpuUtilization.total.avg`
- `os.cpuUtilization.wait.avg`
- `os.cpuUtilization.idle.avg`

Performance Insights segnala l'utilizzo della CPU da parte del motore del database come `os.cpuUtilization.nice.avg`.

Probabili cause di aumento delle attese

Quando questo evento si verifica più del normale, eventualmente indicando un problema di prestazioni, le cause tipiche includono le seguenti:

- Query analitiche
- Transazioni altamente simultanee
- Transazioni di lunga durata
- Un improvviso aumento del numero di connessioni, noto come storm di login
- Un aumento del cambio di contesto

Azioni

Se l'evento di attesa `cpu` domina l'attività del database, non indica necessariamente un problema di prestazioni. Rispondi a questo evento solo quando le prestazioni diminuiscono.

A seconda della causa dell'aumento dell'utilizzo della CPU, considerare le seguenti strategie:

- Aumenta la capacità della CPU dell'host. Questo approccio tipicamente offre un sollievo temporaneo.
- Identifica le principali query per una potenziale ottimizzazione.
- Reindirizza alcuni carichi di lavoro di sola lettura verso i nodi del lettore, se applicabile.

Argomenti

- Identifica le sessioni o le query che causano il problema
- Analizza e ottimizza l'elevato carico di lavoro della CPU

Identifica le sessioni o le query che causano il problema

Per trovare le sessioni e le query, guarda la tabella Prime istruzioni SQL in Performance Insights per le istruzioni SQL che hanno il carico più alto della CPU. Per ulteriori informazioni, consulta Analisi delle metriche utilizzando il pannello di controllo Performance Insights.

In genere, una o due istruzioni SQL consumano la maggior parte dei cicli della CPU. Concentra gli sforzi su queste istruzioni. Supponiamo che l'istanza database disponga di 2 vCPU con un carico del database di 3.1 sessioni attive medie (AAS), tutte nello stato della CPU. In questo caso, la tua istanza è associata alla CPU. Considera le strategie seguenti:

- Esegui l'aggiornamento a una classe di istanza più grande con più vCPU.
- Regola le query per ridurre il carico della CPU.

In questo esempio, le query SQL principali hanno un carico del database di 1,5 AAS, il tutto nello stato della CPU. Un'altra istruzione SQL ha un carico di 0,1 nello stato della CPU. In questo esempio, se si è interrotta l'istruzione SQL con caricamento più basso, non si riduce in modo significativo il carico del database. Tuttavia, se si regolano le due query ad alto carico in modo che siano due volte più efficienti, si elimina il collo di bottiglia della CPU. Se si riduce il carico della CPU di 1,5 AAS del 50%, l'AAS per ogni istruzione diminuisce a 0,75. Il carico totale del databalse speso per la CPU è ora di 1,6 AAS. Questo valore è inferiore alla linea massima di vCPU 2.0.

Per una panoramica utile dell'identificazione e della risoluzione dei problemi con Performance Insights, consulta il post del blogAnalizza i carichi di lavoro di Amazon Aurora MySQL con Performance Insights. Consulta anche l'AWSarticolo di assistenza Come posso identificare e risolvere un elevato utilizzo della CPU sulle istanze di Amazon RDS for MySQL?.

Analizza e ottimizza l'elevato carico di lavoro della CPU

Dopo aver identificato la query o le query che aumentano l'utilizzo della CPU, è possibile ottimizzarle o terminare la connessione. L'esempio seguente mostra come terminare una connessione.

```
CALL mysql.rds_kill(processID);
```

Per ulteriori informazioni, consulta mysql.rds_kill.

Se si termina una sessione, l'azione potrebbe innescare un lungo ripristino dello stato precedente.

Seguire le linee guida per l'ottimizzazione delle query

Per ottimizzare le query, prendere in considerazione le seguenti linee guida:

- Eseguire l'istruzione `EXPLAIN`.

  Questo comando mostra i singoli passaggi coinvolti nell'esecuzione di una query. Per ulteriori informazioni, consulta Ottimizzazione delle query con EXPLAIN nella documentazione di MySQL.

- Eseguire l'istruzione `SHOW PROFILE`.

  Utilizzare questa istruzione per esaminare i dettagli del profilo che possono indicare l'utilizzo delle risorse per le istruzioni eseguite durante la sessione corrente. Per ulteriori informazioni, consulta Istruzione REPAIR TABLE nella documentazione di MySQL.

- Eseguire l'istruzione `ANALYZE TABLE`.

  Utilizzare questa istruzione per riaggiornare le statistiche di indice per le tabelle a cui si accede dalla query che consuma CPU elevata. Analizzando l'istruzione, è possibile aiutare l'ottimizzatore a scegliere un piano di esecuzione appropriato. Per ulteriori informazioni, consulta Istruzione ANALYZE TABLE nella documentazione di MySQL.

Segui le linee guida per migliorare l'utilizzo della CPU

Per migliorare l'utilizzo della CPU in un'istanza database, attenersi a queste linee guida:

- Assicurarsi che tutte le query utilizzino indici appropriati.
- Scopri se è possibile utilizzare query parallele di Aurora. Si può usare questa tecnica per ridurre l'utilizzo della CPU sul nodo head grazie all'elaborazione della funzione di trasferimento, al filtraggio delle righe e alla proiezione delle colonne per la clausola `WHERE`.
- Scopri se il numero di esecuzioni SQL al secondo soddisfa le soglie previste.
- Scopri se la manutenzione dell'indice o la creazione di nuovi indici occupano i cicli della CPU necessari per il carico di lavoro di produzione. Pianifica le attività di manutenzione al di fuori degli orari del picco di attività.
- Scopri se è possibile utilizzare il partizionamento per ridurre la serie dei dati delle query. Per ulteriori informazioni, consultare il post del blog.Come pianificare e ottimizzare Amazon Aurora con compatibilità MySQL per carichi di lavoro consolidati.

Verifica la presenza di congestioni dovute alla connessione

Se il parametro `DBLoadCPU` non è molto alto, ma il parametro `CPUUtilization` è elevato, la causa dell'utilizzo elevato della CPU risiede al di fuori del motore del database. Un esempio classico è una congestione dovuta alla connessione.

Verificare se le condizioni seguenti sono vere:

- C'è un aumento sia del parametro `CPUUtilization` di Performance Insights che del parametro `DatabaseConnections` di Amazon CloudWatch.
- Il numero di thread nella CPU è superiore al numero di vCPU.

Se le condizioni precedenti sono vere, considerare di diminuire il numero di connessioni al database. Ad esempio, è possibile utilizzare un pool di connessione come RDS Proxy. Per conoscere le best practice per una gestione efficace delle connessioni e il dimensionamento, consulta il whitepaper Manuale di Amazon Aurora MySQL DBA per la gestione delle connessioni.

io/aurora_redo_log_flush

L'evento `io/aurora_redo_log_flush` si verifica quando una sessione sta scrivendo dati persistenti nell'archiviazione di Amazon Aurora.

Argomenti

- Versioni del motore supportate
- Contesto
- Probabili cause di aumento delle attese
- Azioni

Versioni del motore supportate

Queste informazioni sull'evento di attesa sono supportate per le seguenti versioni del motore:

- Aurora MySQL versione 2

Contesto

L'evento `io/aurora_redo_log_flush` è per un'operazione di input/output di scrittura (I/O) in Aurora MySQL.

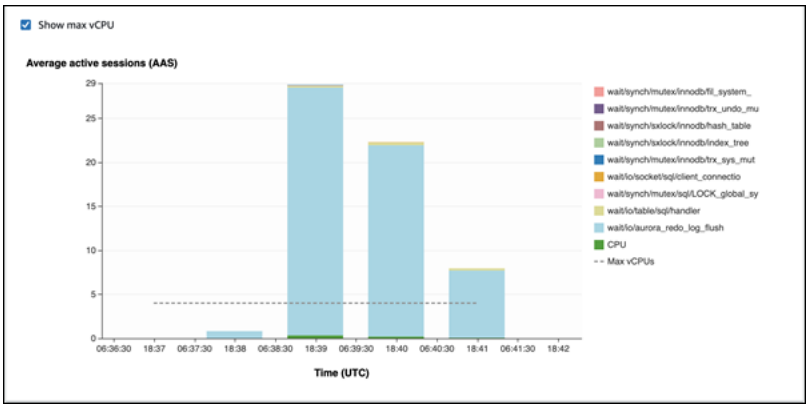
> **Note**
>
> Nella versione 3 di Aurora MySQL, questo evento di attesa è denominato io/redo_log_flush.

Probabili cause di aumento delle attese

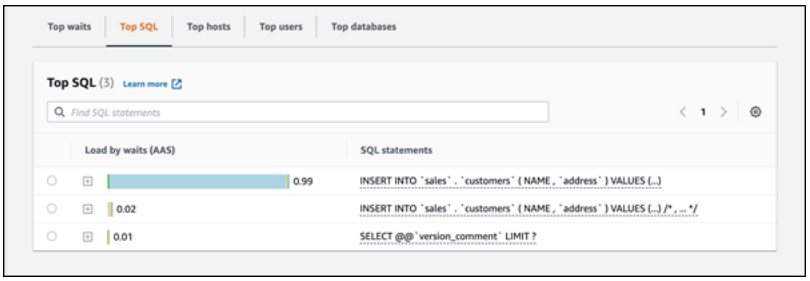
Per la persistenza dei dati, l'esecuzione dei commit richiede una scrittura duratura per un'archiviazione stabile. Se il database sta eseguendo troppi commit, c'è un evento di attesa sull'operazione di I/O di scrittura, l'evento di attesa `io/aurora_redo_log_flush`.

Negli esempi seguenti, 50.000 dati vengono inseriti in un cluster di database di Aurora MySQL utilizzando la classe di istanza database db.r5.xlarge:

- Nel primo esempio, ogni sessione inserisce 10.000 dati riga per riga. Per impostazione predefinita, se un comando del linguaggio di manipolazione dei dati (DML) non si trova all'interno di una transazione, Aurora MySQL utilizza l'esecuzione di commit impliciti. Autocommit è attivo. Ciò significa che per ogni inserimento di riga è presente l'esecuzione di un commit. Performance Insights mostra che le connessioni dedicano la maggior parte del tempo in attesa dell'evento di attesa `io/aurora_redo_log_flush`.

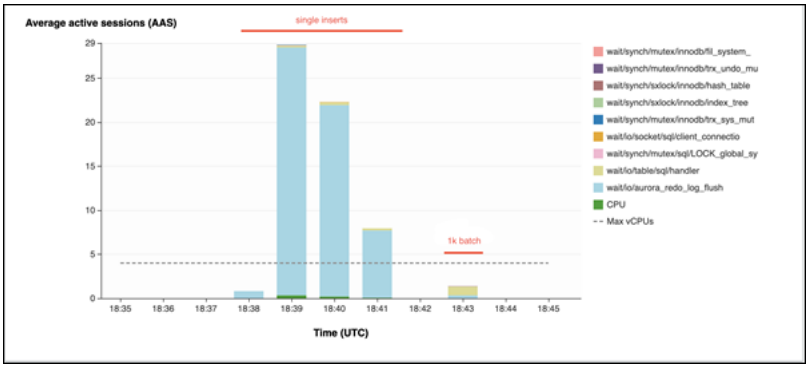


  Ciò è causato dalle semplici istruzioni di inserimento utilizzate.



  I 50.000 dati richiedono 3,5 minuti per essere inseriti.

- Nel secondo esempio, gli inserti sono realizzati in 1.000 batch, ovvero ogni connessione esegue 10 commit anziché 10.000. Performance Insights mostra che le connessioni non dedicano la maggior parte del loro tempo sull'evento di attesa `io/aurora_redo_log_flush`.



I 50.000 dati richiedono 4 secondi per essere inseriti.

Azioni

Consigliamo azioni diverse a seconda delle cause dell'evento di attesa.

Argomenti

- Identificare le sessioni e le query problematiche
- Raggruppa le operazioni di scrittura
- Disattivazione dell'autocommit
- Transazioni di utilizzo
- Utilizza i batch

Identificare le sessioni e le query problematiche

Se l'istanza database sta riscontrando un collo di bottiglia, il primo compito è quello di trovare le sessioni e le query che lo causano. Per un utile Post del blog sul database AWS, consulta Analizza i carichi di lavoro di Amazon Aurora MySQL con Performance Insights.

Per identificare sessioni e query che causano un collo di bottiglia

1. Accedi alla Console di gestione AWS e apri la console Amazon RDS all'indirizzo https://console.aws.amazon.com/rds/.
2. Nel pannello di navigazione scegli Approfondimenti sulle prestazioni.
3. Scegli l'istanza database.
4. In Carico del database, scegli Dividi per attesa.
5. Nella parte inferiore della pagina scegli Prime Instruzioni SQL.

Le query nella parte superiore dell'elenco causano il carico più alto sul database.

Raggruppa le operazioni di scrittura

Gli esempi seguenti attivano l'evento di attesa `io/aurora_redo_log_flush`. (Autocommit è attivo.)

```
INSERT INTO `sampleDB`.`sampleTable` (sampleCol2, sampleCol3) VALUES ('xxxx','xxxxx');
INSERT INTO `sampleDB`.`sampleTable` (sampleCol2, sampleCol3) VALUES ('xxxx','xxxxx');
INSERT INTO `sampleDB`.`sampleTable` (sampleCol2, sampleCol3) VALUES ('xxxx','xxxxx');
....
INSERT INTO `sampleDB`.`sampleTable` (sampleCol2, sampleCol3) VALUES ('xxxx','xxxxx');

UPDATE `sampleDB`.`sampleTable` SET sampleCol3='xxxxx' WHERE id=xx;
UPDATE `sampleDB`.`sampleTable` SET sampleCol3='xxxxx' WHERE id=xx;
UPDATE `sampleDB`.`sampleTable` SET sampleCol3='xxxxx' WHERE id=xx;
....
UPDATE `sampleDB`.`sampleTable` SET sampleCol3='xxxxx' WHERE id=xx;

DELETE FROM `sampleDB`.`sampleTable` WHERE sampleCol1=xx;
DELETE FROM `sampleDB`.`sampleTable` WHERE sampleCol1=xx;
DELETE FROM `sampleDB`.`sampleTable` WHERE sampleCol1=xx;
....
DELETE FROM `sampleDB`.`sampleTable` WHERE sampleCol1=xx;
```

Per ridurre il tempo impiegato in attesa dell'evento di attesa `io/aurora_redo_log_flush`, raggruppa logicamente le operazioni di scrittura in una singola esecuzione di commit per ridurre le chiamate persistenti all'archiviazione.

Disattivazione dell'autocommit

Disattivare l'autocommit prima di apportare modifiche di grandi dimensioni che non si trovano all'interno di una transazione, come illustrato nell'esempio seguente.

```
SET SESSION AUTOCOMMIT=OFF;
UPDATE `sampleDB`.`sampleTable` SET sampleCol3='xxxxx' WHERE sampleCol1=xx;
UPDATE `sampleDB`.`sampleTable` SET sampleCol3='xxxxx' WHERE sampleCol1=xx;
UPDATE `sampleDB`.`sampleTable` SET sampleCol3='xxxxx' WHERE sampleCol1=xx;
....
UPDATE `sampleDB`.`sampleTable` SET sampleCol3='xxxxx' WHERE sampleCol1=xx;
-- Other DML statements here
```

```
COMMIT;

SET SESSION AUTOCOMMIT=ON;
```

Transazioni di utilizzo

È possibile utilizzare le transazioni, come illustrato nell'esempio seguente.

```
BEGIN
INSERT INTO `sampleDB`.`sampleTable` (sampleCol2, sampleCol3) VALUES ('xxxx','xxxxx');
INSERT INTO `sampleDB`.`sampleTable` (sampleCol2, sampleCol3) VALUES ('xxxx','xxxxx');
INSERT INTO `sampleDB`.`sampleTable` (sampleCol2, sampleCol3) VALUES ('xxxx','xxxxx');
....
INSERT INTO `sampleDB`.`sampleTable` (sampleCol2, sampleCol3) VALUES ('xxxx','xxxxx');

DELETE FROM `sampleDB`.`sampleTable` WHERE sampleCol1=xx;
DELETE FROM `sampleDB`.`sampleTable` WHERE sampleCol1=xx;
DELETE FROM `sampleDB`.`sampleTable` WHERE sampleCol1=xx;
....
DELETE FROM `sampleDB`.`sampleTable` WHERE sampleCol1=xx;

-- Other DML statements here
END
```

Utilizza i batch

Si può anche apportare modifiche in batch, come nell'esempio seguente: Tuttavia, l'utilizzo di batch troppo grandi può causare problemi di prestazioni, specialmente nelle repliche di lettura o durante il recupero point-in-time (PITR).

```
INSERT INTO `sampleDB`.`sampleTable` (sampleCol2, sampleCol3) VALUES
('xxxx','xxxxx'),('xxxx','xxxxx'),...,('xxxx','xxxxx'),('xxxx','xxxxx');

UPDATE `sampleDB`.`sampleTable` SET sampleCol3='xxxxx' WHERE sampleCol1 BETWEEN xx AND
 xxx;

DELETE FROM `sampleDB`.`sampleTable` WHERE sampleCol1<xx;
```

io/aurora_respond_to_client

L'evento `io/aurora_respond_to_client` si verifica quando un thread è in attesa di restituire una serie di risultati a un client.

Argomenti

- Versioni del motore supportate
- Context
- Probabili cause di aumento delle attese
- Azioni

Versioni del motore supportate

Queste informazioni sull'evento di attesa sono supportate per le seguenti versioni del motore:

- Aurora MySQL versione 2

Context

L'evento `io/aurora_respond_to_client` indica che un thread è in attesa di restituire una serie di risultati a un client.

L'elaborazione della query è completa e i risultati vengono restituiti al client dell'applicazione. Tuttavia, poiché non c'è abbastanza larghezza di banda di rete nel cluster del database, un thread è in attesa di restituire la serie di risultati.

Probabili cause di aumento delle attese

Quando l'evento `io/aurora_respond_to_client` appare più che normale, possibilmente indicando un problema di prestazioni, le cause tipiche includono le seguenti:

Classe di istanza database insufficiente per il carico di lavoro

: La classe di istanza database utilizzata dal cluster del database non dispone della larghezza di banda di rete necessaria per elaborare il carico di lavoro in modo efficiente.

Serie di risultati di grandi dimensioni

: Si è verificato un aumento delle dimensioni della serie di risultati restituiti, poiché la query restituisce un numero maggiore di righe. La serie di risultati più ampia consuma una maggiore larghezza di banda di rete.

Aumento del carico sul client

Potrebbe esserci pressione della CPU, pressione della memoria o saturazione della rete sul client. Un aumento del carico sul client ritarda la ricezione dei dati dal cluster del database di Aurora MySQL.

Maggiore latenza di rete

Potrebbe esserci una maggiore latenza di rete tra il cluster del databalse di Aurora MySQL e il client. Una maggiore latenza di rete aumenta il tempo necessario per il client per la ricezione dei dati.

Azioni

Consigliamo azioni diverse a seconda delle cause dell'evento di attesa.

Argomenti

- Identificare le sessioni e le query che causano gli eventi
- Dimensionamento della classe dell'istanza database
- Verifica del carico di lavoro per risultati imprevisti
- Distribuisci il carico di lavoro con le istanze del lettore
- Utilizzare il modificatore SQL_BUFFER_RESULT

Identificare le sessioni e le query che causano gli eventi

È possibile utilizzare Performance Insights per mostrare le query bloccate dall'evento di attesa `io/aurora_respond_to_client`. In genere, i database con carico da moderato a significativo hanno eventi di attesa. Gli eventi di attesa possono essere accettabili se le prestazioni sono ottimali. Se le prestazioni non sono ottimali, esaminare dove il database impiega più tempo. Considerare gli eventi di attesa che contribuiscono al carico più elevato per scoprire se è possibile ottimizzare il database e l'applicazione per ridurre tali eventi.

Per trovare query di SQL responsabili del carico elevato

1. Accedi alla Console di gestione AWS e apri la console Amazon RDS all'indirizzo https://console.aws.amazon.com/rds/.
2. Nel pannello di navigazione scegli Approfondimenti sulle prestazioni.

3. Scegli un'istanza database. Viene visualizzato il pannello di controllo di Approfondimenti sulle prestazioni per l'istanza database.
4. Nel grafico Carico del database, scegli Dividi per attesa.
5. Nella parte inferiore della pagina scegli Prime Istruzioni SQL.

   Il grafico elenca le query di SQL responsabili del carico. Quelli in cima all'elenco sono le più responsabili. Per risolvere un collo di bottiglia, occorre concentrarsi su queste istruzioni.

Per una panoramica utile dell'identificazione e della risoluzione dei problemi con Performance Insights, consulta il post del blog AWS Analizza i carichi di lavoro di Amazon Aurora MySQL con Performance Insights.

Dimensionamento della classe dell'istanza database

Verifica l'aumento del valore dei parametri di Amazon CloudWatch relativi alla velocità effettiva di rete, come ad esempio`NetworkReceiveThroughput` e `NetworkTransmitThroughput`. Se viene raggiunta la larghezza di banda di rete della classe di istanza database, è possibile dimensionare la classe di istanza database utilizzata dal cluster del database modificando il cluster del database. Una classe di istanze database con larghezza di banda di rete maggiore restituisce i dati ai client in modo più efficiente.

Per maggiori informazioni sul monitoraggio dei parametri di Amazon CloudWatch, consulta Visualizzazione delle metriche nella console Amazon RDS. Per informazioni sulle classi di istanza database, consulta Classi di istanze database Amazon Aurora. Per ulteriori informazioni sulla modifica di un cluster di database, consulta Modifica di un cluster database Amazon Aurora.

Verifica del carico di lavoro per risultati imprevisti

Controlla il carico di lavoro sul cluster del database e assicurati che non produca risultati imprevisti. Ad esempio, potrebbero esserci query che restituiscono un numero di righe più alto del previsto. In questo caso, puoi utilizzare i parametri del contatore di Performance Insights come `Innodb_rows_read`. Per ulteriori informazioni, consulta Metriche contatore di Performance Insights.

Distribuisci il carico di lavoro con le istanze del lettore

È possibile distribuire il carico di lavoro di sola lettura con le repliche di Aurora. È possibile dimensionare orizzontalmente aggiungendo più repliche di Aurora. Ciò può comportare un aumento dei limiti per la larghezza di banda della rete. Per ulteriori informazioni, consulta Cluster database Amazon Aurora.

Utilizzare il modificatore SQL_BUFFER_RESULT

Puoi aggiungere il modificatore `SQL_BUFFER_RESULT` alle istruzioni `SELECT` per forzare il risultato in una tabella temporanea prima che vengano restituite al client. Questo modificatore può aiutare a risolvere i problemi di prestazioni quando i blocchi InnoDB non vengono liberati perché le query sono presenti nello stato di attesa `io/aurora_respond_to_client`. Per ulteriori informazioni, consulta Istruzione SELECT nella documentazione di MySQL.

io/redo_log_flush

L'evento `io/redo_log_flush` si verifica quando una sessione sta scrivendo dati persistenti nell'archiviazione di Amazon Aurora.

Argomenti

- Versioni del motore supportate
- Contesto
- Probabili cause di aumento delle attese
- Azioni

Versioni del motore supportate

Queste informazioni sull'evento di attesa sono supportate per le seguenti versioni del motore:

- Aurora MySQL versione 3

Contesto

L'evento `io/redo_log_flush` è per un'operazione di input/output di scrittura (I/O) in Aurora MySQL.

> **Note**
>
> In Aurora MySQL versione 2, questo evento di attesa è denominato io/aurora_redo_log_flush.

Probabili cause di aumento delle attese

Per la persistenza dei dati, l'esecuzione dei commit richiede una scrittura duratura per un'archiviazione stabile. Se il database sta eseguendo troppi commit, c'è un evento di attesa sull'operazione di I/O di scrittura, l'evento di attesa `io/redo_log_flush`.

Per esempi del comportamento di questo evento di attesa, consulta io/aurora_redo_log_flush.

Azioni

Consigliamo azioni diverse a seconda delle cause dell'evento di attesa.

Argomenti

- Identificare le sessioni e le query problematiche
- Raggruppa le operazioni di scrittura
- Disattivazione dell'autocommit
- Transazioni di utilizzo
- Utilizza i batch

Identificare le sessioni e le query problematiche

Se l'istanza database sta riscontrando un collo di bottiglia, il primo compito è quello di trovare le sessioni e le query che lo causano. Per un utile Post del blog sul database AWS, consulta Analizza i carichi di lavoro di Amazon Aurora MySQL con Performance Insights.

Per identificare sessioni e query che causano un collo di bottiglia

1. Accedi alla Console di gestione AWS e apri la console Amazon RDS all'indirizzo https://console.aws.amazon.com/rds/.
2. Nel pannello di navigazione scegli Approfondimenti sulle prestazioni.
3. Scegli l'istanza database.
4. In Carico del database, scegli Dividi per attesa.
5. Nella parte inferiore della pagina scegli Prime Instruzioni SQL.

   Le query nella parte superiore dell'elenco causano il carico più alto sul database.

Raggruppa le operazioni di scrittura

Gli esempi seguenti attivano l'evento di attesa `io/redo_log_flush`. (Autocommit è attivo.)

```
INSERT INTO `sampleDB`.`sampleTable` (sampleCol2, sampleCol3) VALUES ('xxxx','xxxxx');
INSERT INTO `sampleDB`.`sampleTable` (sampleCol2, sampleCol3) VALUES ('xxxx','xxxxx');
INSERT INTO `sampleDB`.`sampleTable` (sampleCol2, sampleCol3) VALUES ('xxxx','xxxxx');
....
INSERT INTO `sampleDB`.`sampleTable` (sampleCol2, sampleCol3) VALUES ('xxxx','xxxxx');

UPDATE `sampleDB`.`sampleTable` SET sampleCol3='xxxxx' WHERE id=xx;
UPDATE `sampleDB`.`sampleTable` SET sampleCol3='xxxxx' WHERE id=xx;
UPDATE `sampleDB`.`sampleTable` SET sampleCol3='xxxxx' WHERE id=xx;
....
UPDATE `sampleDB`.`sampleTable` SET sampleCol3='xxxxx' WHERE id=xx;

DELETE FROM `sampleDB`.`sampleTable` WHERE sampleCol1=xx;
DELETE FROM `sampleDB`.`sampleTable` WHERE sampleCol1=xx;
DELETE FROM `sampleDB`.`sampleTable` WHERE sampleCol1=xx;
....
DELETE FROM `sampleDB`.`sampleTable` WHERE sampleCol1=xx;
```

Per ridurre il tempo impiegato in attesa dell'evento di attesa `io/redo_log_flush`, raggruppa logicamente le operazioni di scrittura in una singola esecuzione di commit per ridurre le chiamate persistenti all'archiviazione.

Disattivazione dell'autocommit

Disattivare l'autocommit prima di apportare modifiche di grandi dimensioni che non si trovano all'interno di una transazione, come illustrato nell'esempio seguente.

```
SET SESSION AUTOCOMMIT=OFF;
UPDATE `sampleDB`.`sampleTable` SET sampleCol3='xxxxx' WHERE sampleCol1=xx;
UPDATE `sampleDB`.`sampleTable` SET sampleCol3='xxxxx' WHERE sampleCol1=xx;
UPDATE `sampleDB`.`sampleTable` SET sampleCol3='xxxxx' WHERE sampleCol1=xx;
....
UPDATE `sampleDB`.`sampleTable` SET sampleCol3='xxxxx' WHERE sampleCol1=xx;
-- Other DML statements here
COMMIT;

SET SESSION AUTOCOMMIT=ON;
```

Transazioni di utilizzo

È possibile utilizzare le transazioni, come illustrato nell'esempio seguente.

```
BEGIN
INSERT INTO `sampleDB`.`sampleTable` (sampleCol2, sampleCol3) VALUES ('xxxx','xxxxx');
INSERT INTO `sampleDB`.`sampleTable` (sampleCol2, sampleCol3) VALUES ('xxxx','xxxxx');
INSERT INTO `sampleDB`.`sampleTable` (sampleCol2, sampleCol3) VALUES ('xxxx','xxxxx');
....
INSERT INTO `sampleDB`.`sampleTable` (sampleCol2, sampleCol3) VALUES ('xxxx','xxxxx');

DELETE FROM `sampleDB`.`sampleTable` WHERE sampleCol1=xx;
DELETE FROM `sampleDB`.`sampleTable` WHERE sampleCol1=xx;
DELETE FROM `sampleDB`.`sampleTable` WHERE sampleCol1=xx;
....
DELETE FROM `sampleDB`.`sampleTable` WHERE sampleCol1=xx;

-- Other DML statements here
END
```

Utilizza i batch

Si può anche apportare modifiche in batch, come nell'esempio seguente: Tuttavia, l'utilizzo di batch troppo grandi può causare problemi di prestazioni, specialmente nelle repliche di lettura o durante il recupero point-in-time (PITR).

```
INSERT INTO `sampleDB`.`sampleTable` (sampleCol2, sampleCol3) VALUES
('xxxx','xxxxx'),('xxxx','xxxxx'),...,('xxxx','xxxxx'),('xxxx','xxxxx');

UPDATE `sampleDB`.`sampleTable` SET sampleCol3='xxxxx' WHERE sampleCol1 BETWEEN xx AND
 xxx;

DELETE FROM `sampleDB`.`sampleTable` WHERE sampleCol1<xx;
```

io/socket/sql/client_connection

L'evento `io/socket/sql/client_connection` si verifica quando un thread sta per gestire una nuova connessione.

Argomenti

- Versioni del motore supportate

- Context
- Probabili cause di aumento delle attese
- Azioni

Versioni del motore supportate

Queste informazioni sull'evento di attesa sono supportate per le seguenti versioni del motore:

- Aurora MySQL versioni 2 e 3

Context

L'evento `io/socket/sql/client_connection` indica che mysqld è impegnato a creare thread per gestire le nuove connessioni client in arrivo. In questo scenario, l'elaborazione della manutenzione delle nuove richieste di connessione client rallenta mentre le connessioni attendono l'assegnazione del thread. Per ulteriori informazioni, consulta Server MySQL (mysqld).

Probabili cause di aumento delle attese

Quando questo evento si verifica più del normale, probabilmente indicando un problema di prestazioni, le cause tipiche includono le seguenti:

- C'è un improvviso aumento delle nuove connessioni utente dall'applicazione alla istanza Amazon RDS.
- L'istanza database non è in grado di elaborare nuove connessioni perché la rete, la CPU o la memoria sono state limitate.

Azioni

Se `io/socket/sql/client_connection` domina l'attività del database, non indica necessariamente un problema di prestazioni. In un database che non è inattivo, un evento di attesa è sempre in primo piano. Agisci solo quando le prestazioni diminuiscono. Consigliamo azioni diverse a seconda delle cause dell'evento di attesa.

Argomenti

- Identificare le sessioni e le query problematiche
- Seguire le best practice per la gestione delle connessioni

- Dimensiona verso l’alto la tua istanza se le risorse vengono limitate
- Controlla i principali host e i migliori utenti
- Interrogare le tabelle performance_schema
- Controlla lo stato del thread delle query
- Verifica le tue richieste e le query
- Crea pool delle connessioni al database

Identificare le sessioni e le query problematiche

Se l'istanza database sta riscontrando un collo di bottiglia, il primo compito è quello di trovare le sessioni e le query che lo causano. Per un utile post sul blog, consulta Analizza i carichi di lavoro di Amazon Aurora MySQL con Performance Insights.

Per identificare sessioni e query che causano un collo di bottiglia

1. Accedi alla Console di gestione AWS e apri la console Amazon RDS all'indirizzo https://console.aws.amazon.com/rds/.
2. Nel pannello di navigazione scegli Approfondimenti sulle prestazioni.
3. Scegli l'istanza database.
4. In Carico del database, scegli Dividi per attesa.
5. Nella parte inferiore della pagina scegli Prime Instruzioni SQL.

   Le query nella parte superiore dell'elenco causano il carico più alto sul database.

Seguire le best practice per la gestione delle connessioni

Per gestire le tue connessioni, considera le seguenti strategie:

- Utilizza il pooling di connessioni.

  È possibile aumentare gradualmente il numero di connessioni secondo necessità. Per ulteriori informazioni, consulta il whitepaper Manuale dell'amministratore del database di Amazon Aurora MySQL.
- Usa un nodo lettore per ridistribuire il traffico di sola lettura.

  Per ulteriori informazioni, consultare Repliche di Aurora e Connessioni degli endpoint Amazon Aurora.

Dimensiona verso l'alto la tua istanza se le risorse vengono limitate

Cerca esempi di limitazione nelle seguenti risorse:

- CPU

  Controlla i parametri di Amazon CloudWatch per un elevato utilizzo della CPU.
- Rete

  Verifica un aumento del valore dei parametri di CloudWatch `network receive throughput` e `network transmit throughput`. Se l'istanza ha raggiunto il limite di larghezza di banda di rete per la classe di istanza, è consigliabile dimensionare verso l'alto l'istanza RDS a un tipo di classe di istanza superiore. Per ulteriori informazioni, consulta Classi di istanze database Amazon Aurora.
- Memoria liberabile

  Verifica la presenza di un calo nel parametro di CloudWatch `FreeableMemory`. Inoltre, prendi in considerazione l'attivazione del monitoraggio avanzato. Per ulteriori informazioni, consulta Monitoraggio dei parametri del sistema operativo con il monitoraggio avanzato.

Controlla i principali host e i migliori utenti

Usa Performance Insights per controllare i principali host e i migliori utenti. Per ulteriori informazioni, consulta Analisi delle metriche utilizzando il pannello di controllo Performance Insights.

Interrogare le tabelle performance_schema

Per ottenere un conteggio accurato delle connessioni correnti e totali, eseguire una query sulle tabelle `performance_schema`. Con questa tecnica, si identifica l'utente o l'host di origine responsabile della creazione di un numero elevato di connessioni. Ad esempio, interrogare le tabelle `performance_schema` come indicato di seguito.

```
SELECT * FROM performance_schema.accounts;
SELECT * FROM performance_schema.users;
SELECT * FROM performance_schema.hosts;
```

Controlla lo stato del thread delle query

Se il problema relativo alle prestazioni è continuo, controlla lo stato del thread delle query. Nel client `mysql`, eseguire il comando seguente.

```
show processlist;
```

Verifica le tue richieste e le query

Per verificare la natura delle richieste e delle query dagli account utente, utilizzare la verifica avanzata di AuroraAurora MySQL. Per informazioni su come attivare l'auditing, consulta Utilizzo dell'audit avanzato con un cluster di database Amazon Aurora MySQL.

Crea pool delle connessioni al database

Valuta l'utilizzo di Amazon RDS Proxy per la gestione della connessione. Con RDS Proxy, è possibile consentire alle applicazioni di creare pool delle connessioni di database e condividerle per migliorare la loro capacità di dimensionamento. RDS Proxy rende le applicazioni più resilienti agli errori del database connettendosi automaticamente a un'istanza database di standby, mantenendo al contempo le connessioni delle applicazioni. Per ulteriori informazioni, consulta Server proxy per Amazon RDS per Aurora.

io/table/sql/handler

L'evento `io/table/sql/handler` si verifica quando il lavoro è stato delegato a un motore di archiviazione.

Argomenti

- Versioni del motore supportate
- Contesto
- Probabili cause di aumento delle attese
- Azioni

Versioni del motore supportate

Queste informazioni sull'evento di attesa sono supportate per le seguenti versioni del motore:

- Aurora MySQL versioni 2 e 3

Contesto

L'evento `io/table` indica un'attesa per l'accesso a una tabella. Questo evento si verifica indipendentemente dal fatto che i dati siano memorizzati nella cache nel pool di buffer o se si acceda su disco. L'evento `io/table/sql/handler` indica un aumento dell'attività del carico di lavoro.

Un handler è una routine specializzata in un determinato tipo di dati o incentrata su determinate attività speciali. Ad esempio, un handler di eventi riceve e digerisce eventi e segnali dal sistema operativo o da un'interfaccia utente. Un handler di memoria esegue processi relativi alla memoria. Un handler di input di file è una funzione che riceve l'input di file ed esegue attività speciali sui dati, in base al contesto.

Visualizzazioni come `performance_schema.events_waits_current` spesso mostrano `io/table/sql/handler` quando l'attesa effettiva è un evento di attesa annidato come un blocco. Quando l'attesa effettiva non è `io/table/sql/handler`, Approfondimenti sulle prestazioni segnala l'evento di attesa annidato. Quando Approfondimenti sulle prestazioni segnala `io/table/sql/handler`, rappresenta l'elaborazione InnoDB della richiesta I/O e non un evento di attesa annidato nascosto. Per ulteriori informazioni consulta Performance Schema Atom and Molecule Events nel Manuale di riferimento di MySQL.

L'evento `io/table/sql/handler` appare spesso negli eventi di attesa principali con attese I/O come `io/aurora_redo_log_flush`.

Probabili cause di aumento delle attese

In Approfondimenti sulle prestazioni, picchi improvvisi nell'evento `io/table/sql/handler` indicano un aumento dell'attività del carico di lavoro. Aumento dell'attività significa un aumento dell'I/O.

Approfondimenti sulle prestazioni filtra gli ID degli eventi di nidificazione e non segnala un evento di attesa `io/table/sql/handler` quando l'evento annidato sottostante è un'attesa di blocco. Ad esempio, se l'evento causa principale è synch/mutex/innodb/aurora_lock_thread_slot_futex, Approfondimenti sulle prestazioni visualizza questa attesa nei primi eventi di attesa e non `io/table/sql/handler`.

In visualizzazioni come `performance_schema.events_waits_current`, le attese di `io/table/sql/handler` spesso appaiono quando l'attesa effettiva è un evento di attesa annidato come un blocco. Quando l'attesa effettiva è diversa da `io/table/sql/handler`, Approfondimenti sulle prestazioni cerca l'attesa annidata e segnala l'attesa effettiva anziché `io/table/sql/handler`. Quando Approfondimenti sulle prestazioni segnala `io/table/sql/handler`, la vera attesa è

`io/table/sql/handler` e non un evento di attesa annidato nascosto. Per ulteriori informazioni consulta Performance Schema Atom and Molecule Events nel Manuale di riferimento di MySQL 5.7.

Azioni

Se l'evento di attesa domina l'attività del database, non indica necessariamente un problema di prestazioni. Un evento di attesa è sempre in primo piano quando il database è attivo. È necessario agire solo quando le prestazioni diminuiscono.

Consigliamo azioni diverse a seconda degli altri eventi di attesa visualizzati.

Argomenti

- Identificare le sessioni e le query che causano gli eventi
- Verifica la presenza di una correlazione con i parametri dei contatori di Approfondimenti sulle prestazioni
- Verifica la presenza di altri eventi di attesa correlati

Identificare le sessioni e le query che causano gli eventi

In genere, i database con carico da moderato a significativo hanno eventi di attesa. Gli eventi di attesa possono essere accettabili se le prestazioni sono ottimali. Se le prestazioni non sono ottimali, esaminare dove il database impiega più tempo. Considerare gli eventi di attesa che contribuiscono al carico più elevato per scoprire se è possibile ottimizzare il database e l'applicazione per ridurre tali eventi.

Per trovare query di SQL responsabili del carico elevato

1. Accedi alla Console di gestione AWS e apri la console Amazon RDS all'indirizzo https://console.aws.amazon.com/rds/.
2. Nel pannello di navigazione scegli Approfondimenti sulle prestazioni.
3. Scegli un'istanza database. Viene visualizzato il pannello di controllo di Approfondimenti sulle prestazioni per l'istanza database.
4. Nel grafico Carico del database, scegli Dividi per attesa.
5. Nella parte inferiore della pagina scegli Prime Istruzioni SQL.

   Il grafico elenca le query di SQL responsabili del carico. Quelli in cima all'elenco sono le più responsabili. Per risolvere un collo di bottiglia, occorre concentrarsi su queste istruzioni.

Per una panoramica utile dell'identificazione e della risoluzione dei problemi con Approfondimenti sulle prestazioni, consulta il post del blogAnalizza i carichi di lavoro di Amazon Aurora MySQL con Approfondimenti sulle prestazioni.

Verifica la presenza di una correlazione con i parametri dei contatori di Approfondimenti sulle prestazioni

Verifica la presenza di parametri del contatore di Approfondimenti sulle prestazioni come `Innodb_rows_changed`. Se i parametri del contatore sono correlati con `io/table/sql/handler`, segui questi passaggi:

1. In Approfondimenti sulle prestazioni, cerca le istruzioni SQL che rappresentano l'evento di attesa principale `io/table/sql/handler`. Se possibile, ottimizza questa istruzione in modo che restituisca un numero inferiore di righe.
2. Recupera le tabelle principali dalle visualizzazione `schema_table_statistics` e `x $schema_table_statistics`. Queste visualizzazioni mostrano la quantità di tempo impiegato per tabella. Per ulteriori informazioni, consulta Le visualizzazioni schema_table_statistics e x $schema_table_statistics nel Manuale di riferimento di MySQL.

   Per impostazione predefinita, le righe vengono ordinate in base al tempo di attesa totale discendente. Le tabelle con il maggior numero di contese appaiono per prime. L'output indica se il tempo viene impiegato per le letture, le scritture, il recupero, gli inserimenti, gli aggiornamenti o le eliminazioni.

```
mysql> select * from sys.schema_table_statistics limit 1\G

*************************** 1. row ***************************
     table_schema: read_only_db
       table_name: sbtest41
    total_latency: 54.11 m
     rows_fetched: 6001557
    fetch_latency: 39.14 m
    rows_inserted: 14833
   insert_latency: 5.78 m
     rows_updated: 30470
   update_latency: 5.39 m
     rows_deleted: 14833
   delete_latency: 3.81 m
 io_read_requests: NULL
          io_read: NULL
  io_read_latency: NULL
```

```
io_write_requests: NULL
         io_write: NULL
 io_write_latency: NULL
 io_misc_requests: NULL
  io_misc_latency: NULL
1 row in set (0.11 sec)
```

Verifica la presenza di altri eventi di attesa correlati

Se `synch/sxlock/innodb/btr_search_latch` e `io/table/sql/handler` insieme contribuiscono maggiormente all'anomalia del carico del database, verificare se la variabile `innodb_adaptive_hash_index` è attivata. Se lo è, considera la possibilità di aumentare il valore del parametro `innodb_adaptive_hash_index_parts`.

Se l'indice adattivo Hash è disattivato, considera la possibilità di attivarlo. Per ulteriori informazioni sull'indice adattivo Hash di MySQL, consulta le seguenti risorse:

- L'articolo L'indice adattivo Hash in InnoDB è adatto al mio carico di lavoro? sul sito web di Percona
- Indice adattivo Hash nel Manuale di riferimento di MySQL
- L'articolo Contesa in MySQL InnoDB: informazioni utili dalla sezione Segnalazioni sul sito web di Percona

> **Note**
>
> L'indice adattivo Hash non è supportato nelle istanze database di lettura di Aurora.
> In alcuni casi, le prestazioni potrebbero risultare scadenti su un'istanza di lettura quando `synch/sxlock/innodb/btr_search_latch` e `io/table/sql/handler` sono dominanti. In tal caso, prendere in considerazione il reindirizzamento temporaneo del carico di lavoro all'istanza database di scrittura e l'attivazione dell'indice adattivo Hash.

synch/cond/innodb/row_lock_wait

L'evento `synch/cond/innodb/row_lock_wait` si verifica quando una sessione ha bloccato una riga per un aggiornamento e un'altra sessione tenta di aggiornare la stessa riga. Per ulteriori informazioni, consulta l'argomento sul blocco in InnoDB nella documentazione di MySQL.

Versioni del motore supportate

Queste informazioni sull'evento di attesa sono supportate per le seguenti versioni del motore:

- Aurora MySQL versione 3

Probabili cause di aumento delle attese

Le diverse istruzioni relative al linguaggio di manipolazione dei dati (DML) accedono contemporaneamente alla stessa riga o righe.

Azioni

Consigliamo azioni diverse a seconda degli altri eventi di attesa visualizzati.

Argomenti

- Trova e rispondi alle istruzioni SQL responsabili di questo evento di attesa
- Trova e rispondi alla sessione di blocco

Trova e rispondi alle istruzioni SQL responsabili di questo evento di attesa

Utilizzare Approfondimenti sulle prestazioni per identificare le istruzioni SQL responsabili di questo evento di attesa. Considera le strategie seguenti:

- Se i blocchi di riga sono un problema persistente, considera la possibilità di riscrivere l'applicazione per utilizzare il blocco ottimistico.
- Utilizza istruzioni multiriga.
- Distribuisci il carico di lavoro su diversi oggetti di database. Per farlo, puoi utilizzare il partizionamento.
- Verifica il valore del parametro `innodb_lock_wait_timeout`. Controlla la durata di attesa delle transazioni prima di generare un errore di timeout.

Per una panoramica utile dell'identificazione e della risoluzione dei problemi con Approfondimenti sulle prestazioni, consulta il post del blogAnalizza i carichi di lavoro di Amazon Aurora MySQL con Approfondimenti sulle prestazioni.

Trova e rispondi alla sessione di blocco

Determina se la sessione di blocco è inattiva o attiva. Inoltre, scopri se la sessione proviene da un'applicazione o da un utente attivo.

Per identificare la sessione che tiene il blocco, è possibile eseguire `SHOW ENGINE INNODB STATUS`. Il seguente esempio mostra un output campione.

```
mysql> SHOW ENGINE INNODB STATUS;

---TRANSACTION 1688153, ACTIVE 82 sec starting index read
mysql tables in use 1, locked 1
LOCK WAIT 2 lock struct(s), heap size 1136, 2 row lock(s)
MySQL thread id 4244, OS thread handle 70369524330224, query id 4020834 172.31.14.179
 reinvent executing
select id1 from test.t1 where id1=1 for update
------- TRX HAS BEEN WAITING 24 SEC FOR THIS LOCK TO BE GRANTED:
RECORD LOCKS space id 11 page no 4 n bits 72 index GEN_CLUST_INDEX of table test.t1 trx
 id 1688153 lock_mode X waiting
Record lock, heap no 2 PHYSICAL RECORD: n_fields 5; compact format; info bits 0
```

In alternativa, è possibile usare la seguente query per estrarre i dettagli sui blocchi correnti.

```
mysql> SELECT p1.id waiting_thread,
    p1.user waiting_user,
    p1.host waiting_host,
    it1.trx_query waiting_query,
    ilw.requesting_engine_transaction_id waiting_transaction,
    ilw.blocking_engine_lock_id blocking_lock,
    il.lock_mode blocking_mode,
    il.lock_type blocking_type,
    ilw.blocking_engine_transaction_id blocking_transaction,
    CASE it.trx_state
        WHEN 'LOCK WAIT'
        THEN it.trx_state
        ELSE p.state end blocker_state,
    concat(il.object_schema,'.', il.object_name) as locked_table,
    it.trx_mysql_thread_id blocker_thread,
    p.user blocker_user,
    p.host blocker_host
FROM performance_schema.data_lock_waits ilw
JOIN performance_schema.data_locks il
ON ilw.blocking_engine_lock_id = il.engine_lock_id
```

```
AND ilw.blocking_engine_transaction_id = il.engine_transaction_id
JOIN information_schema.innodb_trx it
ON ilw.blocking_engine_transaction_id = it.trx_id join information_schema.processlist p
ON it.trx_mysql_thread_id = p.id join information_schema.innodb_trx it1
ON ilw.requesting_engine_transaction_id = it1.trx_id join
 information_schema.processlist p1
ON it1.trx_mysql_thread_id = p1.id\G

*************************** 1. row ***************************
waiting_thread: 4244
waiting_user: reinvent
waiting_host: 123.456.789.012:18158
waiting_query: select id1 from test.t1 where id1=1 for update
waiting_transaction: 1688153
blocking_lock: 70369562074216:11:4:2:70369549808672
blocking_mode: X
blocking_type: RECORD
blocking_transaction: 1688142
blocker_state: User sleep
locked_table: test.t1
blocker_thread: 4243
blocker_user: reinvent
blocker_host: 123.456.789.012:18156
1 row in set (0.00 sec)
```

Quando si identifica la sessione, le opzioni includono quanto segue:

- Contattare il proprietario dell'applicazione o l'utente.
- Se la sessione di blocco è inattiva, considerare di terminare la sessione di blocco. Questa azione potrebbe innescare un lungo ripristino dello stato precedente. Per informazioni su come terminare una sessione, consulta Terminare una sessione o una query.

Per ulteriori informazioni su come identificare le transazioni di blocco, consulta Using InnoDB transaction and locking information nella documentazione di MySQL.

synch/cond/innodb/row_lock_wait_cond

L'evento `synch/cond/innodb/row_lock_wait_cond` si verifica quando una sessione ha bloccato una riga per un aggiornamento e un'altra sessione tenta di aggiornare la stessa riga. Per ulteriori informazioni, consulta l'argomento sul blocco in InnoDB nella documentazione di MySQL.

Versioni del motore supportate

Queste informazioni sull'evento di attesa sono supportate per le seguenti versioni del motore:

- Aurora MySQL versione 2

Probabili cause di aumento delle attese

Le diverse istruzioni relative al linguaggio di manipolazione dei dati (DML) accedono contemporaneamente alla stessa riga o righe.

Azioni

Consigliamo azioni diverse a seconda degli altri eventi di attesa visualizzati.

Argomenti

- Trova e rispondi alle istruzioni SQL responsabili di questo evento di attesa
- Trova e rispondi alla sessione di blocco

Trova e rispondi alle istruzioni SQL responsabili di questo evento di attesa

Utilizzare Approfondimenti sulle prestazioni per identificare le istruzioni SQL responsabili di questo evento di attesa. Considera le strategie seguenti:

- Se i blocchi di riga sono un problema persistente, considera la possibilità di riscrivere l'applicazione per utilizzare il blocco ottimistico.
- Utilizza istruzioni multiriga.
- Distribuisci il carico di lavoro su diversi oggetti di database. Per farlo, puoi utilizzare il partizionamento.
- Verifica il valore del parametro `innodb_lock_wait_timeout`. Controlla la durata di attesa delle transazioni prima di generare un errore di timeout.

Per una panoramica utile dell'identificazione e della risoluzione dei problemi con Approfondimenti sulle prestazioni, consulta il post del blogAnalizza i carichi di lavoro di Amazon Aurora MySQL con Approfondimenti sulle prestazioni.

Trova e rispondi alla sessione di blocco

Determina se la sessione di blocco è inattiva o attiva. Inoltre, scopri se la sessione proviene da un'applicazione o da un utente attivo.

Per identificare la sessione che tiene il blocco, è possibile eseguire `SHOW ENGINE INNODB STATUS`. Il seguente esempio mostra un output campione.

```
mysql> SHOW ENGINE INNODB STATUS;

---TRANSACTION 2771110, ACTIVE 112 sec starting index read
mysql tables in use 1, locked 1
LOCK WAIT 2 lock struct(s), heap size 1136, 1 row lock(s)
MySQL thread id 24, OS thread handle 70369573642160, query id 13271336 172.31.14.179
 reinvent Sending data
select id1 from test.t1 where id1=1 for update
------- TRX HAS BEEN WAITING 43 SEC FOR THIS LOCK TO BE GRANTED:
RECORD LOCKS space id 11 page no 3 n bits 0 index GEN_CLUST_INDEX of table test.t1 trx
 id 2771110 lock_mode X waiting
Record lock, heap no 2 PHYSICAL RECORD: n_fields 5; compact format; info bits 0
```

In alternativa, è possibile usare la seguente query per estrarre i dettagli sui blocchi correnti.

```
mysql> SELECT p1.id waiting_thread,
              p1.user waiting_user,
              p1.host waiting_host,
              it1.trx_query waiting_query,
              ilw.requesting_trx_id waiting_transaction,
              ilw.blocking_lock_id blocking_lock,
              il.lock_mode blocking_mode,
              il.lock_type blocking_type,
              ilw.blocking_trx_id blocking_transaction,
              CASE it.trx_state
                WHEN 'LOCK WAIT'
                THEN it.trx_state
                ELSE p.state
              END blocker_state,
              il.lock_table locked_table,
              it.trx_mysql_thread_id blocker_thread,
              p.user blocker_user,
              p.host blocker_host
       FROM information_schema.innodb_lock_waits ilw
       JOIN information_schema.innodb_locks il
```

```
         ON ilw.blocking_lock_id = il.lock_id
        AND ilw.blocking_trx_id = il.lock_trx_id
       JOIN information_schema.innodb_trx it
         ON ilw.blocking_trx_id = it.trx_id
       JOIN information_schema.processlist p
         ON it.trx_mysql_thread_id = p.id
       JOIN information_schema.innodb_trx it1
         ON ilw.requesting_trx_id = it1.trx_id
       JOIN information_schema.processlist p1
         ON it1.trx_mysql_thread_id = p1.id\G

*************************** 1. row ***************************
      waiting_thread: 3561959471
        waiting_user: reinvent
        waiting_host: 123.456.789.012:20485
       waiting_query: select id1 from test.t1 where id1=1 for update
 waiting_transaction: 312337314
       blocking_lock: 312337287:261:3:2
       blocking_mode: X
       blocking_type: RECORD
blocking_transaction: 312337287
       blocker_state: User sleep
        locked_table: `test`.`t1`
      blocker_thread: 3561223876
        blocker_user: reinvent
        blocker_host: 123.456.789.012:17746
1 row in set (0.04 sec)
```

Quando si identifica la sessione, le opzioni includono quanto segue:

- Contattare il proprietario dell'applicazione o l'utente.
- Se la sessione di blocco è inattiva, considerare di terminare la sessione di blocco. Questa azione potrebbe innescare un lungo ripristino dello stato precedente. Per informazioni su come terminare una sessione, consulta Terminare una sessione o una query.

Per ulteriori informazioni su come identificare le transazioni di blocco, consulta Using InnoDB transaction and locking information nella documentazione di MySQL.

synch/cond/sql/MDL_context::COND_wait_status

L'evento `synch/cond/sql/MDL_context::COND_wait_status` si verifica quando ci sono thread in attesa del blocco dei metadati di una tabella.

Argomenti

- Versioni del motore supportate
- Context
- Probabili cause di aumento delle attese
- Azioni

Versioni del motore supportate

Queste informazioni sull'evento di attesa sono supportate per le seguenti versioni del motore:

- Aurora MySQL versioni 2 e 3

Context

L'evento `synch/cond/sql/MDL_context::COND_wait_status` indica che ci sono thread in attesa di un blocco dei metadati di tabella. In alcuni casi, una sessione contiene un blocco di metadati su una tabella e un'altra sessione tenta di ottenere lo stesso blocco sulla stessa tabella. In tal caso, la seconda sessione attende l'evento di attesa `synch/cond/sql/MDL_context::COND_wait_status`.

MySQL utilizza il blocco dei metadati per gestire l'accesso simultaneo agli oggetti del database e garantire la coerenza dei dati. Il blocco dei metadati si applica a tabelle, schemi, eventi pianificati, tablespace e blocchi utente acquisiti con la funzione `get_lock` e i programmi memorizzati. I programmi memorizzati includono procedure, funzioni e attivazioni. Per ulteriori informazioni, consulta Blocco dei metadati nella documentazione di MySQL.

L'elenco dei processi MySQL mostra questa sessione nello stato `waiting for metadata lock`. In Approfondimenti sulle prestazioni, se `Performance_schema` è acceso, l'evento `synch/cond/sql/MDL_context::COND_wait_status` viene visualizzato.

Il timeout predefinito per una query in attesa di un blocco dei metadati si basa sul valore del parametro `lock_wait_timeout`, che viene impostato in modo predefinito a 31.536.000 secondi (365 giorni).

Per ulteriori dettagli sui diversi blocchi InnoDB e sui tipi di blocchi che possono causare conflitti, consultare Blocco InnoDB nella documentazione di MySQL.

Probabili cause di aumento delle attese

Quando l'evento `synch/cond/sql/MDL_context::COND_wait_status` appare più che normale, possibilmente indicando un problema di prestazioni, le cause tipiche includono le seguenti:

Transazioni di lunga durata

Una o più transazioni stanno modificando una grande quantità di dati e mantengono i blocchi sulle tabelle per un tempo molto lungo.

Transazioni inattive

Una o più transazioni rimangono aperte per un lungo periodo, senza che venga eseguito il commit o riprimisto a una precedente versione.

Istruzioni DDL su tabelle grandi

Una o più istruzioni di definizione dei dati (DDL), ad esempio i comandi `ALTER TABLE`, sono state eseguite su tabelle molto grandi.

Blocchi espliciti della tabella

Ci sono blocchi espliciti sulle tabelle che non vengono rilasciate in modo tempestivo. Ad esempio, potrebbe essere eseguita un'applicazione su istruzioni `LOCK TABLE` impropriamente.

Azioni

Consigliamo diverse azioni a seconda delle cause dell'evento di attesa e della versione del cluster del database di Aurora MySQL.

Argomenti

- Identificare le sessioni e le query che causano gli eventi
- Verifica la presenza di eventi passati
- Esegui query su Aurora MySQL versione 2
- Rispondere alla sessione di blocco

Identificare le sessioni e le query che causano gli eventi

È possibile utilizzare Approfondimenti sulle prestazioni per mostrare le query bloccate dall'evento di attesa `synch/cond/sql/MDL_context::COND_wait_status`. Tuttavia, per identificare la

sessione di blocco, eseguire una query sulle tabelle dei metadati da `performance_schema` e `information_schema` sul cluster del database.

In genere, i database con carico da moderato a significativo hanno eventi di attesa. Gli eventi di attesa possono essere accettabili se le prestazioni sono ottimali. Se le prestazioni non sono ottimali, esaminare dove il database impiega più tempo. Considerare gli eventi di attesa che contribuiscono al carico più elevato per scoprire se è possibile ottimizzare il database e l'applicazione per ridurre tali eventi.

Per trovare query di SQL responsabili del carico elevato

1. Accedi alla Console di gestione AWS e apri la console Amazon RDS all'indirizzo https://console.aws.amazon.com/rds/.
2. Nel pannello di navigazione scegli Approfondimenti sulle prestazioni.
3. Scegli un'istanza database. Viene visualizzato il pannello di controllo di Approfondimenti sulle prestazioni per l'istanza database.
4. Nel grafico Carico del database, scegli Dividi per attesa.
5. Nella parte inferiore della pagina scegli Prime Istruzioni SQL.

   Il grafico elenca le query di SQL responsabili del carico. Quelli in cima all'elenco sono le più responsabili. Per risolvere un collo di bottiglia, occorre concentrarsi su queste istruzioni.

Per una panoramica utile della risoluzione dei problemi con Approfondimenti sulle prestazioni, consulta il post del blog di AWS Database Analizzare i carichi di lavoro di Amazon Aurora MySQL con Approfondimenti sulle prestazioni.

Verifica la presenza di eventi passati

È possibile ottenere informazioni dettagliate su questo evento di attesa per verificare la presenza di occorrenze passate. Per fare ciò, completare questa procedura:

- Controllare il linguaggio di manipolazione dei dati (DML) e la velocità effettiva e la latenza del DDL per verificare se sono state apportate modifiche al carico di lavoro.

  È possibile utilizzare Approfondimenti sulle prestazioni per trovare query in attesa di questo evento al momento del problema. Inoltre, è possibile visualizzare il digest delle query eseguite vicino al momento del problema.

- Se i registri di verifica o i registri generali sono attivati per il cluster el database, è possibile verificare la presenza di tutte le query eseguite sugli oggetti (schema.table) coinvolti nella transazione in attesa. È inoltre possibile verificare la presenza delle query completate in esecuzione prima della transazione.

Le informazioni disponibili per l'identificazione e la risoluzione dei problemi degli eventi passati sono limitate. L'esecuzione di queste verifiche non mostra quale oggetto è in attesa di informazioni. Tuttavia, è possibile identificare le tabelle con un carico pesante al momento dell'evento e la serie di righe gestite di frequente che causano conflitti al momento del problema. È quindi possibile utilizzare queste informazioni per riprodurre il problema in un ambiente di test e fornire informazioni dettagliate sulla sua causa.

Esegui query su Aurora MySQL versione 2

In Aurora MySQL versione 2, è possibile identificare direttamente la sessione bloccata eseguendo una query sulle tabelle `performance_schema` o visualizzazione dello schema `sys`. Un esempio può illustrare come interrogare le tabelle per identificare query e sessioni di blocco.

Nel seguente output dell'elenco dei processi, l'ID di connessione 89 è in attesa di un blocco dei metadati e sta eseguendo un comando `TRUNCATE TABLE`. In una query sulle tabelle `performance_schema` o visualizzazioni dello schema `sys`, l'output mostra che la sessione di blocco è 76.

```
MySQL [(none)]> select @@version, @@aurora_version;
+-----------+------------------+
| @@version | @@aurora_version |
+-----------+------------------+
| 5.7.12    | 2.11.5           |
+-----------+------------------+
1 row in set (0.01 sec)

MySQL [(none)]> show processlist;
+----+-----------------+--------------------+-----------+---------+------
+---------------------------------+------------------------------+
| Id | User            | Host               | db        | Command | Time | State
                    | Info                         |
+----+-----------------+--------------------+-----------+---------+------
+---------------------------------+------------------------------+
|  2 | rdsadmin        | localhost          | NULL      | Sleep   |    0 | NULL
                    | NULL                         |
```

```
|  4 | rdsadmin        | localhost          | NULL      | Sleep   |    2 | NULL
                    | NULL                           |
|  5 | rdsadmin        | localhost          | NULL      | Sleep   |    1 | NULL
                    | NULL                           |
| 20 | rdsadmin        | localhost          | NULL      | Sleep   |    0 | NULL
                    | NULL                           |
| 21 | rdsadmin        | localhost          | NULL      | Sleep   |  261 | NULL
                    | NULL                           |
| 66 | auroramysql5712 | 172.31.21.51:52154 | sbtest123 | Sleep   |    0 | NULL
                    | NULL                           |
| 67 | auroramysql5712 | 172.31.21.51:52158 | sbtest123 | Sleep   |    0 | NULL
                    | NULL                           |
| 68 | auroramysql5712 | 172.31.21.51:52150 | sbtest123 | Sleep   |    0 | NULL
                    | NULL                           |
| 69 | auroramysql5712 | 172.31.21.51:52162 | sbtest123 | Sleep   |    0 | NULL
                    | NULL                           |
| 70 | auroramysql5712 | 172.31.21.51:52160 | sbtest123 | Sleep   |    0 | NULL
                    | NULL                           |
| 71 | auroramysql5712 | 172.31.21.51:52152 | sbtest123 | Sleep   |    0 | NULL
                    | NULL                           |
| 72 | auroramysql5712 | 172.31.21.51:52156 | sbtest123 | Sleep   |    0 | NULL
                    | NULL                           |
| 73 | auroramysql5712 | 172.31.21.51:52164 | sbtest123 | Sleep   |    0 | NULL
                    | NULL                           |
| 74 | auroramysql5712 | 172.31.21.51:52166 | sbtest123 | Sleep   |    0 | NULL
                    | NULL                           |
| 75 | auroramysql5712 | 172.31.21.51:52168 | sbtest123 | Sleep   |    0 | NULL
                    | NULL                           |
| 76 | auroramysql5712 | 172.31.21.51:52170 | NULL      | Query   |    0 | starting
                    | show processlist               |
| 88 | auroramysql5712 | 172.31.21.51:52194 | NULL      | Query   |   22 | User sleep
                    | select sleep(10000)            |
| 89 | auroramysql5712 | 172.31.21.51:52196 | NULL      | Query   |    5 | Waiting for
 table metadata lock | truncate table sbtest.sbtest1 |
+----+-----------------+--------------------+-----------+---------+------
+--------------------------------+------------------------------+
18 rows in set (0.00 sec)
```

Successivamente, una query sulle tabelle `performance_schema` o visualizzazioni dello schema `sys` mostra che la sessione di blocco è 76.

```
MySQL [(none)]> select * from sys.schema_table_lock_waits;
```

```
+---------------+-------------+-------------------+-------------
+------------------------------+-------------------+-----------------------
+------------------------------+--------------------+-----------------------------
+------------------------------+--------------------+-------------
+------------------------------+--------------------+-----------------------
+--------------------------+-------------------------------+
| object_schema | object_name | waiting_thread_id | waiting_pid | waiting_account
           | waiting_lock_type | waiting_lock_duration | waiting_query
   | waiting_query_secs | waiting_query_rows_affected | waiting_query_rows_examined |
 blocking_thread_id | blocking_pid | blocking_account             | blocking_lock_type
 | blocking_lock_duration | sql_kill_blocking_query | sql_kill_blocking_connection |
+---------------+-------------+-------------------+-------------
+------------------------------+-------------------+-----------------------
+------------------------------+--------------------+-----------------------------
+------------------------------+--------------------+-------------
+------------------------------+--------------------+-----------------------
+--------------------------+-------------------------------+
| sbtest        | sbtest1     |               121 |          89 |
 auroramysql5712@192.0.2.0    | EXCLUSIVE         | TRANSACTION           | truncate
 table sbtest.sbtest1 |                10 |                            0 |
               0 |                108 |           76 | auroramysql5712@192.0.2.0    |
 SHARED_READ        | TRANSACTION            | KILL QUERY 76           | KILL 76
              |
+---------------+-------------+-------------------+-------------
+------------------------------+-------------------+-----------------------
+------------------------------+--------------------+-----------------------------
+------------------------------+--------------------+-------------
+------------------------------+--------------------+-----------------------
+--------------------------+-------------------------------+
1 row in set (0.00 sec)
```

Rispondere alla sessione di blocco

Quando si identifica la sessione, le opzioni includono quanto segue:

- Contattare il proprietario dell'applicazione o l'utente.
- Se la sessione di blocco è inattiva, considerare di terminare la sessione di blocco. Questa azione potrebbe innescare un lungo ripristino dello stato precedente. Per informazioni su come terminare una sessione, consulta Terminare una sessione o una query.

Per ulteriori informazioni su come identificare le transazioni di blocco, consulta Utilizzo delle informazioni sulle transazioni InnoDB e sul blocco nella documentazione di MySQL.

synch/mutex/innodb/aurora_lock_thread_slot_futex

L'evento `synch/mutex/innodb/aurora_lock_thread_slot_futex` si verifica quando una sessione ha bloccato una riga per un aggiornamento e un'altra sessione tenta di aggiornare la stessa riga. Per ulteriori informazioni, consulta Blocco in InnoDB nel Riferimento di MySQL.

Versioni del motore supportate

Queste informazioni sull'evento di attesa sono supportate per le seguenti versioni del motore:

- Aurora MySQL versione 2

Probabili cause di aumento delle attese

Le diverse istruzioni relative al linguaggio di manipolazione dei dati (DML) accedono contemporaneamente alla stessa riga o righe.

Azioni

Consigliamo azioni diverse a seconda degli altri eventi di attesa visualizzati.

Argomenti

- Trova e rispondi alle istruzioni SQL responsabili di questo evento di attesa
- Trova e rispondi alla sessione di blocco

Trova e rispondi alle istruzioni SQL responsabili di questo evento di attesa

Utilizzare Approfondimenti sulle prestazioni per identificare le istruzioni SQL responsabili di questo evento di attesa. Considera le strategie seguenti:

- Se i blocchi di riga sono un problema persistente, considera la possibilità di riscrivere l'applicazione per utilizzare il blocco ottimistico.
- Utilizza istruzioni multiriga.
- Distribuisci il carico di lavoro su diversi oggetti di database. Per farlo, puoi utilizzare il partizionamento.
- Verifica il valore del parametro `innodb_lock_wait_timeout`. Controlla la durata di attesa delle transazioni prima di generare un errore di timeout.

Per una panoramica utile dell'identificazione e della risoluzione dei problemi con Approfondimenti sulle prestazioni, consulta il post del blogAnalizza i carichi di lavoro di Amazon Aurora MySQL con Approfondimenti sulle prestazioni.

Trova e rispondi alla sessione di blocco

Determina se la sessione di blocco è inattiva o attiva. Inoltre, scopri se la sessione proviene da un'applicazione o da un utente attivo.

Per identificare la sessione che tiene il blocco, è possibile eseguire `SHOW ENGINE INNODB STATUS`. Il seguente esempio mostra un output campione.

```
mysql> SHOW ENGINE INNODB STATUS;

----------------------TRANSACTION 302631452, ACTIVE 2 sec starting index read
mysql tables in use 1, locked 1
LOCK WAIT 2 lock struct(s), heap size 376, 1 row lock(s)
MySQL thread id 80109, OS thread handle 0x2ae915060700, query id 938819 10.0.4.12
 reinvent updating
UPDATE sbtest1 SET k=k+1 WHERE id=503
------- TRX HAS BEEN WAITING 2 SEC FOR THIS LOCK TO BE GRANTED:
RECORD LOCKS space id 148 page no 11 n bits 30 index `PRIMARY` of table
 `sysbench2`.`sbtest1` trx id 302631452 lock_mode X locks rec but not gap waiting
Record lock, heap no 30 PHYSICAL RECORD: n_fields 6; compact format; info bits 0
```

In alternativa, è possibile usare la seguente query per estrarre i dettagli sui blocchi correnti.

```
mysql> SELECT p1.id waiting_thread,
              p1.user waiting_user,
              p1.host waiting_host,
              it1.trx_query waiting_query,
              ilw.requesting_trx_id waiting_transaction,
              ilw.blocking_lock_id blocking_lock,
              il.lock_mode blocking_mode,
              il.lock_type blocking_type,
              ilw.blocking_trx_id blocking_transaction,
              CASE it.trx_state
                WHEN 'LOCK WAIT'
                THEN it.trx_state
                ELSE p.state
              END blocker_state,
              il.lock_table locked_table,
```

```
              it.trx_mysql_thread_id blocker_thread,
              p.user blocker_user,
              p.host blocker_host
       FROM information_schema.innodb_lock_waits ilw
       JOIN information_schema.innodb_locks il
         ON ilw.blocking_lock_id = il.lock_id
        AND ilw.blocking_trx_id = il.lock_trx_id
       JOIN information_schema.innodb_trx it
         ON ilw.blocking_trx_id = it.trx_id
       JOIN information_schema.processlist p
         ON it.trx_mysql_thread_id = p.id
       JOIN information_schema.innodb_trx it1
         ON ilw.requesting_trx_id = it1.trx_id
       JOIN information_schema.processlist p1
         ON it1.trx_mysql_thread_id = p1.id\G

*************************** 1. row ***************************
      waiting_thread: 3561959471
        waiting_user: reinvent
        waiting_host: 123.456.789.012:20485
       waiting_query: select id1 from test.t1 where id1=1 for update
 waiting_transaction: 312337314
       blocking_lock: 312337287:261:3:2
       blocking_mode: X
       blocking_type: RECORD
blocking_transaction: 312337287
       blocker_state: User sleep
        locked_table: `test`.`t1`
      blocker_thread: 3561223876
        blocker_user: reinvent
        blocker_host: 123.456.789.012:17746
1 row in set (0.04 sec)
```

Quando si identifica la sessione, le opzioni includono quanto segue:

- Contattare il proprietario dell'applicazione o l'utente.
- Se la sessione di blocco è inattiva, considerare di terminare la sessione di blocco. Questa azione potrebbe innescare un lungo ripristino dello stato precedente. Per informazioni su come terminare una sessione, consulta Terminare una sessione o una query.

Per ulteriori informazioni su come identificare le transazioni di blocco, consulta Utilizzo delle informazioni sulle transazioni InnoDB e sul blocco nel Manuale di riferimento di MySQL.

synch/mutex/innodb/buf_pool_mutex

L'evento `synch/mutex/innodb/buf_pool_mutex` si verifica quando un thread ha acquisito un blocco nel pool di buffer InnoDB per accedere a una pagina in memoria.

Argomenti

- Versioni del motore rilevanti
- Context
- Probabili cause di aumento delle attese
- Azioni

Versioni del motore rilevanti

Queste informazioni sull'evento di attesa sono supportate per le seguenti versioni del motore:

- Aurora MySQL versione 2

Context

La mutex `buf_pool` è una mutex singola che protegge le strutture di dati di controllo del pool di buffer.

Per ulteriori informazioni, consulta Monitoraggio delle attese di Mutex di InnoDB utilizzando lo schema delle prestazioni nella documentazione di MySQL.

Probabili cause di aumento delle attese

Si tratta di un evento di attesa specifico per il carico di lavoro. Cause comuni perché `synch/mutex/innodb/buf_pool_mutex` appaia tra i primi eventi di attesa includono quanto segue:

- Le dimensioni del pool di buffer non sono abbastanza grandi da contenere la serie di dati funzionante.
- Il carico di lavoro è più specifico per determinate pagine di una tabella specifica del database, causando contese nel pool di buffer.

Azioni

Consigliamo azioni diverse a seconda delle cause dell'evento di attesa.

Argomenti

- Identificare le sessioni e le query che causano gli eventi
- Utilizzo di Performance Insights
- Crea repliche di Aurora
- Analisi delle dimensioni del pool di buffer
- Monitoraggio della cronologia di stato globale

Identificare le sessioni e le query che causano gli eventi

In genere, i database con carico da moderato a significativo hanno eventi di attesa. Gli eventi di attesa possono essere accettabili se le prestazioni sono ottimali. Se le prestazioni non sono ottimali, esaminare dove il database impiega più tempo. Considerare gli eventi di attesa che contribuiscono al carico più elevato per scoprire se è possibile ottimizzare il database e l'applicazione per ridurre tali eventi.

Per visualizzare il grafico SQL principale nella Console di gestione AWS

1. Apri la console Amazon RDS all'indirizzo https://console.aws.amazon.com/rds/.
2. Nel pannello di navigazione scegli Performance Insights.
3. Scegli un'istanza database. Viene visualizzato il pannello di controllo di Approfondimenti sulle prestazioni per l'istanza database.
4. Nel grafico Carico del database, scegli Dividi per attesa.
5. Sotto il grafico Carico del database, seleziona Prime istruzioni SQL.

   Il grafico elenca le query di SQL responsabili del carico. Quelli in cima all'elenco sono le più responsabili. Per risolvere un collo di bottiglia, occorre concentrarsi su queste istruzioni.

Per una panoramica utile dell'identificazione e della risoluzione dei problemi con Performance Insights, consulta il post del blogAnalizza i carichi di lavoro di Amazon Aurora MySQL con Performance Insights.

Utilizzo di Performance Insights

Questo evento è correlato al carico di lavoro. È possibile utilizzare Performance Insights per effettuare quanto segue:

- Identificare l'avvio degli eventi di attesa e se c'è qualche modifica nel carico di lavoro in quel periodo dai registri dell'applicazione o dalle origini correlate.
- Identificare le istruzioni SQL responsabili di questo evento di attesa. Esaminare il piano di esecuzione delle query per accertarsi che queste query siano ottimizzate e utilizzino indici appropriati.

  Se le query principali responsabili dell'evento di attesa sono correlate allo stesso oggetto o tabella di database, prendere in considerazione il partizionamento dell'oggetto o della tabella.

Crea repliche di Aurora

È possibile creare repliche di Aurora per gestire il traffico di sola lettura. È inoltre possibile utilizzare Auto Scaling di Aurora per gestire le sovratensioni nel traffico di lettura. Assicurarsi di eseguire attività di sola lettura pianificate e backup logici sulle repliche di Aurora.

Per ulteriori informazioni, consulta Dimensionamento automatico di Amazon Aurora con le repliche Aurora.

Analisi delle dimensioni del pool di buffer

Verificare se la dimensione del pool di buffer è sufficiente per il carico di lavoro esaminando il parametro `innodb_buffer_pool_wait_free`. Se il valore di questo parametro è alto e aumenta continuamente, ciò indica che la dimensione del pool di buffer non è sufficiente per gestire il carico di lavoro. Se `innodb_buffer_pool_size` è stato impostato correttamente, il valore di `innodb_buffer_pool_wait_free`dovrebbe essere piccolo. Per ulteriori informazioni, consulta Innodb_buffer_pool_wait_free nella documentazione di MySQL.

Aumentare le dimensioni del pool di buffer se l'istanza database dispone di memoria sufficiente per i buffer di sessione e le attività del sistema operativo. In caso contrario, modificare l'istanza database in una classe di istanza database più grande per ottenere memoria aggiuntiva che può essere allocata al pool di buffer.

> **Note**
>
> Aurora MySQL regola automaticamente il valore di `innodb_buffer_pool_instances` basato sul configurato `innodb_buffer_pool_size`.

Monitoraggio della cronologia di stato globale

Monitorando i tassi di modifica delle variabili di stato, è possibile rilevare problemi di blocco o di memoria sull'istanza database. Attivare Global Status History (GoSH) se non è già attivo. Per ulteriori informazioni su GoSH, consulta Gestione della cronologia di stato globale.

Puoi anche creare parametri personalizzati di Amazon CloudWatch per monitorare le variabili di stato. Per ulteriori informazioni, consulta Pubblicazione di parametri personalizzati.

synch/mutex/innodb/fil_system_mutex

L'evento `synch/mutex/innodb/fil_system_mutex` si verifica quando una sessione è in attesa di accedere alla cache di memoria tablespace.

Argomenti

- Versioni del motore supportate
- Context
- Probabili cause di aumento delle attese
- Azioni

Versioni del motore supportate

Queste informazioni sull'evento di attesa sono supportate per le seguenti versioni del motore:

- Aurora MySQL versioni 2 e 3

Context

InnoDB utilizza tablespace per gestire l'area di archiviazione per tabelle e file di registro. La cache di memoria di tablespace è una struttura di memoria globale che mantiene le informazioni sulle tablespace. MySQL utilizza attese `synch/mutex/innodb/fil_system_mutex` per controllare l'accesso simultaneo alla cache di memoria di tablespace.

L'evento `synch/mutex/innodb/fil_system_mutex` indica che attualmente esistono più operazioni che devono recuperare e manipolare le informazioni nella cache di memoria di tablespace per la stessa tablespace.

Probabili cause di aumento delle attese

Quando l'evento `synch/mutex/innodb/fil_system_mutex` appare più che normale, possibilmente indicando un problema di prestazioni, generalmente accade quando sono presenti tutte le seguenti condizioni:

- Un aumento simultaneo delle operazioni relative al linguaggio di manipolazione dei dati (DM) che che aggiornano o eliminano i dati nella stessa tabella.
- La tablespace per questa tabella è molto ampia e ha molte pagine di dati.
- Il fattore di riempimento per queste pagine di dati è basso.

Azioni

Consigliamo azioni diverse a seconda delle cause dell'evento di attesa.

Argomenti

- Identificare le sessioni e le query che causano gli eventi
- Riorganizza tabelle di grandi dimensioni durante le ore non di picco

Identificare le sessioni e le query che causano gli eventi

In genere, i database con carico da moderato a significativo hanno eventi di attesa. Gli eventi di attesa possono essere accettabili se le prestazioni sono ottimali. Se le prestazioni non sono ottimali, esaminare dove il database impiega più tempo. Considerare gli eventi di attesa che contribuiscono al carico più elevato per scoprire se è possibile ottimizzare il database e l'applicazione per ridurre tali eventi.

Per trovare query di SQL responsabili del carico elevato

1. Accedi alla Console di gestione AWS e apri la console Amazon RDS all'indirizzo https://console.aws.amazon.com/rds/.
2. Nel pannello di navigazione scegli Approfondimenti sulle prestazioni.
3. Scegli un'istanza database. Viene visualizzato il pannello di controllo di Performance Insights per l'istanza database.
4. Nel grafico Carico del database, scegli Dividi per attesa.
5. Nella parte inferiore della pagina scegli Prime Instruzioni SQL.

Il grafico elenca le query di SQL responsabili del carico. Quelli in cima all'elenco sono le più responsabili. Per risolvere un collo di bottiglia, occorre concentrarsi su queste istruzioni.

Per una panoramica utile dell'identificazione e della risoluzione dei problemi con Performance Insights, consulta il post del blogAnalizza i carichi di lavoro di Amazon Aurora MySQL con Performance Insights.

Un altro modo per scoprire quali query causano un numero elevato di attese `synch/mutex/innodb/fil_system_mutex` è di controllare `performance_schema`, come nel seguente esempio.

```
mysql> select * from performance_schema.events_waits_current where EVENT_NAME='wait/
synch/mutex/innodb/fil_system_mutex'\G
*************************** 1. row ***************************
            THREAD_ID: 19
             EVENT_ID: 195057
         END_EVENT_ID: 195057
           EVENT_NAME: wait/synch/mutex/innodb/fil_system_mutex
               SOURCE: fil0fil.cc:6700
          TIMER_START: 1010146190118400
            TIMER_END: 1010146196524000
           TIMER_WAIT: 6405600
                SPINS: NULL
        OBJECT_SCHEMA: NULL
          OBJECT_NAME: NULL
           INDEX_NAME: NULL
          OBJECT_TYPE: NULL
OBJECT_INSTANCE_BEGIN: 47285552262176
     NESTING_EVENT_ID: NULL
   NESTING_EVENT_TYPE: NULL
            OPERATION: lock
      NUMBER_OF_BYTES: NULL
                FLAGS: NULL
*************************** 2. row ***************************
            THREAD_ID: 23
             EVENT_ID: 5480
         END_EVENT_ID: 5480
           EVENT_NAME: wait/synch/mutex/innodb/fil_system_mutex
               SOURCE: fil0fil.cc:5906
          TIMER_START: 995269979908800
            TIMER_END: 995269980159200
```

```
           TIMER_WAIT: 250400
                SPINS: NULL
        OBJECT_SCHEMA: NULL
          OBJECT_NAME: NULL
           INDEX_NAME: NULL
          OBJECT_TYPE: NULL
OBJECT_INSTANCE_BEGIN: 47285552262176
     NESTING_EVENT_ID: NULL
   NESTING_EVENT_TYPE: NULL
            OPERATION: lock
      NUMBER_OF_BYTES: NULL
                FLAGS: NULL
*************************** 3. row ***************************
            THREAD_ID: 55
             EVENT_ID: 23233794
         END_EVENT_ID: NULL
           EVENT_NAME: wait/synch/mutex/innodb/fil_system_mutex
               SOURCE: fil0fil.cc:449
          TIMER_START: 1010492125341600
            TIMER_END: 1010494304900000
           TIMER_WAIT: 2179558400
                SPINS: NULL
        OBJECT_SCHEMA: NULL
          OBJECT_NAME: NULL
           INDEX_NAME: NULL
          OBJECT_TYPE: NULL
OBJECT_INSTANCE_BEGIN: 47285552262176
     NESTING_EVENT_ID: 23233786
   NESTING_EVENT_TYPE: WAIT
            OPERATION: lock
      NUMBER_OF_BYTES: NULL
                FLAGS: NULL
```

Riorganizza tabelle di grandi dimensioni durante le ore non di picco

Riorganizza tabelle di grandi dimensioni che identifichi come fonte di un numero elevato di eventi di attesa `synch/mutex/innodb/fil_system_mutex` durante una finestra temporale di manutenzione al di fuori dell'orario di produzione. In questo modo si assicura che la pulizia della mappa delle tablespace interne non si verifichi quando l'accesso rapido alla tabella è fondamentale. Per informazioni sulla riorganizzazione delle tabelle, consulta Istruzione OPTIMIZE TABLE nel Riferimento di MySQL.

synch/mutex/innodb/trx_sys_mutex

L'evento `synch/mutex/innodb/trx_sys_mutex` si verifica quando c'è un'elevata attività del database con un numero elevato di transazioni.

Argomenti

- Versioni del motore rilevanti
- Context
- Probabili cause di aumento delle attese
- Azioni

Versioni del motore rilevanti

Queste informazioni sull'evento di attesa sono supportate per le seguenti versioni del motore:

- Aurora MySQL versioni 2 e 3

Context

Internamente, il motore di database InnoDB utilizza il livello di isolamento della lettura ripetibile con snapshot per fornire coerenza di lettura. In questo modo viene consentita una visualizzazione istantanea del database al momento della creazione dello snapshot.

In InnoDB, tutte le modifiche vengono applicate al database non appena arrivano, indipendentemente dal fatto che sia stato eseguito il commit. Questo approccio significa che senza il controllo della concorrenza multiversione (MVCC), tutti gli utenti connessi al database vedono tutte le modifiche e le righe più recenti. Pertanto, InnoDB richiede un modo per tenere traccia delle modifiche per capire cosa ripristinare allo stato precedente quando necessario.

Per fare ciò, InnoDB utilizza un sistema di transazioni (`trx_sys`) per tenere traccia degli snapshot. Il sistema di transazioni effettua quanto segue:

- Tiene traccia dell'ID della transazione per ogni riga nei registri di annullamento.
- Utilizza una struttura interna InnoDB chiamata `ReadView` che aiuta a identificare quali ID di transazione sono visibili per uno snapshot.

Probabili cause di aumento delle attese

Qualsiasi operazione di database che richiede la gestione coerente e controllata (creazione, lettura, aggiornamento ed eliminazione) degli ID delle transazioni genera una chiamata da `trx_sys` al mutex.

Queste chiamate avvengono all'interno di tre funzioni:

- `trx_sys_mutex_enter` – Crea il mutex.
- `trx_sys_mutex_exit` – Rilascia il mutex.
- `trx_sys_mutex_own` – Verifica se il mutex è di proprietà.

La strumentazione InnoDB Performance Schema tiene traccia di tutte le chiamate mutex `trx_sys`. Il monitoraggio include, a titolo esemplificativo ma non esaustivo, la gestione di `trx_sys` all'avvio o allo spegnimento del database, operazioni di ripristino dello stato precedente, pulizia degli annullamenti, accesso in lettura di righe e carichi del pool di buffer. L'elevata attività del database con un numero elevato di transazioni comporta la comparsa di `synch/mutex/innodb/trx_sys_mutex` tra i principali eventi di attesa.

Per ulteriori informazioni, consulta Monitoraggio delle attese di Mutex di InnoDB utilizzando lo schema delle prestazioni nella documentazione di MySQL.

Azioni

Consigliamo azioni diverse a seconda delle cause dell'evento di attesa.

Argomenti

- Identificare le sessioni e le query che causano gli eventi
- Esamina altri eventi di attesa

Identificare le sessioni e le query che causano gli eventi

In genere, i database con carico da moderato a significativo hanno eventi di attesa. Gli eventi di attesa possono essere accettabili se le prestazioni sono ottimali. Se le prestazioni non sono ottimali, esaminare dove il database impiega più tempo. Guarda gli eventi di attesa che contribuiscono al carico più elevato. Scopri se è possibile ottimizzare il database e l'applicazione per ridurre tali eventi.

Per visualizzare il grafico Istruzioni SQL principali nella Console di gestione AWS

1. Apri la console Amazon RDS all'indirizzo https://console.aws.amazon.com/rds/.
2. Nel pannello di navigazione scegli Performance Insights.
3. Scegli un'istanza database. Viene visualizzato il pannello di controllo di Approfondimenti sulle prestazioni per l'istanza database.
4. Nel grafico Carico del database, scegli Dividi per attesa.
5. Nel grafico Carico del database, seleziona Prime istruzioni SQL.

   Il grafico elenca le query di SQL responsabili del carico. Quelli in cima all'elenco sono le più responsabili. Per risolvere un collo di bottiglia, occorre concentrarsi su queste istruzioni.

Per una panoramica utile dell'identificazione e della risoluzione dei problemi con Performance Insights, consulta il post del blogAnalizza i carichi di lavoro di Amazon Aurora MySQL con Performance Insights.

Esamina altri eventi di attesa

Esamina gli altri eventi di attesa associati all'evento di attesa `synch/mutex/innodb/trx_sys_mutex`. In questo modo, si possono acquisire ulteriori informazioni sulla natura del carico di lavoro. Un numero elevato di transazioni potrebbe ridurre la velocità effettiva, ma anche il carico di lavoro potrebbe renderlo necessario.

Per ulteriori informazioni su come ottimizzare le transazioni, consulta Ottimizzazione della gestione delle transazioni InnoDB nella documentazione di MySQL.

synch/sxlock/innodb/hash_table_locks

L'evento `synch/sxlock/innodb/hash_table_locks` si verifica quando le pagine non trovate nel pool di buffer devono essere lette dall'archiviazione.

Argomenti

- Versioni del motore supportate
- Contesto
- Probabili cause di aumento delle attese
- Azioni

Versioni del motore supportate

Queste informazioni sull'evento di attesa sono supportate per le seguenti versioni:

- Aurora MySQL versioni 2 e 3

Contesto

L'evento `synch/sxlock/innodb/hash_table_locks` indica che un carico di lavoro accede frequentemente a dati che non sono memorizzati nel pool di buffer. Questo evento di attesa è associato ad aggiunte di nuove pagine e a espulsioni di dati vecchi dal pool di buffer. I dati memorizzati nei dati nuovi e vecchi del pool di buffer devono essere memorizzati nella cache, in modo che le pagine vecchie vengano espulse per consentire la memorizzazione nella cache delle nuove pagine. MySQL utilizza un algoritmo utilizzato meno di recente (LRU) per espellere pagine dal pool di buffer. Il carico di lavoro sta tentando di accedere ai dati che non sono stati caricati nel pool di buffer o ai dati che sono stati espulsi dal pool di buffer.

Questo evento di attesa si verifica quando il carico di lavoro deve accedere ai dati nei file su disco o quando i blocchi vengono liberati o aggiunti all'elenco LRU del pool di buffer. Queste operazioni attendono di ottenere un blocco condiviso escluso (SX-Lock). Questo SX-Lock viene utilizzato per la sincronizzazione su tabella hash, che è una tabella in memoria progettata per migliorare le prestazioni di accesso al pool di buffer.

Per ulteriori informazioni, consulta Pool di buffer nella documentazione di MySQL.

Probabili cause di aumento delle attese

Quando l'evento di attesa `synch/sxlock/innodb/hash_table_locks` appare più che normale, probabilmente indicando un problema di prestazioni, le cause tipiche includono le seguenti:

Un pool di buffer sottodimensionato

La dimensione del pool di buffer è troppo piccola per mantenere in memoria tutte le pagine a cui si accede di frequente.

Carico di lavoro pesante

Il carico di lavoro sta causando frequenti espulsioni e ricariche di pagine di dati nella cache del buffer.

Errori di lettura delle pagine

Ci sono errori di lettura delle pagine nel pool di buffer, che potrebbero indicare il danneggiamento dei dati.

Azioni

Consigliamo azioni diverse a seconda delle cause dell'evento di attesa.

Argomenti

- Aumenta le dimensioni del pool del buffer
- Miglioramento dei modelli di accesso ai dati
- Riduci o evita le scansioni complete della tabella
- Controllare i registri degli errori per la presenza del danneggiamento della pagina

Aumenta le dimensioni del pool del buffer

Assicurarsi che il pool del buffer sia ridimensionato in modo appropriato per il carico di lavoro. Per farlo è possibile controllare la percentuale di riscontri nella cache del pool di buffer. In genere, se il valore scende al di sotto del 95%, è consigliabile aumentare la dimensione del pool di buffer. Un pool di buffer più ampio può mantenere più a lungo in memoria le pagine a cui si accede di frequente. Per aumentare le dimensioni del pool di buffer, modificare il valore del parametro `innodb_buffer_pool_size`. Il valore predefinito di questo parametro è basato sulle dimensioni della classe di istanza database. Per ulteriori informazioni, consulta Best practice per la configurazione del database di Amazon Aurora MySQL.

Miglioramento dei modelli di accesso ai dati

Controlla le query interessate da questa attesa e i relativi piani di esecuzione. Considera la possibilità di migliorare i modelli di accesso ai dati. Ad esempio, se si sta utilizzandomysqli_result:: fetch_array, si può provare ad aumentare la dimensione del recupero dell'array.

È possibile utilizzare Performance Insights per visualizzare query e sessioni che potrebbero causare l'evento di attesa `synch/sxlock/innodb/hash_table_locks`.

Per trovare query di SQL responsabili del carico elevato

1. Accedi alla Console di gestione AWS e apri la console Amazon RDS all'indirizzo https://console.aws.amazon.com/rds/.

2. Nel pannello di navigazione scegli Approfondimenti sulle prestazioni.
3. Scegli un'istanza database. Viene visualizzato il pannello di controllo di Approfondimenti sulle prestazioni per l'istanza database.
4. Nel grafico Carico del database, scegli Dividi per attesa.
5. Nella parte inferiore della pagina scegli Prime Istruzioni SQL.

   Il grafico elenca le query di SQL responsabili del carico. Quelli in cima all'elenco sono le più responsabili. Per risolvere un collo di bottiglia, occorre concentrarsi su queste istruzioni.

Per una panoramica utile dell'identificazione e della risoluzione dei problemi con Performance Insights, consulta il post del blog AWS Analizza i carichi di lavoro di Amazon Aurora MySQL con Performance Insights.

Riduci o evita le scansioni complete della tabella

Monitora il carico di lavoro per verificare se sta eseguendo scansioni a tabella completa e, in caso affermativo, ridurle o evitarle. Ad esempio, è possibile monitorare le variabili di stato come ad esempio `Handler_read_rnd_next`. Per ulteriori dettagli, consulta Variabili dello stato del server nella documentazione di MySQL.

Controllare i registri degli errori per la presenza del danneggiamento della pagina

È possibile controllare il mysql-error.log per la presenza di messaggi correlati al danneggiamento che sono stati rilevati in prossimità del momento in cui si è verificato il problema. I messaggi con cui è possibile lavorare per risolvere il problema si trovano nel registro degli errori. Potrebbe essere necessario ricreare oggetti segnalati come danneggiati.

synch/mutex/innodb/temp_pool_manager_mutex

L'evento `synch/mutex/innodb/temp_pool_manager_mutex` wait si verifica quando una sessione è in attesa di acquisire un mutex per la gestione del pool di tablespace temporanee della sessione.

Argomenti

- Versioni del motore supportate
- Contesto
- Probabili cause di aumento delle attese

- Azioni

Versioni del motore supportate

Queste informazioni sull'evento di attesa sono supportate per le seguenti versioni del motore:

- Aurora MySQL versione 3

Contesto

Aurora MySQL versione 3.x e successive consente di controllare più sessioni che accedono `temp_pool_manager_mutex` al pool di tablespace temporaneo contemporaneamente. Aurora MySQL gestisce l'archiviazione tramite un volume cluster Aurora per i dati persistenti e l'archiviazione locale per i file temporanei. Un tablespace temporaneo è necessario quando una sessione crea una tabella temporanea sul volume del cluster Aurora.

Quando una sessione richiede per la prima volta un tablespace temporaneo, MySQL alloca tablespace temporanei della sessione dal pool condiviso. Una sessione può contenere fino a 2 tablespace temporanee alla volta per i seguenti tipi di tabelle:

- Tabelle temporanee create dall'utente
- Tabelle temporanee interne generate dall'ottimizzatore

Il `TempTable` motore predefinito utilizza il seguente meccanismo di overflow per gestire le tabelle temporanee:

- Memorizza le tabelle nella RAM fino al `temptable_max_ram`limite.
- Passa ai file mappati in memoria nella memoria locale quando la RAM è piena.
- Utilizza il volume condiviso del cluster quando i file mappati in memoria raggiungono il limite. `temptable_max_mmap`

Dopo che le tabelle temporanee superano i limiti di RAM e di archiviazione locale, MySQL le gestisce utilizzando tablespace su disco.

Quando una sessione richiede una tabella temporanea su disco, MySQL:

- Cerca i `INACTIVE` tablespace disponibili nel pool da riutilizzare.

- Crea 10 nuove tablespace se non esistono spazi. `INACTIVE`

Quando una sessione si disconnette, MySQL:

- Tronca i tablespace temporanei della sessione.
- Le contrassegna come INATTIVE nel pool per il riutilizzo.
- Mantiene la dimensione attuale del pool fino al riavvio del server.
- Ritorna alla dimensione predefinita del pool (10 tablespace) dopo il riavvio.

Probabili cause di aumento delle attese

Situazioni comuni che causano questo evento di attesa:

- Sessioni simultanee che creano tabelle temporanee interne sul volume del cluster.
- Sessioni simultanee che creano tabelle temporanee utente sul volume del cluster.
- Interruzione improvvisa delle sessioni utilizzando tablespace attivi.
- Espansione del pool di tablespace durante carichi di lavoro di scrittura pesanti.
- Accesso simultaneo alle interrogazioni `INFORMATION_SCHEMA.`

Azioni

Consigliamo azioni diverse a seconda delle cause dell'evento di attesa.

Argomenti

- Monitora e ottimizza l'utilizzo temporaneo delle tabelle
- Esamina le query utilizzando INFORMATION_SCHEMA
- Aumenta il parametro innodb_sync_array_size
- Implementa un pool di connessioni

Monitora e ottimizza l'utilizzo temporaneo delle tabelle

Per monitorare e ottimizzare l'utilizzo delle tabelle temporanee, utilizzate uno di questi metodi:

- Controlla il `Created_tmp_disk_tables` contatore in Performance Insights per tenere traccia della creazione di tabelle temporanee su disco nel cluster Aurora.

- Esegui questo comando nel tuo database per monitorare direttamente la creazione di tabelle temporanee:. `mysql> show status like '%created_tmp_disk%'`

> **Note**
>
> Il comportamento delle tabelle temporanee differisce tra i nodi reader MySQL di Aurora e i nodi writer. Per ulteriori informazioni, consulta Nuovo comportamento della tabella temporanea in Aurora MySQL versione 3.

Dopo aver identificato le query che creano tabelle temporanee, esegui questi passaggi di ottimizzazione:

- `EXPLAIN`Utilizzatelo per esaminare i piani di esecuzione delle query e identificare dove e perché vengono create le tabelle temporanee.
- Modifica le query per ridurre l'utilizzo temporaneo delle tabelle, ove possibile.

Se l'ottimizzazione delle query da sola non risolve i problemi di prestazioni, valuta la possibilità di modificare questi parametri di configurazione:

- `temptable_max_ram`- Controlla l'utilizzo massimo della RAM per le tabelle temporanee.
- `temptable_max_mmap`- Imposta il limite per l'archiviazione di file mappati in memoria.
- `tmp_table_size`- Si applica quando `aurora_tmptable_enable_per_table_limit` è abilitato (disabilitato per impostazione predefinita).

> **Important**
>
> Tieni presente che alcune condizioni di interrogazione richiederanno sempre tabelle temporanee su disco, indipendentemente dalle impostazioni di configurazione. Per ulteriori informazioni sul motore `TempTable` di storage, consulta Utilizzare il motore TempTable di storage su Amazon RDS for MySQL e Amazon Aurora MySQL.

Esamina le query utilizzando INFORMATION_SCHEMA

Quando si interrogano `INFORMATION_SCHEMA` le tabelle, MySQL crea tabelle temporanee InnoDB sul volume del cluster. Ogni sessione necessita di un tablespace temporaneo per queste tabelle, il che può portare a problemi di prestazioni in caso di accesso simultaneo elevato.

Per migliorare le prestazioni:

- Usalo `PERFORMANCE_SCHEMA` al posto di `INFORMATION_SCHEMA` dove possibile.
- Se necessario`INFORMATION_SCHEMA`, riduci la frequenza con cui esegui queste query.

Aumenta il parametro innodb_sync_array_size

Il `innodb_sync_array_size` parametro controlla la dimensione dell'array di mutex/lock attesa in MySQL. Il valore predefinito di 1 funziona per carichi di lavoro generici, ma aumentandolo si può ridurre il conflitto di thread in caso di elevata concorrenza.

Quando il carico di lavoro mostra un numero crescente di thread in attesa:

- Monitora il numero di thread in attesa nel tuo carico di lavoro.
- Imposta un numero `innodb_sync_array_size` pari o superiore al numero di vCPU dell'istanza per suddividere la struttura di coordinamento dei thread e ridurre i conflitti.

> **Note**
>
> Per determinare il numero di v CPUs disponibili sulla tua istanza RDS, consulta le specifiche vCPU nei tipi di istanze Amazon RDS.

Implementa un pool di connessioni

MySQL assegna un tablespace dedicato a ogni sessione che crea una tabella temporanea. Questo tablespace rimane attivo fino al termine della connessione al database. Per gestire le risorse in modo più efficiente:

- Implementa il pool di connessioni per limitare il numero di tablespace temporanee attive.
- Riutilizza le connessioni esistenti invece di crearne di nuove per ogni operazione.

Regolazione di Aurora MySQL con stati del thread

La tabella seguente riassume gli stati del thread generali più comuni per Aurora MySQL.

Stato del thread generale	Descrizione
???	Questo stato del thread indica che un thread sta elaborando un'istruzione `SELECT` che richiede l'uso di una tabella temporanea interna per ordinare i dati.
???	Questo stato del thread indica che un thread sta leggendo e filtrando le righe per cercare una query per determinare la serie di risultati corretti.

creazione di indice di ordinamento

Lo stato del thread `creating sort index` indica che un thread sta elaborando un'istruzione `SELECT` che richiede l'uso di una tabella temporanea interna per ordinare i dati.

Argomenti

- Versioni del motore supportate
- Contesto
- Probabili cause di aumento delle attese
- Azioni

Versioni del motore supportate

Queste informazioni sullo stato del thread sono supportate per le seguenti versioni:

- Aurora MySQL versione 2 fino alla 2.09.2

Contesto

Lo stato `creating sort index` appare quando una query con una clausola `ORDER BY` o `GROUP BY` non può utilizzare un indice esistente per eseguire l'operazione. In questo caso, MySQL deve

eseguire un'operazione `filesort` più costosa. Questa operazione viene generalmente eseguita in memoria se la serie di risultati non è troppo grande. In caso contrario, comporta la creazione di un file su disco.

Probabili cause di aumento delle attese

La comparsa di `creating sort index` non indica di per sé un problema. Se le prestazioni sono scadenti e si vedono frequenti istanze di `creating sort index`, la causa più probabile è query lente con operatori `ORDER BY` o `GROUP BY`.

Azioni

La linea guida generale è trovare query con clausole `ORDER BY` o `GROUP BY` associate agli aumenti dello stato `creating sort index`. Quindi verificare se l'aggiunta di un indice o l'aumento della dimensione del buffer di ordinamento risolve il problema.

Argomenti

- Attiva il Performance Schema se non è attivato
- Identificare le query problematiche
- Esaminare i piani di spiegazione per l'utilizzo di filesort
- Aumenta la dimensione del buffer di ordinamento

Attiva il Performance Schema se non è attivato

Performance Insights segnala gli stati del thread solo se gli strumenti di Performance Schema non sono attivati. Quando gli strumenti di Performance Schema sono attivati, Performance Insights segnala invece gli eventi di attesa. Gli strumenti di Performance Schema forniscono informazioni dettagliate aggiuntive e strumenti migliori quando si esaminano potenziali problemi di prestazione. Pertanto, è consigliabile attivare il Performance Schema. Per ulteriori informazioni, consulta Panoramica dello schema di prestazioni per Approfondimenti sulle prestazioni su Aurora MySQL.

Identificare le query problematiche

Per identificare le query correnti che causano un aumento dello stato `creating sort index`, eseguire `show processlist` e vedi se una qualsiasi delle query ha `ORDER BY` o `GROUP BY`. Facoltativamente, eseguire `explain for connection N`, dove `N` è l'ID dell'elenco dei processi della query con `filesort`.

Per identificare le query precedenti che causano questi aumenti, attiva il registro delle query lente e trova le query con `ORDER BY`. Esegui `EXPLAIN` sulle query lente e cerca “utilizzo di filesort”. Per ulteriori informazioni, consulta Esaminare i piani di spiegazione per l'utilizzo di filesort.

Esaminare i piani di spiegazione per l'utilizzo di filesort

Identifica le istruzioni con clausole `ORDER BY` o `GROUP BY` che si traducono nello stato `creating sort index`.

Negli esempi seguenti viene illustrato come eseguire `explain` su una query. La colonna `Extra` mostra che questa query utilizza `filesort`.

```
mysql> explain select * from mytable order by c1 limit 10\G
*************************** 1. row ***************************
           id: 1
  select_type: SIMPLE
        table: mytable
   partitions: NULL
         type: ALL
possible_keys: NULL
          key: NULL
      key_len: NULL
          ref: NULL
         rows: 2064548
     filtered: 100.00
        Extra: Using filesort
1 row in set, 1 warning (0.01 sec)
```

L'esempio seguente mostra il risultato dell'esecuzione di `EXPLAIN` sulla stessa query dopo la creazione di un indice sulla colonna `c1`.

```
mysql> alter table mytable add index (c1);
```

```
mysql> explain select * from mytable order by c1 limit 10\G
*************************** 1. row ***************************
           id: 1
  select_type: SIMPLE
        table: mytable
   partitions: NULL
         type: index
possible_keys: NULL
          key: c1
```

```
      key_len: 1023
          ref: NULL
         rows: 10
     filtered: 100.00
        Extra: Using index
1 row in set, 1 warning (0.01 sec)
```

Per informazioni sull'utilizzo degli indici per l'ottimizzazione dell'ordinamento, consulta Ottimizzazione ORDER BY nella documentazione di MySQL.

Aumenta la dimensione del buffer di ordinamento

Per verificare se una query specifica richiedeva un processo `filesort` che ha creato un file su disco, controllare il valore della variabile `sort_merge_passes` dopo aver eseguito la query. Di seguito viene riportato un esempio.

```
mysql> show session status like 'sort_merge_passes';
+-------------------+-------+
| Variable_name     | Value |
+-------------------+-------+
| Sort_merge_passes | 0     |
+-------------------+-------+
1 row in set (0.01 sec)

--- run query
mysql> select * from mytable order by u limit 10;
--- run status again:

mysql> show session status like 'sort_merge_passes';
+-------------------+-------+
| Variable_name     | Value |
+-------------------+-------+
| Sort_merge_passes | 0     |
+-------------------+-------+
1 row in set (0.01 sec)
```

Se il valore di `sort_merge_passes` è alto, si consideri di aumentare la dimensione del buffer di ordinamento. Applica l'aumento a livello di sessione, perché l'aumento a livello globale può aumentare significativamente la quantità di RAM utilizzata da MySQL. L'esempio seguente mostra come modificare le dimensioni del buffer di ordinamento prima di eseguire una query.

```
mysql> set session sort_buffer_size=10*1024*1024;
```

```
Query OK, 0 rows affected (0.00 sec)
-- run query
```

invio dei dati

Lo stato del thread `sending data` indica che un thread sta leggendo e filtrando le righe per una query per determinare la serie di risultati corretti. Il nome è fuorviante perché implica che lo stato sta trasferendo i dati, non raccogliendo e preparando i dati da inviare in seguito.

Argomenti

- Versioni del motore supportate
- Context
- Probabili cause di aumento delle attese
- Azioni

Versioni del motore supportate

Queste informazioni sullo stato del thread sono supportate per le seguenti versioni:

- Aurora MySQL versione 2 fino alla 2.09.2

Context

Molti stati del thread sono di breve durata. Operazioni che si verificano durante `sending data` tendono a eseguire un numero elevato di letture su disco o cache. Pertanto, `sending data` è spesso lo stato di esecuzione più lungo per tutta la durata di una determinata query. Questo stato viene visualizzato quando Aurora MySQL sta effettuando le seguenti operazioni:

- Lettura ed elaborazione di righe per un'istruzione `SELECT`
- Esecuzione di un numero elevato di letture da disco o memoria
- Completamento di una lettura completa di tutti i dati da una query specifica
- Lettura di dati da una tabella, da un indice o dal lavoro di una procedura archiviata
- Ordinamento, raggruppamento o ordinamento dei dati

Dopo che lo stato `sending data` termina la preparazione dei dati, lo stato del thread `writing to net` indica il ritorno dei dati al client. Tipicamente `writing to net` viene acquisito solo quando la serie di risultati è molto grande o una latenza di rete grave rallenta il trasferimento.

Probabili cause di aumento delle attese

La comparsa di `sending data` non indica di per sé un problema. Se le prestazioni sono scadenti e si vedono frequenti istanze di `sending data`, le cause più probabili sono le seguenti.

Argomenti

- Query inefficiente
- Configurazione del server non ottimale

Query inefficiente

Nella maggior parte dei casi, ciò che è responsabile di questo stato è una query che non utilizza un indice appropriato per trovare la serie di risultati di una query specifica. Ad esempio, si consideri una query che legge una tabella di dati di 10 milioni per tutti gli ordini effettuati in California, in cui la colonna di stato non è indicizzata o è scarsamente indicizzata. In quest'ultimo caso, l'indice potrebbe esistere, ma l'ottimizzatore lo ignora a causa della bassa cardinalità.

Configurazione del server non ottimale

Se vengono visualizzate diverse query nello stato `sending data`, il server di database potrebbe essere configurato in modo errato. In particolare, il server potrebbe presentare i seguenti problemi:

- Il server di database non dispone di capacità di calcolo sufficiente: I/O del disco, tipo e velocità del disco, CPU o numero di CPU.
- Il server necessita delle risorse allocate, come il pool di buffer di InnoDB per le tabelle InnoDB o il buffer di chiavi per le tabelle MyIsam.
- Impostazioni di memoria per thread come `sort_buffer`, `read_buffer` `ejoin_buffer` consumano più RAM di quanto richiesto, portando il server fisico a necessitare risorse di memoria.

Azioni

La linea guida generale consiste nel trovare query che restituiscono un numero elevato di righe controllando il Performance Schema. Se la registrazione di query che non utilizzano indici è attivata, è anche possibile esaminare i risultati dei registri lenti.

Argomenti

- Attiva il Performance Schema se non è attivato
- Analisi delle impostazioni della memoria
- Esaminare i piani di spiegazione per l'utilizzo dell'indice
- Controllare il volume di dati restituiti
- Verifica la presenza di problemi di concorrenza
- Verificare la struttura delle query

Attiva il Performance Schema se non è attivato

Performance Insights segnala gli stati del thread solo se gli strumenti di Performance Schema non sono attivati. Quando gli strumenti di Performance Schema sono attivati, Performance Insights segnala invece gli eventi di attesa. Gli strumenti di Performance Schema forniscono informazioni dettagliate aggiuntive e strumenti migliori quando si esaminano potenziali problemi di prestazione. Pertanto, è consigliabile attivare il Performance Schema. Per ulteriori informazioni, consulta Panoramica dello schema di prestazioni per Approfondimenti sulle prestazioni su Aurora MySQL.

Analisi delle impostazioni della memoria

Esaminare le impostazioni di memoria per i pool di buffer primari. Assicurarsi che questi pool siano ridimensionati in modo appropriato per il carico di lavoro. Se il database utilizza più istanze del pool di buffer, assicurati che non siano divisi in molti piccoli pool di buffer. I thread possono utilizzare solo un pool di buffer alla volta.

Assicurarsi che le seguenti impostazioni di memoria utilizzate per ciascun thread siano dimensionate correttamente:

- read_buffer
- read_rnd_buffer
- sort_buffer
- join_buffer
- binlog_cache

A meno che non si abbiano ragioni specifiche per modificare le impostazioni, utilizzare i valori predefiniti.

Esaminare i piani di spiegazione per l'utilizzo dell'indice

Per queri nello stato del thread `sending data`, esaminare il piano per determinare se vengono utilizzati indici appropriati. Se una query non utilizza un indice utile, si prenda in considerazione l'aggiunta di suggerimenti come `USE INDEX` o `FORCE INDEX`. I suggerimenti possono aumentare o diminuire notevolmente il tempo necessario per eseguire una query, quindi occorre prestare attenzione prima di aggiungerli.

Controllare il volume di dati restituiti

Controllare le tabelle sottoposte a query e la quantità di dati che contengono. È possibile archiviare qualcuno di questi dati? In molti casi, la causa di tempi di esecuzione delle query scadenti non è il risultato del piano della query, ma il volume di dati da elaborare. Molti sviluppatori sono molto efficienti nell'aggiungere dati a un database, ma raramente considerano il ciclo di vita del set di dati nelle fasi di progettazione e sviluppo.

Si consiglia di cercare query che funzionino bene nei database con volumi ridotti ma che funzionino male nel sistema attuale. A volte gli sviluppatori che progettano query specifiche potrebbero non rendersi conto che queste query restituiscono 350.000 righe. Gli sviluppatori potrebbero aver sviluppato le query in un ambiente con volumi inferiori con set di dati più piccoli rispetto agli ambienti di produzione.

Verifica la presenza di problemi di concorrenza

Verifica se sono in esecuzione contemporaneamente più query dello stesso tipo. Alcune forme di query vengono eseguite in modo efficiente quando vengono eseguite da sole. Tuttavia, se forme di query simili vengono eseguite insieme o in volume elevato, possono causare problemi di concorrenza. Spesso questi problemi sono causati quando il database utilizza tabelle temporanee per restituire dei risultati. Un livello di isolamento restrittivo delle transazioni può anche causare problemi di concorrenza.

Se le tabelle vengono lette e scritte contemporaneamente, il database potrebbe utilizzare i blocchi. Per aiutare a identificare i periodi di prestazioni scadenti, esaminare l'uso dei database attraverso processi batch su larga scala. Per visualizzare i blocchi e i ripristini degli stati precedenti recenti, esaminare l'output del comando `SHOW ENGINE INNODB STATUS`.

Verificare la struttura delle query

Verifica se le query acquisite da questi stati utilizzano sottoquery. Questo tipo di query spesso porta a prestazioni scadenti perché il database compila i risultati internamente e li sostituisce nuovamente

nella query per la restituzione dei dati. Questo processo è un passaggio aggiuntivo per il database. In molti casi, questo passaggio può causare prestazioni scadenti in condizioni di caricamento altamente concomitante.

Controlla anche se le tue query utilizzano un numero elevato di clausole `ORDER BY` e `GROUP BY`. In tali operazioni, spesso il database deve prima formare l'intero set di dati in memoria. Quindi deve ordinarlo o raggrupparlo in modo specifico prima di restituirlo al client.

Ottimizzazione di Aurora MySQL con approfondimenti proattivi di Amazon DevOps Guru

Gli approfondimenti proattivi di DevOps Guru rilevano le condizioni problematiche note nel cluster di database Aurora MySQL prima che si verifichino. Con DevOps Guru è possibile:

- Evitare molti problemi comuni relativi al database controllando la configurazione del database rispetto alle impostazioni consigliate comuni.
- Ricevere gli avvisi per le criticità relative al parco istanze che, se non controllate, possono portare a problemi più gravi in seguito.
- Ricevere gli avvisi per i nuovi problemi individuati.

Ogni approfondimento proattivo contiene un'analisi della causa del problema e i suggerimenti per le azioni correttive.

Argomenti

- La lunghezza dell'elenco della cronologia di InnoDB è aumentata in modo significativo
- Il database sta creando tabelle temporanee su disco

La lunghezza dell'elenco della cronologia di InnoDB è aumentata in modo significativo

A partire dalla *`data`*, l'elenco della cronologia delle modifiche alle righe è aumentato in modo significativo, fino alla *`lunghezza`* di *`db-instance`*. Questo aumento influisce sulle prestazioni di chiusura delle query e arresto del database.

Argomenti

- Versioni del motore supportate
- Contesto

- Probabili cause di questo problema
- Azioni
- Parametri rilevanti

Versioni del motore supportate

Queste informazioni approfondite sono supportate per tutte le versioni di Aurora MySQL.

Contesto

Il sistema di transazioni InnoDB gestisce il controllo della concorrenza multiversione (MVCC). Quando una riga viene modificata, la versione precedente alla modifica dei dati da modificare viene archiviata come record di annullamento in un log di annullamenti. Ogni record di annullamento ha un riferimento al record di ripristino precedente, formando un elenco collegato.

L'elenco della cronologia di InnoDB è un elenco globale dei log di annullamento per le transazioni sottoposte a commit. MySQL utilizza l'elenco della cronologia per eliminare i record e le pagine di log quando le transazioni non richiedono più la cronologia. La lunghezza dell'elenco della cronologia è il numero totale di log di annullamento che contengono modifiche nell'elenco della cronologia. Ogni log contiene una o più modifiche. Se la lunghezza dell'elenco della cronologia di InnoDB aumenta in modo significativo, indicando un numero elevato di precedenti versioni di riga, la chiusura delle query e l'arresto del database diventano più lenti.

Probabili cause di questo problema

Le cause tipiche di un lungo elenco della cronologia sono:

- Transazioni di lunga durata, in lettura o scrittura
- Carico elevato in scrittura

Azioni

Consigliamo azioni diverse a seconda delle cause degli approfondimenti.

Argomenti

- Non iniziare alcuna operazione che comporti l'arresto del database finché non diminuiscono le dimensioni dell'elenco della cronologia di InnoDB
- Identificare e terminare le transazioni di lunga durata

- Usa Approfondimenti sulle prestazioni per individuare i principali host e i migliori utenti.

Non iniziare alcuna operazione che comporti l'arresto del database finché non diminuiscono le dimensioni dell'elenco della cronologia di InnoDB

Poiché un lungo elenco della cronologia di InnoDB rallenta l'arresto del database, riduci le dimensioni dell'elenco prima di iniziare le relative operazioni. Queste operazioni includono gli aggiornamenti della versione principale del database.

Identificare e terminare le transazioni di lunga durata

Puoi individuare le transazioni di lunga durata eseguendo la query `information_schema.innodb_trx`.

> **Note**
>
> Assicurarsi anche di cercare transazioni di lunga durata nelle repliche di lettura.

Per identificare e terminare le transazioni di lunga durata

1. Nel client SQL esegui la seguente query:

```
SELECT a.trx_id,
      a.trx_state,
      a.trx_started,
      TIMESTAMPDIFF(SECOND,a.trx_started, now()) as "Seconds Transaction Has Been
 Open",
      a.trx_rows_modified,
      b.USER,
      b.host,
      b.db,
      b.command,
      b.time,
      b.state
FROM  information_schema.innodb_trx a,
      information_schema.processlist b
WHERE a.trx_mysql_thread_id=b.id
  AND TIMESTAMPDIFF(SECOND,a.trx_started, now()) > 10
ORDER BY trx_started
```

2. Terminare ogni transazione di lunga durata con la stored procedure mysql.rds_kill.

Usa Approfondimenti sulle prestazioni per individuare i principali host e i migliori utenti.

Ottimizza le transazioni in modo che un numero elevato di righe modificate venga immediatamente sottoposto a commit.

Parametri rilevanti

A questo approfondimento è correlato il seguente parametro:

- `trx_rseg_history_len`: questa metrica del contatore può essere visualizzata in Approfondimenti sulle prestazioni, oltre che nella tabella `INFORMATION_SCHEMA.INNODB_METRICS`. Per ulteriori informazioni, consulta InnoDB INFORMATION_SCHEMA metrics table nella documentazione di MySQL.
- `RollbackSegmentHistoryListLength`: questa metrica di Amazon CloudWatch misura i log di undo che registrano le transazioni sottoposte a commit con record contrassegnati per l'eliminazione. Questi record sono pianificati per essere elaborati dall'operazione di rimozione InnoDB. La metrica `trx_rseg_history_len` ha lo stesso valore di `RollbackSegmentHistoryListLength`.
- `PurgeBoundary`: numero di transazione fino al quale è consentita l'eliminazione da InnoDB. Se questa metrica CloudWatch non avanza per lunghi periodi di tempo, è una buona indicazione che l'eliminazione da InnoDB è bloccata da transazioni di lunga durata. Per indagare, controlla le transazioni attive sul cluster di database Aurora MySQL. Questa metrica è disponibile solo per Aurora MySQL versione 2.11 e successive e versione 3.08 e successive.
- `PurgeFinishedPoint`: numero di transazione fino al quale viene eseguita l'eliminazione da InnoDB. Questa metrica CloudWatch consente di esaminare la velocità di eliminazione di InnoDB. Questa metrica è disponibile solo per Aurora MySQL versione 2.11 e successive e versione 3.08 e successive.
- `TransactionAgeMaximum`: età della transazione attiva in esecuzione meno recente. Questa metrica CloudWatch è disponibile solo per Aurora MySQL versione 3.08 e successive.
- `TruncateFinishedPoint`: numero di transazione fino al quale viene eseguito l'annullamento del troncamento. Questa metrica CloudWatch è disponibile solo per Aurora MySQL versione 2.11 e successive e versione 3.08 e successive.

Per ulteriori informazioni sulle metriche CloudWatch, consulta Parametri a livello di istanza per Amazon Aurora.

Il database sta creando tabelle temporanee su disco

L'utilizzo recente della tabella temporanea su disco è aumentato in modo significativo, fino al valore di *percentage*. Il database crea le tabelle temporanee al secondo come specificato in *number*. Ciò potrebbe influire sulle prestazioni e aumentare le operazioni su disco in *db-instance*.

Argomenti

- Versioni del motore supportate
- Context
- Probabili cause di questo problema
- Azioni
- Parametri rilevanti

Versioni del motore supportate

Queste informazioni approfondite sono supportate per tutte le versioni di Aurora MySQL.

Context

A volte è necessario che il server MySQL crei una tabella temporanea interna durante l'elaborazione di una query. Aurora MySQL può contenere una tabella temporanea interna in memoria, dove può essere elaborata dal motore di archiviazione TempTable o MEMORY o archiviata su disco da InnoDB. Per ulteriori informazioni, consulta Uso della tabella temporanea interna in MySQL nel Manuale di riferimento di MySQL.

Probabili cause di questo problema

Un aumento delle tabelle temporanee su disco indica l'uso di query complesse. Se la memoria configurata non è sufficiente per archiviare le tabelle temporanee in memoria, Aurora MySQL crea le tabelle su disco. Ciò può influire sulle prestazioni e aumentare le operazioni del disco.

Azioni

Consigliamo azioni diverse a seconda delle cause degli approfondimenti.

- Per Aurora MySQL versione 3, consigliamo di utilizzare il motore di archiviazione TempTable.
- Ottimizza le query in modo da restituire meno dati selezionando solo le colonne necessarie.

Se attivi lo schema delle prestazioni con tutti gli strumenti `statement` abilitati e temporizzati, puoi eseguire la query `SYS.statements_with_temp_tables` per recuperare l'elenco delle query che utilizzano le tabelle temporanee. Per ulteriori informazioni, consulta Prerequisiti per l'utilizzo dello schema sys nella documentazione di MySQL.

- Prendi in considerazione l'indicizzazione delle colonne coinvolte nelle operazioni di ordinamento e raggruppamento.
- Riscrivi le query per evitare le colonne `BLOB` e `TEXT`. Queste colonne utilizzano sempre il disco.
- Ottimizza i parametri del database `tmp_table_size` e `max_heap_table_size`.

  Il valore predefinito di questi parametri è 16 MiB. Quando si utilizza il motore di archiviazione MEMORY per tabelle temporanee in memoria, la dimensione massima è definita dal valore `tmp_table_size` o `max_heap_table_size`, a seconda di quale sia il più piccolo. Quando viene raggiunta questa dimensione massima, MySQL converte automaticamente la tabella temporanea interna in memoria in una tabella temporanea interna di InnoDB su disco. Per ulteriori informazioni, consulta Uso del motore di archiviazione TempTable in Amazon RDS per MySQL e Amazon Aurora MySQL.

  > **Note**
  >
  > Quando si creano in modo esplicito tabelle MEMORY con CREATE TABLE, solo la variabile `max_heap_table_size` determina la dimensione massima di una tabella. Inoltre, non è prevista la conversione in un formato su disco.

Parametri rilevanti

A questo approfondimento sono correlati i seguenti parametri di Approfondimenti sulle prestazioni:

- Created_tmp_disk_tables
- Created_tmp_tables

Per ulteriori informazioni, consulta Created_tmp_disk_tables nella documentazione di MySQL.

Query parallela per Amazon Aurora MySQL

Questo argomento illustra l'ottimizzazione delle prestazioni mediante query parallele per Amazon Aurora edizione compatibile con MySQL. Questa funzionalità utilizza un percorso di elaborazione speciale per alcune query che implicano grandi quantità di dati grazie all'architettura di storage condivisa di Aurora. Le query in parallelo forniscono i migliori risultati con cluster di database Aurora MySQL che includono tabelle con milioni di righe e query analitiche la cui elaborazione dura vari minuti o addirittura ore.

Argomenti

- Panoramica delle query in parallelo per Aurora MySQL
- Creazione di un cluster di database per query in parallelo in Aurora MySQL
- Attivazione e disattivazione di query in parallelo in Aurora MySQL
- Ottimizzazione delle query in parallelo in Aurora MySQL
- Identificazione delle istruzioni che utilizzano la funzione di query in parallelo per Aurora MySQL
- Monitoraggio di query in parallelo per Aurora MySQL
- Costrutti SQL per query in parallelo in Aurora MySQL

Panoramica delle query in parallelo per Aurora MySQL

Le query in parallelo Aurora MySQL sono un'ottimizzazione che parallelizza alcune operazioni di I/O e di calcolo utilizzate nell'elaborazione di query che implicano grandi quantità di dati. Il lavoro che viene parallelizzato include il recupero di righe dalla memoria, l'estrazione dei valori della colonna e la determinazione di quali righe corrispondono alle condizioni nelle clausole join e nella clausola `WHERE`. Questa attività che implicano grandi quantità di dati sono delegate (in termini di ottimizzazione del database, trasferite) a più nodi nel livello di storage distribuito di Aurora. Senza query parallela, ogni query porta tutti i dati scansionati a un singolo nodo all'interno del cluster Aurora MySQL (il nodo head) ed esegue l'elaborazione in quel nodo.

> **Tip**
>
> Anche il motore di database PostgreSQL ha una funzionalità chiamata "query parallela". Tale caratteristica non è correlata alla query Aurora parallela.

Quando la funzionalità di query in parallelo è abilitata, il motore Aurora MySQL determina automaticamente quando utilizzarla senza richiedere modifiche SQL come hint o attributi di tabella. Nelle sezioni seguenti, viene descritto quando una query in parallelo è applicata a una query. Scoprirai inoltre come assicurarti che una query in parallelo sia applicata dove necessario.

> **Note**
>
> L'ottimizzazione delle query parallele offre il massimo vantaggio per le query a esecuzione prolungata che richiedono minuti o ore per il completamento. Aurora MySQL generalmente non esegue l'ottimizzazione delle query parallele per query economiche. In genere, inoltre, non esegue l'ottimizzazione delle query parallele se un'altra tecnica di ottimizzazione ha più senso, come la memorizzazione nella cache delle query, la memorizzazione nella cache del pool di buffer o le ricerche degli indici. Consulta se ritieni che le query in parallelo non sono utilizzate dove dovrebber Identificazione delle istruzioni che utilizzano la funzione di query in parallelo per Aurora MySQL.

Argomenti

- Vantaggi
- Architettura
- Prerequisiti
- Limitazioni
- Costi di I/O con la query parallela

Vantaggi

Con la query parallela è possibile eseguire query analitiche a uso intensivo di dati sulle tabelle Aurora MySQL. In molti casi, è possibile ottenere un miglioramento delle prestazioni dell'ordine di grandezza rispetto alla divisione tradizionale della manodopera per l'elaborazione delle query.

Alcuni dei vantaggi derivanti dall'uso delle query in parallelo sono:

- Miglioramento delle prestazioni di I/O a seguito delle parallelizzazione delle richieste di lettura fisiche in molteplici nodi di storage.
- Traffico di rete ridotto. Aurora non trasmette intere pagine di dati dai nodi di storage al nodo head e quindi non filtra le righe e le colonne superflue in seguito. Aurora trasmette invece tuple compatte contenenti solo i valori di colonna necessari per il set di risultati.

- Riduzione dell'utilizzo della CPU sul nodo head grazie dell'elaborazione della funzione di trasferimento, al filtraggio delle righe e alla proiezione delle colonne per la clausola `WHERE`.
- Riduzione della sollecitazione della memoria a livello del pool di buffer. Le pagine elaborate dalla query parallela non vengono aggiunte al pool di buffer. Questo approccio riduce la possibilità che una scansione intensiva di dati sgomberi i dati utilizzati di frequente dal pool di buffer.
- Riduzione potenziale della duplicazione dei dati nella pipeline ETL (extract, transform, load), rendendo più pratica l'esecuzione di query analitiche di lunga durata su dati esistenti.

Architettura

La funzionalità di query in parallelo utilizza i principi architetturali chiave di Aurora MySQL: disaccoppiamento del motore di database dal sottosistema di storage e riduzione del traffico di rete mediante la razionalizzazione dei protocolli di comunicazione. Aurora MySQL utilizza queste tecniche per velocizzare le operazioni ad uso intensivo di scrittura, come l'elaborazione dei log di ripristino. La query in parallelo applica gli stessi principi alle operazioni di lettura.

> **Note**
>
> L'architettura delle query in parallelo Aurora MySQL differisce da quella delle funzionalità con nomi simili in altri sistemi di database. La query parallela Aurora MySQL non comporta l'utilizzo di multiprocessori simmetrici e quindi non dipende dalla capacità della CPU del server di database. L'elaborazione in parallelo avviene nel livello di storage, indipendentemente dal server Aurora MySQL utilizzato come coordinatore di query.

Per impostazione predefinita, senza le query in parallelo, l'elaborazione di una query Aurora comporta la trasmissione di dati non elaborati a un singolo nodo nel cluster Aurora (il nodo head). Aurora esegue quindi tutte le ulteriori elaborazioni per quella query in un singolo thread su quel singolo nodo. Con le query in parallelo, una grande parte di queste attività che implicano un numero elevato di operazioni di I/O e un utilizzo intensivo della CPU viene delegata ai nodi nel livello di storage. Solo le righe compatte del set di risultati sono ritrasmesse al nodo head, con righe già filtrate e valori di colonna già estratti e trasformati. Il miglioramento delle prestazioni risulta dalla riduzione del traffico di rete, da un minore utilizzo della CPU sul nodo head e dalla parallelizzazione delle operazioni di I/O nei nodi di storage. Il volume di operazioni di filtraggio, proiezione e I/O in parallelo non dipende dal numero di istanze database nel cluster Aurora che esegue la query.

Prerequisiti

Per l'utilizzo di tutte le funzionalità della query parallela è necessario un cluster di database Aurora MySQL che esegue la versione 2.09 o successiva. Se si dispone già di un cluster che si desidera utilizzare con query parallela, è possibile aggiornarlo a una versione compatibile e abilitare la query parallela in seguito. In questo caso, assicurarsi di seguire la procedura di aggiornamento in Considerazioni relative agli aggiornamenti per le query in parallelo perché i nomi delle impostazioni di configurazione e i valori predefiniti sono diversi in queste versioni più recenti.

Le istanze database nel cluster devono utilizzare le classi di istanza `db.r*`.

Assicurati che l'ottimizzazione del join hash sia attivata per il cluster. Per scoprire come, consulta Abilitazione dell'hash join per cluster di query parallele.

Per personalizzare parametri quali `aurora_parallel_query` e `aurora_disable_hash_join`, è necessario disporre di un gruppo di parametri personalizzato da utilizzare con il cluster. È possibile specificare questi parametri singolarmente per ogni istanza DB utilizzando un gruppo di parametri DB. Tuttavia, si consiglia di specificarli in un gruppo di parametri cluster DB. In questo modo, tutte le istanze DB nel cluster ereditano le stesse impostazioni per questi parametri.

Limitazioni

Le seguenti limitazioni si applicano alla funzionalità di query in parallelo:

- La query parallela non è supportata con la configurazione dell'archiviazione del cluster database Aurora I/O-Optimized.
- Non è possibile utilizzare la query parallela con le classi di istanza db.t2 o db.t3. Questa limitazione si applica anche se si richiede la query parallela utilizzando la variabile di sessione `aurora_pq_force`.
- La query parallela non si applica alle tabelle che utilizzano i formati di riga `COMPRESSED` o `REDUNDANT`. Utilizzare i formati di riga `COMPACT` o `DYNAMIC` per le tabelle che si intende utilizzare con la query parallela.
- Aurora utilizza un algoritmo basato sui costi per determinare se utilizzare il meccanismo di query parallela per ogni istruzione SQL. L'utilizzo di determinati costrutti SQL in un'istruzione può impedire query parallele o rendere la query parallela improbabile per tale istruzione. Per informazioni sulla compatibilità dei costrutti SQL con la query parallela, vedere Costrutti SQL per query in parallelo in Aurora MySQL.

- Ogni istanza database Aurora può eseguire solo un numero specifico di sessioni di query in parallelo simultanee. Se una query comporta più parti che utilizzano una query in parallelo, come sottoquery, join o operatori `UNION`, quelle fasi sono eseguite in sequenza. L'istruzione viene considerata solo come singola sessione di query in parallelo. Puoi monitorare il numero di sessioni attive mediante le variabili di stato di query in parallelo. Puoi verificare il limite delle sessioni simultanee per una determinata istanza database eseguendo una query sulla variabile di stato `Aurora_pq_max_concurrent_requests`.
- La query parallela è disponibile in tutte le Regioni Regioni AWS supportate da Aurora. Per la maggior parte delle Regioni Regioni AWS, la versione di Aurora MySQL minima richiesta per utilizzare la query parallela è 2.09.
- La query parallela è progettata per migliorare le prestazioni delle query a uso intensivo di dati. Non è progettata per eseguire query leggere.
- Ti consigliamo di utilizzare i nodi di lettura per le istruzioni SELECT, in particolare quelle che richiedono un uso intensivo di dati.

Costi di I/O con la query parallela

Se il tuo cluster Aurora MySQL utilizza una query parallela, potresti vedere un aumento dei valori di `VolumeReadIOPS`. Le query parallele non utilizzano il pool di buffer. Pertanto, sebbene le query siano veloci, questa elaborazione ottimizzata può comportare un aumento delle operazioni di lettura e degli addebiti associati.

I costi di I/O delle query parallele vengono misurati a livello di archiviazione e sono uguali o maggiori con la query parallela attivata. Il vantaggio sta nel miglioramento delle prestazioni delle query. I motivi per i quali i costi di I/O sono potenzialmente più elevati con la query parallela sono due:

- Anche se alcuni dei dati di una tabella si trovano nel pool di buffer, la query parallela richiede che tutti i dati vengano scansionati a livello di archiviazione, con conseguenti costi di I/O.
- L'esecuzione di una query parallela non prepara il pool di buffer. Di conseguenza, le successive esecuzioni della stessa query parallela comportano l'intero costo di I/O.

Creazione di un cluster di database per query in parallelo in Aurora MySQL

Per creare un cluster Aurora MySQL con una query in parallelo, aggiungervi nuove istanze o eseguire operazioni amministrative, utilizzi le stesse tecniche Console di gestione AWS e AWS CLI utilizzate con altri cluster Aurora MySQL. Puoi creare un cluster che sia compatibile con le query in parallelo.

Puoi anche creare un cluster di database compatibile con le query in parallelo ripristinando uno snapshot di un database Aurora compatibile con MySQL 5.6. Se non sai come creare un cluster Aurora MySQL, consulta i prerequisiti e altre informazioni utili in Creazione di un cluster database Amazon Aurora.

Scegli sempre la versione del motore Aurora MySQL più recente. Attualmente, tutte le versioni disponibili di Aurora MySQL supportano la query in parallelo. Hai la possibilità di attivare e disattivare la query in parallelo oppure utilizzare la query in parallelo con i cluster esistenti, se usi le versioni più recenti.

Per aggiungere nuove istanze database a un nuovo cluster o a un cluster ripristinato da uno snapshot, devi utilizzare le stesse tecniche utilizzate con altri cluster Aurora MySQL.

Puoi creare un cluster di query in parallelo utilizzando la console Amazon RDS o il AWS CLI.

Indice

- Creazione di un cluster abilitato per le query in parallelo mediante la console
- Creazione di un cluster abilitato per le query in parallelo mediante l'interfaccia a riga di comando

Creazione di un cluster abilitato per le query in parallelo mediante la console

Per creare un cluster abilitato per le query in parallelo mediante la console, procedi come descritto di seguito.

Per creare un cluster abilitato per le query in parallelo mediante l Console di gestione AWS

1. Seguire la procedura generale dell'Console di gestione AWS descritta in Creazione di un cluster database Amazon Aurora.
2. In Tipo di motore, seleziona Aurora MySQL.
3. Per Configurazione aggiuntiva, scegli un gruppo di parametri creato per il gruppo di parametri cluster di database. L'utilizzo di tale gruppo di parametri personalizzato è richiesto per Aurora MySQL 2.09 e versioni successive. Nel gruppo di parametri del cluster DB, specificare le impostazioni dei parametri `aurora_parallel_query=ON` e `aurora_disable_hash_join=OFF`. In questo modo si attiva la query parallela per il cluster e si abilita l'ottimizzazione dell'hash join che funziona in combinazione con la query parallela.

Per verificare che un nuovo cluster è compatibile con le query in parallelo

1. Creare un cluster utilizzando la tecnica precedente.
2. (Per Aurora MySQL versione 2 o 3) Verifica che l'impostazione di configurazione `aurora_parallel_query` sia true.

```
mysql> select @@aurora_parallel_query;
+-------------------------+
| @@aurora_parallel_query |
+-------------------------+
|                       1 |
+-------------------------+
```

3. (Per Aurora MySQL versione 2) Verifica che l'impostazione di configurazione di `aurora_disable_hash_join` sia false.

```
mysql> select @@aurora_disable_hash_join;
+----------------------------+
| @@aurora_disable_hash_join |
+----------------------------+
|                          0 |
+----------------------------+
```

4. Con alcune tabelle di grandi dimensioni e query a uso intensivo di dati, controllare i piani di query per confermare che alcune query utilizzano l'ottimizzazione delle query parallele. A tale scopo, segui la procedura in Identificazione delle istruzioni che utilizzano la funzione di query in parallelo per Aurora MySQL.

Creazione di un cluster abilitato per le query in parallelo mediante l'interfaccia a riga di comando

Per creare un cluster abilitato per le query in parallelo mediante l'interfaccia a riga di comando, procedi come descritto di seguito.

Per creare un cluster abilitato per le query in parallelo mediante l AWS CLI

1. (Facoltativo) Verificare quali versioni Aurora MySQL sono compatibili con i cluster di query parallele. Per fare ciò, utilizzare il comando `describe-db-engine-versions` e controllare il valore del campo `SupportsParallelQuery`. Per un esempio, consulta Verifica della compatibilità della versione Aurora MySQL per la query parallela.

2. (Facoltativo) Creare un gruppo di parametri cluster DB personalizzato con le impostazioni `aurora_parallel_query=ON` e `aurora_disable_hash_join=OFF`. Utilizzare comandi come i seguenti.

```
aws rds create-db-cluster-parameter-group --db-parameter-group-family aurora-
mysql8.0 --db-cluster-parameter-group-name pq-enabled-80-compatible
aws rds modify-db-cluster-parameter-group --db-cluster-parameter-group-name pq-
enabled-80-compatible \
  --parameters
 ParameterName=aurora_parallel_query,ParameterValue=ON,ApplyMethod=pending-reboot
aws rds modify-db-cluster-parameter-group --db-cluster-parameter-group-name pq-
enabled-80-compatible \
  --parameters
 ParameterName=aurora_disable_hash_join,ParameterValue=OFF,ApplyMethod=pending-
reboot
```

   Se si esegue questo passaggio, specificare l'opzione `--db-cluster-parameter-group-name` *`my_cluster_parameter_group`* nell'istruzione `create-db-cluster` successiva. Sostituire il nome del proprio gruppo di parametri. Se si omette questo passaggio, si crea il gruppo di parametri e lo si associa al cluster in seguito, come descritto in Attivazione e disattivazione di query in parallelo in Aurora MySQL.

3. Seguire la procedura generale dell'AWS CLI descritta in Creazione di un cluster database Amazon Aurora.

4. Specificare il set di opzioni seguente:

   - Per l'opzione `--engine`, utilizzare `aurora-mysql`. Questi valori producono cluster di query parallele compatibili con MySQL 5.7 o 8.0.
   - Per l' opzione `--db-cluster-parameter-group-name`, specificare il nome di un gruppo di parametri cluster DB creato e specificato per il valore del parametro `aurora_parallel_query=ON`. Se si omette questa opzione, è possibile creare il cluster con un gruppo di parametri predefinito e successivamente modificarlo per utilizzare tale gruppo di parametri personalizzato.
   - Per l'opzione `--engine-version`, utilizzare una versione Aurora MySQL compatibile con la query parallela. Utilizzare la procedura da Ottimizzazione delle query in parallelo in Aurora MySQL per ottenere un elenco di versioni, se necessario.

     Il codice di esempio seguente mostra come fare. Sostituire il proprio valore per ciascuna delle variabili di ambiente, ad esempio *`$CLUSTER_ID`*. In questo esempio è specificata anche

l'opzione `--manage-master-user-password` per generare la password dell'utente master e gestirla in Secrets Manager. Per ulteriori informazioni, consulta Gestione delle password con Amazon Aurora e Gestione dei segreti AWS. In alternativa, puoi utilizzare l'opzione `--master-password` per specificare e gestire personalmente la password.

```
aws rds create-db-cluster --db-cluster-identifier $CLUSTER_ID \
  --engine aurora-mysql --engine-version 8.0.mysql_aurora.3.04.1 \
  --master-username $MASTER_USER_ID --manage-master-user-password \
  --db-cluster-parameter-group-name $CUSTOM_CLUSTER_PARAM_GROUP

aws rds create-db-instance --db-instance-identifier ${INSTANCE_ID}-1 \
  --engine same_value_as_in_create_cluster_command \
  --db-cluster-identifier $CLUSTER_ID --db-instance-class $INSTANCE_CLASS
```

5. Verificare che la funzionalità di query in parallelo sia disponibile per il cluster creato o ripristinato.

   Verifica che l'impostazione di configurazione `aurora_parallel_query` sia presente. Se questa impostazione ha il valore 1, la query parallela è pronta per l'uso. Se questa impostazione ha il valore 0, impostarla su 1 prima di poter utilizzare la query parallela. In entrambi i casi, il cluster è in grado di eseguire query parallele.

   ```
   mysql> select @@aurora_parallel_query;
   +------------------------+
   | @@aurora_parallel_query|
   +------------------------+
   |                      1 |
   +------------------------+
   ```

Per ripristinare una snapshot in un cluster abilitato per le query in parallelo mediante l AWS CLI

1. Verificare quali versioni Aurora MySQL sono compatibili con i cluster di query parallele. Per fare ciò, utilizzare il comando `describe-db-engine-versions` e controllare il valore del campo `SupportsParallelQuery`. Per un esempio, consulta Verifica della compatibilità della versione Aurora MySQL per la query parallela. Decidere quale versione utilizzare per il cluster ripristinato.
2. Individuare uno snapshot del cluster compatibile con Aurora MySQL.
3. Seguire la procedura generale dell'AWS CLI descritta in Ripristino da uno snapshot cluster database.

   ```
   aws rds restore-db-cluster-from-snapshot \
   ```

```
  --db-cluster-identifier mynewdbcluster \
  --snapshot-identifier mydbclustersnapshot \
  --engine aurora-mysql
```

4. Verificare che la funzionalità di query in parallelo sia disponibile per il cluster creato o ripristinato. Utilizzare la stessa procedura di verifica descritta in Creazione di un cluster abilitato per le query in parallelo mediante l'interfaccia a riga di comando.

Attivazione e disattivazione di query in parallelo in Aurora MySQL

Quando la funzionalità di query in parallelo è abilitata, Aurora MySQL determina se utilizzarla al tempo di esecuzione per ogni query. In caso di join, unioni, sottoquery, ecc., Aurora MySQL determina se utilizzare le query in parallelo al runtime per ogni blocco di query. Per informazioni dettagliate, consulta Identificazione delle istruzioni che utilizzano la funzione di query in parallelo per Aurora MySQL e Costrutti SQL per query in parallelo in Aurora MySQL.

Puoi attivare o disattivare dinamicamente le query parallele di un'istanza database a livello globale e a livello di sessione mediante l'opzione aurora_parallel_query. È possibile modificare l'impostazione `aurora_parallel_query` nel gruppo di cluster DB per attivare o disattivare la query parallela per impostazione predefinita.

```
mysql> select @@aurora_parallel_query;
+------------------------+
| @@aurora_parallel_query|
+------------------------+
|                      1 |
+------------------------+
```

Per attivare o disattivare il parametro `aurora_parallel_query` a livello di sessione, utilizzare i metodi standard per modificare un'impostazione di configurazione client. Ad esempio, è possibile farlo tramite la riga `mysql` di comando o all'interno di un'applicazione JDBC o ODBC. Ad esempio, il comando sul client MySQL standard è `set session aurora_parallel_query = {'ON'/'OFF'}`. Puoi anche aggiungere il parametro a livello di sessione alla configurazione JDBC o al codice dell'applicazione per abilitare o disabilitare dinamicamente le query in parallelo.

È possibile modificare in modo permanente l'impostazione per il parametro `aurora_parallel_query`, per un'istanza DB specifica o per l'intero cluster. Se si specifica il valore del parametro in un gruppo di parametri DB, tale valore si applica solo a un'istanza DB specifica nel cluster. Se si specifica il valore del parametro in un gruppo di parametri cluster DB, tutte le istanze DB

del cluster ereditano la stessa impostazione. Per attivare il parametro `aurora_parallel_query`, utilizza le tecniche applicabili ai gruppi di parametri, come descritto in Gruppi di parametri per Amazon Aurora. Effettua la procedura riportata di seguito.

1. Creare un gruppo di parametri cluster personalizzato (scelta consigliata) o un gruppo di parametri DB personalizzato.
2. In questo gruppo di parametri, aggiornare `parallel_query` al valore desiderato.
3. A seconda che sia stato creato un gruppo di parametri cluster DB o un gruppo di parametri DB, collegare il gruppo di parametri al cluster Aurora o alle istanze DB specifiche in cui si prevede di utilizzare la funzionalità di query parallela.

> **Tip**
>
> Poiché `aurora_parallel_query` è un parametro dinamico, non richiede il riavvio del cluster dopo aver modificato questa impostazione. Tuttavia, qualsiasi connessione che utilizzava una query parallela prima di attivare l'opzione continuerà a farlo fino a quando la connessione non viene chiusa o l'istanza viene riavviata.

Puoi modificare il parametro di query in parallelo utilizzando l'operazione API ModifyDBClusterParameterGroup o l'operazione API ModifyDBParameterGroup o Console di gestione AWS.

Puoi attivare l'hash join per i cluster di query parallele, attivare e disattivare le query parallele utilizzando la console Amazon RDS o ilAWS CLI e sovrascrivere lo strumento di ottimizzazione di query parallele.

Indice

- Abilitazione dell'hash join per cluster di query parallele
- Attivazione e disattivazione della query parallela utilizzando la console
- Attivazione e disattivazione della query parallela utilizzando la CLI
- Sostituzione dell'ottimizzatore di query parallele

Abilitazione dell'hash join per cluster di query parallele

La query in parallelo viene in genere utilizzata per quei tipi di query che richiedono un uso intensivo di risorse che traggono vantaggio dall'ottimizzazione dell'hash join. Pertanto, è utile assicurarsi che i

join hash siano abilitati per i cluster in cui si prevede di utilizzare la query parallela. Per informazioni su come utilizzare i join hash in modo efficace, vedere Ottimizzazione di grandi query di join Aurora MySQL con hash join.

Attivazione e disattivazione della query parallela utilizzando la console

Puoi abilitare o disabilitare le query in parallelo a livello dell'istanza database utilizzando gruppi di parametri.

Per attivare o disattivare la query parallela per un cluster di database con Console di gestione AWS

1. Creare un gruppo di parametri personalizzato, come descritto in Gruppi di parametri per Amazon Aurora.
2. Imposta aurora_parallel_query su 1 (attivato) o 0 (disattivato). Per impostazione predefinita, il parametro aurora_parallel_query è abilitato sui cluster dove la funzionalità di query in parallelo è disponibile.
3. Se si utilizza un gruppo di parametri cluster personalizzato, collegarlo al cluster Aurora DB in cui si prevede di utilizzare la funzionalità di query parallela. Se usi un gruppo di parametri di database personalizzato, collegalo a una o più istanze database nel cluster. Si consiglia di utilizzare un gruppo di parametri cluster. In questo modo si assicura che tutte le istanze DB nel cluster abbiano le stesse impostazioni per la query parallela e le funzionalità associate, ad esempio hash join.

Attivazione e disattivazione della query parallela utilizzando la CLI

Puoi modificare il parametro di query in parallelo utilizzando il comando `modify-db-cluster-parameter-group` o `modify-db-parameter-group`. Scegliere il comando appropriato a seconda che si specifichi il valore di `aurora_parallel_query` tramite un gruppo di parametri cluster DB o un gruppo di parametri DB.

Per abilitare o disabilitare le query in parallelo per un cluster di database con CLI

- Modificare il parametro di query in parallelo utilizzando il comando `modify-db-cluster-parameter-group`. Utilizzare un comando come il seguente. Sostituire il nome appropriato per il proprio gruppo di parametri personalizzato. Sostituire `ON` o `OFF` per la parte `ParameterValue` dell'opzione `--parameters`.

```
$ aws rds modify-db-cluster-parameter-group --db-cluster-parameter-group-
name cluster_param_group_name \
  --parameters
 ParameterName=aurora_parallel_query,ParameterValue=ON,ApplyMethod=pending-reboot
{
    "DBClusterParameterGroupName": "cluster_param_group_name"
}

aws rds modify-db-cluster-parameter-group --db-cluster-parameter-group-
name cluster_param_group_name \
  --parameters ParameterName=aurora_pq,ParameterValue=ON,ApplyMethod=pending-reboot
```

Puoi anche abilitare o disabilitare le query in parallelo a livello di sessione, ad esempio tramite la riga di comando `mysql` o nell'applicazione JDBC o ODBC. A questo proposito, utilizza i metodi standard impiegati per modificare un'impostazione della configurazione client. Ad esempio, il comando sul client MySQL standard è `set session aurora_parallel_query = {'ON'/'OFF'}` per Aurora MySQL.

Puoi anche aggiungere il parametro a livello di sessione alla configurazione JDBC o al codice dell'applicazione per abilitare o disabilitare dinamicamente le query in parallelo.

Sostituzione dell'ottimizzatore di query parallele

È possibile utilizzare la variabile di sessione `aurora_pq_force` per sostituire l'ottimizzatore di query parallele e richiedere una query parallela per ogni query. Ti consigliamo di eseguire questa operazione solo a scopo di test. L'esempio seguente mostra come utilizzare `aurora_pq_force` in una sessione.

```
set SESSION aurora_parallel_query = ON;
set SESSION aurora_pq_force = ON;
```

Per disattivare la sostituzione, procedi come segue:

```
set SESSION aurora_pq_force = OFF;
```

Ottimizzazione delle query in parallelo in Aurora MySQL

Per ottimizzare il tuo cluster di database per le query parallele, considera quali cluster di database trarrebbero vantaggio dalle query parallele e se eseguire l'aggiornamento per le query parallele. Quindi, ottimizza il carico di lavoro e crea oggetti dello schema per le query parallele.

Indice

- Pianificazione di un cluster di query parallele
  - Verifica della compatibilità della versione Aurora MySQL per la query parallela
- Considerazioni relative agli aggiornamenti per le query in parallelo
  - Aggiornare cluster di query paralleli a Aurora MySQL versione 3
  - Aggiornamento ad Aurora MySQL 2.09 e versioni successive
- Ottimizzazione delle prestazioni per le query in parallelo
- Creazione di oggetti di schema per usufruire dei vantaggi delle query in parallelo

Pianificazione di un cluster di query parallele

La pianificazione di un cluster di database abilitato per la query parallela richiede di effettuare alcune scelte. Queste includono l'esecuzione di passaggi di installazione (ovvero la creazione o il ripristino di un cluster Aurora MySQL completo), e la decisione in quale misura abilitare la query parallela nel cluster di database.

Considerare quanto segue nell'ambito della pianificazione:

- Se si utilizza Aurora MySQL che è compatibile con MySQL 5.7, è necessario creare un cluster con provisioning. Quindi si attiva la query parallela utilizzando il parametro `aurora_parallel_query`.

  Se disponi di un cluster Aurora MySQL, non è necessario creare un nuovo cluster per utilizzare la query parallela. È possibile associare il cluster o istanze DB specifiche nel cluster a un gruppo di parametri con il parametro `aurora_parallel_query` abilitato. In questo modo, è possibile ridurre il tempo e gli sforzi necessari per impostare i dati rilevanti da utilizzare con query parallele.
- Pianificare le tabelle di grandi dimensioni che è necessario riorganizzare in modo da poter utilizzare la query parallela quando si accede ad esse. Potrebbe essere necessario creare nuove versioni di alcune tabelle di grandi dimensioni in cui la query parallela è utile. Ad esempio, potrebbe essere necessario rimuovere gli indici di ricerca full-text. Per informazioni dettagliate, consultare Creazione di oggetti di schema per usufruire dei vantaggi delle query in parallelo.

Verifica della compatibilità della versione Aurora MySQL per la query parallela

Per verificare quali versioni Aurora MySQL sono compatibili con i cluster di query parallele, utilizzare il comando `describe-db-engine-versions` AWS CLI e controllare il valore del

campo `SupportsParallelQuery`. L'esempio di codice seguente mostra come verificare quali combinazioni sono disponibili per i cluster di query paralleli in una regione AWS specificata. Assicurarsi di specificare la stringa di parametro `--query` completa su una singola riga.

```
aws rds describe-db-engine-versions --region us-east-1 --engine aurora-mysql \
--query '*[]|[?SupportsParallelQuery == `true`].[EngineVersion]' --output text
```

Il comando precedente genera un output simile al seguente: L'output potrebbe variare a seconda delle versioni Aurora MySQL disponibili nella regione AWS specificata.

```
5.7.mysql_aurora.2.11.1
5.7.mysql_aurora.2.11.2
5.7.mysql_aurora.2.11.3
5.7.mysql_aurora.2.11.4
5.7.mysql_aurora.2.11.5
5.7.mysql_aurora.2.11.6
5.7.mysql_aurora.2.12.0
5.7.mysql_aurora.2.12.1
5.7.mysql_aurora.2.12.2
5.7.mysql_aurora.2.12.3
5.7.mysql_aurora.2.12.4
8.0.mysql_aurora.3.04.0
8.0.mysql_aurora.3.04.1
8.0.mysql_aurora.3.04.2
8.0.mysql_aurora.3.04.3
8.0.mysql_aurora.3.05.2
8.0.mysql_aurora.3.06.0
8.0.mysql_aurora.3.06.1
8.0.mysql_aurora.3.07.0
8.0.mysql_aurora.3.07.1
```

Dopo aver iniziato a utilizzare query parallele con un cluster, è possibile monitorare le prestazioni e rimuovere gli ostacoli all'utilizzo delle query parallele. Per quelle istruzioni, Ottimizzazione delle prestazioni per le query in parallelo.

Considerazioni relative agli aggiornamenti per le query in parallelo

A seconda delle versioni originali e di destinazione quando si aggiorna un cluster di query parallelo, è possibile trovare miglioramenti nei tipi di query che la query parallela può ottimizzare. È inoltre possibile che non sia necessario specificare un parametro speciale della modalità motore per la

query parallela. Le sezioni seguenti spiegano le considerazioni quando si aggiorna un cluster in cui è attivata la query parallela.

Aggiornare cluster di query paralleli a Aurora MySQL versione 3

Diverse istruzioni SQL, clausole e tipi di dati hanno un supporto per query parallele nuove o migliorate a partire da Aurora MySQL versione 3. Quando si esegue l'aggiornamento da una versione precedente alla versione 3, verificare se ulteriori query possono trarre vantaggio dalle ottimizzazioni parallele delle query. Per informazioni su questi miglioramenti delle query parallele, vedere Tipi di dati di colonna,Tabelle partizionate, eFunzioni di aggregazione, clausole GROUP BY e clausole HAVING.

Se stai aggiornando un cluster di query parallele da Aurora MySQL 2.08 o versione precedente, sono disponibili anche informazioni sulle modifiche per attivare la query parallela. A questo scopo, leggere Aggiornamento ad Aurora MySQL 2.09 e versioni successive.

L'ottimizzazione del hash join è disattivata per impostazione predefinita in Aurora MySQL versione 3. L'opzione di configurazione `aurora_disable_hash_join` delle versioni precedenti non viene utilizzata.

Aggiornamento ad Aurora MySQL 2.09 e versioni successive

In Aurora MySQL 2.09 e versioni successive, la query parallela funziona per i cluster allocati e non richiede il parametro della modalità del motore `parallelquery`. Pertanto, non è necessario creare un nuovo cluster o ripristinare da uno snapshot esistente per utilizzare query parallele con queste versioni. È possibile utilizzare le procedure di aggiornamento descritte in Aggiornamento della versione secondaria o del livello di patch di un cluster di database Aurora MySQL per aggiornare il cluster a tale versione. È possibile aggiornare un cluster meno recente indipendentemente dal fatto che si trattasse di un cluster di query parallelo o di un cluster di cui è stato eseguito il provisioning. Per ridurre il numero di scelte nel menu Versione motore è possibile scegliere Mostra versioni che supportano la funzionalità di query parallela per filtrare le voci di tale menu. Quindi scegli Aurora MySQL 2.09 o versione successiva.

Dopo aver aggiornato un cluster di query parallele meno recente ad Aurora MySQL 2.09 o versione successiva, puoi attivare la query parallela nel cluster aggiornato. La query parallela è disattivata per impostazione predefinita in queste versioni e la procedura per abilitarla è diversa. Anche l'ottimizzazione hash join è disattivata per impostazione predefinita e deve essere abilitata separatamente. Pertanto, assicurarsi di attivare nuovamente queste impostazioni dopo l'aggiornamento. Per istruzioni su questa operazione, vedere Attivazione e disattivazione di query in parallelo in Aurora MySQL e Abilitazione dell'hash join per cluster di query parallele.

In particolare, è possibile abilitare la query parallela utilizzando i parametri di configurazione `aurora_parallel_query=ON` e `aurora_disable_hash_join=OFF` invece di `aurora_pq_supported` e `aurora_pq`. I parametri `aurora_pq_supported` e `aurora_pq` sono deprecati nelle versioni Aurora MySQL più recenti.

Nel cluster aggiornato, l'attributo `EngineMode` ha il valore `provisioned` invece di `parallelquery`. Per verificare se la query parallela è disponibile per una versione del motore specificata, ora si controlla il valore del campo `SupportsParallelQuery` nell'output del comando `describe-db-engine-versions` AWS CLI. Nelle versioni precedenti di Aurora MySQL, è stata verificata la presenza di `parallelquery` nell'elenco `SupportedEngineModes`.

Dopo l'aggiornamento ad Aurora MySQL 2.09 o versione successiva, puoi utilizzare le funzionalità indicate di seguito. Queste funzionalità non sono disponibili per i cluster di query parallele che eseguono versioni Aurora MySQL precedenti.

- Approfondimenti sulle prestazioni. Per ulteriori informazioni, consulta Monitoraggio del carico DB con Performance Insights su Amazon Aurora.
- Backtrack in corso Per ulteriori informazioni, consulta Backtrack di un cluster database Aurora.
- Arresto e avvio del cluster. Per ulteriori informazioni, consulta Avvio e arresto di un cluster di database Amazon Aurora.

Ottimizzazione delle prestazioni per le query in parallelo

Per gestire le prestazioni di un carico di lavoro con la funzionalità di query in parallelo, assicurati che questa sia utilizzata per le query per le quali tale ottimizzazione risulta più utile.

A questo proposito, puoi eseguire le operazioni elencate di seguito:

- Assicurarsi che le tabelle più grandi siano compatibili con la query parallela. È possibile modificare le proprietà della tabella o ricreare alcune tabelle in modo che le query per tali tabelle possano trarre vantaggio dall'ottimizzazione delle query parallele. Per scoprire come, consulta Creazione di oggetti di schema per usufruire dei vantaggi delle query in parallelo.
- Monitorare le query che utilizzano la funzionalità di query in parallelo. Per scoprire come, consulta Monitoraggio di query in parallelo per Aurora MySQL.
- Verificare che la query parallela venga utilizzata per le query a esecuzione prolungata e a uso intensivo di dati e con il giusto livello di concorrenza per il carico di lavoro. Per scoprire come, consulta Identificazione delle istruzioni che utilizzano la funzione di query in parallelo per Aurora MySQL.

- Ottimizzare il codice SQL per consentire la query parallela da applicare alle query previste. Per scoprire come, consulta Costrutti SQL per query in parallelo in Aurora MySQL.

Creazione di oggetti di schema per usufruire dei vantaggi delle query in parallelo

Prima di creare o modificare tabelle che si intende utilizzare per query parallele, assicurarsi di familiarizzare con i requisiti descritti in Prerequisiti e Limitazioni.

Poiché per le query in parallelo è necessario che le tabelle utilizzino l'impostazione `ROW_FORMAT=Compact` o `ROW_FORMAT=Dynamic`, verifica le impostazioni di configurazione Aurora per eventuali modifiche all'opzione di configurazione `INNODB_FILE_FORMAT`. Esegui l'istruzione `SHOW TABLE STATUS` per confermare il formato di riga per tutte le tabelle in un database.

Prima di modificare lo schema per consentire alla query parallela di lavorare con più tabelle, assicurarsi di eseguire il test. I test devono confermare se la query parallela determina un aumento netto delle prestazioni per tali tabelle. Assicurati inoltre che i requisiti di schema per le query in parallelo coincidano con i tuoi obiettivi.

Ad esempio, prima di passare da `ROW_FORMAT=Compressed` a `ROW_FORMAT=Compact` o `ROW_FORMAT=Dynamic`, compara le prestazioni dei carichi di lavoro rispetto alle tabelle di origine e a quelle nuove. Considera inoltre altri potenziali effetti come un aumento del volume di dati.

Identificazione delle istruzioni che utilizzano la funzione di query in parallelo per Aurora MySQL

In regola generale, non è necessario eseguire specifiche operazioni per utilizzare le query in parallelo. Quando una query soddisfa i requisiti essenziali della funzionalità di query in parallelo, l'ottimizzatore di query decide automaticamente se utilizzare tale funzionalità per ogni query.

Se esegui dei test in un ambiente di sviluppo o di test, è possibile che la funzione di query in parallelo non sia utilizzata in quanto le tabelle non contengono una quantità sufficiente di righe o volumi di dati. I dati associati alle tabelle, soprattutto quelle create recentemente per effettuare dei test, possono anche trovarsi interamente nel pool di buffer.

Durante il monitoraggio o l'ottimizzazione delle prestazioni del cluster, devi determinare se le query in parallelo sono utilizzate nei contesti appropriati. È possibile che sia necessario modificare lo schema di database, le impostazioni, le query SQL o persino la topologia del cluster e le impostazioni di connessione dell'applicazione per utilizzare questa funzionalità.

Per verificare se una query sta utilizzando la funzionalità di query in parallelo, consulta il piano di esecuzione della query (denominato anche "piano explain") eseguendo l'istruzione EXPLAIN. Per esempi su come espressioni, clausole e istruzioni SQL alterano l'output EXPLAIN per le query parallele, consulta Costrutti SQL per query in parallelo in Aurora MySQL.

L'esempio seguente illustra la differenza tra un piano query tradizionale e un piano di query in parallelo. Questo piano di spiegazione è da Query 3 del benchmark TPC-H. Molti degli esempi di query in questa sezione utilizzano tabelle del set di dati TPC-H. È possibile ottenere le definizioni di tabella, le query e il programma dbgen che genera dati di esempio dal sito Web TPC-H.

```
EXPLAIN SELECT l_orderkey,
  sum(l_extendedprice * (1 - l_discount)) AS revenue,
  o_orderdate,
  o_shippriority
FROM customer,
  orders,
  lineitem
WHERE c_mktsegment = 'AUTOMOBILE'
AND c_custkey = o_custkey
AND l_orderkey = o_orderkey
AND o_orderdate < date '1995-03-13'
AND l_shipdate > date '1995-03-13'
GROUP BY l_orderkey,
  o_orderdate,
  o_shippriority
ORDER BY revenue DESC,
  o_orderdate LIMIT 10;
```

Per impostazione predefinita, la query potrebbe avere un piano simile al seguente. Se il join hash non viene visualizzato nel piano di query, assicurarsi che l'ottimizzazione sia abilitata per prima.

```
+----+-------------+----------+------------+------+---------------+------+---------
+------+----------+----------+----------------------------------------------------+
| id | select_type | table    | partitions | type | possible_keys | key  | key_len |
 ref  | rows     | filtered | Extra                                              |
+----+-------------+----------+------------+------+---------------+------+---------
+------+----------+----------+----------------------------------------------------+
|  1 | SIMPLE      | customer | NULL       | ALL  | NULL          | NULL | NULL    |
 NULL |  1480234 |    10.00 | Using where; Using temporary; Using filesort       |
|  1 | SIMPLE      | orders   | NULL       | ALL  | NULL          | NULL | NULL    |
 NULL | 14875240 |     3.33 | Using where; Using join buffer (Block Nested Loop) |
```

```
|  1 | SIMPLE      | lineitem | NULL       | ALL  | NULL          | NULL | NULL    |
 NULL | 59270573 |     3.33 | Using where; Using join buffer (Block Nested Loop) |
+----+-------------+----------+------------+------+---------------+------+---------
+------+----------+----------+-----------------------------------------------------+
```

Per Aurora MySQL versione 3, è possibile abilitare il join hash a livello di sessione eseguendo la seguente istruzione.

```
SET optimizer_switch='block_nested_loop=on';
```

Per Aurora MySQL versione 2.09 e successive, è necessario impostare il parametro di database `aurora_disable_hash_join` o il parametro del cluster di database su `0` (disattivato). La disattivazione di `aurora_disable_hash_join` imposta il valore di `optimizer_switch` su `hash_join=on`.

Dopo aver attivato il join hash, prova a eseguire nuovamente l'istruzione `EXPLAIN`. Per informazioni su come utilizzare i join hash in modo efficace, vedere Ottimizzazione di grandi query di join Aurora MySQL con hash join.

Quando la funzionalità di query in parallelo di join hash è disabilitata, la query può avere un piano simile al seguente, che utilizza un hash join ma non le query in parallelo.

```
+----+-------------+----------+...+-----------
+-------------------------------------------------------------------+
| id | select_type | table    |...| rows      | Extra
                         |
+----+-------------+----------+...+-----------
+-------------------------------------------------------------------+
|  1 | SIMPLE      | customer |...|   5798330 | Using where; Using index; Using
 temporary; Using filesort       |
|  1 | SIMPLE      | orders   |...| 154545408 | Using where; Using join buffer (Hash
 Join Outer table orders)   |
|  1 | SIMPLE      | lineitem |...| 606119300 | Using where; Using join buffer (Hash
 Join Outer table lineitem) |
+----+-------------+----------+...+-----------
+-------------------------------------------------------------------+
```

Quando la funzionalità di query in parallelo è abilitata, due fasi di questo piano possono utilizzare l'ottimizzazione mediante query in parallelo, come indicato dalla colonna `Extra` nell'output `EXPLAIN`. L'elaborazione che implica un numero elevato di operazioni I/O e un utilizzo intensivo della CPU per tali fasi viene trasferita al livello di storage.

```
+----+...
+-----------------------------------------------------------------------------------------------
+
| id |...| Extra
                                                    |
+----+...
+-----------------------------------------------------------------------------------------------
+
|  1 |...| Using where; Using index; Using temporary; Using filesort
                                                    |
|  1 |...| Using where; Using join buffer (Hash Join Outer table orders); Using
 parallel query (4 columns, 1 filters, 1 exprs; 0 extra)   |
|  1 |...| Using where; Using join buffer (Hash Join Outer table lineitem); Using
 parallel query (4 columns, 1 filters, 1 exprs; 0 extra) |
+----+...
+-----------------------------------------------------------------------------------------------
+
```

Per informazioni su come interpretare l'output EXPLAIN per una query in parallelo e sulle parti delle istruzioni SQL a cui le query in parallelo possono essere applicate, consulta Costrutti SQL per query in parallelo in Aurora MySQL.

L'esempio di output seguente mostra i risultati dell'esecuzione della query precedente su un'istanza db.r4.2xlarge con un pool di buffer a freddo. L'esecuzione della query risulta molto più rapida con la funzionalità di query in parallelo.

> **Note**
>
> Poiché la durata dipende da molti fattori ambientali, i risultati potrebbero essere differenti. Esegui sempre dei test di prestazioni personali per confermare i risultati con il tuo ambiente, carico di lavoro e così via.

```
-- Without parallel query
+------------+-------------+-------------+----------------+
| l_orderkey | revenue     | o_orderdate | o_shippriority |
+------------+-------------+-------------+----------------+
|   92511430 | 514726.4896 | 1995-03-06  |              0 |
.
.
|   28840519 | 454748.2485 | 1995-03-08  |              0 |
```

```
+------------+-------------+-------------+----------------+
10 rows in set (24 min 49.99 sec)
```

```
-- With parallel query
+------------+-------------+-------------+----------------+
| l_orderkey | revenue     | o_orderdate | o_shippriority |
+------------+-------------+-------------+----------------+
|   92511430 | 514726.4896 | 1995-03-06  |              0 |
.
.
|   28840519 | 454748.2485 | 1995-03-08  |              0 |
+------------+-------------+-------------+----------------+
10 rows in set (1 min 49.91 sec)
```

Molti degli esempi di query in questa sezione utilizzano le tabelle di questo set di dati TPC-H, in particolare la tabella PART, che comporta 20 milioni di righe e la definizione seguente.

```
+---------------+---------------+------+-----+---------+-------+
| Field         | Type          | Null | Key | Default | Extra |
+---------------+---------------+------+-----+---------+-------+
| p_partkey     | int(11)       | NO   | PRI | NULL    |       |
| p_name        | varchar(55)   | NO   |     | NULL    |       |
| p_mfgr        | char(25)      | NO   |     | NULL    |       |
| p_brand       | char(10)      | NO   |     | NULL    |       |
| p_type        | varchar(25)   | NO   |     | NULL    |       |
| p_size        | int(11)       | NO   |     | NULL    |       |
| p_container   | char(10)      | NO   |     | NULL    |       |
| p_retailprice | decimal(15,2) | NO   |     | NULL    |       |
| p_comment     | varchar(23)   | NO   |     | NULL    |       |
+---------------+---------------+------+-----+---------+-------+
```

Esegui dei test con il tuo carico di lavoro per determinare se le singole istruzioni SQL possono beneficiare della funzionalità di query in parallelo. Utilizza quindi le tecniche di monitoraggio seguenti per determinare la frequenza d'uso delle query in parallelo con i carichi di lavoro reali nel tempo. Per tali carichi di lavoro, vengono applicati dei fattori supplementari come i limiti di simultaneità.

Monitoraggio di query in parallelo per Aurora MySQL

Se il tuo cluster Aurora MySQL utilizza una query parallela, potresti vedere un aumento dei valori di `VolumeReadIOPS`. Le query parallele non utilizzano il pool di buffer. Pertanto, sebbene le query

siano veloci, questa elaborazione ottimizzata può comportare un aumento delle operazioni di lettura e degli addebiti associati.

Oltre ai parametri di Amazon CloudWatch descritti in Visualizzazione delle metriche nella console Amazon RDS, Aurora fornisce altre variabili di stato globali. È possibile utilizzare queste variabili di stato globali per monitorare l'esecuzione di query parallele. Possono darti informazioni sul motivo per cui l'ottimizzatore potrebbe utilizzare o meno la query parallela in una determinata situazione. Per accedere a tali variabili, puoi utilizzare il comando `SHOW GLOBAL STATUS`. Queste variabili sono descritte nella tabella seguente.

Una sessione di query parallela non è necessariamente una mappatura uno a uno con le query eseguite dal database. Supponiamo ad esempio che tuo il piano di query comporti due fasi che utilizzano la funzionalità di query in parallelo. In tal caso, la query comporta due sessioni in parallelo e i contatori dei tentativi di richieste e delle richieste riuscite vengono incrementati di due.

Quando esegui dei test con la funzionalità di query in parallelo mediante l'istruzione `EXPLAIN`, prevedi un aumento dei contatori designati come "non scelti" anche se le query non sono in esecuzione. Quando utilizzi la funzionalità di query in parallelo in un ambiente di produzione, puoi verificare se i contatori "non scelti" aumentano più rapidamente del previsto. A questo punto, è possibile regolare in modo che la query parallela venga eseguita per le query previste. A tale scopo, è possibile modificare le impostazioni del cluster, il mix di query, le istanze database in cui è abilitata la query parallela e così via.

Questi contatori sono monitorati a livello di istanza database. Quando ti connetti a un endpoint differente, è possibile che i parametri siano differenti in quanto ogni istanza database esegue il relativo set di query in parallelo. I parametri possono variare anche quando l'endpoint di lettura si connette a un'istanza database differente per ogni sessione.

Nome	Descrizione
`Aurora_pq_bytes_returned`	Il numero di byte per le strutture di dati tuple trasmesse al nodo head nel corso delle query in parallelo. Dividi questo valore per 16.384 per compararlo a `Aurora_pq_pages_pushed_down` .
`Aurora_pq_max_concurrent_requests`	Il numero massimo di sessioni di query in parallelo eseguibili simultaneamente su questa

Nome	Descrizione
	istanza database Aurora. È un numero fisso che dipende dalla classe di istanza DB AWS.
`Aurora_pq_pages_pushed_down`	Il numero di pagine di dati (ognuna con una dimensione fissa di 16 KiB) per le quali una query in parallelo ha evitato una trasmissione di rete al nodo head.
`Aurora_pq_request_attempted`	Il numero di sessioni di query in parallelo necessarie. Questo valore può rappresentare più di una sessione per query, a seconda dei costrutti SQL come sottoquery e join.
`Aurora_pq_request_executed`	Il numero di sessioni di query in parallelo riuscite.
`Aurora_pq_request_failed`	Il numero di sessioni di query in parallelo che hanno restituito un errore al client. In alcuni casi, una richiesta di query in parallelo può non riuscire, ad esempio a causa di un problema nel livello di storage. In questi casi, viene eseguito un nuovo tentativo per la parte di query non riuscita utilizzando il meccanismo di query non in parallelo. Se anche il nuovo tentativo non riesce, viene restituito un errore al client e questo contatore viene incrementato.
`Aurora_pq_request_in_progress`	Il numero di sessioni di query in parallelo attualmente in corso. Questo numero si applica all'istanza database Aurora a cui sei connesso e non all'intero cluster di database Aurora. Per determinare se un'istanza database è prossima al relativo limite di simultaneità, compara questo valore a `Aurora_pq_max_concurrent_requests` .

Nome	Descrizione
`Aurora_pq_request_not_chosen`	Il numero di volte che una query in parallelo non è stata scelta per una query. Questo valore è la somma di vari altri contatori più granulari . Un'istruzione `EXPLAIN` può incrementare questo contatore anche se la query non viene effettivamente eseguita.
`Aurora_pq_request_not_chosen_below_min_rows`	Il numero di volte che una query in parallelo non è stata scelta a causa del numero di righe nella tabella. Un'istruzione `EXPLAIN` può incrementare questo contatore anche se la query non viene effettivamente eseguita.
`Aurora_pq_request_not_chosen_column_bit`	Numero di richieste di query parallele che utilizzano il percorso di elaborazione delle query non parallele a causa di un tipo di dati non supportato nell'elenco delle colonne proiettate.
`Aurora_pq_request_not_chosen_column_geometry`	Numero di richieste di query parallele che utilizzano il percorso di elaborazione delle query non parallele perché la tabella contiene colonne con il tipo di `GEOMETRY` dati. Per informazioni sulle versioni di Aurora MySQL che rimuovono questa limitazione, consulta Aggiornare cluster di query paralleli a Aurora MySQL versione 3.

Nome	Descrizione
`Aurora_pq_request_not_chosen_column_lob`	Numero di richieste di query parallele che utilizzano il percorso di elaborazione delle query non parallele perché la tabella contiene colonne con un tipo di dati `LOB` o colonne `VARCHAR` memorizzate esternamente a causa della lunghezza dichiarata. Per informazioni sulle versioni di Aurora MySQL che rimuovono questa limitazione, consulta Aggiornare cluster di query paralleli a Aurora MySQL versione 3.
`Aurora_pq_request_not_chosen_column_virtual`	Numero di richieste di query parallele che utilizzano il percorso di elaborazione delle query non parallele perché la tabella contiene una colonna virtuale.
`Aurora_pq_request_not_chosen_custom_charset`	Numero di richieste di query parallele che utilizzano il percorso di elaborazione delle query non parallele perché la tabella dispone di colonne con un set di caratteri personalizzato.
`Aurora_pq_request_not_chosen_fast_ddl`	Il numero di richieste di query parallele che utilizzano il percorso di elaborazione delle query non parallele perché la tabella è attualmente in fase di modifica da un'istruzione `ALTER` DDL veloce.
`Aurora_pq_request_not_chosen_ few_pages_outside_buffer_pool`	Il numero di volte che una query in parallelo non è stata scelta anche se meno del 95% dei dati della tabella era già nel pool di buffer, in quanto non c'erano dati non memorizzati nel buffer sufficienti per giustificare l'utilizzo della funzione di query in parallelo.

Nome	Descrizione
`Aurora_pq_request_not_chosen_full_text_index`	Numero di richieste di query parallele che utilizzano il percorso di elaborazione delle query non parallele perché la tabella dispone di indici full-text.
`Aurora_pq_request_not_chosen_high_buffer_pool_pct`	Il numero di volte che una query in parallelo non è stata scelta in quanto un'elevata percentuale di dati di tabella (attualmente maggiore del 95%) era già nel pool di buffer. In questi casi, l'ottimizzatore determina che la lettura dei dati del pool di buffer è più efficace. Un'istruzione `EXPLAIN` può incrementare questo contatore anche se la query non viene effettivamente eseguita.
`Aurora_pq_request_not_chosen_index_hint`	Numero di richieste di query parallele che utilizzano il percorso di elaborazione delle query non parallele perché la query include un hint di indice.
`Aurora_pq_request_not_chosen_innodb_table_format`	Il numero di richieste di query parallele che utilizzano il percorso di elaborazione delle query non parallele perché la tabella utilizza un formato di riga InnoDB non supportato. La query parallela Aurora si applica solo ai formati di riga `COMPACT`, `REDUNDANT` e `DYNAMIC`.
`Aurora_pq_request_not_chosen_long_trx`	Il numero di richieste di query in parallelo che hanno utilizzato il percorso di elaborazione di query non in parallelo in seguito all'avvio della query in una transazione di lunga durata. Un'istruzione `EXPLAIN` può incrementare questo contatore anche se la query non viene effettivamente eseguita.

Nome	Descrizione
`Aurora_pq_request_not_chosen_no_where_clause`	Numero di richieste di query parallele che utilizzano il percorso di elaborazione delle query non parallele perché la query non include alcuna clausola `WHERE`.
`Aurora_pq_request_not_chosen_range_scan`	Numero di richieste di query parallele che utilizzano il percorso di elaborazione delle query non parallele perché la query utilizza una scansione di intervallo su un indice.
`Aurora_pq_request_not_chosen_row_length_too_long`	Numero di richieste di query parallele che utilizzano il percorso di elaborazione delle query non parallele perché la lunghezza totale combinata di tutte le colonne è troppo lunga.
`Aurora_pq_request_not_chosen_small_table`	Il numero di volte che una query in parallelo non è stata scelta a causa della dimensione globale della tabella, come determinata dal numero di righe e dalla lunghezza media delle stesse. Un'istruzione `EXPLAIN` può increment are questo contatore anche se la query non viene effettivamente eseguita.
`Aurora_pq_request_not_chosen_temporary_table`	Numero di richieste di query parallele che utilizzano il percorso di elaborazione delle query non parallele perché la query fa riferimen to a tabelle temporanee che utilizzano i tipi di tabella `MyISAM` o `memory` non supportati.

Nome	Descrizione
`Aurora_pq_request_not_chosen_tx_isolation`	Numero di richieste di query parallele che utilizzano il percorso di elaborazione delle query non parallele perché la query utilizza un livello di isolamento delle transazioni non supportato. Sulle istanze del database di lettura, la query parallela si applica solo ai livelli di isolamento `REPEATABLE READ` e `READ COMMITTED` .
`Aurora_pq_request_not_chosen_update_delete_stmts`	Numero di richieste di query parallele che utilizzano il percorso di elaborazione delle query non parallele perché la query fa parte di un'istruzione `UPDATE` o `DELETE`.
`Aurora_pq_request_not_chosen_unsupported_access`	Il numero di richieste di query in parallelo che utilizzano il percorso di elaborazione di query non in parallelo in quanto la clausola `WHERE` non soddisfa i criteri delle query in parallelo . Ciò può avvenire se la query non richiede l'analisi di un grande volume di dati oppure se la query è un'istruzione `DELETE` o `UPDATE`.
`Aurora_pq_request_not_chosen_unsupported_storage_type`	Il numero di richieste di query in parallelo che utilizzano il percorso di elaborazione delle query non in parallelo perché il cluster database Aurora MySQL non utilizza una configurazione di archiviazione del cluster Aurora supportata. Questo parametro è disponibile in Aurora MySQL versione 3.04 e versioni successive. Per ulteriori informazioni, consulta Limitazioni.

Nome	Descrizione
`Aurora_pq_request_throttled`	Il numero di volte che una query in parallelo non è stata scelta a causa del numero massimo di query in parallelo simultanee già in esecuzione su una particolare istanza database Aurora.

Costrutti SQL per query in parallelo in Aurora MySQL

Nella sezione seguente è possibile trovare ulteriori dettagli sul motivo per cui particolari istruzioni SQL utilizzano o non utilizzano query parallele. Questa sezione illustra inoltre come le feature Aurora MySQL interagiscono con la query parallela. Queste informazioni dettagliate possono consentirti di diagnosticare i problemi relativi alle prestazioni di un cluster che utilizza le query in parallelo o di comprendere il modo in cui queste vengono applicate a un particolare carico di lavoro.

La decisione di utilizzare la funzione di query in parallelo dipende da molti fattori che intervengono al momento dell'esecuzione dell'istruzione. Di conseguenza, per certe query, tale funzionalità può essere utilizzata sempre, mai o solo in determinate circostanze.

> **Tip**
>
> Quando si visualizzano questi esempi in HTML, è possibile utilizzare il widget Copia nell'angolo in alto a destra di ogni elenco di codici per copiare il codice SQL e provarlo. L'utilizzo del widget Copia evita di copiare i caratteri aggiuntivi attorno alle righe del prompt `mysql>` e della continuazione `->`.

Argomenti

- Istruzioni EXPLAIN
- Clausola WHERE
- DDL (Data Definition Language)
- Tipi di dati di colonna
- Tabelle partizionate
- Funzioni di aggregazione, clausole GROUP BY e clausole HAVING

- Chiamate di funzione nella clausola WHERE
- Clausola LIMIT
- Operatori di confronto
- Join
- Sottoquery
- UNION
- Visualizzazioni
- Istruzioni DML
- Transazioni e blocco
- Indici B-Tree
- Indici di ricerca full-text (FTS)
- Colonne virtuali
- Meccanismi di caching integrati
- Suggerimenti per l'ottimizzazione
- Tabelle temporanee MyISAM

Istruzioni EXPLAIN

Come mostrato negli esempi in questa sezione, l'istruzione `EXPLAIN` indica se ogni fase di una query è idonea alla funzionalità di query in parallelo. Indica inoltre quali aspetti di una query possono essere trasferiti a un livello di storage. Di seguito sono riportati gli elementi più importanti del piano query:

- Un valore differente da `NULL` per la colonna `key` suggerisce che la query può essere eseguita in modo efficace mediante ricerche di indice e che l'utilizzo della funzionalità di query in parallelo è improbabile.
- Un valore basso della colonna `rows` (ovvero un valore non in milioni) suggerisce che la query non ha accesso a dati sufficienti per giustificare l'utilizzo della funzionalità di query in parallelo e che quindi è improbabile. Ciò significa che la query parallela è improbabile.
- La colonna `Extra` indica se l'uso della funzionalità di query in parallelo è previsto. Questo output è simile all'esempio seguente.

  ```
  Using parallel query (A columns, B filters, C exprs; D extra)
  ```

  Il numero `columns` rappresenta il numero di colonne a cui il blocco di query fa riferimento.

Il numero `filters` rappresenta il numero di predicati `WHERE` che rappresentano un semplice confronto tra un valore di colonna e una costante. Il confronto può ricercare un'uguaglianza, una disuguaglianza o un intervallo. Aurora può parallelizzare questi tipi di predicati in maniera più efficace.

Il numero `exprs` rappresenta il numero di espressioni come chiamate di funzione, operatori o altre espressioni che possono anche essere parallelizzate, sebbene meno efficacemente rispetto a una condizione di filtro.

Il numero `extra` rappresenta il numero di espressioni che non è possibile trasferire e che sono eseguite dal nodo head.

Consideriamo ad esempio il seguente output `EXPLAIN`.

```
mysql> explain select p_name, p_mfgr from part
    -> where p_brand is not null
    -> and upper(p_type) is not null
    -> and round(p_retailprice) is not null;
+----+-------------+-------+...+----------
+-------------------------------------------------------------------------------+
| id | select_type | table |...| rows     | Extra
                                 |
+----+-------------+-------+...+----------
+-------------------------------------------------------------------------------+
|  1 | SIMPLE      | part  |...| 20427936 | Using where; Using parallel query (5
 columns, 1 filters, 2 exprs; 0 extra) |
+----+-------------+-------+...+----------
+-------------------------------------------------------------------------------+
```

Le informazioni nella colonna `Extra` mostrano che cinque colonne sono estratte da ogni riga per valutare le condizioni della query e costruire il set di risultati. Un predicato `WHERE` comporta un filtro, ovvero una colonna che è testata direttamente nella clausola `WHERE`. Due clausole `WHERE` richiedono la valutazione di espressioni più complesse, che in questo caso implicano chiamate di funzione. Il campo `0 extra` conferma che tutte le operazioni nella clausola `WHERE` sono trasferite al livello di storage nell'ambito dell'elaborazione di query in parallelo.

Nei casi in cui una query in parallelo non viene scelta, puoi in genere dedurre il motivo dalle altre colonne dell'output `EXPLAIN`. Ad esempio, il valore `rows` può essere troppo basso oppure la colonna `possible_keys` può indicare che la query è in grado di utilizzare una ricerca di indice anziché

l'analisi di una grande quantità di dati. Nell'esempio seguente viene illustrata una query in cui l'ottimizzatore può stimare che la query eseguirà la scansione solo di un numero ridotto di righe. Lo fa in base alle caratteristiche della chiave primaria. In tal caso, nessuna query in parallelo è necessaria.

```
mysql> explain select count(*) from part where p_partkey between 1 and 100;
+----+-------------+-------+-------+---------------+---------+---------+------+------
+--------------------------+
| id | select_type | table | type  | possible_keys | key     | key_len | ref  | rows |
 Extra                    |
+----+-------------+-------+-------+---------------+---------+---------+------+------
+--------------------------+
|  1 | SIMPLE      | part  | range | PRIMARY       | PRIMARY | 4       | NULL |   99 |
 Using where; Using index |
+----+-------------+-------+-------+---------------+---------+---------+------+------
+--------------------------+
```

L'output che indica se una query in parallelo sarà utilizzata prende in considerazione tutti i fattori presenti al momento dell'esecuzione dell'istruzione `EXPLAIN`. L'ottimizzatore può effettuare una scelta differente quando la query è in esecuzione, se la situazione è cambiata nel frattempo. Ad esempio, `EXPLAIN` può indicare che un'istruzione utilizzerà una query in parallelo. Tuttavia, quando successivamente la query viene eseguita, può non usare la query in parallelo in base alle condizioni in quel momento. Tali condizioni possono includere diverse altre query parallele in esecuzione contemporaneamente. Possono anche includere righe che vengono eliminate dalla tabella, un nuovo indice in fase di creazione, troppo tempo che passa all'interno di una transazione aperta e così via.

Clausola WHERE

Per utilizzare l'ottimizzazione mediante query in parallelo, una query deve includere una clausola `WHERE`.

Questa ottimizzazione accelera molti tipi di espressioni utilizzati nella clausola `WHERE`:

- Confronti semplici tra un valore di colonna e una costante, noti come filtri. Questi confronti sono ottimali quando trasferiti al livello di storage. Il numero di espressioni di filtro in una query è indicato nell'output `EXPLAIN`.
- Anche altri tipi di espressioni nella clausola `WHERE` sono trasferiti al livello di storage quando possibile. Il numero di tali espressioni in una query è indicato nell'output `EXPLAIN`. Queste espressioni possono essere chiamate di funzione, operatori `LIKE`, espressioni `CASE` e così via.

- Alcune funzioni e operatori non sono attualmente trasferiti dalla query in parallelo. Il numero di tali espressioni in una query è indicato come contatore `extra` nell'output `EXPLAIN`. Il resto della query può ancora utilizzare la funzionalità di query in parallelo.
- Sebbene le espressioni nell'elenco di selezione non siano trasferite, le query contenenti tali funzioni possono ancora beneficiare di una riduzione del traffico di rete per i risultati intermedi delle query in parallelo. Ad esempio, le query che chiamano funzioni di aggregazione nell'elenco di selezione possono beneficiare delle query in parallelo, anche se le funzioni di aggregazione non sono trasferite.

Ad esempio, la query seguente esegue un'analisi dell'intera tabella ed elabora tutti i valori per la colonna `P_BRAND`: Tuttavia, non utilizza la funzionalità di query in parallelo in quanto la query non include alcuna clausola `WHERE`.

```
mysql> explain select count(*), p_brand from part group by p_brand;
+----+-------------+-------+------+---------------+------+---------+------+----------
+----------------------------------+
| id | select_type | table | type | possible_keys | key  | key_len | ref  | rows     |
 Extra                            |
+----+-------------+-------+------+---------------+------+---------+------+----------
+----------------------------------+
|  1 | SIMPLE      | part  | ALL  | NULL          | NULL | NULL    | NULL | 20427936 |
 Using temporary; Using filesort |
+----+-------------+-------+------+---------------+------+---------+------+----------
+----------------------------------+
```

Al contrario, la query seguente include predicati `WHERE` che filtrano i risultati, di conseguenza la funzionalità di query in parallelo può essere utilizzata:

```
mysql> explain select count(*), p_brand from part where p_name is not null
    ->   and p_mfgr in ('Manufacturer#1', 'Manufacturer#3') and p_retailprice > 1000
    -> group by p_brand;
+----+...+----------
+-------------------------------------------------------------------------------------------
+
| id |...| rows     | Extra
                                          |
+----+...+----------
+-------------------------------------------------------------------------------------------
+
```

```
|  1 |...| 20427936 | Using where; Using temporary; Using filesort; Using parallel
 query (5 columns, 1 filters, 2 exprs; 0 extra) |
+----+...+----------
+-----------------------------------------------------------------------------------------
+
```

Se l'ottimizzatore stima che il numero di righe restituite per un blocco di query è basso, la funzionalità di query in parallelo non viene utilizzata per quel blocco. L'esempio seguente mostra uno scenario in cui un operatore "maggiore di" nella colonna di chiave primaria viene applicato a milioni di righe, comportando l'utilizzo della funzionalità di query in parallelo. Per l'operatore inverso "minore di" viene stimata l'applicazione soltanto ad alcune righe e tale funzionalità non è quindi utilizzata.

```
mysql> explain select count(*) from part where p_partkey > 10;
+----+...+----------
+-------------------------------------------------------------------------------+
| id |...| rows     | Extra
          |
+----+...+----------
+-------------------------------------------------------------------------------+
|  1 |...| 20427936 | Using where; Using parallel query (1 columns, 1 filters, 0 exprs;
 0 extra) |
+----+...+----------
+-------------------------------------------------------------------------------+

mysql> explain select count(*) from part where p_partkey < 10;
+----+...+------+--------------------------+
| id |...| rows | Extra                    |
+----+...+------+--------------------------+
|  1 |...|    9 | Using where; Using index |
+----+...+------+--------------------------+
```

DDL (Data Definition Language)

In Aurora MySQL versione 2, le query parallele sono disponibili solo per le tabelle che non hanno in sospeso alcuna operazione DDL (Data Definition Language) veloce. In Aurora MySQL versione 3, è possibile utilizzare una query parallela su una tabella contemporaneamente a un'operazione DDL istantanea.

Il DDL istantaneo in Aurora MySQL versione 3 sostituisce la funzionalità DDL veloce in Aurora MySQL versione 2. Per informazioni sulle istruzioni DDL istantanee, consultare DDL istantaneo (Aurora MySQL versione 3).

Tipi di dati di colonna

In Aurora MySQL versione 3, la query parallela può funzionare con tabelle contenenti colonne con tipi di dati `TEXT`,`BLOB`,`JSON`, e `GEOMETRY`. Può funzionare anche con colonne `VARCHAR` e `CHAR` con una lunghezza massima dichiarata superiore a 768 byte. Se la query fa riferimento a colonne contenenti tipi di oggetti così grandi, il lavoro aggiuntivo per recuperarli aggiunge un sovraccarico all'elaborazione delle query. In tal caso, verificare se la query può omettere i riferimenti a tali colonne. In caso contrario, eseguire i valori di riferimento per confermare se tali query sono più veloci con la query parallela attivata o disattivata.

In Aurora MySQL versione 2, la query parallela presenta le seguenti limitazioni per i tipi di oggetti di grandi dimensioni:

- I tipi di dati `TEXT`, `BLOB`, `JSON` e `GEOMETRY` non sono supportati con le query in parallelo. Una query che fa riferimento alle colonne di questi tipi non può utilizzare le query in parallelo.
- Le colonne a lunghezza variabile (tipi di dati `VARCHAR` e `CHAR`) sono compatibili con le query in parallelo fino a una lunghezza dichiarata massima di 768 byte. Una query che fa riferimento a qualsiasi colonna dei tipi dichiarati con una lunghezza massima superiore non può utilizzare le query in parallelo. Per le colonne che utilizzano set di caratteri multibyte, il limite di byte prende in considerazione il numero massimo di byte nel set di caratteri. Ad esempio, per il set di caratteri `utf8mb4` (che ha un limite di lunghezza di 4 byte), una colonna `VARCHAR(192)` è compatibile con le query in parallelo mentre una colonna `VARCHAR(193)` non lo è.

Tabelle partizionate

È possibile utilizzare tabelle partizionate con query in parallelo in Aurora MySQL versione 3. Poiché le tabelle partizionate sono rappresentate internamente come più tabelle più piccole, una query che utilizza query parallele su una tabella non partizionata potrebbe non utilizzare query parallele su una tabella partizionata identica. Aurora MySQL considera se ogni partizione è abbastanza grande da poter essere idonea per l'ottimizzazione delle query parallele, anziché valutare le dimensioni dell'intera tabella. Controllare se la variabile di stato `Aurora_pq_request_not_chosen_small_table` viene incrementata se una query in una tabella partizionata non utilizza una query parallela quando ci si aspetta.

Ad esempio, si consideri una tabella partizionata con `PARTITION BY HASH (`*`column`*`) PARTITIONS 2` e un'altra tabella partizionata con `PARTITION BY HASH (`*`column`*`) PARTITIONS 10`. Nella tabella con due partizioni, le partizioni sono cinque volte più grandi della tabella con dieci partizioni. Pertanto, è più probabile che la query parallela venga utilizzata per le query sulla tabella

con meno partizioni. Nell'esempio seguente, la tabella PART_BIG_PARTITIONS ha due partizioni e la PART_SMALL_PARTITIONS ha dieci partizioni. Con dati identici, è più probabile che la query parallela venga utilizzata per la tabella con meno partizioni di grandi dimensioni.

```
mysql> explain select count(*), p_brand from part_big_partitions where p_name is not
 null
    ->   and p_mfgr in ('Manufacturer#1', 'Manufacturer#3') and p_retailprice > 1000
 group by p_brand;
+----+-------------+---------------------+------------
+-------------------------------------------------------------------------------------
+
| id | select_type | table               | partitions | Extra
                                                                                      |
+----+-------------+---------------------+------------
+-------------------------------------------------------------------------------------
+
|  1 | SIMPLE      | part_big_partitions | p0,p1      | Using where; Using temporary;
 Using parallel query (4 columns, 1 filters, 1 exprs; 0 extra; 1 group-bys, 1 aggrs) |
+----+-------------+---------------------+------------
+-------------------------------------------------------------------------------------
+

mysql> explain select count(*), p_brand from part_small_partitions where p_name is not
 null
    ->   and p_mfgr in ('Manufacturer#1', 'Manufacturer#3') and p_retailprice > 1000
 group by p_brand;
+----+-------------+-----------------------+------------------------------
+------------------------------+
| id | select_type | table                 | partitions                   | Extra
                   |
+----+-------------+-----------------------+------------------------------
+------------------------------+
|  1 | SIMPLE      | part_small_partitions | p0,p1,p2,p3,p4,p5,p6,p7,p8,p9 | Using
 where; Using temporary |
+----+-------------+-----------------------+------------------------------
+------------------------------+
```

Funzioni di aggregazione, clausole GROUP BY e clausole HAVING

Le query che implicano funzioni di aggregazione sono spesso adatte alle query in parallelo, in quanto comportano l'analisi di un gran numero di righe in tabelle di grandi dimensioni.

In Aurora MySQL 3, la query parallela può ottimizzare le query che coinvolgono chiamate di funzioni aggregate nell'elenco di selezione e nella clausola `HAVING`.

Prima di Aurora MySQL 3 le chiamate di funzione di aggregazione presenti nell'elenco di selezione o nella clausola `HAVING` non sono trasferite al livello di archiviazione . Tuttavia, la funzionalità di query in parallelo può sempre migliorare le prestazioni di queste query con funzioni di aggregazione. A questo proposito, estrae dapprima in parallelo i valori di colonna dalle pagine di dati non elaborati nel livello di storage. Ritrasmette quindi quei valori al nodo head in un formato di tupla compatto anziché come pagine di dati complete. Come sempre, la query richiede almeno un predicato `WHERE` affinché la funzione di query in parallelo sia attivata.

Gli esempi semplici seguenti illustrano i tipi di query di aggregazione che possono beneficiare della funzionalità di query in parallelo restituendo risultati intermedi in formato compatto al nodo head, filtrando le righe non corrispondenti dai risultati intermedi oppure con entrambe le operazioni.

```
mysql> explain select sql_no_cache count(distinct p_brand) from part where p_mfgr =
 'Manufacturer#5';
+----+...+-------------------------------------------------------------------------+
| id |...| Extra                                                                   |
+----+...+-------------------------------------------------------------------------+
|  1 |...| Using where; Using parallel query (2 columns, 1 filters, 0 exprs; 0 extra) |
+----+...+-------------------------------------------------------------------------+

mysql> explain select sql_no_cache p_mfgr from part where p_retailprice > 1000 group by
 p_mfgr having count(*) > 100;
+----+...
+-------------------------------------------------------------------------------------------------------
+
| id |...| Extra
                                  |
+----+...
+-------------------------------------------------------------------------------------------------------
+
|  1 |...| Using where; Using temporary; Using filesort; Using parallel query (3
 columns, 0 filters, 1 exprs; 0 extra) |
+----+...
+-------------------------------------------------------------------------------------------------------
+
```

Chiamate di funzione nella clausola WHERE

Aurora consente di applicare l'ottimizzazione mediante query in parallelo per le chiamate alla maggior parte delle funzioni integrate nella clausola `WHERE`. La parallelizzazione di queste chiamate di funzione consente di ridurre il carico di lavoro della CPU mediante il nodo head. La valutazione delle funzioni di predicato in parallelo durante le prime fasi di una query consente a Aurora di ridurre al minimo la quantità di dati trasmessi ed elaborati nelle fasi successive.

Attualmente, la parallelizzazione non viene applicata alle chiamate di funzione nell'elenco di selezione. Queste funzioni sono valutate dal nodo head anche se chiamate di funzione identiche sono presenti nella clausola `WHERE`. I valori di origine delle colonne pertinenti sono inclusi nelle tuple ritrasmesse dai nodi di storage al nodo head. Il nodo head esegue tutte le trasformazioni come `UPPER`, `CONCATENATE`, ecc., allo scopo di generare i valori finali del set di risultati.

Nell'esempio seguente, una query in parallelo parallelizza la chiamata a `LOWER` in quanto è presente nella clausola `WHERE`. La query in parallelo non viene applicata alle chiamate a `SUBSTR` e `UPPER` in quanto queste sono presenti nell'elenco di selezione.

```
mysql> explain select sql_no_cache distinct substr(upper(p_name),1,5) from part
    -> where lower(p_name) like '%cornflower%' or lower(p_name) like '%goldenrod%';
+----+...
+-------------------------------------------------------------------------------------------
+
| id |...| Extra
                |
+----+...
+-------------------------------------------------------------------------------------------
+
|  1 |...| Using where; Using temporary; Using parallel query (2 columns, 0 filters, 1
 exprs; 0 extra) |
+----+...
+-------------------------------------------------------------------------------------------
+
```

Le stesse considerazioni sono valide per altre espressioni, come espressioni `CASE` o operatori `LIKE`. L'esempio seguente mostra una query in parallelo che valuta l'espressione `CASE` e gli operatori `LIKE` nella clausola `WHERE`.

```
mysql> explain select p_mfgr, p_retailprice from part
    -> where p_retailprice > case p_mfgr
```

```
    ->   when 'Manufacturer#1' then 1000
    ->   when 'Manufacturer#2' then 1200
    ->   else 950
    -> end
    -> and p_name like '%vanilla%'
    -> group by p_retailprice;
+----+...
+-------------------------------------------------------------------------------------------------
+
| id |...| Extra
                                |
+----+...
+-------------------------------------------------------------------------------------------------
+
|  1 |...| Using where; Using temporary; Using filesort; Using parallel query (4
 columns, 0 filters, 2 exprs; 0 extra) |
+----+...
+-------------------------------------------------------------------------------------------------
+
```

Clausola LIMIT

Le query in parallelo non sono correntemente utilizzate per blocchi di query che includono una clausola `LIMIT`. Le query in parallelo possono ancora essere utilizzate per fasi di query precedenti con le clausole `GROUPBY`, `ORDER BY` oppure con join.

Operatori di confronto

L'ottimizzatore stima il numero di righe da analizzare per valutare gli operatori di confronto e determina se utilizzare la funzionalità di query in parallelo in base a quella stima.

Il primo esempio qui sotto mostra che un confronto delle uguaglianze con la colonna di chiave primaria può essere eseguito in modo efficace senza la funzionalità di query in parallelo. Il secondo esempio mostra che un confronto simile con una colonna non indicizzata richiede l'analisi di milioni di righe e che di conseguenza può beneficiare della funzionalità di query in parallelo.

```
mysql> explain select * from part where p_partkey = 10;
+----+...+------+-------+
| id |...| rows | Extra |
+----+...+------+-------+
|  1 |...|    1 | NULL  |
+----+...+------+-------+
```

```
mysql> explain select * from part where p_type = 'LARGE BRUSHED BRASS';
+----+...+----------
+-------------------------------------------------------------------------------+
| id |...| rows     | Extra
          |
+----+...+----------
+-------------------------------------------------------------------------------+
|  1 |...| 20427936 | Using where; Using parallel query (9 columns, 1 filters, 0 exprs;
 0 extra) |
+----+...+----------
+-------------------------------------------------------------------------------+
```

Le stesse considerazioni sono valide per i test di diseguaglianza e i confronti di intervalli come "minore di", "maggiore di" o "uguale a" oppure `BETWEEN`. L'ottimizzatore stima il numero di righe da analizzare e in base al volume globale di operazioni di I/O determina se l'utilizzo della funzionalità di query in parallelo è giustificato.

Join

Le query join con tabelle di grandi dimensioni comportano in genere operazioni con grandi quantità di dati che beneficiano dell'ottimizzazione mediante query in parallelo. I confronti dei valori di colonna tra molteplici tabelle (ovvero i predicati di join stessi) non sono attualmente parallelizzati. La funzionalità di query in parallelo può tuttavia trasferire una parte dell'elaborazione interna per altre fasi di join, come la costruzione del filtro Bloom durante un hash join. Inoltre, può essere applicata a query join anche senza una clausola `WHERE`. Di conseguenza, una query join è un'eccezione alla regola secondo cui una clausola `WHERE` è necessaria per utilizzare la query in parallelo.

Ogni fase dell'elaborazione di un join è valutata per determinare se è idonea per la funzionalità di query in parallelo. Se più fasi possono utilizzare tale funzionalità, sono eseguite in sequenza. Ogni query join viene pertanto contabilizzata come singola sessione di query in parallelo in termini di limiti di simultaneità.

Ad esempio, quando una query join include predicati `WHERE` per filtrare le righe di una delle tabelle unite in join, tale opzione di filtraggio può utilizzare la funzionalità di query in parallelo. Un altro esempio consiste in una query join che utilizza il meccanismo di hash join per unire in join una tabella di grandi dimensioni e una di piccole dimensioni. In tal caso, la funzionalità di query in parallelo potrebbe essere applicata all'analisi delle tabelle per generare la struttura di dati del filtro Bloom.

> **Note**
>
> La query in parallelo viene in genere utilizzata per quei tipi di query che richiedono un uso intensivo di risorse che traggono vantaggio dall'ottimizzazione dell'hash join. Il metodo per attivare l'ottimizzazione dell'hash join dipende dalla versione di Aurora MySQL. Per informazioni dettagliate su ogni versione, consultare Abilitazione dell'hash join per cluster di query parallele. Per informazioni su come utilizzare i join hash in modo efficace, vedere Ottimizzazione di grandi query di join Aurora MySQL con hash join.

```
mysql> explain select count(*) from orders join customer where o_custkey = c_custkey;
+----+...+----------+-------+--------------+-------------+...+-----------
+-------------------------------------------------------------------------------------------
+
| id |...| table    | type  | possible_keys | key         |...| rows      | Extra

              |
+----+...+----------+-------+--------------+-------------+...+-----------
+-------------------------------------------------------------------------------------------
+
|  1 |...| customer | index | PRIMARY       | c_nationkey |...|  15051972 | Using index

              |
|  1 |...| orders   | ALL   | o_custkey     | NULL        |...| 154545408 | Using join
 buffer (Hash Join Outer table orders); Using parallel query (1 columns, 0 filters, 1
 exprs; 0 extra) |
+----+...+----------+-------+--------------+-------------+...+-----------
+-------------------------------------------------------------------------------------------
+
```

Per una query join che utilizza il meccanismo di ciclo nidificato, il blocco di ciclo nidificato più esterno può utilizzare la funzionalità di query in parallelo. L'utilizzo delle query in parallelo dipende dai fattori usuali, come la presenza di condizioni di filtro supplementari nella clausola `WHERE`.

```
mysql> -- Nested loop join with extra filter conditions can use parallel query.
mysql> explain select count(*) from part, partsupp where p_partkey != ps_partkey and
 p_name is not null and ps_availqty > 0;
+----+-------------+----------+...+----------
+-------------------------------------------------------------------------+
| id | select_type | table    |...| rows     | Extra
                                  |
```

```
+----+-------------+----------+...+----------
+-------------------------------------------------------------------------------+
|  1 | SIMPLE      | part     |...| 20427936 | Using where; Using parallel query (2
 columns, 1 filters, 0 exprs; 0 extra) |
|  1 | SIMPLE      | partsupp |...| 78164450 | Using where; Using join buffer (Block
 Nested Loop)                          |
+----+-------------+----------+...+----------
+-------------------------------------------------------------------------------+
```

Sottoquery

Il blocco di query esterno e il blocco di sottoquery interno possono utilizzare ciascuna query parallela o meno. Se lo fanno si basa sulle solite caratteristiche della tabella, clausola `WHERE` e così via, per ogni blocco. Ad esempio, la query seguente utilizza tale funzionalità per il blocco di sottoquery ma non per il blocco esterno.

```
mysql> explain select count(*) from part where
   --> p_partkey < (select max(p_partkey) from part where p_name like '%vanilla%');
+----+-------------+...+----------
+-------------------------------------------------------------------------------+
| id | select_type |...| rows     | Extra
                        |
+----+-------------+...+----------
+-------------------------------------------------------------------------------+
|  1 | PRIMARY     |...|     NULL | Impossible WHERE noticed after reading const tables
                       |
|  2 | SUBQUERY    |...| 20427936 | Using where; Using parallel query (2 columns, 0
 filters, 1 exprs; 0 extra) |
+----+-------------+...+----------
+-------------------------------------------------------------------------------+
```

Attualmente, le sottoquery correlate non possono utilizzare l'ottimizzazione mediante query in parallelo.

UNION

Ogni blocco di query in una query `UNION` può utilizzare o meno la funzionalità di query in parallelo in base alle caratteristiche abituali della tabella, della clausola `WHERE`, ecc., per ogni parte della query `UNION`.

```
mysql> explain select p_partkey from part where p_name like '%choco_ate%'
    -> union select p_partkey from part where p_name like '%vanil_a%';
```

```
+----+---------------+...+----------
+-------------------------------------------------------------------------------+
| id | select_type    |...| rows     | Extra
                           |
+----+---------------+...+----------
+-------------------------------------------------------------------------------+
|  1 | PRIMARY        |...| 20427936 | Using where; Using parallel query (2 columns, 0
 filters, 1 exprs; 0 extra) |
|  2 | UNION          |...| 20427936 | Using where; Using parallel query (2 columns, 0
 filters, 1 exprs; 0 extra) |
| NULL | UNION RESULT | <union1,2> |...|     NULL | Using temporary
                        |
+----+-------------+...+----------
+-------------------------------------------------------------------------------+
```

> **Note**
>
> Ogni clausola `UNION` nella query viene eseguita in modo sequenziale. Anche se la query include molteplici fasi che utilizzano la funzionalità di query in parallelo, esegue sempre un'unica query in parallelo. Di conseguenza, anche una query complessa con più fasi viene contabilizzata come una sola query per il limite di query in parallelo simultanee.

Visualizzazioni

L'ottimizzatore riscrive le query con una visualizzazione come query più lunga che utilizza le tabelle sottostanti. La query in parallelo funziona quindi nello stesso modo indipendentemente dal fatto che i riferimenti delle tabelle siano visualizzazioni o tabelle reali. Tutte le considerazioni sull'utilizzo o meno della funzionalità di query in parallelo per una query nonché su quali parti vengono trasferite, si applicano alla query riscritta finale.

Ad esempio, il piano query seguente mostra la definizione di una visualizzazione che in genere non utilizza le query in parallelo. Quando viene eseguita la query della visualizzazione con ulteriori clausole `WHERE`, Aurora MySQL utilizza una query in parallelo.

```
mysql> create view part_view as select * from part;
mysql> explain select count(*) from part_view where p_partkey is not null;
+----+...+----------
+-------------------------------------------------------------------------------+
| id |...| rows     | Extra
          |
```

```
+----+...+----------
+-------------------------------------------------------------------------+
|  1 |...| 20427936 | Using where; Using parallel query (1 columns, 0 filters, 0 exprs;
 1 extra) |
+----+...+----------
+-------------------------------------------------------------------------+
```

Istruzioni DML

L'istruzione `INSERT` può utilizzare la funzionalità di query in parallelo per la fase `SELECT` dell'elaborazione, se la parte `SELECT` soddisfa le altre condizioni relative a tale funzionalità.

```
mysql> create table part_subset like part;
mysql> explain insert into part_subset select * from part where p_mfgr =
 'Manufacturer#1';
+----+...+----------
+-------------------------------------------------------------------------+
| id |...| rows     | Extra
          |
+----+...+----------
+-------------------------------------------------------------------------+
|  1 |...| 20427936 | Using where; Using parallel query (9 columns, 1 filters, 0 exprs;
 0 extra) |
+----+...+----------
+-------------------------------------------------------------------------+
```

> **Note**
>
> In genere, dopo un'istruzione `INSERT`, i dati delle righe appena inserite si trovano nel pool di buffer. Di conseguenza, una tabella può non essere idonea per la funzionalità di query in parallelo subito dopo l'inserimento di un gran numero di righe. Successivamente, dopo la rimozione dei dati dal pool di buffer durante il funzionamento normale, le query sulla tabella possono iniziare a utilizzare di nuovo la funzionalità di query in parallelo.

L'istruzione `CREATE TABLE AS SELECT` non utilizza la funzionalità di query in parallelo anche se la porzione `SELECT` dell'istruzione fosse idonea a farlo. L'aspetto DDL di questa istruzione la rende incompatibile con l'elaborazione di query in parallelo. Nell'istruzione `INSERT ... SELECT`, la porzione `SELECT` può invece utilizzare la funzionalità di query in parallelo.

Le query in parallelo non sono mai utilizzate per le istruzioni `DELETE` o `UPDATE`, indipendentemente dalla dimensione della tabella e dei predicati nella clausola `WHERE`.

```
mysql> explain delete from part where p_name is not null;
+----+-------------+...+----------+-------------+
| id | select_type |...| rows     | Extra       |
+----+-------------+...+----------+-------------+
|  1 | SIMPLE      |...| 20427936 | Using where |
+----+-------------+...+----------+-------------+
```

Transazioni e blocco

Puoi utilizzare tutti i livelli di isolamento sull'istanza primaria Aurora.

Sulle istanze database di lettura Aurora, la query parallela si applica alle istruzioni eseguite nel livello di isolamento `REPEATABLE READ`. Aurora MySQL versione 2.09 o successiva può utilizzare anche il livello di isolamento `READ COMMITTED` sulle istanze database di lettura. `REPEATABLE READ` è il livello di isolamento predefinito per le istanze database di lettura Aurora. Per utilizzare il livello di isolamento `READ COMMITTED` sulle istanze DB del lettore è necessario impostare l'opzione di configurazione `aurora_read_replica_read_committed` a livello di sessione. Il livello di isolamento `READ COMMITTED` per le istanze di lettura è conforme al comportamento standard SQL. Tuttavia, l'isolamento è meno rigoroso per le istanze di lettura rispetto a quando le query utilizzano il livello di isolamento `READ COMMITTED` sull'istanza di scrittura.

Per ulteriori informazioni sui livelli di isolamento Aurora, in particolare sulle differenze su `READ COMMITTED` fra le istanze di scrittura e lettura, consultare Livelli di isolamento di Aurora MySQL.

Al termine di una transazione di grandi dimensioni, le statistiche della tabella possono non essere aggiornate. Queste statistiche possono necessitare di un'istruzione `ANALYZE TABLE` affinché Aurora sia in grado di stimare accuratamente il numero di righe. Un'istruzione DML su larga scala può anche comportare lo storage di una porzione significativa dei dati della tabella nel pool di buffer. La presenza di questi dati nel pool di buffer può implicare un utilizzo meno frequente della funzionalità di query in parallelo per quella tabella fino a che i dati non vengono rimossi dal pool.

Quando la tua sessione è in una transazione di lunga durata (per impostazione predefinita, 10 minuti), le altre query in quella sessione non utilizzano la funzionalità di query in parallelo. È anche possibile che si verifichi un timeout durante una singola query di lunga durata. Questo tipo di timeout può verificarsi se la durata della query è più lunga dell'intervallo massimo (attualmente 10 minuti) prima dell'inizio dell'elaborazione di query in parallelo.

Puoi limitare il rischio di un'esecuzione accidentale delle transazioni di lunga durata impostando `autocommit=1` nelle sessioni `mysql` in cui esegui query ad hoc (uniche). Anche un'istruzione `SELECT` relativa a una tabella inizia una transazione creando una visualizzazione di lettura. Una visualizzazione di lettura è un set di dati coerente per le query successive che viene conservato fino al commit della transazione. Tieni presente questa restrizione anche quando utilizzi applicazioni JDBC o ODBC con Aurora, in quanto possono essere eseguite con l'impostazione `autocommit` disattivata.

L'esempio seguente mostra come, con l'impostazione `autocommit` disattivata, l'esecuzione di una query su una tabella crea una visualizzazione di lettura che implicitamente avvia una transazione. Le query eseguite subito dopo possono ancora utilizzare la funzionalità di query in parallelo. Tuttavia, dopo una pausa di alcuni minuti, le query non sono più idonee per tale funzionalità. Termina la transazione con `COMMIT` o `ROLLBACK` per ripristinare l'idoneità.

```
mysql> set autocommit=0;

mysql> explain select sql_no_cache count(*) from part where p_retailprice > 10.0;
+----+...+---------
+-------------------------------------------------------------------------------+
| id |...| rows    | Extra
         |
+----+...+---------
+-------------------------------------------------------------------------------+
|  1 |...| 2976129 | Using where; Using parallel query (1 columns, 1 filters, 0 exprs;
 0 extra) |
+----+...+---------
+-------------------------------------------------------------------------------+

mysql> select sleep(720); explain select sql_no_cache count(*) from part where
 p_retailprice > 10.0;
+------------+
| sleep(720) |
+------------+
|          0 |
+------------+
1 row in set (12 min 0.00 sec)

+----+...+---------+-------------+
| id |...| rows    | Extra       |
+----+...+---------+-------------+
|  1 |...| 2976129 | Using where |
+----+...+---------+-------------+
```

```
mysql> commit;

mysql> explain select sql_no_cache count(*) from part where p_retailprice > 10.0;
+----+...+---------
+-------------------------------------------------------------------------------+
| id |...| rows    | Extra
         |
+----+...+---------
+-------------------------------------------------------------------------------+
|  1 |...| 2976129 | Using where; Using parallel query (1 columns, 1 filters, 0 exprs;
 0 extra) |
+----+...+---------
+-------------------------------------------------------------------------------+
```

Per determinare quante volte le query non erano idonee per le query in parallelo perché parte di transazioni di lunga durata, verifica la variabile di stato `Aurora_pq_request_not_chosen_long_trx`.

```
mysql> show global status like '%pq%trx%';
+--------------------------------------+-------+
| Variable_name                        | Value |
+--------------------------------------+-------+
| Aurora_pq_request_not_chosen_long_trx | 4     |
+---------------------------------+-------+
```

Qualsiasi istruzione `SELECT` che acquisisce blocchi, come la sintassi `SELECT FOR UPDATE` o `SELECT LOCK IN SHARE MODE`, non può utilizzare la funzionalità di query in parallelo.

Le query in parallelo possono essere utilizzate con una tabella bloccata da un'istruzione `LOCK TABLES`.

```
mysql> explain select o_orderpriority, o_shippriority from orders where o_clerk =
 'Clerk#000095055';
+----+...+-----------
+-------------------------------------------------------------------------------+
| id |...| rows      | Extra
           |
+----+...+-----------
+-------------------------------------------------------------------------------+
|  1 |...| 154545408 | Using where; Using parallel query (3 columns, 1 filters, 0
 exprs; 0 extra) |
```

```
+----+...+-----------
+-------------------------------------------------------------------------------+

mysql> explain select o_orderpriority, o_shippriority from orders where o_clerk =
 'Clerk#000095055' for update;
+----+...+-----------+-------------+
| id |...| rows      | Extra       |
+----+...+-----------+-------------+
|  1 |...| 154545408 | Using where |
+----+...+-----------+-------------+
```

Indici B-Tree

Le statistiche raccolte da un'istruzione `ANALYZE TABLE` consentono all'ottimizzatore di stabilire quando utilizzare la funzionalità di query in parallelo o le ricerche di indice in funzione delle caratteristiche dei dati di ogni colonna. Mantieni aggiornate le statistiche eseguendo `ANALYZE TABLE` dopo le operazioni DML che apportano modifiche sostanziali ai dati in una tabella.

Se le ricerche di indice sono in grado di eseguire efficacemente una query senza l'analisi di un grande volume di dati, Aurora può utilizzare le ricerche di indice. In questo modo, si evita l'overhead dell'elaborazione di query in parallelo. Esistono inoltre limiti di simultaneità sul numero di query in parallelo che possono essere eseguite simultaneamente su qualsiasi cluster di database Aurora. Assicurati di seguire le best practice per l'indicizzazione delle tabelle, di modo che le query più frequenti e che utilizzano maggiormente la simultaneità ricorrano alle ricerche di indice.

Indici di ricerca full-text (FTS)

Al momento, le query in parallelo non sono utilizzate per le tabelle che contengono un indice di ricerca full-text, indipendentemente dal fatto che la query faccia riferimento a tali colonne indicizzate o che utilizzi l'operatore `MATCH`.

Colonne virtuali

Attualmente, la query parallela non viene utilizzata per le tabelle che contengono una colonna virtuale, indipendentemente dal fatto che la query faccia riferimento a colonne virtuali.

Meccanismi di caching integrati

Aurora include meccanismi di caching integrati, ovvero il pool di buffer e la cache di query L'ottimizzatore Aurora sceglie tra questi meccanismi di caching e la funzionalità di query in parallelo a seconda di quale è più efficace per una determinata query.

Quando una query in parallelo filtra le righe e trasforma ed estrae i valori di colonna, i dati sono ritrasmessi al nodo head come tuple anziché pagine di dati. Di conseguenza, l'esecuzione di una query in parallelo non aggiunge pagine al pool di buffer o rimuove pagine presenti nel pool di buffer.

Aurora verifica il numero di pagine di dati della tabella presenti nel pool di buffer e quale proporzione di quei dati quel numero rappresenta. Aurora si basa su tali informazioni per determinare se è più efficace utilizzare la funzionalità di query parallela (e ignorare i dati nel pool di buffer). In alternativa, Aurora può utilizzare il percorso di elaborazione di query non in parallelo, che si avvale dei dati nella cache del pool di buffer. Le pagine memorizzate nella cache e l'impatto delle query con una grande quantità di dati sulle operazioni di caching e rimozione dipendono dalle impostazioni di configurazione relative al pool di buffer. Può quindi rivelarsi difficile prevedere se una specifica query utilizza la funzionalità di query in parallelo, poiché la scelta dipende dai dati che si trovano nel pool di buffer e che cambiano costantemente.

Inoltre, Aurora impone limiti di simultaneità per le query in parallelo. Poiché non tutte le query utilizzano la funzionalità di query in parallelo, una parte importante dei dati delle tabelle a cui hanno accesso più query simultaneamente si trova nel pool di buffer. Di conseguenza, Aurora spesso non sceglie queste tabelle per le query in parallelo.

Quando esegui una sequenza di query non in parallelo sulla stessa tabella, la prima query può risultare lenta in quanto i dati non sono presenti nel pool di buffer. La seconda query e quelle successive saranno molto più rapide in quanto il pool di buffer conterrà già dei dati. Le query in parallelo in genere forniscono prestazioni costanti sin dalla prima query su una tabella. Quando esegui test di prestazioni, compara le query non in parallelo con un pool di buffer a caldo e uno a freddo. In alcuni casi, i risultati con un pool di buffer a caldo sono comparabili a quelli delle query in parallelo. In questi casi, prendere in considerazione fattori quali la frequenza delle query rispetto a tale tabella. Considerare anche se vale la pena mantenere i dati per quella tabella nel pool di buffer.

La cache di query evita di dover rieseguire una query quando si invia una query identica e i dati di tabella sottostante non vengono modificati. Le query ottimizzate mediante la funzionalità di query in parallelo possono essere memorizzate nella cache di query, in modo da generare risultati istantanei all'esecuzione successiva.

> **Note**
>
> Quando si comparano le prestazioni, la cache di query può generare valori di durata artificialmente bassi. Di conseguenza, nelle comparazioni puoi utilizzare l'hint `sql_no_cache`. Questo hint impedisce la generazione dei risultati a partire dalla cache di query, anche se la stessa query è stata eseguita in precedenza. L'hint è posizionato subito

dopo l'istruzione `SELECT` in una query. Molti esempi di query in parallelo in questo argomento includono questo suggerimento per consentire la comparazione, in termini di durata, delle versioni della stessa query con e senza la funzionalità di query in parallelo abilitata. Assicurati di rimuovere questo hint dal codice sorgente quando utilizzi la funzionalità di query in parallelo in un ambiente di produzione.

Suggerimenti per l'ottimizzazione

Un altro modo per controllare l'ottimizzazione consiste nell'utilizzare i suggerimenti che possono essere specificati in singole istruzioni. Ad esempio, è possibile attivare l'ottimizzazione per una tabella in un'istruzione e quindi disattivare l'ottimizzazione per un'altra tabella. Per ulteriori informazioni su questi suggerimenti, consulta Suggerimenti di ottimizzazione nel Manuale di riferimento MySQL.

È possibile utilizzare i suggerimenti SQL con le query Aurora MySQL per ottimizzare le prestazioni. È inoltre possibile utilizzare i suggerimenti per impedire che i piani di esecuzione per query importanti vengano modificati a causa di condizioni imprevedibili.

Abbiamo esteso la funzionalità dei suggerimenti SQL per controllare meglio le scelte di ottimizzazione per i tuoi piani di query. Questi suggerimenti si applicano alle query che utilizzano l'ottimizzazione delle query parallele. Per ulteriori informazioni, consulta Suggerimenti di Aurora MySQL.

Tabelle temporanee MyISAM

L'ottimizzazione mediante query in parallelo è applicabile solo alle tabelle InnoDB. Poiché Aurora MySQL utilizza MyISAM in background per le tabelle temporanee, le fasi di query interne che implicano tabelle temporanee non utilizzano mai la funzionalità di query in parallelo. Queste fasi di query sono indicate da `Using temporary` nell'output `EXPLAIN`.

Utilizzo dell'audit avanzato con un cluster di database Amazon Aurora MySQL

Puoi utilizzare la funzionalità di audit avanzato ad alte prestazioni in Amazon Aurora MySQL per controllare l'attività del database. A questo proposito, devi abilitare la raccolta dei registri di controllo impostando vari parametri del cluster di database. Quando l'audit avanzato è abilitato, puoi utilizzarlo per registrare qualsiasi combinazione di eventi supportati.

Puoi visualizzare o scaricare i registri di verifica per esaminare le informazioni di verifica per un'istanza database alla volta. Per farlo, puoi utilizzare le procedure in Monitoraggio dei file di log di Amazon Aurora.

> **Tip**
>
> Per un cluster Aurora DB contenente più istanze database, potrebbe essere più conveniente esaminare i registri di verifica per tutte le istanze del cluster. Per farlo, puoi utilizzare CloudWatch Logs. Puoi attivare un'impostazione a livello di cluster per pubblicare i dati del registro di verifica di Aurora MySQL in un gruppo di registri in CloudWatch. Quindi puoi visualizzare, filtrare e cercare i registri di verifica tramite l'interfaccia di CloudWatch. Per ulteriori informazioni, consulta Pubblicazione di log Amazon Aurora MySQL in Amazon CloudWatch Logs.

Abilitazione dell'audit avanzato

Utilizza i parametri descritti in questa sezione per abilitare e configurare l'audit avanzato per il cluster di database.

Utilizza il parametro `server_audit_logging` per abilitare o disabilitare Advanced Auditing.

Utilizza il parametro `server_audit_events` per specificare quali eventi registrare.

Utilizzare i parametri `server_audit_incl_users` e `server_audit_excl_users` per specificare chi deve essere controllato. Per impostazione predefinita, vengono verificati tutti gli utenti. Per informazioni dettagliate su come funzionano questi parametri quando uno o entrambi sono lasciati vuoti o gli stessi nomi utente sono specificati in entrambi, consulta server_audit_incl_users e server_audit_excl_users.

Configura l'audit avanzato impostando questi parametri nel gruppo di parametri utilizzato dal tuo cluster di database. È possibile utilizzare la procedura visualizzata in Modifica dei parametri in un gruppo di parametri database in Amazon Aurora per modificare i parametri del cluster di database con la Console di gestione AWS. Per modificare i parametri del cluster database a livello di programmazione, puoi utilizzare il comando della AWS CLI modify-db-cluster-parameter-group o l'API Amazon RDS ModifyDBClusterParameterGroup.

La modifica di questi parametri non richiede un riavvio del cluster database quando il gruppo di parametri è già associato al cluster. Quando associ il gruppo di parametri al cluster per la prima volta, devi riavviare il cluster.

Argomenti

- server_audit_logging
- server_audit_events
- server_audit_incl_users
- server_audit_excl_users

server_audit_logging

Abilita o disabilita l'audit avanzato. L'impostazione predefinita di questo parametro è OFF; impostarlo su ON per abilitare la verifica avanzata.

Nessun dato di verifica viene visualizzato nei registri a meno che non si definiscano anche uno o più tipi di eventi da sottoporre a verifica utilizzando il parametro `server_audit_events`.

Per confermare che i dati di audit siano registrati per un'istanza database, verifica che alcuni file di log per quell'istanza abbiano i nomi del modulo `audit/audit.log.`*`other_identifying_information`*. Per visualizzare i nomi dei file di log, segui la procedura descritta in Visualizzazione ed elenco dei file di log del database.

server_audit_events

Contiene un elenco di eventi delimitati da virgola da registrare. Gli eventi devono essere in maiuscolo e non devono esserci spazi bianchi tra gli elementi dell'elenco (ad esempio,: `CONNECT,QUERY_DDL`. L'impostazione predefinita di questo parametro è una stringa vuota.

Puoi registrare qualsiasi combinazione dei seguenti eventi:

- CONNECT – Registra le connessioni riuscite e quelle non riuscite nonché le disconnessioni. Questo evento include informazioni sull'utente.
- QUERY – Registra tutte le query in testo normale, incluse quelle non riuscite a causa di errori di sintassi o di autorizzazione.

> **Tip**
>
> Con questo tipo di evento attivato, i dati di verifica includono informazioni sul monitoraggio continuo e sul controllo dell'integrità eseguiti automaticamente da Aurora. Se sei interessato solo a particolari tipi di operazioni, puoi utilizzare i tipi di eventi più specifici. Puoi inoltre utilizzare l'interfaccia CloudWatch per cercare nei log eventi relativi a database, tabelle o utenti specifici.

- QUERY_DCL – Simile all'evento QUERY, ma restituisce solo le query DCL (Data Control Language), ad esempio GRANT, REVOKE e così via.
- QUERY_DDL –Simile all'evento QUERY, ma restituisce solo le query DDL (Data Definition Language), ad esempio CREATE, ALTER e così via.
- QUERY_DML – Simile all'evento QUERY, ma restituisce solo le query DML (Data Manipulation Language), ad esempio INSERT, UPDATE e così via.
- TABLE –Registra le tabelle interessate dall'esecuzione della query.

> **Note**
>
> In Aurora non esiste alcun filtro che escluda determinate query dai log di audit. Per escludere le query `SELECT`, è necessario escludere tutte le istruzioni DML.
> Se un determinato utente riporta queste query `SELECT` interne nei log di audit, puoi escludere tale utente impostando il parametro del cluster di database server_audit_excl_users. Tuttavia, se quell'utente viene utilizzato anche in altre attività e non può essere escluso, non esiste altra opzione per escludere le query `SELECT`.

server_audit_incl_users

Contiene l'elenco di nomi utente delimitati da virgole per gli utenti la cui attività è registrata. Non ci devono essere spazi bianchi tra gli elementi dell'elenco. Ad esempio: `user_3,user_4`. L'impostazione predefinita di questo parametro è una stringa vuota. La lunghezza massima è 1024

caratteri. I nomi utente specificati devono corrispondere ai valori nella colonna `User` della tabella `mysql.user`. Per ulteriori informazioni sui nomi utente, consulta Nomi e password dell'account utente nella documentazione di MySQL.

Se `server_audit_incl_users` e `server_audit_excl_users` sono vuoti (impostazione predefinita), l'audit riguarderà tutti gli utenti.

Se aggiungi utenti a `server_audit_incl_users` e lasci vuoto `server_audit_excl_users`, saranno controllati solo quegli utenti.

Se aggiungi utenti a `server_audit_excl_users` e lasci vuoto `server_audit_incl_users`, saranno controllati tutti gli utenti, eccetto quelli elencati in `server_audit_excl_users`.

Se aggiungi gli stessi utenti a `server_audit_excl_users` e `server_audit_incl_users`, quegli utenti saranno controllati. Quando lo stesso utente è elencato in entrambe le impostazioni, a `server_audit_incl_users`viene data una priorità maggiore.

Gli eventi di connessione e disconnessione non sono interessati da questa variabile; sono sempre registrati se specificato. Un utente è registrato anche se è specificato nel parametro `server_audit_excl_users`, perché `server_audit_incl_users` ha una priorità maggiore.

server_audit_excl_users

Contiene l'elenco di nomi utente delimitati da virgole per gli utenti la cui attività non è registrata. Non ci devono essere spazi bianchi tra gli elementi dell'elenco. Ad esempio: `rdsadmin,user_1,user_2`. L'impostazione predefinita di questo parametro è una stringa vuota. La lunghezza massima è 1024 caratteri. I nomi utente specificati devono corrispondere ai valori nella colonna `User` della tabella `mysql.user`. Per ulteriori informazioni sui nomi utente, consulta Nomi e password dell'account utente nella documentazione di MySQL.

Se `server_audit_incl_users` e `server_audit_excl_users` sono vuoti (impostazione predefinita), l'audit riguarderà tutti gli utenti.

Se aggiungi utenti a `server_audit_excl_users` e lasci vuoto `server_audit_incl_users`, solo quegli utenti elencati in `server_audit_excl_users` non saranno controllati, mentre tutti gli altri lo saranno.

Se aggiungi gli stessi utenti a `server_audit_excl_users` e `server_audit_incl_users`, quegli utenti saranno controllati. Quando lo stesso utente è elencato in entrambe le impostazioni, a `server_audit_incl_users`viene data una priorità maggiore.

Gli eventi di connessione e disconnessione non sono interessati da questa variabile; sono sempre registrati se specificato. Un utente è registrato se è anche specificato nel parametro `server_audit_incl_users`, in quanto tale impostazione è prioritaria rispetto a `server_audit_excl_users`.

Visualizzazione dei log di audit

Puoi visualizzare e scaricare i log di audit utilizzando la console. Nella pagina Databases (Database), scegliere l'istanza database per visualizzarne i dettagli, quindi scorrere verso la sezione Logs (Log). I registri di verifica prodotti dalla caratteristica Advanced Auditing hanno i nomi del modulo `audit/audit.log.`*`other_identifying_information`*.

Per scaricare un file di log, selezionare il file nella sezione Logs (Log), quindi scegliere Download (Scarica).

Puoi anche ottenere un elenco di file di log utilizzando il comando describe-db-log-files dell'AWS CLI. Puoi scaricare il contenuto di un file di log utilizzando il comando download-db-log-file-portion dell'AWS CLI. Per ulteriori informazioni, consulta Visualizzazione ed elenco dei file di log del database e Download di un file di log di database.

Dettagli dei log di audit

I file di log sono rappresentati come file con valori delimitati da virgole (CSV) in formato UTF-8. Le query sono inoltre racchiuse tra virgolette singole (').

Il log di audit viene memorizzato separatamente nell'archiviazione locale di ciascuna istanza database Aurora MySQL. Ogni istanza distribuisce le operazioni di scrittura su quattro file di log alla volta. Il file di log può avere dimensioni massime di 100 MB. Quando viene raggiunto questo limite non configurabile, Aurora effettua la rotazione dei file e genera un nuovo file.

> **Tip**
>
> Le voci dei log non sono in ordine sequenziale. Per ordinare le voci, utilizza il valore del timestamp. Per visualizzare gli ultimi eventi, potrebbe essere necessario esaminare tutti i file di log. Per una maggiore flessibilità nell'ordinamento e nella ricerca dei dati dei registri, attiva l'impostazione per caricare i registri di verifica su CloudWatch e visualizzarli utilizzando l'interfaccia CloudWatch.
> Per visualizzare i dati di verifica con più tipi di campi e con output in formato JSON, puoi inoltre utilizzare la caratteristica Database Activity Streams (Flussi di attività di database). Per

ulteriori informazioni, consulta Monitoraggio di Amazon Aurora tramite i flussi di attività del database.

I file dei log di audit includono le seguenti informazioni delimitate da virgola, in righe, nell'ordine specificato:

Campo	Descrizione
timestamp	Il timestamp Unix per l'evento registrato con una precisione al microsecondo.
serverhost	Il nome dell'istanza per cui l'evento viene registrato.
username	Il nome utente connesso dell'utente.
host	L'host da cui l'utente ha effettuato la connessione.
connectionid	Il numero di ID di connessione per l'operazione registrata.
queryid	Il numero di ID di query che può essere utilizzato per trovare gli eventi di tabella relazionale e le query correlate. Per gli eventi `TABLE`, vengono aggiunte più righe.
operation	Il tipo di operazione registrata. I valori possibili sono `CONNECT`, `QUERY`, `READ`, `WRITE`, `CREATE`, `ALTER`, `RENAME` e `DROP`.
database	Il database attivo, come impostato dal comando `USE`.
oggetto	Per gli eventi `QUERY`, questo valore indica la query eseguita dal database. Per gli eventi `TABLE`, indica il nome di tabella.
retcode	Il codice restituito dell'operazione di registrazione.

Replica con Amazon Aurora MySQL

Le funzionalità di replica Aurora MySQL sono essenziali per garantire l'elevata disponibilità e le prestazioni avanzate del cluster. Aurora facilita la creazione o il ridimensionamento dei cluster con un massimo di 15 repliche Aurora.

Tutte le repliche utilizzano gli stessi dati sottostanti. Se alcune istanze database passano alla modalità offline, altre rimangono disponibili per continuare a elaborare le query o per essere utilizzate come scrittore, se necessario. Aurora ripartisce automaticamente le connessioni di sola lettura in più istanze database, consentendo a un cluster Aurora di supportare carichi di lavoro che comportano un grande numero di query.

Nei seguenti argomenti vengono fornite informazioni sul funzionamento della replica Aurora MySQL nonché sulla regolazione delle impostazioni di replica per garantire una disponibilità e prestazioni ottimali.

Argomenti

- Utilizzo delle repliche di Aurora
- Opzioni di replica per Amazon Aurora MySQL
- Considerazioni sulle prestazioni per la replica Amazon Aurora MySQL
- Configurazione dei filtri di replica con Aurora MySQL
- Monitoraggio della replica Amazon Aurora MySQL.
- Repliche di cluster di database Amazon Aurora MySQL tra Regioni AWS
- Replica tra Aurora e MySQL o tra Aurora e un altro cluster di database Aurora (replica dei log binari)
- Utilizzo della replica basata su GTID

Utilizzo delle repliche di Aurora

Le repliche Aurora sono endpoint indipendenti in un cluster DB Aurora, utilizzati al meglio per operazioni di dimensionamento della lettura e maggiore disponibilità. È possibile distribuire fino a 15 repliche di Aurora nelle zone di disponibilità sulle quali si estende un cluster di database in una Regione AWS. Sebbene il volume cluster database sia fatto di copie multiple di dati per il cluster database, i dati nel volume cluster sono rappresentati come un volume singolo e logico nell'istanza primaria e nelle repliche Aurora nel cluster database. Per ulteriori informazioni sulle repliche di Aurora, consulta Repliche di Aurora.

Le repliche Aurora funzionano bene per il dimensionamento della lettura perché sono dedicate completamente a operazioni di lettura nel volume cluster. Le operazioni di lettura sono gestite dall'istanza primaria. Poiché il volume del cluster viene condiviso tra tutte le istanze database nel tuo cluster database Aurora MySQL, non è necessaria alcuna ulteriore azione per replicare una copia dei dati per ciascuna replica Aurora. Al contrario, le repliche di lettura MySQL devono riprodurre, su un singolo thread, tutte le operazioni di scrittura dall'istanza database master al datastore locale. Questa limitazione può influire sulla capacita delle repliche di lettura MySQL di supportare grandi volumi di traffico in lettura.

Con Aurora MySQL, quando una replica Aurora viene eliminata, l'endpoint dell'istanza viene rimosso immediatamente e la replica Aurora viene rimossa dall'endpoint di lettura. Se vi sono istruzioni in esecuzione nella replica Aurora da eliminare, si ha un periodo di tolleranza di tre minuti. Le istruzioni esistenti possono finire correttamente durante un periodo di tolleranza. Quando finisce il periodo di tolleranza, la replica Aurora viene chiusa ed eliminata.

Important

Le repliche Aurora per Aurora MySQL utilizzano sempre il livello di isolamento della transazione predefinita `REPEATABLE READ` per operazioni nelle tabelle InnoDB. Puoi utilizzare il comando `SET TRANSACTION ISOLATION LEVEL` per modificare il livello di transazione solo per l'istanza primaria di un cluster database Aurora MySQL. Questa restrizione evita blocchi a livello di utente nelle repliche Aurora e permette alle repliche Aurora di dimensionarsi per supportare migliaia di connessioni utente attive mantenendo il ritardo di replica al minimo.

Note

Le istruzioni DDL che vengono eseguite sull'istanza primaria possono interrompere le connessioni di database sulle repliche Aurora associate. Se una connessione di replica Aurora sta utilizzando attivamente un oggetto di database, come una tabella e quell'oggetto è modificato nell'istanza primaria utilizzando un'istruzione DDL, la connessione di replica Aurora viene interrotta.

> **Note**
>
> La regione Cina (Ningxia) non supporta le repliche di lettura fra regioni.

Opzioni di replica per Amazon Aurora MySQL

Puoi impostare repliche tra qualsiasi delle seguenti opzioni:

- Due cluster di database Aurora MySQL in Regioni AWS diverse mediante la creazione di una replica di lettura di un cluster di database Aurora MySQL tra regioni.

  Per ulteriori informazioni, consulta Repliche di cluster di database Amazon Aurora MySQL tra Regioni AWS.
- Due cluster di database Aurora MySQL nella stessa Regione AWS mediante l'utilizzo di una replica del log binario (binlog) MySQL.

  Per ulteriori informazioni, consulta Replica tra Aurora e MySQL o tra Aurora e un altro cluster di database Aurora (replica dei log binari).
- Una istanza database RDS for MySQL come origine e un cluster database Aurora MySQL, creando una replica di lettura Aurora di una istanza database RDS for MySQL.

  È possibile utilizzare questo approccio per apportare modifiche ai dati esistenti e in corso in Aurora MySQL durante la migrazione ad Aurora. Per ulteriori informazioni, consulta Migrazione di dati da un'istanza database RDS per MySQL a un cluster database Amazon Aurora MySQL utilizzando una replica di lettura Aurora.

  È inoltre possibile utilizzare questo approccio per aumentare la scalabilità delle query di lettura per i dati. A tale scopo, eseguire query sui dati utilizzando una o più istanze database all'interno di un cluster Aurora MySQL di sola lettura. Per ulteriori informazioni, consulta Dimensionamento delle letture per il database MySQL con Amazon Aurora.
- Un cluster di database Aurora MySQL in una Regione AWS e fino a cinque cluster di database Aurora MySQL di sola lettura in regioni diverse mediante la creazione di un database globale Aurora.

  È possibile utilizzare un database globale Aurora per supportare applicazioni con un footprint globale. Il cluster database Aurora MySQL primario dispone di un'istanza di scrittura e fino a 15 repliche Aurora. I cluster database Aurora MySQL secondari di sola lettura possono essere

composti di un massimo di 16 repliche Aurora. Per ulteriori informazioni, consulta Utilizzo di Database globale Amazon Aurora.

> **Note**
>
> Il riavvio dell'istanza primaria di un cluster database Amazon Aurora riavvia automaticamente anche le repliche Aurora per il cluster database, per ristabilire un punto di ingresso che garantisce coerenza di lettura/scrittura nel cluster database.

Considerazioni sulle prestazioni per la replica Amazon Aurora MySQL

Le funzionalità seguenti consentono di ottimizzare le prestazioni della replica Aurora MySQL.

La funzionalità di compressione dei log di replica riduce automaticamente la larghezza di banda di rete per i messaggi di replica. Poiché ogni messaggio è trasmesso a tutte le repliche Aurora, i vantaggi sono maggiori per i cluster più voluminosi. Questa funzionalità implica un certo overhead della CPU sul nodo scrittore per eseguire la compressione. È sempre abilitato in Aurora MySQL versione 2 e versione 3.

La funzionalità di filtraggio binlog riduce automaticamente la larghezza di banda di rete per i messaggi di replica. Poiché le repliche Aurora non utilizzano informazioni di binlog incluse nei messaggi di replica, quei dati sono omessi dai messaggi inviati a tali nodi.

In Aurora MySQL versione 2, puoi controllare questa funzionalità modificando il parametro `aurora_enable_repl_bin_log_filtering`. Per impostazione predefinita, questo parametro è abilitato. Poiché questa ottimizzazione è concepita per essere trasparente, puoi disattivare questa impostazione solo durante la diagnosi o la risoluzione dei problemi relativi alla replica. Ad esempio per corrispondere al comportamento di un cluster Aurora MySQL meno recente dove questa funzionalità non era disponibile.

In Aurora MySQL versione 3, il filtraggio dei file binlog è sempre abilitato.

Configurazione dei filtri di replica con Aurora MySQL

Puoi utilizzare i filtri di replica per specificare quali database e tabelle vengono replicati con una replica di lettura. I filtri di replica possono includere database e tabelle nella replica o escluderli dalla replica.

Di seguito sono riportati alcuni casi d'uso per i filtri di replica:

- Per ridurre le dimensioni di una replica di lettura. Con il filtro di replica è possibile escludere i database e le tabelle che non sono necessari nella replica di lettura.
- Per escludere database e tabelle dalle repliche di lettura per motivi di sicurezza.
- Per replicare database e tabelle diversi per casi d'uso specifici in repliche di lettura diverse. Ad esempio, è possibile utilizzare repliche di lettura specifiche per l'analisi o la condivisione.
- Per un cluster di database che dispone di repliche di lettura in più Regioni AWS, per replicare database o tabelle diversi in differenti Regioni AWS.
- Per specificare quali database e tabelle vengono replicati con un cluster di database Aurora MySQL configurato come replica in una topologia di replica in entrata. Per ulteriori informazioni su questa configurazione, consulta Replica tra Aurora e MySQL o tra Aurora e un altro cluster di database Aurora (replica dei log binari).

Argomenti

- Impostazione dei parametri dei filtri di replica per Aurora MySQL
- Limitazioni dei filtri di replica per Aurora MySQL
- Esempi di filtri di replica per Aurora MySQL
- Visualizzazione dei filtri di replica per una replica di lettura

Impostazione dei parametri dei filtri di replica per Aurora MySQL

Per configurare i filtri di replica, imposta i seguenti parametri:

- `binlog-do-db` – Replica le modifiche ai log binari specificati. Quando si imposta questo parametro per un cluster di origine dei log binari, vengono replicati solo i log binari specificati nel parametro.
- `binlog-ignore-db` – Non replicare le modifiche ai log binari specificati. Quando il parametro `binlog-do-db` è impostato per un cluster di origine dei log binari, questo parametro non viene valutato.
- `replicate-do-db` – Replicare le modifiche ai database specificati. Quando si imposta questo parametro per un cluster di replica dei log binari, vengono replicati solo i database specificati nel parametro.

- `replicate-ignore-db` – Non replicare le modifiche ai database specificati. Quando il parametro `replicate-do-db` è impostato per un cluster di replica dei log binari, questo parametro non viene valutato.
- `replicate-do-table` – Replicare le modifiche alle tabelle specificate. Quando si imposta questo parametro per una replica di lettura, vengono replicate solo le tabelle specificate nel parametro. Inoltre, quando viene impostato il parametro `replicate-do-db` o `replicate-ignore-db`, assicurati di includere il database che contiene le tabelle specificate nella replica con il cluster di replica dei log binari.
- `replicate-ignore-table` – Non replicare le modifiche alle tabelle specificate. Quando il parametro `replicate-do-table` è impostato per un cluster di replica dei log binari, questo parametro non viene valutato.
- `replicate-wild-do-table` – Replicare le tabelle in base ai modelli di nome del database e della tabella specificati. I caratteri jolly `%` e `_` sono supportati. Quando viene impostato il parametro `replicate-do-db` o `replicate-ignore-db`, assicurati di includere il database che contiene le tabelle specificate nella replica con il cluster di replica dei log binari.
- `replicate-wild-ignore-table` – Non replicare le tabelle in base ai modelli di nomi di database e tabella specificati. I caratteri jolly `%` e `_` sono supportati. Quando il parametro `replicate-do-table` o `replicate-wild-do-table` è impostato per un cluster di replica dei log binari, questo parametro non viene valutato.

I parametri vengono valutati nell'ordine in cui sono elencati. Per ulteriori informazioni sul funzionamento di questi parametri, consulta la documentazione di MySQL:

- Per informazioni generali, consulta Opzioni e variabili del server di replica.
- Per informazioni sulla modalità di valutazione dei parametri di filtro della replica del database, consulta Valutazione delle opzioni di replica a livello di database e registrazione binaria.
- Per informazioni sulla modalità di valutazione dei parametri di filtro replica delle tabelle, consulta Valutazione delle opzioni di replica a livello di tabella.

Per impostazione predefinita, ognuno di questi parametri ha un valore vuoto. In ogni cluster di log binari, è possibile utilizzare questi parametri per impostare, modificare ed eliminare i filtri di replica. Quando viene impostato uno di questi parametri, è necessario separare ogni filtro dagli altri con una virgola.

È possibile utilizzare i caratteri jolly `%` e `_` nei parametri `replicate-wild-do-table` e `replicate-wild-ignore-table`. Il carattere jolly `%` corrisponde a un numero qualsiasi di caratteri e il carattere jolly `_` corrisponde a un solo carattere.

Il formato di registrazione binaria dell'istanza database di origine è importante per la replica perché determina il record delle modifiche ai dati. L'impostazione del parametro `binlog_format` determina se la replica è basata su righe o basata su dichiarazione. Per ulteriori informazioni, consulta Configurazione della registrazione di log binari Aurora MySQL per database Single-AZ.

> **Note**
>
> Tutte le istruzioni DDL (Data Definition Language) vengono replicate come istruzioni, indipendentemente dall'impostazione `binlog_format` dell'istanza database di origine.

Limitazioni dei filtri di replica per Aurora MySQL

Le seguenti limitazioni si applicano ai filtri di replica per Aurora MySQL:

- I filtri di replica sono supportati solo per Aurora MySQL versione 3.
- Ogni parametro di filtro della replica ha un limite di 2.000 caratteri.
- Le virgole non sono supportate nei filtri di replica.
- Il filtro delle repliche non supporta le transazioni XA.

  Per ulteriori informazioni, consulta Restrizioni sulle transazioni XA nella documentazione di MySQL.

Esempi di filtri di replica per Aurora MySQL

Per configurare il filtro per una replica di lettura, modifica i parametri di filtro replica nel gruppo di parametri del cluster di database associato alla replica di lettura.

> **Note**
>
> Non è consentito modificare un gruppo di parametri del cluster di database predefinito. Se la replica di lettura usa un gruppo di parametri predefinito, creare un nuovo gruppo di parametri e associarlo alla replica di lettura. Per ulteriori informazioni sui gruppi di parametri del cluster di database, consulta Gruppi di parametri per Amazon Aurora.

È possibile impostare parametri in un gruppo di parametri del cluster di database utilizzando la Console di gestione AWS, la AWS CLI o l'API RDS. Per informazioni sull'estensione dei parametri consulta Modifica dei parametri in un gruppo di parametri database in Amazon Aurora. Quando si impostano parametri in un gruppo di parametri del cluster di database, tutti i cluster di database associati al gruppo di parametri utilizzano le impostazioni dei parametri. Se imposti i parametri di filtro replica in un gruppo di parametri del cluster di database, assicurati che il gruppo di parametri sia associato solo alle repliche di lettura. Lasciare vuoti i parametri di filtro di replica per le istanze database di origine.

Negli esempi seguenti vengono impostati i parametri utilizzando AWS CLI. In questi esempi si imposta `ApplyMethod` su `immediate` in modo che le modifiche ai parametri avvengano immediatamente dopo il completamento del comando della CLI. Se si desidera applicare una modifica in sospeso dopo il riavvio della replica di lettura, impostare `ApplyMethod` su `pending-reboot`.

Gli esempi seguenti impostano i filtri di replica:

- Including databases in replication
- Including tables in replication
- Including tables in replication with wildcard characters
- Excluding databases from replication
- Excluding tables from replication
- Excluding tables from replication using wildcard characters

Example Inclusione dei database nella replica

Nell'esempio seguente sono inclusi i database `mydb1` e `mydb2` nella replica.

Per Linux, macOS o Unix:

```
aws rds modify-db-cluster-parameter-group \
  --db-cluster-parameter-group-name myparametergroup \
  --parameters "ParameterName=replicate-do-
db,ParameterValue='mydb1,mydb2',ApplyMethod=immediate"
```

Per Windows:

```
aws rds modify-db-cluster-parameter-group ^
```

```
  --db-cluster-parameter-group-name myparametergroup ^
  --parameters "ParameterName=replicate-do-
db,ParameterValue='mydb1,mydb2',ApplyMethod=immediate"
```

Example Inclusione delle tabelle nella replica

Nell'esempio seguente sono incluse le tabelle `table1` e `table2` nel database `mydb1` nella replica.

Per Linux, macOS o Unix:

```
aws rds modify-db-cluster-parameter-group \
  --db-cluster-parameter-group-name myparametergroup \
  --parameters "ParameterName=replicate-do-
table,ParameterValue='mydb1.table1,mydb1.table2',ApplyMethod=immediate"
```

Per Windows:

```
aws rds modify-db-cluster-parameter-group ^
  --db-cluster-parameter-group-name myparametergroup ^
  --parameters "ParameterName=replicate-do-
table,ParameterValue='mydb1.table1,mydb1.table2',ApplyMethod=immediate"
```

Example Inclusione di tabelle nella replica utilizzando caratteri jolly

Nell'esempio seguente sono incluse tabelle con nomi che iniziano con `order` e `return` nel database `mydb` nella replica.

Per Linux, macOS o Unix:

```
aws rds modify-db-cluster-parameter-group \
  --db-cluster-parameter-group-name myparametergroup \
  --parameters "ParameterName=replicate-wild-do-table,ParameterValue='mydb.order
%,mydb.return%',ApplyMethod=immediate"
```

Per Windows:

```
aws rds modify-db-cluster-parameter-group ^
  --db-cluster-parameter-group-name myparametergroup ^
```

```
  --parameters "ParameterName=replicate-wild-do-table,ParameterValue='mydb.order
%,mydb.return%',ApplyMethod=immediate"
```

Example Esclusione di database dalla replica

Nell'esempio seguente vengono esclusi i database `mydb5` e `mydb6` dalla replica.

Per Linux, macOS o Unix:

```
aws rds modify-db-cluster-parameter-group \
  --db-cluster-parameter-group-name myparametergroup \
  --parameters "ParameterName=replicate-ignore-
db,ParameterValue='mydb5,mydb6',ApplyMethod=immediate"
```

Per Windows:

```
aws rds modify-db-cluster-parameter-group ^
  --db-cluster-parameter-group-name myparametergroup ^
  --parameters "ParameterName=replicate-ignore-
db,ParameterValue='mydb5,mydb6,ApplyMethod=immediate"
```

Example Esclusione di tabelle dalla replica

Nell'esempio seguente vengono escluse dalla replica le tabelle `table1` nel database `mydb5` e `table2` nel database `mydb6`.

Per Linux, macOS o Unix:

```
aws rds modify-db-cluster-parameter-group \
  --db-cluster-parameter-group-name myparametergroup \
  --parameters "ParameterName=replicate-ignore-
table,ParameterValue='mydb5.table1,mydb6.table2',ApplyMethod=immediate"
```

Per Windows:

```
aws rds modify-db-cluster-parameter-group ^
  --db-cluster-parameter-group-name myparametergroup ^
  --parameters "ParameterName=replicate-ignore-
table,ParameterValue='mydb5.table1,mydb6.table2',ApplyMethod=immediate"
```

Example Esclusione di tabelle dalla replica utilizzando caratteri jolly

Nell'esempio seguente vengono escluse le tabelle con nomi che iniziano con `order` e `return` nel database `mydb7` dalla replica.

Per Linux, macOS o Unix:

```
aws rds modify-db-cluster-parameter-group \
  --db-cluster-parameter-group-name myparametergroup \
  --parameters "ParameterName=replicate-wild-ignore-table,ParameterValue='mydb7.order
%,mydb7.return%',ApplyMethod=immediate"
```

Per Windows:

```
aws rds modify-db-cluster-parameter-group ^
  --db-cluster-parameter-group-name myparametergroup ^
  --parameters "ParameterName=replicate-wild-ignore-table,ParameterValue='mydb7.order
%,mydb7.return%',ApplyMethod=immediate"
```

Visualizzazione dei filtri di replica per una replica di lettura

È possibile visualizzare i filtri di replica per una replica di lettura nei seguenti modi:

- Controllare le impostazioni dei parametri di filtro replica nel gruppo di parametri associato alla replica di lettura.

  Per istruzioni, consulta Visualizzazione dei valori dei parametri per un gruppo di parametri del database in Amazon Aurora.
- In un client MySQL, connettersi alla replica di lettura ed eseguire l'istruzione `SHOW REPLICA STATUS`.

  Nell'output, i campi seguenti mostrano i filtri di replica per la replica di lettura:
  - `Binlog_Do_DB`
  - `Binlog_Ignore_DB`
  - `Replicate_Do_DB`
  - `Replicate_Ignore_DB`
  - `Replicate_Do_Table`
  - `Replicate_Ignore_Table`

- `Replicate_Wild_Do_Table`
- `Replicate_Wild_Ignore_Table`

Per ulteriori informazioni su questi campi, consulta Verifica dello stato della replica nella documentazione di MariaDB.

Monitoraggio della replica Amazon Aurora MySQL.

Il dimensionamento di lettura e l'alta disponibilità dipendono da un periodo di ritardo minimo. Puoi monitorare il ritardo di una replica Aurora rispetto all'istanza principale del cluster database Aurora MySQL, monitorando il parametro `AuroraReplicaLag` di Amazon CloudWatch. La metrica `AuroraReplicaLag` viene registrata in ogni replica Aurora.

L'istanza DB primaria registra anche le metriche `AuroraReplicaLagMaximum` e `AuroraReplicaLagMinimum` Amazon CloudWatch. La metrica `AuroraReplicaLagMaximum` registra la quantità massima di ritardo tra l'istanza DB primaria e ogni replica Aurora nel cluster DB. La metrica `AuroraReplicaLagMinimum` registra la quantità minima di ritardo tra l'istanza DB primaria e ogni replica Aurora nel cluster DB.

Se è necessario il valore più attuale del ritardo replica di Aurora, puoi controllare la metrica `AuroraReplicaLag` in Amazon CloudWatch. Il ritardo replica di Aurora viene anche registrato su ogni replica Aurora del cluster database Aurora MySQL nella tabella `information_schema.replica_host_status`. Per ulteriori informazioni su questa tabella, consultare information_schema.replica_host_status.

Per ulteriori informazioni sul monitoraggio delle istanze RDS e dei parametri CloudWatch, consulta Monitoraggio dei parametri in un cluster di database Amazon Aurora.

Repliche di cluster di database Amazon Aurora MySQL tra Regioni AWS

Puoi creare un cluster di database Amazon Aurora MySQL come una replica di lettura in una Regione AWS diversa rispetto al cluster di database di origine. L'adozione di questo approccio può migliorare le capacità di ripristino di emergenza, permettere di dimensionare le operazioni di lettura in una Regione AWS più vicina agli utenti e semplificare la migrazione da una Regione AWS a un'altra.

Puoi creare repliche di lettura dei cluster di database crittografati e non crittografati. La replica di lettura deve essere crittografata se il cluster di database di origine è crittografato.

Per ogni cluster di database di origine, puoi avere fino a cinque cluster database tra regioni che sono repliche di lettura.

> **Note**
>
> In alternativa alle repliche di lettura tra regioni, è possibile dimensionare le operazioni di lettura con ritardi minimi utilizzando un database globale Aurora. Un Database globale Aurora ha un cluster di database Aurora primario in una Regione AWS e fino a dieci cluster di database di sola lettura secondari in Regioni differenti. Ogni cluster database secondario può includere fino a 16 repliche Aurora (anziché 15). La replica dal cluster database primario a tutti i secondari viene gestita dal livello di archiviazione di Aurora anziché dal motore del database, pertanto il ritardo per la replica delle modifiche è minimo, in genere inferiore a 1 secondo. Mantenere il motore di database fuori dal processo di replica significa che il motore di database è dedicato all'elaborazione dei carichi di lavoro. Significa inoltre che non è necessario configurare o gestire la replica binlog (registrazione dei log binari) di Aurora MySQL. Per ulteriori informazioni, consulta Utilizzo di Database globale Amazon Aurora.

Quando crei una replica di lettura del cluster di database Aurora MySQL in un'altra Regione AWS, devi tenere presente quanto segue:

- Il cluster di database di origine e il cluster di database di replica di lettura tra regione può avere fino a 15 repliche Aurora assieme all'istanza primaria per il cluster database. Con questa funzionalità, puoi dimensionare le operazioni di lettura per la Regione AWS di origine e la Regione AWS di destinazione della replica.
- In uno scenario con più regioni, si verifica un ritardo maggiore tra il cluster di database di origine e la replica di lettura a causa della maggiore lunghezza dei canali di rete tra Regioni AWS.

- I dati trasferiti per repliche tra regioni comportano costo di trasferimento dei dati Amazon RDS. Le seguenti operazioni di replica tra regioni generano addebiti per i dati trasferiti fuori dalla Regione AWS di origine:
  - Quando si crea la replica di lettura, Amazon RDS acquisisce uno snapshot del cluster di origine e lo trasferisce alla Regione AWS che contiene la replica di lettura.
  - Per ogni modifica dei dati apportata nei database di origine, Amazon RDS trasferisce dati dalla regione di origine alla Regione AWS che contiene la replica di lettura.

  Per ulteriori informazioni sui prezzi del trasferimento dati Amazon RDS, consulta Prezzi di Amazon Aurora.
- Puoi eseguire più operazioni di creazione o eliminazione simultanee per repliche di lettura che fanno riferimento allo stesso cluster di database di origine. Tuttavia, devi rimanere entro il limite di cinque repliche di lettura per ogni cluster di database di origine.
- Per un efficace funzionamento della replica, ciascuna replica di lettura deve avere la stessa quantità di risorse di calcolo e storage del cluster di database di origine. Se si dimensiona il cluster di database di origine, si devono dimensionare anche le repliche di lettura.

Argomenti

- Prima di iniziare
- Creazione di un cluster di database per una replica di lettura tra Regioni per Aurora MySQL
- Promozione di una replica di lettura a cluster di database per Aurora MySQL
- Risoluzione dei problemi delle repliche tra Regioni per Amazon Aurora MySQL

Prima di iniziare

Prima di poter creare un cluster di database di Aurora MySQL che è una replica di lettura tra regioni, è necessario abilitare l'accesso binario al cluster di database fonte di Aurora MySQL. La replica tra regioni per Aurora MySQL utilizza la replica binaria MySQL per riprodurre le modifiche nel cluster di database di replica di lettura tra regioni.

Per abilitare l'accesso binario a un cluster database Aurora MySQL, aggiornare il parametro `binlog_format` per il cluster database fonte. Il parametro `binlog_format` è un parametro a livello di cluster che si trova nel gruppo di parametri cluster predefinito. Se il cluster database utilizza il gruppo di parametri cluster database predefinito, crea un nuovo gruppo di parametri cluster database per modificare le impostazioni `binlog_format`. Consigliamo di impostare `binlog_format` su

`MIXED`. Tuttavia, puoi anche impostare `binlog_format` su `ROW` o `STATEMENT` se hai bisogno di un formato binlog specifico. Riavvia il cluster database Aurora affinché venga applicata la modifica.

Per ulteriori informazioni sull'utilizzo di registrazione binaria con Aurora, consulta Replica tra Aurora e MySQL o tra Aurora e un altro cluster di database Aurora (replica dei log binari). Per ulteriori informazioni sulla modifica dei parametri di configurazione di Aurora MySQL, consulta Parametri dell'istanza database e del cluster database di Amazon Aurora e Gruppi di parametri per Amazon Aurora.

Creazione di un cluster di database per una replica di lettura tra Regioni per Aurora MySQL

Puoi creare un cluster database Aurora che è una replica di lettura tra regioni utilizzando la Console di gestione AWS, la AWS Command Line Interface (AWS CLI) o l'API di Amazon RDS. Puoi creare repliche di lettura tra regioni da cluster di database crittografati e non crittografati.

Quando crei una replica di lettura tra Regioni per Aurora MySQL utilizzando la Console di gestione AWS, Amazon RDS crea un cluster database nella Regione AWS di destinazione e poi crea automaticamente un'istanza database che è l'istanza primaria per quel cluster database.

Quando crei una replica di lettura tra Regioni utilizzando la AWS CLI o l'API RDS, prima crea il cluster database nella Regione AWS di destinazione e attendi che diventi attivo. Quando è attivo, crei un'istanza database che è l'istanza primaria per il cluster database.

La replica inizia quando l'istanza primaria del cluster di database di replica di lettura diventa disponibile.

Utilizza le seguenti procedure per creare una replica di lettura tra regioni da un cluster di database di Aurora MySQL. Queste procedure funzionano per la creazione di repliche di lettura da cluster di database crittografati o non crittografati.

Console

Come creare un cluster database Aurora MySQL che è una replica di lettura tra regioni con la Console di gestione AWS

1. Accedi alla Console di gestione AWS e apri la console Amazon RDS all'indirizzo https://console.aws.amazon.com/rds/.
2. Nell'angolo in alto a destra della Console di gestione AWS, selezionare la Regione AWS che ospita il cluster di database di origine.

3. Nel riquadro di navigazione, scegliere Databases (Database).
4. Seleziona il cluster database per cui vuoi creare una replica di lettura tra Regioni.
5. In Actions (Operazioni), scegli Create cross region read replica (Crea replica di lettura tra regioni).
6. Nella pagina Create cross region read replica (Crea replica di lettura tra regioni), scegliere le impostazioni di opzione per il cluster di database di replica di lettura tra regioni, come descritto nella seguente tabella.

Opzione	Descrizione
Regione di destinazione	Seleziona la Regione AWS che ospita il nuovo cluster di database di replica di lettura tra Regioni.
Destination DB subnet group (Gruppo di sottoreti del database di destinazione)	Seleziona il gruppo di sottoreti di database da utilizzare per il cluster di database di replica di lettura tra regioni.
Accessibile pubblicamente	Seleziona Yes (Sì) per assegnare al cluster di database di replica di lettura tra regioni un indirizzo IP pubblico; altrimenti, seleziona No.
Encryption (Crittografia)	Selezionare Enable encryption (Abilita crittografia) per abilitare la crittografia dei dati inattivi del cluster database. Per ulteriori informazioni, consulta Crittografia delle risorse Amazon Aurora.
AWS KMS key	Disponibile solo se Encryption (Crittografia) è impostato su Enable encryption (Abilita crittografia). Selezionare la AWS KMS key da utilizzare per crittografare questo cluster DB. Per ulteriori informazioni, consulta Crittografia delle risorse Amazon Aurora.
DB instance class (Classe istanza database)	Scegliere una classe di istanza database che definisca i requisiti di elaborazione e di memoria per l'istanza primaria nel cluster DB. Per ulteriori informazioni sulle opzioni di classe di istanza database, consulta Classi di istanze database Amazon Aurora.

Opzione	Descrizione
Multi-AZ deployment (Implementazione Multi-AZ)	Seleziona Yes (Sì) per creare una replica di lettura del nuovo cluster di database in un'altra zona di disponibilità nella Regione AWS di destinazione per il supporto per failover. Per altre informazioni sulle zone di disponibilità multiple, consulta Regioni e zone di disponibilità.
Read replica source (Origine replica di lettura)	Seleziona il cluster di database di origine per il quale creare una replica di lettura tra regioni.
DB instance identifier (Identificatore istanze DB)	Digita un nome per l'istanza primaria nel cluster di database di replica di lettura tra regioni. Questo identificatore viene utilizzato nell'indirizzo dell'endpoint per l'istanza primaria del nuovo cluster database. L'identificatore istanze database presenta i seguenti vincoli: • Deve contenere da 1 a 63 caratteri alfanumerici o trattini. • Il primo carattere deve essere una lettera. • Non può terminare con un trattino o contenere due trattini consecutivi. • Deve essere univoco per tutte le istanze database, per ogni Account AWS e in ogni Regione AWS. In quanto il cluster di database di replica di lettura tra regione viene creato da una snapshot del cluster database origine, il nome utente master e la password master per la replica di lettura corrispondono al nome utente master e alla password master per il cluster di database di origine.

Opzione	Descrizione
DB Cluster Identifier (Identificatore cluster DB)	Digita un nome per il cluster database di replica di lettura tra Regioni che sia univoco per l'account nella Regione AWS di destinazione per la replica. Questo identificatore viene utilizzato nell'indirizzo dell'endpoint del cluster per il cluster database. Per informazioni sull'endpoint del cluster, consulta Connessioni degli endpoint Amazon Aurora. L'identificatore del cluster DB prevede i seguenti vincoli: • Deve contenere da 1 a 63 caratteri alfanumerici o trattini. • Il primo carattere deve essere una lettera. • Non può terminare con un trattino o contenere due trattini consecutivi. • Deve essere univoco per tutti i cluster database, per ogni Account AWS e in ogni Regione AWS.
Priorità	Seleziona una priorità di failover per l'istanza primaria del nuovo cluster database. Questa priorità determina l'ordine di promozione delle repliche di Aurora durante il recupero da un errore dell'istanza principale. Se non si specifica alcun valore, l'impostazione predefinita è tier-1 (livello 1). Per ulteriori informazioni, consulta Tolleranza ai guasti di un cluster DB Aurora.
Database port (Porta del database)	Specifica la porta per applicazioni e utilità da utilizzare per accedere al database. I cluster database Aurora hanno come porta predefinita MySQL, 3306. I firewall presso alcune aziende bloccano le connessioni a questa porta. Se il firewall dell'azienda blocca la porta predefinita, scegliere un'altra porta per il nuovo cluster database.

Opzione	Descrizione
Enhanced Monitoring (Monitoraggio avanzato)	Scegliere Enable enhanced monitoring (Abilita monitoraggio avanzato) per abilitare la raccolta di parametri in tempo reale per il sistema operativo su cui viene eseguito il cluster database. Per ulteriori informazioni, consulta Monitoraggio dei parametri del sistema operativo con il monitoraggio avanzato.
Monitoring Role (Ruolo monitoraggio)	Disponibile solo se Enhanced Monitoring (Monitoraggio avanzato) è impostato su Enable enhanced monitoring (Abilita monitoraggio avanzato). Selezionare il ruolo IAM creato per consentire a Amazon RDS di comunicare con Amazon CloudWatch Logs oppure scegliere Default (Predefinito) per fare in modo che RDS crei automaticamente un ruolo denominato `rds-monitoring-role` . Per ulteriori informazioni, consulta Monitoraggio dei parametri del sistema operativo con il monitoraggio avanzato.
Granularity (Granularità)	Disponibile solo se Enhanced Monitoring (Monitoraggio avanzato) è impostato su Enable enhanced monitoring (Abilita monitoraggio avanzato). Impostare l'intervallo, in secondi, tra le operazioni di raccolta dei parametri per il cluster DB.
Auto minor version upgrade (Aggiornamento automatico della versione secondaria)	Queste impostazioni non si applicano ai cluster di database Aurora MySQL. Per ulteriori informazioni sugli aggiornamenti del motore per Aurora MySQL, consultare Aggiornamenti del motore del database per Amazon Aurora MySQL.

7. Seleziona Create (Crea) per creare la replica di lettura tra regioni per Aurora.

AWS CLI

Per creare un cluster di database Aurora MySQL che è una replica di lettura tra regioni con CLI

1. Chiamare il comando AWS CLI create-db-cluster nella Regione AWS in cui si desidera creare il cluster di database di replica di lettura. Includi l'opzione `--replication-source-identifier` e specifica l' Amazon Resource Name (ARN) del cluster di database per il quale creare una replica di lettura.

   Per la replica tra regioni dove il cluster di database identificato tramite `--replication-source-identifier` è crittografato, è necessario specificare anche le opzioni `--kms-key-id` e `--storage-encrypted`.

   > **Note**
   >
   > Puoi impostare repliche tra regioni da un cluster di database crittografato a una replica di lettura crittografata specificando `--storage-encrypted` e fornendo un valore per `--kms-key-id`.

   Non puoi specificare i parametri `--master-username` e `--master-user-password`. Tali valori vengono presi dal cluster di database di origine.

   Il seguente esempio di codice crea una replica di lettura nella regione us-east-1 da uno snapshot di cluster di database non crittografato nella regione us-west-2. Il comando viene chiamato nella regione Stati Uniti orientali 1. In questo esempio è specificata l'opzione `--manage-master-user-password` per generare la password dell'utente master e gestirla in Secrets Manager. Per ulteriori informazioni, consulta Gestione delle password con Amazon Aurora e Gestione dei segreti AWS. In alternativa, puoi utilizzare l'opzione `--master-password` per specificare e gestire personalmente la password.

   Per Linux, macOS o Unix:

```
aws rds create-db-cluster \
  --db-cluster-identifier sample-replica-cluster \
  --engine aurora-mysql \
  --engine-version 8.0.mysql_aurora.3.08.0 \
  --replication-source-identifier arn:aws:rds:us-
west-2:123456789012:cluster:sample-master-cluster
```

Per Windows:

```
aws rds create-db-cluster ^
  --db-cluster-identifier sample-replica-cluster ^
  --engine aurora-mysql ^
  --engine-version 8.0.mysql_aurora.3.08.0 ^
  --replication-source-identifier arn:aws:rds:us-
west-2:123456789012:cluster:sample-master-cluster
```

Il seguente esempio di codice crea una replica di lettura nella regione us-east-1 da uno snapshot di cluster di database crittografato nella regione us-west-2. Il comando viene chiamato nella regione Stati Uniti orientali 1.

Per Linux, macOS o Unix:

```
aws rds create-db-cluster \
  --db-cluster-identifier sample-replica-cluster \
  --engine aurora-mysql \
  --engine-version 8.0.mysql_aurora.3.08.0 \
  --replication-source-identifier arn:aws:rds:us-
west-2:123456789012:cluster:sample-master-cluster \
  --kms-key-id my-us-east-1-key \
  --storage-encrypted
```

Per Windows:

```
aws rds create-db-cluster ^
  --db-cluster-identifier sample-replica-cluster ^
  --engine aurora-mysql ^
  --engine-version 8.0.mysql_aurora.3.08.0 ^
  --replication-source-identifier arn:aws:rds:us-
west-2:123456789012:cluster:sample-master-cluster ^
  --kms-key-id my-us-east-1-key ^
  --storage-encrypted
```

L'opzione `--source-region` è obbligatoria per la replica tra la regione AWS GovCloud (Stati Uniti orientali) e la regione AWS GovCloud (Stati Uniti occidentali), dove il cluster di database identificato da `--replication-source-identifier` è crittografato. Per `--source-region`, specificare la Regione AWS del cluster di database di origine.

Se non si specifica `--source-region`, è necessario specificare un valore per `--pre-signed-url`. Un URL prefirmato è un URL che contiene una richiesta firmata Signature Versione 4 per il comando `create-db-cluster` chiamato nella Regione AWS di origine. Per ulteriori informazioni sull'opzione `pre-signed-url`, consulta create-db-cluster in Riferimento ai comandi AWS CLI.

2. Controlla che il cluster di database sia diventato disponibile utilizzando il comando AWS CLI describe-db-clusters, come visualizzato nell'esempio seguente.

```
aws rds describe-db-clusters --db-cluster-identifier sample-replica-cluster
```

Quando i risultati **`describe-db-clusters`** mostrano uno stato `available`, crea l'istanza primaria per il cluster database affinché la replica possa iniziare. A tale scopo, utilizza il comando AWS CLI create-db-instance, come visualizzato nell'esempio seguente.

Per Linux, macOS o Unix:

```
aws rds create-db-instance \
  --db-cluster-identifier sample-replica-cluster \
  --db-instance-class db.r5.large \
  --db-instance-identifier sample-replica-instance \
  --engine aurora-mysql
```

Per Windows:

```
aws rds create-db-instance ^
  --db-cluster-identifier sample-replica-cluster ^
  --db-instance-class db.r5.large ^
  --db-instance-identifier sample-replica-instance ^
  --engine aurora-mysql
```

Quando l'istanza database viene creata e diventa disponibile, inizia la replica. Puoi determinare se l'istanza database è disponibile chiamando il comando AWS CLI describe-db-instances.

API RDS

Per creare un cluster di database di Aurora MySQL che è una replica di lettura tra regioni con l'API

1. Chiama l'operazione dell'API RDS CreateDBCluster nella Regione AWS in cui desideri creare il cluster di database di replica di lettura. Includi il parametro `ReplicationSourceIdentifier` e specifica l' Amazon Resource Name (ARN) del cluster di database per il quale creare una replica di lettura.

   Per la replica tra regioni dove il cluster di database identificato da `ReplicationSourceIdentifier` è crittografato, specifica il parametro `KmsKeyId` e impostare il parametro `StorageEncrypted` su `true`.

   > **Note**
   >
   > Puoi configurare repliche tra regioni da un cluster di database crittografato a una replica di lettura crittografata specificando `StorageEncrypted` come **`true`** e fornendo un valore per `KmsKeyId`. In questo caso, non devi specificare `PreSignedUrl`.

   Non devi includere i parametri `MasterUsername` e `MasterUserPassword`, perché quei valori sono presi dal cluster database origine.

   Il seguente esempio di codice crea una replica di lettura nella regione us-east-1 da uno snapshot di cluster di database non crittografato nella regione us-west-2. Questa operazione viene chiamata nella regione us-east-1.

```
https://rds.us-east-1.amazonaws.com/
  ?Action=CreateDBCluster
  &ReplicationSourceIdentifier=arn:aws:rds:us-west-2:123456789012:cluster:sample-
master-cluster
  &DBClusterIdentifier=sample-replica-cluster
  &Engine=aurora-mysql
  &SignatureMethod=HmacSHA256
  &SignatureVersion=4
  &Version=2014-10-31
  &X-Amz-Algorithm=AWS4-HMAC-SHA256
  &X-Amz-Credential=AKIADQKE4SARGYLE/20161117/us-east-1/rds/aws4_request
  &X-Amz-Date=20160201T001547Z
  &X-Amz-SignedHeaders=content-type;host;user-agent;x-amz-content-sha256;x-amz-date
```

```
  &X-Amz-Signature=a04c831a0b54b5e4cd236a90dcb9f5fab7185eb3b72b5ebe9a70a4e95790c8b7
```

Il seguente esempio di codice crea una replica di lettura nella regione us-east-1 da uno snapshot di cluster di database crittografato nella regione us-west-2. Questa operazione viene chiamata nella regione us-east-1.

```
https://rds.us-east-1.amazonaws.com/
  ?Action=CreateDBCluster
  &KmsKeyId=my-us-east-1-key
  &StorageEncrypted=true
  &PreSignedUrl=https%253A%252F%252Frds.us-west-2.amazonaws.com%252F
         %253FAction%253DCreateDBCluster
         %2526DestinationRegion%253Dus-east-1
         %2526KmsKeyId%253Dmy-us-east-1-key
         %2526ReplicationSourceIdentifier%253Darn%25253Aaws%25253Ards%25253Aus-
west-2%25253A123456789012%25253Acluster%25253Asample-master-cluster
         %2526SignatureMethod%253DHmacSHA256
         %2526SignatureVersion%253D4
         %2526Version%253D2014-10-31
         %2526X-Amz-Algorithm%253DAWS4-HMAC-SHA256
         %2526X-Amz-Credential%253DAKIADQKE4SARGYLE%252F20161117%252Fus-
west-2%252Frds%252Faws4_request
         %2526X-Amz-Date%253D20161117T215409Z
         %2526X-Amz-Expires%253D3600
         %2526X-Amz-SignedHeaders%253Dcontent-type%253Bhost%253Buser-agent%253Bx-
amz-content-sha256%253Bx-amz-date
         %2526X-Amz-Signature
%253D255a0f17b4e717d3b67fad163c3ec26573b882c03a65523522cf890a67fca613
  &ReplicationSourceIdentifier=arn:aws:rds:us-west-2:123456789012:cluster:sample-
master-cluster
  &DBClusterIdentifier=sample-replica-cluster
  &Engine=aurora-mysql
  &SignatureMethod=HmacSHA256
  &SignatureVersion=4
  &Version=2014-10-31
  &X-Amz-Algorithm=AWS4-HMAC-SHA256
  &X-Amz-Credential=AKIADQKE4SARGYLE/20161117/us-east-1/rds/aws4_request
  &X-Amz-Date=20160201T001547Z
  &X-Amz-SignedHeaders=content-type;host;user-agent;x-amz-content-sha256;x-amz-date
  &X-Amz-Signature=a04c831a0b54b5e4cd236a90dcb9f5fab7185eb3b72b5ebe9a70a4e95790c8b7
```

Per la replica tra regioni tra la regione AWS GovCloud (Stati Uniti orientali) e la regione AWS GovCloud (Stati Uniti occidentali), in cui il cluster di database identificato da `ReplicationSourceIdentifier` è crittografato, specificare anche il parametro `PreSignedUrl`. L'URL prefirmato deve essere una richiesta valida per l'operazione API `CreateDBCluster` che può essere eseguita nella Regione AWS di origine contenente il cluster di database crittografato da replicare. L'identificatore della chiave KMS viene utilizzato per crittografare la replica di lettura e deve essere una chiave KMS valida per la Regione AWS di destinazione. Per generare automaticamente invece che manualmente un URL prefirmato, utilizzare il comando AWS CLI create-db-cluster, ma con l'opzione `--source-region`.

2. Controllare che il cluster di database sia diventato disponibile utilizzando l'operazione DescribeDBClusters dell'API RDS, come visualizzato nell'esempio seguente.

```
https://rds.us-east-1.amazonaws.com/
  ?Action=DescribeDBClusters
  &DBClusterIdentifier=sample-replica-cluster
  &SignatureMethod=HmacSHA256
  &SignatureVersion=4
  &Version=2014-10-31
  &X-Amz-Algorithm=AWS4-HMAC-SHA256
  &X-Amz-Credential=AKIADQKE4SARGYLE/20161117/us-east-1/rds/aws4_request
  &X-Amz-Date=20160201T002223Z
  &X-Amz-SignedHeaders=content-type;host;user-agent;x-amz-content-sha256;x-amz-date
  &X-Amz-Signature=84c2e4f8fba7c577ac5d820711e34c6e45ffcd35be8a6b7c50f329a74f35f426
```

Quando lo stato dei risultati di `DescribeDBClusters` è `available`, creare l'istanza primaria per il cluster di database affinché la replica possa iniziare. A tale scopo, utilizzare l'operazione CreateDBInstance dell'API RDS come visualizzato nell'esempio seguente.

```
https://rds.us-east-1.amazonaws.com/
  ?Action=CreateDBInstance
  &DBClusterIdentifier=sample-replica-cluster
  &DBInstanceClass=db.r5.large
  &DBInstanceIdentifier=sample-replica-instance
  &Engine=aurora-mysql
  &SignatureMethod=HmacSHA256
  &SignatureVersion=4
  &Version=2014-10-31
  &X-Amz-Algorithm=AWS4-HMAC-SHA256
  &X-Amz-Credential=AKIADQKE4SARGYLE/20161117/us-east-1/rds/aws4_request
```

```
&X-Amz-Date=20160201T003808Z
&X-Amz-SignedHeaders=content-type;host;user-agent;x-amz-content-sha256;x-amz-date
&X-Amz-Signature=125fe575959f5bbcebd53f2365f907179757a08b5d7a16a378dfa59387f58cdb
```

Quando l'istanza database viene creata e diventa disponibile, inizia la replica. Puoi determinare se l'istanza database è disponibile chiamando il comando AWS CLI DescribeDBInstances.

Visualizzazione di repliche tra regioni Amazon Aurora MySQL

Puoi visualizzare le relazioni di replica tra regioni per i cluster di database di Amazon Aurora MySQL chiamando il comando AWS CLI describe-db-clusters o l'operazione API RDS DescribeDBClusters. Nella risposta, fai riferimento al campo `ReadReplicaIdentifiers` per gli identificatori di cluster di database di cluster di qualsiasi cluster di database di replica di lettura tra regioni. Fai riferimento all'elemento `ReplicationSourceIdentifier` per l'ARN del cluster di database di origine corrispondente all'origine della replica.

Promozione di una replica di lettura a cluster di database per Aurora MySQL

Puoi promuovere una replica di lettura Aurora MySQL a un cluster database standalone. Quando promuovi una replica di lettura Aurora MySQL, le relative istanze database vengono riavviate per essere disponibili.

In genere, promuovi una replica di lettura Aurora MySQL a un cluster database standalone come schema di ripristino dei dati in caso di errore del cluster database di origine.

A tale scopo, crea prima di tutto una replica di lettura e quindi monitora il cluster di database di origine per individuare eventuali errori. In caso di errore, sono necessarie le operazioni seguenti:

1. Promuovi la replica di lettura.
2. Indirizzare il traffico di database al cluster database promosso.
3. Creare una replica di lettura sostitutiva con il cluster di database promosso come origine.

Quando promuovi una replica di lettura, questa diventa un cluster di database Aurora standalone. Il processo di promozione può richiedere alcuni minuti per il completamento, che possono aumentare a seconda delle dimensioni della replica di lettura. Dopo aver promosso la replica di lettura a nuovo cluster di database, il cluster si comporta come qualsiasi altro cluster di database. Ad esempio, è possibile creare repliche di lettura dal cluster ed eseguire operazioni di ripristino point-in-time. È anche possibile creare repliche Aurora per il cluster database.

Poiché il cluster di database promosso non è più una replica di lettura, non è possibile usarlo come target di replica.

Le fasi seguenti illustrano il processo generale di promozione di una replica di lettura a cluster di database:

1. Arrestare la scrittura delle transazioni nel cluster di database di origine della replica di lettura e quindi attendere il completamento di tutti gli aggiornamenti nella replica di lettura. Gli aggiornamenti del database vengono eseguiti nella replica di lettura dopo essere stati completati nel cluster di database di origine e questo ritardo di replica può variare significativamente. Utilizzare il parametro `ReplicaLag` per determinare quando sono stati applicati tutti gli aggiornamenti alla replica di lettura. Il parametro `ReplicaLag` consente di registrare il tempo di ritardo di un'istanza database di replica di lettura rispetto all'istanza database di origine. Quando il parametro `ReplicaLag` diventa `0`, la replica di lettura ha raggiunto l'istanza database di origine.
2. Promuovi la replica di lettura utilizzando l'opzione Promuovi sulla console Amazon RDS, il comando AWS CLI promote-read-replica-db-cluster o l'operazione API Amazon RDS PromoteReadReplicaDBCluster.

   È possibile scegliere un'istanza database Aurora MySQL per promuovere la replica di lettura. Dopo la promozione della replica di lettura, il cluster di database di Aurora MySQL viene promosso a cluster di database standalone. L'istanza database con la priorità di failover maggiore viene promossa a istanza database primaria per il cluster database. Le altre istanze database diventano repliche Aurora.

   > **Note**
   >
   > Per il completamento del processo di promozione sono necessari alcuni minuti. Quando promuovi una replica di lettura, la replica viene arrestata e le istanze database vengono riavviate. Al termine del riavvio, la replica di lettura è disponibile come un nuovo cluster di database.

Console

Per promuovere una replica di lettura Aurora MySQL a cluster database

1. Accedi alla Console di gestione AWS e apri la console Amazon RDS all'indirizzo https://console.aws.amazon.com/rds/.

2. Nella console scegliere Instances (Istanze).

   Viene visualizzato il riquadro Instance (Istanza).
3. Nel riquadro Instances (Istanze), scegliere la replica di lettura che si vuole promuovere.

   Le repliche di lettura vengono visualizzate come istanze database Aurora MySQL.
4. Per Actions (Operazioni), scegliere Promote read replica (Promuovere replica di lettura).
5. Nella pagina di conferma scegliere Promote read replica (Promuovi replica di lettura).

AWS CLI

Per promuovere una replica di lettura a un cluster di database, utilizza il comando AWS CLI promote-read-replica-db-cluster.

Example

Per Linux, macOS o Unix:

```
aws rds promote-read-replica-db-cluster \
    --db-cluster-identifier mydbcluster
```

Per Windows:

```
aws rds promote-read-replica-db-cluster ^
    --db-cluster-identifier mydbcluster
```

API RDS

Per promuovere una replica di lettura a cluster di database, chiama PromoteReadReplicaDBCluster.

Risoluzione dei problemi delle repliche tra Regioni per Amazon Aurora MySQL

Di seguito è riportato un elenco di messaggi di errore comuni che si possono verificare durante la creazione di una replica di lettura tra regioni Amazon Aurora e le istruzioni per risolvere gli errori specificati.

Il cluster origine [cluster database ARN] non ha binlog abilitati

Per risolvere questo problema, abilitare l'accesso binario nel cluster database origine. Per ulteriori informazioni, consulta Prima di iniziare.

Il cluster origine [ARN cluster database] non ha il gruppo di parametri del cluster sincronizzato nello scrittore

Ricevi questo errore se hai aggiornato il parametro `binlog_format` del cluster database, ma non hai riavviato l'istanza primaria per il cluster database. Riavvia l'istanza primaria (ovvero, lo scrittore) per il cluster database e prova di nuovo.

Il cluster origine [ARN cluster database] ha già una replica di lettura in questa regione

Per ogni cluster di database di origine in qualsiasi Regione AWS, puoi avere fino a cinque cluster di database tra regioni che sono repliche di lettura. Se disponi già del numero massimo di repliche di lettura per un cluster di database in una particolare Regione AWS, devi eliminarne una esistente prima di poter creare un nuovo cluster di database tra regioni in tale regione.

Il cluster di database [ARN cluster di database] richiede l'aggiornamento di un motore di database per il supporto delle repliche tra regioni

Per risolvere questo problema, aggiorna la versione del motore di database per tutte le istanze nel cluster di database di origine alla versione di motore di database più recente e prova a creare di nuovo la replica di lettura tra regioni.

Replica tra Aurora e MySQL o tra Aurora e un altro cluster di database Aurora (replica dei log binari)

In quanto Amazon Aurora MySQL è compatibile con MySQL, puoi impostare la replica tra un database MySQL e un cluster di database Amazon Aurora MySQL. Questo tipo di replica utilizza la replica dei log binari MySQL ed è comunemente indicato come replica binlog. Se si utilizza la replica dei log binari con Aurora, si consiglia di eseguire sul database MySQL versione 5.5 o successiva. È possibile impostare la replica in cui il cluster Aurora MySQL del database è il master di replica o la replica. È possibile eseguire la replica con un'istanza DB Amazon RDS MySQL, un database MySQL esterno su Amazon RDS o un altro cluster DB Aurora MySQL.

> **Note**
>
> Non è possibile utilizzare la replica binlog da o verso determinati tipi di cluster di database Aurora. In particolare, la replica binlog non è disponibile per i cluster Aurora Serverless v1. Se l'istruzione `SHOW MASTER STATUS` e `SHOW SLAVE STATUS` (Aurora MySQL versione 2) o `SHOW REPLICA STATUS` (Aurora MySQL versione 3) non restituisce alcun output, controlla che il cluster in uso supporti la replica binlog.

Puoi anche eseguire una replica con un'istanza database RDS per MySQL o un cluster di database Aurora MySQL in un'altra Regione AWS. Quando esegui la replica tra Regioni AWS, assicurati che i cluster di database e le istanze database siano accessibili pubblicamente. Se i cluster di database Aurora MySQL si trovano in sottoreti private nel VPC, usa il peering VPC tra le Regioni AWS. Per ulteriori informazioni, consulta Un cluster database in un VPC a cui accede un'istanza EC2 in un VPC diverso.

Se intendi configurare la replica tra un cluster di database Aurora MySQL e un cluster di database Aurora MySQL in un'altra Regione AWS, puoi creare un cluster di database Aurora MySQL come replica di lettura in una Regione AWS differente da quella del cluster di database di origine. Per ulteriori informazioni, consulta Repliche di cluster di database Amazon Aurora MySQL tra Regioni AWS.

Con Aurora MySQL versione 2 e 3, è possibile effettuare la replica tra Aurora MySQL e un'origine o una destinazione esterna che utilizza gli identificatori globali di transazione (GTID) per la replica. Assicurati che i parametri basati su GTID nel cluster di database Aurora MySQL presentino impostazioni compatibili con lo stato GTID del database esterno. Per informazioni su come effettuare questa operazione, consulta Utilizzo della replica basata su GTID. In Aurora MySQL versione 3.01 e successive, puoi scegliere come assegnare GTID alle transazioni replicate da una fonteche non utilizza GTID. Per informazioni sulla procedura archiviata che controlla tale impostazione, vedere mysql.rds_assign_gtids_to_anonymous_transactions (Aurora MySQL versione 3).

> **Warning**
>
> Quando esegui la replica tra Aurora MySQL e MySQL, assicurati di utilizzare solo tabelle InnoDB. Se hai tabelle MyISAM, che desideri replicare, puoi convertirle in InnoDB prima di impostare la replica con il seguente comando.
>
> ```
> alter table <schema>.<table_name> engine=innodb, algorithm=copy;
> ```

Nelle sezioni seguenti, viene descritto come configurare la replica, interrompere la replica, scalare le letture per il database in uso, ottimizzare la replica binlog e configurare il file di log binario avanzato.

Argomenti

- Configurazione del processo di replica dei log binari per Aurora MySQL
- Arresto del processo di replica dei log binari per Aurora MySQL

- Dimensionamento delle letture per il database MySQL con Amazon Aurora
- Ottimizzazione della replica dei log binari per Aurora MySQL
- Configurazione del file di log binario avanzato per Aurora MySQL

Configurazione del processo di replica dei log binari per Aurora MySQL

L'impostazione della replica MySQL con Aurora MySQL prevede le seguenti fasi che vengono discusse in dettaglio:

Indice

- 1. Abilitare l'accesso binario nella fonte di replica
- 2. Mantieni i log binari nel master di replica fino a quando non sono più necessari
- 3. Creazione di una copia o un dump dell'origine della replica
- 4. Caricamento del dump nella destinazione di replica (se necessario)
- 5. Creazione di un utente di replica sull'origine di replica
- 6. Abilitare la replica nel target di replica
  - Definire una posizione per arrestare la replica su una replica di lettura
- 7. Monitora la replica
- Sincronizzazione delle password tra origine di replica e destinazione

1. Abilitare l'accesso binario nella fonte di replica

Seguono le istruzioni su come abilitare l'accesso binario nella fonte di replica per il motore del database.

Motore del database	Istruzioni
Aurora MySQL	Per abilitare l'accesso binario in un cluster di database Aurora MySQL Imposta il parametro del cluster di database `binlog_format` su `ROW`, `STATEMENT` o `MIXED`. `MIXED` è consigliato a meno che tu abbia bisogno di un formato binlog specifico. Il valore predefinito è `OFF`.

Motore del database	Istruzioni
	Per modificare il parametro `binlog_format` , crea un gruppo di parametri del cluster di database personalizzato e associalo al tuo cluster di database. Non puoi modificare i parametri in un gruppo di parametri del cluster di database predefinito. Se stai modificando il parametro `binlog_format` da `OFF` a un altro valore, riavvia il cluster di database Aurora per rendere effettiva la modifica. Per ulteriori informazioni, consultare Parametri dell'istanza database e del cluster database di Amazon Aurora e Gruppi di parametri per Amazon Aurora.
RDS for MySQL	Per abilitare l'accesso binario in un'istanza database Amazon RDS Non è possibile abilitare l'accesso binario direttamente per un'istanza database Amazon RDS, ma si può abilitarlo procedendo in uno dei seguenti modi: • Abilitare i backup automatici per l'istanza database. È possibile abilitare i backup automatici quando si crea un'istanza database o si possono abilitare i backup modificando un'istanza database esistente. Per ulteriori informazioni, consulta la pagina relativa alla creazione di un'istanza database nella Guida per l'utente di Amazon RDS. • Crea una replica di lettura per l'istanza database. Per ulteriori informazioni, consulta Uso di repliche di lettura nella Guida per l'utente di Amazon RDS.

Motore del database	Istruzioni
MySQL (esterno)	Per impostare la replica crittografata Per replicare i dati in maniera sicura con Aurora MySQL versione 2, puoi utilizzare la replica crittografata. **Note** Se non hai bisogno di utilizzare la replica crittografata, puoi saltare queste fasi. Seguono i prerequisiti per l'utilizzo della replica crittografata: • Secure Sockets Layer (SSL) deve essere abilitato su un database master MySQL esterno. • Una chiave e un certificato client devono essere preparati per il cluster di database Aurora MySQL. Durante la replica crittografata, il cluster di database Aurora MySQL agisce come un client per il server di database MySQL. I certificati e le chiavi per il client Aurora MySQL sono in file in formato .pem. 1. Assicurati di essere preparato per la replica crittografata: • Se SSL non è abilitato sul database MySQL fonte esterno e non si dispone di una chiave client e di un certificato client preparato, abilitare SSL sul server di database MySQL e generare la chiave client e il certificato client necessari. • Se SSL è abilitato sul master esterno, fornisci un certificato e una chiave client per il cluster di database Aurora MySQL. Se non disponi di questi elementi, genera una nuova chiave e un nuovo certificato per il cluster di database Aurora MySQL. Per firmare il certificato client, devi avere la chiave autorità certificato che hai utilizzato per configurare SSL nel database master esterno MySQL.

Motore del database	Istruzioni

Per ulteriori informazioni, consulta Creating SSL Certificates and Keys Using openssl nella documentazione MySQL.

Hai bisogno del certificato autorità certificato, della chiave client e del certificato client.

2. Connettersi al cluster di database Aurora MySQL come il master utilizzando SSL.

   Per informazioni sulla connessione a un cluster di database Aurora MySQL con SSL, consulta Connessioni TLS a cluster di database Aurora MySQL.

3. Esegui la procedura archiviata `mysql.rds_import_binlog_ssl_material` per importare le informazioni SSL nel cluster di database Aurora MySQL.

   Per il parametro `ssl_material_value` inserisci le informazioni dai file in formato .pem per il cluster di database Aurora MySQL nel payload JSON corretto.

   L'esempio seguente importa le informazioni SSL in un cluster di database Aurora MySQL. Nei file in formato .pem, il codice del corpo è in genere più lungo del codice del corpo riportato nell'esempio.

```
call mysql.rds_import_binlog_ssl_material(
'{"ssl_ca":"-----BEGIN CERTIFICATE-----
AAAAB3NzaC1yc2EAAAADAQABAAABAQClKsfkNkuSevGj3eYhCe53pcj
qP3maAhDFcvBS7O6V
hz2ItxCih+PnDSUaw+WNQn/mZphTk/a/gU8jEzoOWbkM4yxyb/wB96
xbiFveSFJuOp/d6RJhJOI0iBXr
lsLnBItntckiJ7FbtxJMXLvvwJryDUilBMTjYtwB+QhYXUMOzce5Pjz5/
i8SeJtjnV3iAoG/cQk+0FzZ
qaeJAAHco+CY/5WrUBkrHmFJr6HcXkvJdWPkYQS3xqC0+FmUZofz22
1CBt5IMucxXPkX4rWi+z7wB3Rb
BQoQzd8v7yeb7OzlPnWOyN0qFU0XA246RA8QFYiCNYwI3f05p6KLxEXAMPLE
-----END CERTIFICATE-----\n","ssl_cert":"-----BEGIN CERTIFICA
TE-----
```

Motore del database	Istruzioni

```
AAAAB3NzaC1yc2EAAAADAQABAAABAQClKsfkNkuSevGj3eYhCe53pcj
qP3maAhDFcvBS7O6V
hz2ItxCih+PnDSUaw+WNQn/mZphTk/a/gU8jEzoOWbkM4yxyb/wB96
xbiFveSFJuOp/d6RJhJOI0iBXr
lsLnBItntckiJ7FbtxJMXLvvwJryDUilBMTjYtwB+QhYXUMOzce5Pjz5/
i8SeJtjnV3iAoG/cQk+0FzZ
qaeJAAHco+CY/5WrUBkrHmFJr6HcXkvJdWPkYQS3xqC0+FmUZofz22
1CBt5IMucxXPkX4rWi+z7wB3Rb
BQoQzd8v7yeb7OzlPnWOyN0qFU0XA246RA8QFYiCNYwI3f05p6KLxEXAMPLE
-----END CERTIFICATE-----\n","ssl_key":"-----BEGIN RSA PRIVATE
 KEY-----
AAAAB3NzaC1yc2EAAAADAQABAAABAQClKsfkNkuSevGj3eYhCe53pc
jqP3maAhDFcvBS7O6V
hz2ItxCih+PnDSUaw+WNQn/mZphTk/a/gU8jEzoOWbkM4yxyb/wB96xbiFveSF
JuOp/d6RJhJOI0iBXr
lsLnBItntckiJ7FbtxJMXLvvwJryDUilBMTjYtwB+QhYXUMOzce5Pjz5/i8SeJ
tjnV3iAoG/cQk+0FzZ
qaeJAAHco+CY/5WrUBkrHmFJr6HcXkvJdWPkYQS3xqC0+FmUZofz221CBt5IMu
cxXPkX4rWi+z7wB3Rb
BQoQzd8v7yeb7OzlPnWOyN0qFU0XA246RA8QFYiCNYwI3f05p6KLxEXAMPLE
-----END RSA PRIVATE KEY-----\n"}');
```

Per ulteriori informazioni, consultare mysql.rds_import_binlog_ssl_material e Connessioni TLS a cluster di database Aurora MySQL.

> **Note**
>
> Dopo aver eseguito la procedura, i segreti vengono archiviati in file. Per eliminare i file in un secondo momento, puoi eseguire la stored procedure mysql.rds_remove_binlog_ssl_material.

Per abilitare l'accesso binario a un database esterno MySQL

1. Da un shell comando, interrompi il servizio mysql.

Motore del database	Istruzioni

```
sudo service mysqld stop
```

2. Modificare il file `my.cnf` (posto in genere sotto `/etc`).

```
sudo vi /etc/my.cnf
```

Aggiungere le opzioni `log_bin` e `server_id` alla sezione `[mysqld]`. L'opzione `log_bin` fornisce un identificatore di nome file per i file di log binari. L'opzione `server_id` fornisce un identificatore univoco per il server in relazioni master-replica.

Se la replica crittografata non è necessaria, assicurarsi che il database MySQL esterno venga avviato con binlog abilitati e SSL disabilitato.

Seguono le voci rilevanti nel file `/etc/my.cnf` per dati non crittografati.

```
log-bin=mysql-bin
server-id=2133421
innodb_flush_log_at_trx_commit=1
sync_binlog=1
```

Se la replica crittografata è necessaria, assicurati che il database MySQL esterno venga avviato con SSL e i binlog abilitati.

Le voci rilevanti nel file `/etc/my.cnf` includono le posizioni del file .pem per il server di database MySQL.

```
log-bin=mysql-bin
server-id=2133421
innodb_flush_log_at_trx_commit=1
sync_binlog=1

# Setup SSL.
ssl-ca=/home/sslcerts/ca.pem
ssl-cert=/home/sslcerts/server-cert.pem
```

Motore del database	Istruzioni

```
ssl-key=/home/sslcerts/server-key.pem
```

In aggiunta, l'opzione `sql_mode` per l'istanza database MySQL deve essere impostata su 0 o non deve essere inclusa nel file my.cnf.

Durante la connessione al database esterno MySQL, registra la posizione del log binario del database esterno MySQL.

```
mysql> SHOW MASTER STATUS;
```

L'output visualizzato dovrebbe essere simile al seguente:

```
+------------------+----------+--------------+------------------
+-------------------+
| File             | Position | Binlog_Do_DB | Binlog_Ignore_DB |
 Executed_Gtid_Set |
+------------------+----------+--------------+------------------
+-------------------+
| mysql-bin.000031 |      107 |              |                  |
              |
+------------------+----------+--------------+------------------
+-------------------+
1 row in set (0.00 sec)
```

Per ulteriori informazioni, vedere Setting the replication source configuration nella documentazione di MySQL.

3. Avvia il servizio mysql.

```
sudo service mysqld start
```

2. Mantieni i log binari nel master di replica fino a quando non sono più necessari

Quando si utilizza la replica dei log binari MySQL, Amazon RDS non gestisce il processo di replica. Come risultato, devi assicurarti che i file binlog nel master di replica siano mantenuti finché le

modifiche vengono applicate alla replica. Questa manutenzione aiuta a ripristinare il database di origine in caso di errore.

Utilizza le istruzioni seguenti per mantenere i registri binari per il motore di database.

Motore del database	Istruzioni
Aurora MySQL	Per mantenere i log binari in un cluster di database Aurora MySQL Non hai accesso ai file binlog per un cluster di database Aurora MySQL. Come risultato, devi selezionare un intervallo di tempo per mantenere i file binlog nel master di replica abbastanza a lungo per assicurare che le modifiche vengano applicate alla replica prima che il file binlog venga eliminato da Amazon RDS. Puoi mantenere i file binlog in un cluster di database Aurora MySQL fino a 90 giorni. Se stai impostando la replica con un database MySQL o un'istanza database RDS per MySQL come replica e il database per il quale stai creando la replica è di grandi dimensioni, seleziona un intervallo di tempo ampio per mantenere i file binlog finché la copia iniziale del database nella replica sia completa e il ritardo di replica sia pari a 0. Per impostare il periodo di conservazione dei log binari, usa la procedura mysql.rds_set_configuration e specifica il parametro di configurazione `'binlog retention hours'` insieme al numero di ore di conservazione dei file binlog nel cluster di database. Il valore massimo per Aurora MySQL versione 2.11.0 e successive e versione 3 è 2160 (90 giorni). L'esempio seguente imposta il periodo di conservazione dei file binlog su 6 giorni: `CALL mysql.rds_set_configuration('binlog retention hours', 144);` Dopo l'inizio della replica, è possibile verificare che le modifiche siano state applicate alla replica eseguendo il comando `SHOW SLAVE STATUS` (Aurora MySQL versione 2) o `SHOW REPLICA STATUS` (Aurora MySQL versione 3) nella replica e controllando il campo `Seconds behind master`. Se il campo `Seconds behind master` è 0, non vi è alcun ritardo di replica. Quando non c'é ritardo di replica, riduci il periodo

Motore del database	Istruzioni
	di tempo di conservazione dei file binlog impostando il parametro di configurazione `binlog retention hours` su un intervallo di tempo più breve. Se questa impostazione non è specificata, verrà utilizzato il valore predefinito per Aurora MySQL ovvero 24 (1 giorno). Se si specifica un valore per `'binlog retention hours'` che è superiore al valore massimo, allora Aurora MySQL utilizza il valore massimo.
RDS per MySQL	Per mantenere i log binari in un'istanza database Amazon RDS Puoi mantenere i file dei registri binari in un'istanza database di Amazon RDS impostando le ore di conservazione del binlog allo stesso modo di un cluster di database Aurora MySQL, come descritto nella riga precedente. Puoi anche mantenere i file binlog in un'istanza database Amazon RDS creando una replica di lettura per l'istanza database. La replica di lettura è temporanea e ha solamente l'obiettivo di mantenere i file binlog. Dopo che la replica di lettura è stata creata, chiama la procedura mysql.rds_stop_replication nella replica di lettura. Mentre la replica è interrotta, Amazon RDS non elimina nessuno dei file binlog nel master di replica. Dopo aver impostato la replica con la replica permanente, puoi eliminare la replica di lettura quando il ritardo di replica (campo `Seconds behind master`) tra il master di replica a la replica permanente diventa 0.
MySQL (esterno)	Per mantenere i log binari in un database esterno MySQL Siccome i file binlog in un database MySQL non sono gestiti da Amazon RDS, vengono mantenuti finché li elimini. Dopo l'inizio della replica, è possibile verificare che le modifiche siano state applicate alla replica eseguendo il comando `SHOW SLAVE STATUS` (Aurora MySQL versione 2) o `SHOW REPLICA STATUS` (Aurora MySQL versione 3) nella replica e controllando il campo `Seconds behind master`. Se il campo `Seconds behind master` è 0, non vi è alcun ritardo di replica. Quando non c'é ritardo di replica, puoi eliminare i vecchi file binlog.

3. Creazione di una copia o un dump dell'origine della replica

È possibile utilizzare uno snapshot, un clone o un dump dell'origine della replica per caricare una copia di base dei dati nella replica. Quindi si inizia a replicare da quel punto.

Utilizza le istruzioni seguenti per creare una copia o un dump dell'origine della replica per il motore di database.

Motore del database	Istruzioni
Aurora MySQL	Come creare una copia di un cluster di database Aurora MySQL Seleziona uno dei seguenti metodi: • Ripristino da uno snapshot di un cluster di database: 1. Crea una snapshot di un cluster di database del cluster di database Amazon Aurora. Per ulteriori informazioni, consulta Creazione di uno snapshot del cluster database. 2. Crea un nuovo cluster di database Aurora ripristinandolo dalla snapshot cluster di database che hai appena creato. Assicurati di mantenere il gruppo di parametri database per il cluster di database ripristinato come con il cluster di database originale. Ciò assicura che la copia del cluster di database abbia l'accesso binario abilitato. Per ulteriori informazioni, consulta Ripristino da uno snapshot cluster database. • Clona il tuo cluster di database. Per ulteriori informazioni, consulta Clonazione di un volume per un cluster di database Amazon Aurora. Come determinare il nome e la posizione del file binlog Seleziona uno dei seguenti metodi: • Nella Console di gestione AWS: 1. Scegli Database, quindi scegli l'istanza primaria (di scrittura) per fare in modo che il nuovo cluster di database Aurora mostri i dettagli.

Motore del database	Istruzioni

2. Scorri fino a Recent Events (Eventi recenti). Viene visualizzato un messaggio di evento che include il nome e la posizione del file binlog. Il messaggio evento è nel formato seguente.

   ```
   Binlog position from crash recovery is binlog-file-name  binlog-position
   ```

3. Salva i valori del nome e della posizione del file binlog per quando avvierai la replica.

- Chiama il comando describe-events dell'interfaccia AWS CLI, come nell'esempio seguente.

  ```
  aws rds describe-events

  {
      "Events": [
          {
              "EventCategories": [],
              "SourceType": "db-instance",
              "SourceArn": "arn:aws:rds:us-west-2:123456789012:
  db:sample-restored-instance",
              "Date": "2016-10-28T19:43:46.862Z",
              "Message": "Binlog position from crash recovery is mysql-
  bin-changelog.000003 4278",
              "SourceIdentifier": "sample-restored-instance"
          }
      ]
  }
  ```

- Controlla nel log degli errori di MySQL l'ultima posizione del file binlog MySQL.

Come creare un dump di un cluster di database Aurora MySQL

Se la destinazione di replica è un database MySQL esterno o un'istanza database RDS per MySQL, devi creare un file dump dal cluster di database Aurora.

Motore del database	Istruzioni
	Assicurati che il comando `mysqldump` venga eseguito nella copia del cluster di database di origine che hai creato. In questo modo si evitano considerazioni bloccanti quando si esegue il dump. Se il dump fosse eseguito direttamente sul cluster di database di origine, sarebbe necessario bloccare le tabelle di origine per evitare scritture simultanee su di esse mentre il dump è in corso. 1. Effettua la connessione al tuo cluster di database utilizzando un client MySQL. 2. Emetti il comando `mysqldump` . Per esempio: `PROMPT> mysqldump --databases` *`database_name`* `--single-transaction --order-by-primary -r backup.sql -u` *`local_user`* `s -p` 3. Dopo aver creato il file dump, puoi eliminare la copia del cluster di database.
RDS per MySQL	Per creare una snapshot di un'istanza database Amazon RDS Crea una replica di lettura dell'istanza database Amazon RDS. Per ulteriori informazioni, consulta Creazione di una replica di lettura nella Guida per l'utente di Amazon Relational Database Service. 1. Esegui la connessione alla replica di lettura e interrompi la replica eseguendo la procedura mysql.rds_stop_replication. 2. Mentre la replica di lettura è Arrestata, esegui la connessione alla replica di lettura e quindi il comando `SHOW SLAVE STATUS` (Aurora MySQL versione 2) o `SHOW REPLICA STATUS` (Aurora MySQL versione 3). Recupera il nome dei file di log binario corrente dal campo `Relay_Master_Log_File` e la posizione del file di log dal campo `Exec_Master_Log_Pos` . Salva questi valori per quando avvierai la replica. 3. Mentre la replica di lettura rimane nello stato Stopped (Interrotto), crea una snapshot DB di una replica di lettura. Per ulteriori informazioni, consulta Creazione di uno snapshot DB nella Guida per l'utente di Amazon Relational Database Service. 4. Elimina la replica di lettura.

Motore del database	Istruzioni

MySQL (esterno)	Per creare il dump di un database MySQL esterno
	1. Prima di creare un dump, devi assicurarti che la posizione del binlog per il dump sia aggiornata con i dati nell'istanza di origine. Per fare ciò, devi prima fermare qualsiasi operazione di scrittura all'istanza con il seguente comando:
	`mysql> FLUSH TABLES WITH READ LOCK;`
	2. Crea un dump di un database MySQL utilizzando il comando `mysqldump` come illustrato di seguito:
	`PROMPT> sudo mysqldump --databases database_name --master-data=2 --single-transaction \ --order-by-primary -r backup.sql -u local_user -p`
	3. Dopo aver creato il dump, sblocca le tabelle nel database MySQL con il seguente comando:
	`mysql> UNLOCK TABLES;`

4. Caricamento del dump nella destinazione di replica (se necessario)

Se intendi caricare i dati da un dump di un database MySQL che è esterno ad Amazon RDS, è consigliabile creare un'istanza EC2 in cui copiare i file dump. Quindi puoi caricare i dati nel tuo cluster di database o nell'istanza database da quell'istanza EC2. Utilizzando questo approccio, puoi comprimere i file dump prima di copiarli nell'istanza EC2 per ridurre i costi di rete associati con la copia dei dati in Amazon RDS. Puoi anche crittografare il file o i file dump per assicurare i dati mentre vengono trasferiti nella rete.

> **Note**
>
> Se crei un nuovo cluster di database Aurora MySQL come destinazione di replica, non è necessario che tu carichi un file dump:

- È possibile eseguire il ripristino da uno snapshot del cluster di database per creare un nuovo cluster di database. Per ulteriori informazioni, consulta Ripristino da uno snapshot cluster database.
- Puoi clonare il tuo cluster di database di origine per creare un nuovo cluster di database. Per ulteriori informazioni, consulta Clonazione di un volume per un cluster di database Amazon Aurora.
- Puoi eseguire la migrazione dei dati da uno snapshot di un'istanza database in un nuovo cluster di database. Per ulteriori informazioni, consulta Migrazione di dati a un cluster di database Amazon Aurora MySQL.

Utilizza le istruzioni seguenti per caricare il dump dell'origine di replica nella destinazione di replica per il motore di database.

Motore del database	Istruzioni

Aurora MySQL	Come caricare un dump in un cluster di database Aurora MySQL
	1. Copia l'output del comando `mysqldump` dal master di replica in una posizione che può anche connettersi al cluster di database Aurora MySQL.
	2. Connettersi al cluster di database Aurora MySQL utilizzando il comando `mysql`. Di seguito è riportato un esempio.
	`PROMPT> mysql -h host_name -port=3306 -u db_master_user -p`
	3. Al prompt `mysql`, esegui il comando `source` e passagli il nome del file dump del database per caricare i dati nel cluster di database Aurora MySQL, ad esempio:
	`mysql> source backup.sql;`

RDS per MySQL	Per caricare un dump in un'istanza database Amazon RDS 1. Copia l'output del comando `mysqldump` dal master di replica in una posizione che può anche connettersi all'istanza database MySQL.

Motore del database	Istruzioni

	2. Connettersi all'istanza database MySQL utilizzando il comando `mysql`. Di seguito è riportato un esempio.
	`PROMPT> mysql -h host_name -port=3306 -u db_master_user -p`
	3. Al prompt `mysql`, esegui il comando `source` e passagli il nome del file dump del database per caricare i dati nell'istanza database MySQL, ad esempio:
	`mysql> source backup.sql;`

MySQL (esterno)	Per caricare un dump in un database MySQL esterno
	Non è possibile caricare uno snapshot di database o di cluster di database in un database MySQL esterno. Devi utilizzare invece l'output dal comando `mysqldump` .
	1. Copia l'output del comando `mysqldump` dal master di replica in una posizione che può anche connettersi al database MySQL.
	2. Connettersi al database MySQL utilizzando il comando `mysql`. Di seguito è riportato un esempio.
	`PROMPT> mysql -h host_name -port=3306 -u db_master_user -p`
	3. Al prompt `mysql`, esegui il comando `source` e passagli il nome del file dump del database per caricare i dati nel database MySQL. Di seguito è riportato un esempio.
	`mysql> source backup.sql;`

5. Creazione di un utente di replica sull'origine di replica

Crea un ID utente sull'origine utilizzato solamente per la replica. L'esempio seguente riguarda database RDS per MySQL o database di origine MySQL esterni.

```
mysql> CREATE USER 'repl_user'@'domain_name' IDENTIFIED BY 'password';
```

Per i database di origine Aurora MySQL, il parametro del cluster di database `skip_name_resolve` è impostato su 1 (0N) e non può essere modificato, quindi è necessario utilizzare un indirizzo IP per l'host anziché un nome di dominio. Per ulteriori informazioni, consulta skip_name_resolve nella documentazione di MySQL.

```
mysql> CREATE USER 'repl_user'@'IP_address' IDENTIFIED BY 'password';
```

L'utente richiede i privilegi `REPLICATION CLIENT` e `REPLICATION SLAVE`. Concedi questi privilegi all'utente.

Se non hai bisogno di utilizzare la replica crittografata, richiedi le connessioni SSL per l'utente replica. Ad esempio, puoi utilizzare una delle seguenti istruzioni per richiedere connessioni SSL per l'account utente `repl_user`.

```
GRANT REPLICATION CLIENT, REPLICATION SLAVE ON *.* TO 'repl_user'@'IP_address';
```

```
GRANT USAGE ON *.* TO 'repl_user'@'IP_address' REQUIRE SSL;
```

> **Note**
>
> Se `REQUIRE SSL` non è incluso, la connessione di replica potrebbe ridiventare una connessione non crittografata.

6. Abilitare la replica nel target di replica

Raccomandiamo di fare una snapshot manuale del cluster di database Aurora MySQL o del target di replica dell'istanza database RDS for MySQL, prima di abilitare la replica. Se c'è un problema e devi ristabilire la replica con il cluster di database o il target di replica dell'istanza database, puoi ripristinare il cluster di database o l'istanza database da questa snapshot invece di dover importare di nuovo i dati nel target di replica.

Utilizza le istruzioni seguenti per attivare la replica del motore di database.

Motore del database	Istruzioni
Aurora MySQL	Per abilitare la replica da un cluster di database Aurora MySQL 1. Individua il punto di partenza per la replica. Hai bisogno del nome e della posizione del file binlog. Se la destinazione della replica del cluster di database è stata creata da: • Snapshot o clone del cluster di database: recupera il nome e la posizione del file binlog dagli eventi recenti per il tuo nuovo cluster di database, come mostrato in 3. Creazione di una copia o un dump dell'origine della replica. • Snapshot di database: hai recuperato il nome e la posizione del file binlog con il comando `SHOW SLAVE STATUS` (Aurora MySQL versione 2) o `SHOW REPLICA STATUS` (Aurora MySQL versione 3) quando hai creato lo snapshot dell'origine della replica. 2. Connettiti al cluster di database e richiama le seguenti procedure per avviare la replica con l'origine utilizzando il nome e il percorso del file di log binario del passaggio precedente: • mysql.rds_set_external_source (Aurora MySQL versione 3) • mysql.rds_set_external_master (Aurora MySQL versione 2) • mysql.rds_start_replication (tutte le versioni) L'esempio seguente si applica ad Aurora MySQL versione 3. `CALL mysql.rds_set_external_source ('mydbinstance.123456789012 .us-east-1.rds.amazonaws.com', 3306, 'repl_user', 'password', 'mysql-bin-changelog.000031', 107, 0); CALL mysql.rds_start_replication;` Per utilizzare la crittografia SSL, imposta il valore finale su 1 anziché `0`.
RDS per MySQL	Per abilitare la replica da un'istanza database Amazon RDS

Motore del database	Istruzioni
	1. Se il target di replica dell'istanza database è stato creato da una snapshot DB, hai bisogno del file e della posizione binlog che sono il punto di partenza per la replica. Hai recuperato questi valori con il comando `SHOW SLAVE STATUS` (Aurora MySQL versione 2) o `SHOW REPLICA STATUS` (Aurora MySQL versione 3), quando hai creato lo snapshot dell'origine della replica. 2. Esegui la connessione all'istanza database e chiama le procedure mysql.rds_set_external_master (Aurora MySQL versione 2) o mysql.rds_set_external_source (Aurora MySQL versione 3) e mysql.rds_start_replication per avviare la replica con l'origine. Utilizza il nome e il percorso del file di log binario del passaggio precedente. Di seguito è riportato un esempio. `CALL mysql.rds_set_external_master ('mydbcluster.cluster-12345 6789012.us-east-1.rds.amazonaws.com', 3306, 'repl_user', 'password', 'mysql-bin-changelog.000031', 107, 0); CALL mysql.rds_start_replication;` Per utilizzare la crittografia SSL, imposta il valore finale su 1 anziché `0`.

Motore del database	Istruzioni

MySQL (esterno)	Per abilitare la replica da un database esterno MySQL;
	1. Recupera il file e la posizione binlog che sono il punto di inizio per la replica. Hai recuperato questi valori con il comando `SHOW SLAVE STATUS` (Aurora MySQL versione 2) o `SHOW REPLICA STATUS` (Aurora MySQL versione 3), quando hai creato lo snapshot dell'origine della replica. Se il target di replica MySQL esterno è stato popolato dall'output del comando `mysqldump` con l'opzione `--master-data=2` , il file e la posizione binlog sono inclusi nell'output. Di seguito è riportato un esempio.
	`-- -- Position to start replication or point-in-time recovery from -- -- CHANGE MASTER TO MASTER_LOG_FILE='mysql-bin-changelog.000031', MASTER_LOG_POS=107;`
	2. Esegui la connessione alla destinazione della replica MySQL esterna ed emetti `CHANGE MASTER TO` e `START SLAVE` (Aurora MySQL versione 2) o `START REPLICA` (Aurora MySQL versione 3) per avviare la replica con l'origine della replica utilizzando il nome e il percorso del file di log binario dalla fase precedente, ad esempio:
	`CHANGE MASTER TO MASTER_HOST = 'mydbcluster.cluster-123456789012.us-east-1.rds.amazonaws.com', MASTER_PORT = 3306, MASTER_USER = 'repl_user', MASTER_PASSWORD = 'password', MASTER_LOG_FILE = 'mysql-bin-changelog.000031', MASTER_LOG_POS = 107; -- And one of these statements depending on your engine version: START SLAVE; -- Aurora MySQL version 2 START REPLICA; -- Aurora MySQL version 3`

Se la replica ha esito negativo, può verificarsi un notevole aumento di I/O non intenzionale sulla replica, che può compromettere le prestazioni. Se la replica non riesce o non è più necessaria, è possibile eseguire la stored procedure mysql.rds_reset_external_master (Aurora MySQL versione 2) o mysql.rds_reset_external_source (Aurora MySQL versione 3) per rimuovere la configurazione della replica.

Definire una posizione per arrestare la replica su una replica di lettura

In Aurora MySQL versione 3.04 e successive, puoi avviare una replica e quindi arrestarla in una posizione di file di log binario specifica mediante la stored procedure mysql.rds_start_replication_until (Aurora MySQL versione 3).

Per avviare la replica su una replica di lettura e arrestare la replica in corrispondenza di una posizione specifica

1. Utilizzando un client MySQL, stabilisci una connessione al cluster di database Aurora MySQL di replica come utente master.
2. Eseguire la procedura archiviata mysql.rds_start_replication_until (Aurora MySQL versione 3).

   L'esempio seguente avvia la replica e replica le modifiche fino a raggiungere la posizione `120` nel file di log binario `mysql-bin-changelog.000777`. In caso di disaster recovery, presumere che la posizione `120` si riferisca al momento immediatamente precedente l'errore.

   ```
   call mysql.rds_start_replication_until(
     'mysql-bin-changelog.000777',
     120);
   ```

La replica si arresta automaticamente quando viene raggiunto il punto di arresto. Viene generato il seguente evento RDS: `Replication has been stopped since the replica reached the stop point specified by the rds_start_replication_until stored procedure`.

Se si utilizza la replica basata su GTID, scegliere la stored procedure mysql.rds_start_replication_until_gtid (Aurora MySQL versione 3) invece della mysql.rds_start_replication_until (Aurora MySQL versione 3). Per ulteriori informazioni sulla replica basata su GTID, consultare Utilizzo della replica basata su GTID.

7. Monitora la replica

Quando imposti la replica MySQL con un cluster di database Aurora MySQL, devi monitorare gli eventi di failover per il cluster di database Aurora MySQL quando è il target di replica. Se accade un failover, il cluster di database che è il target di replica potrebbe essere ricreato in un nuovo host con un indirizzo di rete diverso. Per informazioni su come monitorare gli eventi di failover, consulta Utilizzo della notifica degli eventi di Amazon RDS.

È anche possibile monitorare quanto è indietro la destinazione della replica rispetto all'origine eseguendo la connessione alla destinazione della replica e il comando `SHOW SLAVE STATUS` (Aurora MySQL versione 2) o `SHOW REPLICA STATUS` (Aurora MySQL versione 3). Nell'output del comando, il campo `Seconds Behind Master` indica quanto è indietro il target di replica rispetto al master di replica.

> **Important**
>
> Se aggiorni il tuo cluster di database e specifichi un gruppo di parametri personalizzati, assicurati di riavviare manualmente il cluster al termine dell'aggiornamento. In questo modo il cluster utilizza le nuove impostazioni dei parametri personalizzati e riavvia la replica dei log binari.

Sincronizzazione delle password tra origine di replica e destinazione

Quando si modificano gli account utente e le password nell'origine di replica utilizzando le istruzioni SQL, tali modifiche vengono replicate automaticamente nella destinazione di replica.

Se utilizzi la Console di gestione AWS, la AWS CLIo l'API RDS per modificare la password principale nell'origine di replica, tali modifiche non vengono replicate automaticamente nella destinazione di replica. Se si desidera sincronizzare l'utente principale e la password principale tra il sistema di origine e quello di destinazione, è necessario apportare la stessa modifica alla destinazione di replica.

Arresto del processo di replica dei log binari per Aurora MySQL

Per fermare la replica dei log binari con un'istanza database MySQL, un database MySQL esterno o un altro cluster di database Aurora, segui questa procedura, che viene illustrata nel dettaglio nel seguente argomento.

1. Arrestare la replica dei log binari nella destinazione di replica

2. Disabilitare l'accesso binario alla fonte di replica

1. Arrestare la replica dei log binari nella destinazione di replica

Usa le seguenti istruzioni per arrestare la replica dei log binari del motore di database.

Motore del database	Istruzioni
Aurora MySQL	Per arrestare la replica dei log binari in una destinazione di replica del cluster di database Aurora MySQL Esegui la connessione al cluster di database Aurora che è la destinazione della replica e chiama la procedura mysql.rds_stop_replication.
RDS per MySQL	Per arrestare la replica binlog in un'istanza database Amazon RDS Esegui la connessione all'istanza database RDS che è la destinazione della replica e chiama la procedura mysql.rds_stop_replication.
MySQL (esterno)	Per fermare la replica dei log binari in un database MySQL esterno Connettiti al database MySQL ed esegui il comando`STOP SLAVE` (versione 5.7) o `STOP REPLICA` (versione 8.0).

2. Disabilitare l'accesso binario alla fonte di replica

Usa le seguenti istruzioni per disattivare la registrazione dei log binari nell'origine della replica per il motore del database.

Motore del database	Istruzioni
Aurora MySQL	Per disabilitare l'accesso binario in un cluster di database Amazon Aurora 1. Connettiti al cluster di database Aurora che è l'origine della replica. 2. Utilizza la procedura mysql.rds_set_configuration e specifica il parametro di configurazione `binlog retention hours`, con il valore `NULL`, come illustrato nell'esempio seguente.

Motore del database	Istruzioni
	``` CALL mysql.rds_set_configuration('binlog retention hours', NULL); ``` **Note** Non puoi utilizzare il valore `0` per `binlog retention hours`. 3. Imposta il parametro `binlog_format` su `OFF` nel master di replica. Il parametro `binlog_format` è nel gruppo di parametri di database del cluster di database personalizzato associato al cluster di database. Dopo aver modificato il valore del parametro `binlog_format` , riavvia il cluster di database per rendere effettiva la modifica. Per ulteriori informazioni, consultare Parametri dell'istanza database e del cluster database di Amazon Aurora e Modifica dei parametri in un gruppo di parametri database in Amazon Aurora.
RDS per MySQL	Per disabilitare l'accesso binario in un'istanza database Amazon RDS Non è possibile abilitare l'accesso binario direttamente per un'istanza database Amazon RDS, ma è possibile abilitarlo procedendo come di seguito: 1. Disabilitare i backup automatici per l'istanza database. Puoi disabilitare i backup automatici modificando un'istanza database esistente e impostando Backup Retention Period (Tempo di conservazione del backup) su 0. Per ulteriori informazioni, consulta Modifica di un'istanza database Amazon RDS e Utilizzo dei backup nella Guida per l'utente di Amazon Relational Database Service. 2. Elimina tutte le repliche di lettura per l'istanza database. Per ulteriori informazioni, consulta Funzionamento delle repliche di lettura con istanze database MariaDB, MySQL e PostgreSQL nella Guida per l'utente di Amazon Relational Database Service.

Motore del database	Istruzioni
MySQL (esterno)	Per disabilitare l'accesso binario a un database esterno MySQL Esegui la connessione al database MySQL e chiama il comando `STOP REPLICATION` . 1. Da una shell dei comandi, arrestare il servizio `mysqld`, ``` sudo service mysqld stop ``` 2. Modificare il file `my.cnf` (posto in genere sotto `/etc`). ``` sudo vi /etc/my.cnf ``` Elimina le opzioni `log_bin` e `server_id` dalla sezione `[mysqld]`. Per ulteriori informazioni, vedere Setting the replication source configuration nella documentazione di MySQL. 3. Avvia il servizio mysql. ``` sudo service mysqld start ```

Dimensionamento delle letture per il database MySQL con Amazon Aurora

Puoi utilizzare Amazon Aurora con l'istanza database MySQL per usufruire delle funzionalità di dimensionamento della lettura di Amazon Aurora ed espandere il reale carico di lavoro della tua istanza database MySQL. Per utilizzare Aurora per ridimensionare le operazioni di lettura dell'istanza database MySQL, crea un cluster di database Amazon Aurora MySQL e rendilo una replica di lettura per l'istanza database MySQL. Ciò si applica a un'istanza database RDS per MySQL o a un database MySQL in esecuzione esternamente a Amazon RDS.

Per informazioni su come creare un cluster di database Amazon Aurora, consulta Creazione di un cluster database Amazon Aurora.

Quando si configura la replica tra l'istanza database MySQL e il cluster di database Amazon Aurora, assicurarsi di seguire queste linee guida:

- Utilizzare l'indirizzo dell'endpoint del cluster di database di Amazon Aurora quando si fa riferimento al cluster di database Amazon Aurora MySQL. Se si verifica un failover, la replica di Aurora promossa a istanza principale per il cluster di database Aurora MySQL continua a utilizzare l'indirizzo dell'endpoint del cluster di database.
- Conservare i binlog sull'istanza di scrittura finché non si ha la conferma che siano stati applicati alla replica di Aurora. Questa manutenzione assicura il ripristino dell'istanza di lettura nel caso di errori.

Important

Quando si utilizza la replica auto-gestita, si ha la responsabilità di monitorare e risolvere eventuali problemi di replica. Per ulteriori informazioni, consulta Diagnosi e risoluzione del ritardo tra repliche di lettura.

Note

Le autorizzazioni richieste per avviare la replica su un cluster di database Aurora MySQL sono limitate e non disponibili per l'utente master Amazon RDS. Pertanto sarà necessario utilizzare le procedure mysql.rds_set_external_master (Aurora MySQL versione 2) o mysql.rds_set_external_source (Aurora MySQL versione 3) e mysql.rds_start_replication per impostare la replica tra il cluster di database Aurora MySQL e l'istanza database MySQL.

Avvio della replica tra un'istanza di origine esterna e un cluster di database Aurora MySQL

1. Rendere di sola lettura l'istanza database MySQL di origine:

```
mysql> FLUSH TABLES WITH READ LOCK;
mysql> SET GLOBAL read_only = ON;
```

2. Eseguire il comando `SHOW MASTER STATUS` nell'istanza database MySQL di origine per determinare la posizione di binlog. Viene restituito un output simile all'esempio seguente:

```
File                        Position
-----------------------------------
```

```
 mysql-bin-changelog.000031      107
-----------------------------------
```

3. Copia il database dall'istanza database MySQL esterna al cluster di database Amazon Aurora MySQL utilizzando `mysqldump`. Per database di dimensioni particolarmente elevate, è possibile utilizzare la procedura descritta in Importazione dei dati in un database Amazon RDS per MySQL con tempo di inattività ridotto nella Guida per l'utente di Amazon Relational Database Service.

   Per Linux, macOS o Unix:

```
mysqldump \
    --databases <database_name> \
    --single-transaction \
    --compress \
    --order-by-primary \
    -u local_user \
    -p local_password | mysql \
        --host aurora_cluster_endpoint_address \
        --port 3306 \
        -u RDS_user_name \
        -p RDS_password
```

   Per Windows:

```
mysqldump ^
    --databases <database_name> ^
    --single-transaction ^
    --compress ^
    --order-by-primary ^
    -u local_user ^
    -p local_password | mysql ^
        --host aurora_cluster_endpoint_address ^
        --port 3306 ^
        -u RDS_user_name ^
        -p RDS_password
```

> **Note**
>
> Assicurati che non siano presenti spazi tra l'opzione `-p` e la password immessa.

Utilizza le opzioni `--host`, `--user (-u)`, `--port` e `-p` nel comando `mysql` per specificare il nome host, il nome utente e la password per eseguire la connessione al cluster di database Aurora. Il nome host è il nome DNS per l'endpoint del cluster di database Amazon Aurora, ad esempio, `mydbcluster.cluster-123456789012.us-east-1.rds.amazonaws.com`. Il valore dell'endpoint è disponibile nei dettagli del cluster nella console di gestione Amazon RDS.

4. Rendere nuovamente scrivibile l'istanza database MySQL di origine:

```
mysql> SET GLOBAL read_only = OFF;
mysql> UNLOCK TABLES;
```

Per ulteriori informazioni sulla creazione di backup da utilizzare con la replica, vedere Backing up a source or replica by making it read only nella documentazione di MySQL.

5. Nella console di gestione Amazon RDS aggiungi l'indirizzo IP del server che ospita il database MySQL di origine al gruppo di sicurezza VPC per il cluster di database Amazon Aurora. Per ulteriori informazioni sulla modifica di un gruppo di sicurezza VPC, consulta Gruppi di sicurezza per il VPC nella Guida per l'utente di Amazon Virtual Private Cloud.

Potrebbe anche essere necessario configurare la rete locale per consentire le connessioni dall'indirizzo IP del cluster di database Amazon Aurora, affinché possa comunicare con l'istanza di MySQL di origine. Per individuare l'indirizzo IP del cluster di database Amazon Aurora, utilizzare il comando `host`.

```
host aurora_endpoint_address
```

Il nome host è il nome DNS dell'endpoint del cluster di database Amazon Aurora.

6. Utilizzando il client scelto, eseguire la connessione all'istanza di MySQL esterna e creare un utente MySQL da utilizzare per la replica. Questo account viene utilizzato unicamente per la replica e deve essere limitato al dominio personale per aumentare la sicurezza. Di seguito è riportato un esempio.

```
CREATE USER 'repl_user'@'example.com' IDENTIFIED BY 'password';
```

7. Per un'istanza di MySQL esterna, concedere i privilegi `REPLICATION CLIENT` e `REPLICATION SLAVE` all'utente della replica. Per concedere ad esempio i privilegi `REPLICATION CLIENT` e `REPLICATION SLAVE` su tutti i database per l'utente "`repl_user`" del proprio dominio, eseguire questo comando.

```
GRANT REPLICATION CLIENT, REPLICATION SLAVE ON *.* TO 'repl_user'@'example.com'
    IDENTIFIED BY 'password';
```

8. Acquisire uno snapshot manuale del cluster di database Aurora MySQL da impostare come replica di lettura prima di impostare la replica. Se è necessario ridefinire la replica con il cluster di database come replica di lettura, è possibile ripristinare il cluster di database Aurora MySQL da tale snapshot anziché dover importare i dati dall'istanza database MySQL in un nuovo cluster di database Aurora MySQL.
9. Configurare il cluster di database Amazon Aurora come replica. Esegui la connessione al cluster di database Amazon Aurora come utente master e identifica il database MySQL di origine come origine di replica utilizzando mysql.rds_set_external_master (Aurora MySQL versione 2) o le procedure mysql.rds_set_external_source (Aurora MySQL versione 3) e mysql.rds_start_replication.

   Utilizza il nome e la posizione del file binlog, recuperati nella fase 2. Di seguito è riportato un esempio di :

   ```
   For Aurora MySQL version 2:
   CALL mysql.rds_set_external_master ('mymasterserver.example.com', 3306,
       'repl_user', 'password', 'mysql-bin-changelog.000031', 107, 0);

   For Aurora MySQL version 3:
   CALL mysql.rds_set_external_source ('mymasterserver.example.com', 3306,
       'repl_user', 'password', 'mysql-bin-changelog.000031', 107, 0);
   ```

10. Nel cluster di database Amazon Aurora chiama la procedura mysql.rds_start_replication per avviare la replica.

    ```
    CALL mysql.rds_start_replication;
    ```

Dopo aver definito la replica tra l'istanza database MySQL di origine e il cluster di database Amazon Aurora sarà possibile aggiungere le repliche di Aurora al cluster di database Amazon Aurora. È possibile quindi connettere le repliche di Aurora per il dimensionamento della lettura dei dati. Per informazioni sulla creazione di una replica di Aurora, consulta Aggiunta di repliche di Aurora a un cluster di database.

Ottimizzazione della replica dei log binari per Aurora MySQL

Di seguito, è possibile imparare a ottimizzare le prestazioni di replica dei log binari e risolvere i problemi correlati in Aurora MySQL.

> **Tip**
>
> Questa discussione presuppone che tu abbia familiarità con il meccanismo di replica dei log binari MySQL e del suo funzionamento. Per informazioni di base, consulta Implementazione della replica nella documentazione di MySQL.

Replica dei log binari multithread

Con la replica del log binario multithread, un thread SQL legge gli eventi dal log di inoltro e li mette in coda per l'applicazione dei thread SQL worker. I thread di lavoro SQL sono gestiti dal thread coordinatore. Gli eventi di log binari vengono applicati in parallelo quando possibile. Il livello di parallelismo dipende da fattori quali versione, parametri, progettazione dello schema e caratteristiche del carico di lavoro.

La replica dei log binari multithread è supportata in Aurora MySQL versione 3 e Aurora MySQL 2.12.1 e versioni successive. Affinché una replica multithread elabori in modo efficiente gli eventi dei log binari in parallelo, è necessario configurare l'origine per la replica dei log binari multithread, la quale dovrà utilizzare una versione che includa le informazioni sul parallelismo nei file di log binari.

Quando un'istanza database Aurora MySQL è configurata per l'utilizzo della replica dei log binari, per impostazione predefinita l'istanza di replica utilizza la replica a thread singolo per le versioni precedenti alla 3.04 di Aurora MySQL. Per abilitare la replica multithread, si aggiorna il parametro `replica_parallel_workers` a un valore maggiore di 1 nel gruppo di parametri personalizzati.

Per Aurora MySQL 3.04 e versioni successive, la replica è multithread per impostazione predefinita, con `replica_parallel_workers` impostato su 4. È possibile modificare questo parametro nel gruppo di parametri personalizzato.

Per aumentare la resilienza del database in caso di arresti imprevisti, è consigliabile abilitare la replica GTID sull'origine e consentire i GTID sulla replica. Per consentire la replica GTID, si imposta `gtid_mode` su `ON_PERMISSIVE` sia sull'origine sia sulla replica. Per ulteriori informazioni sulla replica basata su GTID, consultare Utilizzo della replica basata su GTID.

Le opzioni di configurazione seguenti consentono di ottimizzare la replica multithread. Per informazioni sull'utilizzo, consulta Opzioni e variabili di replica e registrazione binaria nel Manuale di riferimento MySQL. Per ulteriori informazioni sulla replica multithread, consulta il blog MySQL Improving the Parallel Applier with Writeset-based Dependency Tracking.

I valori ottimali dei parametri dipendono da diversi fattori. Ad esempio, le prestazioni per la replica dei log binari sono influenzate dalle caratteristiche del carico di lavoro del database e dalla classe di istanza database su cui è in esecuzione la replica. Pertanto, si consiglia di testare con attenzione tutte le modifiche apportate a questi parametri di configurazione prima di applicare nuove impostazioni di parametro a un'istanza di produzione:

- `binlog_format recommended value` impostato su `ROW`
- `binlog_group_commit_sync_delay`
- `binlog_group_commit_sync_no_delay_count`
- `binlog_transaction_dependency_history_size`
- `binlog_transaction_dependency_tracking`, il valore consigliato è `WRITESET`
- `replica_preserve_commit_order`
- `replica_parallel_type`, il valore consigliato è `LOGICAL_CLOCK`
- `replica_parallel_workers`
- `replica_pending_jobs_size_max`
- `transaction_write_set_extraction`, il valore consigliato è `XXHASH64`

Le caratteristiche dello schema e del carico di lavoro sono fattori che influiscono sulla replica in parallelo. Di seguito sono riportati i fattori più comuni.

- Assenza di chiavi primarie: RDS non è in grado di stabilire la dipendenza writeset per le tabelle senza chiavi primarie. Con il formato `ROW`, è possibile eseguire una singola istruzione su più righe con un'unica scansione completa della tabella sull'origine, ma si ottiene una scansione completa della tabella per ogni riga modificata sulla replica. L'assenza di chiavi primarie riduce in modo significativo il throughput di replica.
- Presenza di chiavi esterne: se sono presenti chiavi esterne, Amazon RDS non può utilizzare la dipendenza writeset per il parallelismo delle tabelle con la relazione delle chiavi esterne.
- Dimensione delle transazioni: se una singola transazione si estende su dozzine o centinaia di megabyte o gigabyte, il thread coordinatore e uno dei thread di lavoro potrebbero impiegare molto

tempo solo per elaborare quella transazione. Durante questo periodo, tutti gli altri thread di lavoro potrebbero rimanere inattivi dopo aver completato l'elaborazione delle transazioni precedenti.

In Aurora MySQL 3.06 e versioni successive, è possibile migliorare le prestazioni delle repliche di log binari durante la replica delle transazioni per tabelle di grandi dimensioni con più di un indice secondario. Questa funzionalità introduce un pool di thread per applicare le modifiche dell'indice secondario in parallelo su una replica di log binari. La funzionalità è controllata dal parametro del cluster di database `aurora_binlog_replication_sec_index_parallel_workers`, che controlla il numero totale di thread paralleli disponibili per applicare le modifiche dell'indice secondario. Per impostazione predefinita, il parametro è impostato su `0` (disabilitato). L'abilitazione di questa funzionalità non richiede il riavvio dell'istanza. Per abilitare questa funzionalità, arrestare la replica in corso, impostare il numero desiderato di thread di lavoro paralleli e quindi riavviare la replica.

Ottimizzazione della replica dei log binari

In Aurora MySQL 2.10 e versioni successive, Aurora applica automaticamente un'ottimizzazione nota come cache I/O binlog alla replica del log binario. Inserendo nella cache gli eventi binlog più recenti, questa ottimizzazione è progettata per migliorare le prestazioni del thread di dump di binlog limitando al contempo l'impatto sulle transazioni in primo piano sull'istanza di origine binlog.

> **Note**
>
> Questa memoria utilizzata per questa funzione è indipendente dall'impostazione `binlog_cache` MySQL.
> Questa funzione non si applica alle istanze DB Aurora che utilizzano le classi di istanza `db.t2` e `db.t3`.

Non è necessario regolare alcun parametro di configurazione per attivare questa ottimizzazione. In particolare, se il parametro di configurazione `aurora_binlog_replication_max_yield_seconds` è stato impostato su un valore diverso da zero nelle versioni precedenti di Aurora MySQL, impostarlo su zero per le versioni attualmente disponibili.

Queste variabili di stato `aurora_binlog_io_cache_reads` e `aurora_binlog_io_cache_read_requests` consentono di monitorare la frequenza con cui i dati vengono letti dalla cache I/O dei log binari.

- `aurora_binlog_io_cache_read_requests`: mostra il numero di richieste di lettura I/O binlog dalla cache.
- `aurora_binlog_io_cache_reads`: mostra il numero di letture I/O binlog che recuperano informazioni dalla cache.

La seguente query SQL calcola la percentuale di richieste di lettura binlog che sfruttano le informazioni memorizzate nella cache. In questo caso, più il rapporto è vicino a 100, migliore è.

```
mysql> SELECT
  (SELECT VARIABLE_VALUE FROM INFORMATION_SCHEMA.GLOBAL_STATUS
    WHERE VARIABLE_NAME='aurora_binlog_io_cache_reads')
  / (SELECT VARIABLE_VALUE FROM INFORMATION_SCHEMA.GLOBAL_STATUS
    WHERE VARIABLE_NAME='aurora_binlog_io_cache_read_requests')
  * 100
  as binlog_io_cache_hit_ratio;
+---------------------------+
| binlog_io_cache_hit_ratio |
+---------------------------+
|         99.99847949080622 |
+---------------------------+
```

La funzione di cache I/O binlog include anche nuovi parametri relativi ai thread di dump di binlog. I thread di dump sono i thread creati quando le nuove repliche di binlog sono collegate all'istanza fonte binlog.

I parametri del thread di dump vengono stampati nel log del database ogni 60 secondi con il prefisso `[Dump thread metrics]`. I parametri includono informazioni per ogni replica binlog, ad esempio `Secondary_id`, `Secondary_uuid`, il nome del file binlog e la posizione in cui ogni replica sta leggendo. I parametri includono anche `Bytes_behind_primary` che rappresenta la distanza in byte tra l'origine della replica e la replica. Questo parametro misura il ritardo del thread I/O di replica. Tale cifra è diversa dal ritardo del thread dell'applicatore SQL della replica, rappresentato dal parametro `seconds_behind_master` sulla replica binlog. È possibile determinare se le repliche di binlog stiano recuperando l'origine o rimangano indietro controllando se la distanza diminuisce o aumenta.

Log di inoltro in memoria

In Aurora MySQL versione 3.10 e versioni successive, Aurora introduce un'ottimizzazione nota come log di inoltro in memoria per migliorare il throughput della replica. Questa ottimizzazione migliora

le prestazioni di I/O del log di inoltro memorizzando nella cache tutto il contenuto del log di inoltro intermedio presente in memoria. Di conseguenza, diminuisce la latenza di commit riducendo al minimo le operazioni di I/O di archiviazione, poiché il contenuto del log di inoltro rimane facilmente accessibile in memoria.

Per impostazione predefinita, la funzionalità di log di inoltro in memoria è abilitata automaticamente per gli scenari di replica gestiti da Aurora (incluse implementazioni blu/verdi, replica Aurora-Aurora e repliche tra Regioni) quando la replica soddisfa una di queste configurazioni:

- Modalità di replica a thread singolo (replica_parallel_workers = 0)
- Replica multithread con modalità GTID abilitata:
  - Posizionamento automatico abilitato
  - Modalità GTID impostata su ON per la replica
- Replica basata su file con replica_preserve_commit_order = ON

La funzionalità di log di inoltro in memoria è supportata su classi di istanza più grandi di t3.large, ma non è disponibile sulle istanze Aurora Serverless. Il buffer circolare del log di inoltro ha una dimensione fissa di 128 MB. Per monitorare il consumo di memoria di questa funzionalità, è possibile eseguire la seguente query:

```
SELECT event_name, current_alloc FROM sys.memory_global_by_current_bytes WHERE
 event_name = 'memory/sql/relaylog_io_cache';
```

La funzionalità di log di inoltro in memoria è controllata dal parametro aurora_in_memory_relaylog, che può essere impostato a livello di istanza o di cluster di database. È possibile abilitare o disabilitare questa funzionalità in modo dinamico senza riavviare l'istanza:

1. Arrestare la replica in corso
2. Impostare aurora_in_memory_relaylog su ON (per abilitare) o OFF (per disabilitare) nel gruppo di parametri
3. Riavviare la replica

Esempio:

```
CALL mysql.rds_stop_replication;
set aurora_in_memory_relaylog to ON to enable or OFF to disable in cluster parameter
 group
```

```
CALL mysql.rds_start_replication;
```

Anche quando aurora_in_memory_relaylog è impostato su ON, la funzionalità di log di inoltro in memoria potrebbe comunque essere disabilitata in determinate condizioni. Per verificare lo stato corrente della funzionalità, è possibile utilizzare il seguente comando:

```
SHOW GLOBAL STATUS LIKE 'Aurora_in_memory_relaylog_status';
```

Se la funzionalità viene disabilitata in modo imprevisto, è possibile identificarne il motivo eseguendo il seguente comando:

```
SHOW GLOBAL STATUS LIKE 'Aurora_in_memory_relaylog_disabled_reason';
```

Questo comando restituisce un messaggio che spiega perché la funzionalità è attualmente disabilitata.

Configurazione del file di log binario avanzato per Aurora MySQL

Il file di log binario avanzato riduce il sovraccarico delle prestazioni di elaborazione causato dall'attivazione del file di log binario, che in alcuni casi può arrivare fino al 50%. Con il file di log binario avanzato, questo sovraccarico può essere ridotto a circa il 13%. Per ridurre il sovraccarico, il file di log binario avanzato scrive i log binari e i log delle transazioni nello spazio di archiviazione in parallelo, il che riduce al minimo i dati scritti al momento del commit della transazione.

L'utilizzo del file di log binario avanzato migliora anche i tempi di ripristino del database dopo riavvii e failover fino al 99% rispetto al file di log binario della community MySQL. Il file di log binario avanzato è compatibile con i carichi di lavoro esistenti basati sul file di log binario e viene utilizzato nello stesso modo in cui si utilizza il file di log binario della community MySQL.

Il file di log binario avanzato è disponibile in Aurora MySQL versione 3.03.1 e successive.

Argomenti

- Configurazione dei parametri del file di log binario avanzato
- Altri parametri correlati
- Differenze tra file di log binario avanzato e file di log binario avanzato della community MySQL
- Metriche Amazon CloudWatch per il file di log binario avanzato
- Limitazioni del file di log binario avanzato

Configurazione dei parametri del file di log binario avanzato

È possibile passare dal file di log binario della community MySQL al file di log binario avanzato attivando/disattivando i relativi parametri. Gli utenti esistenti di file di log binario possono continuare a leggere e consumare i file di log binario senza interruzioni nella sequenza di file di log binario.

Per attivare il file di log binario avanzato, imposta i seguenti parametri:

Parametro	Predefinito	Descrizione
`binlog_format`	–	Impostare il parametro `binlog_format` sul formato di registrazione binaria desiderato per attivare il file di log binario avanzato. Assicurarsi che `binlog_format parameter` non sia impostato su OFF. Per ulteriori informazioni, consultare Configurazione del log binario di Aurora MySQL.
`aurora_enhanced_binlog`	`0`	Impostare il valore di questo parametro su 1 nel gruppo di parametri del cluster database associato al cluster Aurora MySQL. Quando si modifica il valore di questo parametro, è necessario riavviare l'istanza di scrittura quando il valore di `DBClusterParameterGroupStatus` viene visualizzato come `pending-reboot` .
`binlog_backup`	`1`	Disattivare questo parametro per attivare il file di log binario avanzato. A tale scopo,

Parametro	Predefinito	Descrizione
		impostare il valore di questo parametro su `0`.
`binlog_replication_globaldb`	`1`	Disattivare questo parametro per attivare il file di log binario avanzato. A tale scopo, impostare il valore di questo parametro su `0`.

> **Important**
>
> È possibile disattivare i parametri `binlog_backup` e `binlog_replication_globaldb` solo in caso di utilizzo del file di log binario avanzato.

Per disattivare il file di log binario avanzato, imposta i seguenti parametri:

Parametro	Descrizione
`aurora_enhanced_binlog`	Impostare il valore di questo parametro su `0` nel gruppo di parametri del cluster database associato al cluster Aurora MySQL. Ogni volta che si modifica il valore di questo parametro, è necessario riavviare l'istanza di scrittura quando il valore di `DBClusterParameterGroupStatus` viene visualizzato come `pending-reboot`.
`binlog_backup`	Attivare questo parametro quando si disattiva il file di log binario avanzato. A tale scopo, impostare il valore di questo parametro su 1.
`binlog_replication_globaldb`	Attivare questo parametro quando si disattiva il file di log binario avanzato. A tale scopo, impostare il valore di questo parametro su 1.

Per verificare se il file di log binario avanzato è attivo, usare il seguente comando nel client MySQL:

```
mysql>show status like 'aurora_enhanced_binlog';

+------------------------+--------+
| Variable_name          | Value  |
+------------------------+--------+
| aurora_enhanced_binlog | ACTIVE |
+------------------------+--------+
1 row in set (0.00 sec)
```

Quando il file di log binario avanzato è attivo, l'output mostra `ACTIVE` per `aurora_enhanced_binlog`.

Altri parametri correlati

Quando si attiva il file di log binario avanzato, questa operazione interessa i seguenti parametri:

- Il parametro `max_binlog_size` è visibile ma non modificabile. Il relativo valore predefinito `134217728` viene impostato automaticamente su `268435456` quando il file di log binario avanzato è attivato.
- A differenza file di log binario avanzato della community MySQL, `binlog_checksum` non agisce come parametro dinamico quando il file di log binario avanzato è attivato. Affinché la modifica a questo parametro abbia effetto, è necessario riavviare manualmente il cluster database anche quando `ApplyMethod` è `immediate`.
- Il valore impostato per il parametro `binlog_order_commits` non ha alcun effetto sull'ordine dei commit quando il file di log binario avanzato è attivato. I commit vengono sempre ordinati senza ulteriori implicazioni in termini di prestazioni.

Differenze tra file di log binario avanzato e file di log binario avanzato della community MySQL

Il file di log binario avanzato interagisce in modo diverso con i cloni, i backup e il database globale Aurora rispetto al file di log binario avanzato di Community MySQL. Si consiglia di analizzare le seguenti differenze prima di utilizzare il file di log binario avanzato.

- I file di log binario avanzati del cluster di database di origine non sono disponibili su un cluster di database clonato.

- I file di log binario avanzati non sono inclusi nei backup di Aurora. Pertanto, i file di log binario avanzati del cluster database di origine non sono disponibili dopo il ripristino di un cluster database nonostante il relativo periodo di conservazione impostato.
- Se utilizzati con un database globale Aurora, i file di log binario avanzati del cluster database primario non vengono replicati nel cluster database nelle regioni secondarie.

Examples (Esempi)

Negli esempi seguenti vengono illustrate le differenze tra file di log binario avanzati e file di log binario della community MySQL.

Su un cluster database ripristinato o clonato

Quando il file di log binario avanzato è attivato, i file di log binario storici non sono disponibili nel cluster database ripristinato o clonato. Dopo un'operazione di ripristino o clonazione, se il file di log binario è attivato, il nuovo cluster database inizia a scrivere la propria sequenza di file di log binario a partire da 1 (mysql-bin-changelog.000001).

Per attivare il file di log binario avanzato dopo un'operazione di ripristino o clonazione, impostare i parametri del cluster database richiesti sul cluster database ripristinato o clonato. Per ulteriori informazioni, consulta Configurazione dei parametri del file di log binario avanzato.

Example Esempio: operazione di clonazione o ripristino eseguita con il file di log binario avanzato attivato

Cluster database di origine:

```
mysql> show binary logs;

+----------------------------+-----------+-----------+
| Log_name                   | File_size | Encrypted |
+----------------------------+-----------+-----------+
| mysql-bin-changelog.000001 |       156 | No        |
| mysql-bin-changelog.000002 |       156 | No        |
| mysql-bin-changelog.000003 |       156 | No        |
| mysql-bin-changelog.000004 |       156 | No        | --> Enhanced Binlog turned on
| mysql-bin-changelog.000005 |       156 | No        | --> Enhanced Binlog turned on
| mysql-bin-changelog.000006 |       156 | No        | --> Enhanced Binlog turned on
+----------------------------+-----------+-----------+
6 rows in set (0.00 sec)
```

In un cluster database ripristinato o clonato, non viene eseguito il backup dei file di log binario quando è attivato il file di log binario avanzato. Per evitare discontinuità nei dati del file di log binario, non sono disponibili nemmeno i file di log binario scritti prima di attivare il file di log binario avanzato.

```
mysql>show binary logs;

+----------------------------+-----------+-----------+
| Log_name                   | File_size | Encrypted |
+----------------------------+-----------+-----------+
| mysql-bin-changelog.000001 |       156 | No        | --> New sequence of Binlog files
+----------------------------+-----------+-----------+
1 row in set (0.00 sec)
```

Example Esempio: operazione di clonazione o ripristino eseguita con il file di log binario avanzato disattivato

Cluster di database di origine:

```
mysql>show binary logs;

+----------------------------+-----------+-----------+
| Log_name                   | File_size | Encrypted |
+----------------------------+-----------+-----------+
| mysql-bin-changelog.000001 |       156 | No        |
| mysql-bin-changelog.000002 |       156 | No        | --> Enhanced Binlog enabled
| mysql-bin-changelog.000003 |       156 | No        | --> Enhanced Binlog enabled
| mysql-bin-changelog.000004 |       156 | No        |
| mysql-bin-changelog.000005 |       156 | No        |
| mysql-bin-changelog.000006 |       156 | No        |
+----------------------------+-----------+-----------+
6 rows in set (0.00 sec)
```

Il file di log binario avanzato viene disabilitato dopo `mysql-bin-changelog.000003`. In un cluster database ripristinato o clonato, sono disponibili file di log binario scritti dopo aver disattivato il file di log binario avanzato.

```
mysql>show binary logs;

+----------------------------+-----------+-----------+
| Log_name                   | File_size | Encrypted |
+----------------------------+-----------+-----------+
| mysql-bin-changelog.000004 |       156 | No        |
| mysql-bin-changelog.000005 |       156 | No        |
| mysql-bin-changelog.000006 |       156 | No        |
+----------------------------+-----------+-----------+
1 row in set (0.00 sec)
```

Su un database globale Amazon Aurora

Su un database globale Amazon Aurora, i dati del file di log binario del cluster database primario non vengono replicati nei cluster database secondari. Dopo un processo di failover tra regioni, i dati del file di log binario non sono disponibili nel cluster database primario appena promosso. Se il file di log binario è attivato, il cluster database appena promosso avvia la propria sequenza di file di log binario, a partire da 1 (mysql-bin-changelog.000001).

Per attivare il file di log binario avanzato dopo il failover, è necessario impostare i parametri del cluster database richiesti sul cluster database secondario. Per ulteriori informazioni, consulta Configurazione dei parametri del file di log binario avanzato.

Example Esempio: operazione di failover globale del database eseguita con il file di log binario avanzato attivato

Vecchio cluster database primario (prima del failover):

```
mysql>show binary logs;

+----------------------------+-----------+-----------+
| Log_name                   | File_size | Encrypted |
+----------------------------+-----------+-----------+
| mysql-bin-changelog.000001 |       156 | No        |
| mysql-bin-changelog.000002 |       156 | No        |
| mysql-bin-changelog.000003 |       156 | No        |
| mysql-bin-changelog.000004 |       156 | No        | --> Enhanced Binlog enabled
| mysql-bin-changelog.000005 |       156 | No        | --> Enhanced Binlog enabled
| mysql-bin-changelog.000006 |       156 | No        | --> Enhanced Binlog enabled
```

```
+---------------------------+-----------+-----------+
6 rows in set (0.00 sec)
```

Nuovo cluster database primario (dopo il failover):

I file di log binario non vengono replicati nelle regioni secondarie quando il file di log binario avanzato è attivato. Per evitare discontinuità nei dati del file di log binario, i file di log binario scritti prima di attivare il file di log binario avanzato non sono disponibili.

```
mysql>show binary logs;

+---------------------------+-----------+-----------+
| Log_name                  | File_size | Encrypted |
+---------------------------+-----------+-----------+
| mysql-bin-changelog.000001 |       156 | No        | --> Fresh sequence of Binlog
 files
+---------------------------+-----------+-----------+
1 row in set (0.00 sec)
```

Example Esempio: operazione di failover globale del database eseguita con il file di log binario avanzato disattivato

Cluster database di origine:

```
mysql>show binary logs;

+---------------------------+-----------+-----------+
| Log_name                  | File_size | Encrypted |
+---------------------------+-----------+-----------+
| mysql-bin-changelog.000001 |       156 | No        |
| mysql-bin-changelog.000002 |       156 | No        | --> Enhanced Binlog enabled
| mysql-bin-changelog.000003 |       156 | No        | --> Enhanced Binlog enabled
| mysql-bin-changelog.000004 |       156 | No        |
| mysql-bin-changelog.000005 |       156 | No        |
| mysql-bin-changelog.000006 |       156 | No        |
+---------------------------+-----------+-----------+
6 rows in set (0.00 sec)
```

Cluster database ripristinato o clonato:

Il file di log binario avanzato viene disabilitato dopo `mysql-bin-changelog.000003`. I file di log binario scritti dopo aver disattivato il file di log binario avanzato vengono replicati e sono disponibili nel cluster database appena promosso.

```
mysql>show binary logs;

+----------------------------+-----------+-----------+
| Log_name                   | File_size | Encrypted |
+----------------------------+-----------+-----------+
| mysql-bin-changelog.000004 |       156 | No        |
| mysql-bin-changelog.000005 |       156 | No        |
| mysql-bin-changelog.000006 |       156 | No        |
+----------------------------+-----------+-----------+
3 rows in set (0.00 sec)
```

Metriche Amazon CloudWatch per il file di log binario avanzato

Le seguenti metriche di Amazon CloudWatch vengono pubblicate solo quando il file di log binario avanzato è attivato.

Parametro CloudWatch	Descrizione	unità
ChangeLogBytesUsed	Quantità di spazio di archiviazione in byte utilizzato dal file di log binario avanzato.	Byte
ChangeLogReadIOPs	Numero di operazioni I/O di lettura eseguite nel file di log binario avanzato in intervalli di 5 minuti.	Conteggio per 5 minuti
ChangeLogWriteIOPs	Numero di operazioni I/O di scrittura su disco eseguite nel file di log binario avanzato in intervalli di 5 minuti.	Conteggio per 5 minuti

Limitazioni del file di log binario avanzato

Le seguenti limitazioni si applicano ai cluster database Amazon Aurora quando il file di log binario avanzato è attivato.

- Il file di log binario avanzato è supportato solo in Aurora MySQL versione 3.03.1 e successive.
- I file di log binario avanzati scritti sul cluster database primario non vengono copiati nei cluster database clonati o ripristinati.
- Se utilizzati con un database globale Amazon Aurora, i file di log binario avanzati del cluster database primario non vengono replicati nei cluster database secondari. Pertanto, dopo il processo di failover, i dati dei file di log binario storici non sono disponibili nel nuovo cluster database primario.
- I seguenti parametri di configurazione del file di log binario vengono ignorati:
  - `binlog_group_commit_sync_delay`
  - `binlog_group_commit_sync_no_delay_count`
  - `binlog_max_flush_queue_time`
- Non è possibile eliminare o rinominare una tabella danneggiata in un database. Per eliminare queste tabelle, è possibile contattare Supporto.
- La cache I/O del file di log binario è disabilitata quando il file di log binario avanzato è attivato. Per ulteriori informazioni, consulta Ottimizzazione della replica dei log binari per Aurora MySQL.

> **Note**
>
> Il file di log binario avanzato è caratterizzato da miglioramenti delle prestazioni di lettura simili a quelli della cache I/O del file di log binario e da ulteriori ottimizzazioni delle prestazioni di scrittura.

- La funzionalità Backtrack non è supportata. Il file di log binario avanzato non può essere attivato in un cluster database nelle seguenti condizioni:
  - Cluster database con la funzionalità Backtrack abilitata.
  - Cluster di database con la funzionalità Backtrack precedentemente abilitata, ma non disattivata.
  - Cluster database ripristinato da un cluster database di origine o da uno snapshot con la funzionalità Backtrack abilitata.

Utilizzo della replica basata su GTID

Di seguito sono riportate le indicazioni per utilizzare gli identificatori di transazione globali (GTID) con la replica basata sui log binari (binlog) tra un cluster Aurora MySQL e un'origine esterna.

> **Note**
>
> Per Aurora, puoi utilizzare questa caratteristica solo con i cluster Aurora MySQL che utilizzano la replica basata sui registri binari verso o da un database MySQL esterno. L'altro database può essere un'istanza Amazon RDS MySQL, un database MySQL On-Premise o un cluster di database Aurora in una Regione AWS diversa. Per sapere come configurare questo tipo di replica, consultare Replica tra Aurora e MySQL o tra Aurora e un altro cluster di database Aurora (replica dei log binari).

Se si utilizza la replica basata su binlog e non si ha familiarità con la replica basata su GTID con MySQL, consulta Replication with global transaction identifiers nella documentazione MySQL.

La replica basata su GTID è supportata per Aurora MySQL versione 2 e 3.

Argomenti

- Identificatori globali di transazione (GTID)
- Parametri per la replica basata su GTID
- Abilitazione della replica basata su GTID per un cluster Aurora MySQL
- Disabilitazione della replica basata su GTID per un cluster DB Aurora MySQL

Identificatori globali di transazione (GTID)

Gli identificatori globali di transazione (GTID) sono identificatori univoci generati per le transazioni MySQL sottoposte a commit. Puoi utilizzare i GTID per semplificare la replica basata sui log binari e facilitare la risoluzione dei problemi.

> **Note**
>
> Quando Aurora sincronizza i dati tra le istanze database in un cluster, tale meccanismo di replica non coinvolge i log binari (binlog). Per Aurora MySQL, la replica basata su GTID si

applica solo quando si utilizza anche la replica basata sui log binari per replicare un database esterno compatibile con MySQL verso o da un cluster di database di Aurora MySQL.

MySQL utilizza due diversi tipi di transazioni per la replica basata sui log binari:

- Transazioni GTID – Transazioni identificate da un GTID.
- Transazioni anonime – Transazioni a cui non è assegnato un GTID.

In una configurazione di replica, i GTID sono univoci in tutte le istanze database. I GTID semplificano la configurazione della replica perché, quando vengono utilizzati, non è necessario fare riferimento alle posizioni nel file di log. I GTID semplificano anche la registrazione delle transazioni replicate e verificano che l'istanza di origine e le repliche siano coerenti.

In genere si utilizza la replica basata su GTID con Aurora durante la replica da un database esterno compatibile con MySQL verso un cluster Aurora. Puoi configurare questa configurazione di replica come parte di una migrazione da un database locale o Amazon RDS verso Aurora MySQL. Se il database esterno utilizza già i GTID, l'abilitazione della replica basata su GTID per il cluster Aurora semplifica il processo di replica.

Configuri la replica basata su GTID per un cluster Aurora MySQL impostando innanzitutto i parametri di configurazione rilevanti in un gruppo di parametri del cluster di database. Quindi associ tale gruppo di parametri al cluster.

Parametri per la replica basata su GTID

Utilizzare i parametri seguenti per configurare la replica basata su GTID.

Parametro	Valori validi	Descrizione
`gtid_mode`	`OFF`, `OFF_PERMISSIVE` , `ON_PERMISSIVE` , `ON`	`OFF` indica che le nuove transazioni sono anonime, ovvero non hanno GTID, e che una transazione deve essere anonima per poter essere replicata. `OFF_PERMISSIVE` indica che le nuove transazioni sono anonime, ma tutte le transazio ni possono essere replicate.

Parametro	Valori validi	Descrizione
		`ON_PERMISSIVE` indica che le nuove transazioni hanno GTID assegnati, ma tutte le transazioni possono essere replicate. `ON` indica che le nuove transazioni hanno GTID assegnati e che una transazione deve avere un GTID per poter essere replicata.
`enforce_gtid_consistency`	`OFF`, `ON`, `WARN`	`OFF` consente alle transazioni di violare la coerenza GTID. `ON` impedisce alle transazioni di violare la coerenza GTID. `WARN` consente alle transazioni di violare la consistenza GTID, ma genera un avviso quando si verifica una violazione.

> **Note**
>
> Nella Console di gestione AWS, il parametro `gtid_mode` viene visualizzato come `gtid-mode`.

Per la replica basata su GTID, utilizza queste impostazioni per il gruppo di parametri del cluster di database per il cluster di database Aurora MySQL:

- `ON` e `ON_PERMISSIVE` si applicano solo alla replica in uscita da un cluster Aurora MySQL. Entrambi questi valori fanno sì che il cluster di database Aurora utilizzi i GTID per le transazioni replicate su un database esterno. `ON` richiede che anche il database esterno utilizzi la replica basata su GTID. Per `ON_PERMISSIVE` la replica basata su GTID è opzionale sul database esterno.
- `OFF_PERMISSIVE`, se impostato, significa che il cluster di database Aurora può accettare la replica in ingresso da un database esterno. Possono farlo indipendentemente dal fatto che il database esterno utilizzi la replica basata su GTID o meno.

- `OFF`, se impostato, significa che il cluster di database Aurora accetta solo la replica in ingresso da database esterni che non utilizzano la replica basata su GTID.

> **Tip**
>
> La replica in ingresso è lo scenario di replica basata sui log binari più comune per i cluster Aurora MySQL. Per la replica in ingresso, ti consigliamo di impostare la modalità GTID su `OFF_PERMISSIVE`. Questa impostazione consente la replica in ingresso da database esterni indipendentemente dalle impostazioni GTID dell'origine della replica.

Per ulteriori informazioni sui gruppi di parametri, consultare Gruppi di parametri per Amazon Aurora.

Abilitazione della replica basata su GTID per un cluster Aurora MySQL

Quando la replica basata su GTID è abilitata per un cluster di database Aurora MySQL, le impostazioni GTID si applicano sia alla replica basata sui log binari in ingresso che a quella in uscita.

Per abilitare la replica basata su GTID per un cluster Aurora MySQL

1. Creare o modificare un gruppo di parametri del cluster di database utilizzando le seguenti impostazioni dei parametri:
   - `gtid_mode` – `ON` o `ON_PERMISSIVE`
   - `enforce_gtid_consistency` – `ON`
2. Associare il gruppo di parametri del cluster di database al cluster Aurora MySQL. A tale scopo, seguire la procedura descritta in Gruppi di parametri per Amazon Aurora.
3. (Facoltativo) Specifica come assegnare i GTID alle transazioni che non li includono. Per eseguire questa operazione, chiama la procedura archiviata in mysql.rds_assign_gtids_to_anonymous_transactions (Aurora MySQL versione 3).

Disabilitazione della replica basata su GTID per un cluster DB Aurora MySQL

Puoi disabilitare la replica basata su GTID per un cluster di database Aurora MySQL. Ciò significa che il cluster Aurora non può eseguire la replica basata sui log binari in ingresso o in uscita con database esterni che utilizzano la replica basata su GTID. .

> **Note**
>
> Nella seguente procedura, replica di lettura indica la destinazione della replica in una configurazione Aurora con replica basata sui log binari verso o da un database esterno. Non indica istanze database di replica a sola lettura di Aurora. Ad esempio, quando un cluster Aurora accetta la replica in ingresso da un'origine esterna, l'istanza primaria Aurora funge da replica di lettura per la replica basata sui log binari.

Per ulteriori dettagli sulle procedure memorizzate citate in questa sezione, consultare Informazioni di riferimento sulle stored procedure Aurora MySQL.

Per disabilitare la replica basata su GTID per un cluster database Aurora MySQL

1. Nelle repliche di Aurora, eseguire questa procedura:

   Per la versione 3

   ```
   CALL mysql.rds_set_source_auto_position(0);
   ```

   Per la versione 2

   ```
   CALL mysql.rds_set_master_auto_position(0);
   ```

2. Reimpostare `gtid_mode` su `ON_PERMISSIVE`.

   a. Verificare che il gruppo di parametri del cluster di database associato al cluster di database Aurora MySQL abbia il parametro `gtid_mode` impostato su `ON_PERMISSIVE`.

      Per ulteriori informazioni sull'impostazione dei parametri di configurazione mediante i gruppi di parametri, consultare Gruppi di parametri per Amazon Aurora.

   b. Riavvio del cluster di database Aurora MySQL

3. Reimpostare `gtid_mode` su `OFF_PERMISSIVE`.

   a. Verificare che il gruppo di parametri del cluster di database associato al cluster di database Aurora MySQL abbia il parametro `gtid_mode` impostato su `OFF_PERMISSIVE`.

   b. Riavvio del cluster di database Aurora MySQL

4. Attendere che tutte le transazioni GTID vengano applicate sull'istanza primaria di Aurora. Per verificare che le transazioni siano applicate, eseguire queste operazioni:

   a. Sull'istanza database primaria di Aurora, esegui il comando `SHOW MASTER STATUS`.

      L'output visualizzato dovrebbe essere simile al seguente.

      ```
      File                       Position
      -----------------------------------
      mysql-bin-changelog.000031      107
      -----------------------------------
      ```

      Annotare il file e la posizione nell'output.

   b. In ogni replica di lettura, utilizzare le informazioni su file e posizione presenti nell'istanza di origine menzionata nella fase precedente per eseguire la query seguente.

      Per la versione 3

      ```
      SELECT SOURCE_POS_WAIT('file', position);
      ```

      Per la versione 2

      ```
      SELECT MASTER_POS_WAIT('file', position);
      ```

      Ad esempio, se il nome del file è `mysql-bin-changelog.000031` e la posizione è `107`, eseguire questa istruzione.

      Per la versione 3

      ```
      SELECT SOURCE_POS_WAIT('mysql-bin-changelog.000031', 107);
      ```

      Per la versione 2

      ```
      SELECT MASTER_POS_WAIT('mysql-bin-changelog.000031', 107);
      ```

5. Reimpostare i parametri GTID per disabilitare la replica basata su GTID.

a. Verificare che il gruppo di parametri del cluster di database associato al cluster Aurora MySQL abbia le seguenti impostazioni dei parametri:

   - `gtid_mode` – `OFF`
   - `enforce_gtid_consistency` – `OFF`

b. Riavvio del cluster di database Aurora MySQL

Utilizzo dell'inoltro di scrittura locale in un cluster database Amazon Aurora MySQL

L'inoltro di scrittura locale (all'interno del cluster) consente alle applicazioni di eseguire transazioni di lettura/scrittura direttamente su una replica di Aurora. Queste transazioni vengono quindi inoltrate all'istanza database di scrittura in attesa che venga eseguito il commit. Puoi utilizzare l'inoltro di scrittura locale quando le applicazioni richiedono la coerenza lettura dopo scrittura, che è la capacità di leggere l'ultima scrittura in una transazione.

Le repliche di lettura ricevono aggiornamenti in modo asincrono dall'istanza di scrittura. Senza l'inoltro di scrittura, è necessario eseguire la transazione delle eventuali letture che richiedono la coerenza lettura dopo scrittura sull'istanza database di scrittura. Oppure è necessario sviluppare una logica dell'applicazione personalizzata complessa per sfruttare più repliche di lettura per la scalabilità. Le applicazioni devono suddividere completamente tutto il traffico in lettura e in scrittura, mantenendo due gruppi di connessioni al database per inviare il traffico all'endpoint corretto. Questi costi di sviluppo complicano la progettazione dell'applicazione quando le query fanno parte di una singola sessione logica, o transazione, all'interno dell'applicazione. Inoltre, poiché il ritardo di replica può variare tra le repliche di lettura, è difficile ottenere una coerenza di lettura globale tra tutte le istanze nel database.

L'inoltro di scrittura evita la necessità di suddividere tali transazioni o inviarle esclusivamente all'istanza di scrittura, semplificando lo sviluppo dell'applicazione. Questa nuova funzionalità consente di ottenere facilmente il dimensionamento della lettura per carichi di lavoro che devono leggere l'ultima scrittura in una transazione e non sono sensibili alla latenza di scrittura.

L'inoltro di scrittura locale è diverso dall'inoltro di scrittura globale, che inoltra le scritture da un cluster database secondario al cluster database primario in un database globale Aurora. Puoi utilizzare l'inoltro di scrittura locale in un cluster database che fa parte di un database globale Aurora. Per ulteriori informazioni, consulta Utilizzo dell'inoltro di scrittura in un database globale Amazon Aurora.

L'inoltro di scrittura locale richiede Aurora MySQL versione 3.04 o successive.

Argomenti

- Abilitazione dell'inoltro di scrittura locale
- Verifica se l'inoltro di scrittura è abilitato un cluster database
- Compatibilità delle applicazioni e di SQL con l'inoltro di scrittura
- Livelli di isolamento per l'inoltro di scrittura

- Coerenza di lettura per l'inoltro di scrittura
- Esecuzione di istruzioni a più parti con inoltro scrittura
- Transazioni con inoltro di scrittura
- Parametri di configurazione per l'inoltro di scrittura
- Parametri di Amazon CloudWatch e variabili di stato di Aurora MySQL per l'inoltro di scrittura
- Identificazione delle transazioni e delle query inoltrate

Abilitazione dell'inoltro di scrittura locale

Per impostazione predefinita, l'inoltro di scrittura locale non è abilitato per i cluster database Aurora MySQL. Abilitare l'inoltro di scrittura locale a livello di cluster, non a livello di istanza.

> **Important**
>
> Inoltre, è possibile abilitare l'inoltro di scrittura locale per le repliche di lettura tra regioni che utilizzano la registrazione binaria, ma le operazioni di scrittura non vengono inoltrate all'origine Regione AWS. Vengono inoltrate all'istanza database di scrittura del cluster di replica di lettura binlog.
> Si consiglia di usare questo metodo solo se si dispone di un caso d'uso per scrivere nella replica di lettura binlog nella Regione AWS secondaria. In caso contrario, si potrebbe verificare uno scenario "split-brain" in cui i set di dati replicati non sono tra loro coerenti.
> Si consiglia di utilizzare l'inoltro di scrittura globale con database globali, anziché l'inoltro di scrittura locale su repliche di lettura tra regioni, a meno che non sia assolutamente necessario. Per ulteriori informazioni, consulta Utilizzo dell'inoltro di scrittura in un database globale Amazon Aurora.

Console

Usando la Console di gestione AWS, seleziona la casella di controllo Attiva l'inoltro di scrittura locale in Inoltro di scrittura di repliche di lettura quando si crea o si modifica un cluster database.

AWS CLI

Per abilitare l'inoltro di scrittura con la AWS CLI, usa l'opzione `--enable-local-write-forwarding`. Questa opzione funziona quando si crea un nuovo cluster database secondario tramite il comando `create-db-cluster`. Inoltre, funziona quando si modifica un cluster database esistente

tramite il comando `modify-db-cluster`. Puoi disattivare l'inoltro di scrittura mediante l'opzione `--no-enable-local-write-forwarding` con questi stessi comandi CLI.

Nell'esempio seguente viene creato un cluster DB Aurora MySQL con l'inoltro di scrittura abilitato.

```
aws rds create-db-cluster \
  --db-cluster-identifier write-forwarding-test-cluster \
  --enable-local-write-forwarding \
  --engine aurora-mysql \
  --engine-version 8.0.mysql_aurora.3.04.0 \
  --master-username myuser \
  --master-user-password mypassword \
  --backup-retention 1
```

Vengono quindi create istanze database di scrittura e lettura in modo da poter utilizzare l'inoltro di scrittura. Per ulteriori informazioni, consulta Creazione di un cluster database Amazon Aurora.

API RDS

Per abilitare l'inoltro di scrittura utilizzando l'API Amazon RDS, impostare il parametro `EnableLocalWriteForwarding` su `true`. Questo parametro funziona quando si crea un nuovo cluster database secondario utilizzando l'operazione `CreateDBCluster`. Funziona anche quando si modifica un cluster database esistente utilizzando l'operazione `ModifyDBCluster`. È possibile disattivare l'inoltro di scrittura impostando il parametro `EnableLocalWriteForwarding` su `false`.

Abilitazione dell'inoltro di scrittura per le sessioni di database

Il parametro `aurora_replica_read_consistency` è un parametro database e un parametro del cluster database che abilita l'inoltro di scrittura. È possibile specificare `EVENTUAL`, `SESSION` o `GLOBAL` per il livello di coerenza di lettura. Per ulteriori informazioni sui livelli di coerenza, consulta Coerenza di lettura per l'inoltro di scrittura.

A questo parametro si applicano le seguenti regole:

- Il valore predefinito è '' (null).
- L'inoltro di scrittura è disponibile solo se `aurora_replica_read_consistency` è impostato su `EVENTUAL`, `SESSION` o `GLOBAL`. Questo parametro è rilevante solo nelle istanze di lettura di cluster database secondari con l'inoltro di scrittura abilitato.
- Non è possibile importare questo parametro (quando vuoto) o annullarne l'impostazione (quando già impostato) all'interno di una transazione con più istruzioni. È possibile modificarlo da un valore

valido a un altro valore valido durante una transazione di questo tipo, ma questa azione non è consigliata.

Verifica se l'inoltro di scrittura è abilitato un cluster database

Per determinare se è possibile utilizzare l'inoltro di scrittura in un cluster database, verifica che l'attributo `LocalWriteForwardingStatus` del cluster sia impostato su `enabled`.

In Console di gestione AWS, nella scheda Configurazione della pagina dei dettagli del cluster, viene visualizzato lo stato Abilitato per Inoltro di scrittura locale di replica di lettura.

Per visualizzare lo stato dell'impostazione di inoltro di scrittura per tutti i cluster, esegui il comando AWS CLI seguente.

Example

```
aws rds describe-db-clusters \
--query '*[].
{DBClusterIdentifier:DBClusterIdentifier,LocalWriteForwardingStatus:LocalWriteForwardingStatus}

[
    {
        "LocalWriteForwardingStatus": "enabled",
        "DBClusterIdentifier": "write-forwarding-test-cluster-1"
    },
    {
        "LocalWriteForwardingStatus": "disabled",
        "DBClusterIdentifier": "write-forwarding-test-cluster-2"
    },
    {
        "LocalWriteForwardingStatus": "requested",
        "DBClusterIdentifier": "test-global-cluster-2"
    },
    {
        "LocalWriteForwardingStatus": "null",
        "DBClusterIdentifier": "aurora-mysql-v2-cluster"
    }
]
```

Un cluster database può avere i seguenti valori per `LocalWriteForwardingStatus`:

- `disabled`: l'inoltro di scrittura è disabilitato.

- `disabling`: l'inoltro di scrittura è in fase di disabilitazione.
- `enabled`: l'inoltro di scrittura è abilitato.
- `enabling`: l'inoltro di scrittura è in fase di abilitazione.
- `null`: l'inoltro di scrittura non è disponibile per questo cluster database.
- `requested`: l'inoltro di scrittura è stato richiesto, ma non è ancora attivo.

Compatibilità delle applicazioni e di SQL con l'inoltro di scrittura

È possibile utilizzare i seguenti tipi di istruzioni SQL con l'inoltro di scrittura:

- Istruzioni DML (Data Manipulation Language), ad esempio `INSERT`, `DELETE` e `UPDATE`. Esistono alcune restrizioni sulle proprietà di queste istruzioni che è possibile utilizzare con l'inoltro di scrittura, come descritto di seguito.
- Istruzioni `SELECT ... LOCK IN SHARE MODE` e `SELECT FOR UPDATE`.
- Istruzioni `PREPARE` e `EXECUTE`.

Alcune istruzioni non sono consentite o possono produrre risultati non aggiornati quando vengono utilizzate in un cluster database con inoltro di scrittura. Inoltre, le funzioni e le procedure definite dall'utente non sono supportate. Pertanto, l'impostazione `EnableLocalWriteForwarding` è disabilitata per impostazione predefinita per i cluster database. Prima di abilitarla, verificare che il codice dell'applicazione non sia interessato da nessuna di queste restrizioni.

Le seguenti restrizioni si applicano alle istruzioni SQL utilizzate con l'inoltro di scrittura. In alcuni casi, è possibile utilizzare le istruzioni su cluster database con inoltro di scrittura abilitato. Questo approccio funziona se l'inoltro di scrittura non è abilitato all'interno della sessione dal parametro di configurazione `aurora_replica_read_consistency`. Se si prova a utilizzare un'istruzione quando non è consentita a causa dell'inoltro di scrittura, verrà visualizzato un messaggio di errore simile al seguente:

```
ERROR 1235 (42000): This version of MySQL doesn't yet support 'operation with write
 forwarding'.
```

DDL (Data Definition Language)

Connettersi all'istanza database di scrittura per eseguire le istruzioni DDL. Non è possibile eseguirle da istanze database di lettura.

Aggiornamento di una tabella permanente utilizzando i dati di una tabella temporanea

È possibile utilizzare tabelle temporanee in cluster database con l'inoltro di scrittura abilitato. Tuttavia, non è possibile utilizzare un'istruzione DML per modificare una tabella permanente se l'istruzione fa riferimento a una tabella temporanea. Ad esempio, non è possibile utilizzare un'istruzione `INSERT ... SELECT` che accetta i dati da una tabella temporanea.

Transazioni XA

Non è possibile utilizzare le istruzioni seguenti in un cluster database quando l'inoltro di scrittura è abilitato all'interno della sessione. È possibile utilizzare queste istruzioni su cluster database per i quali non è abilitato l'inoltro di scrittura o all'interno di sessioni in cui l'impostazione `aurora_replica_read_consistency` è vuota. Prima di abilitare l'inoltro di scrittura all'interno di una sessione, occorre verificare se il codice utilizza queste istruzioni.

```
XA {START|BEGIN} xid [JOIN|RESUME]
XA END xid [SUSPEND [FOR MIGRATE]]
XA PREPARE xid
XA COMMIT xid [ONE PHASE]
XA ROLLBACK xid
XA RECOVER [CONVERT XID]
```

Istruzioni LOAD per tabelle permanenti

Non è possibile utilizzare le istruzioni seguenti in un cluster database con l'inoltro di scrittura abilitato.

```
LOAD DATA INFILE 'data.txt' INTO TABLE t1;
LOAD XML LOCAL INFILE 'test.xml' INTO TABLE t1;
```

Istruzioni plugin

Non è possibile utilizzare le istruzioni seguenti in un cluster database con l'inoltro di scrittura abilitato.

```
INSTALL PLUGIN example SONAME 'ha_example.so';
UNINSTALL PLUGIN example;
```

Istruzioni SAVEPOINT

Non è possibile utilizzare le istruzioni seguenti in un cluster database quando l'inoltro di scrittura è abilitato all'interno della sessione. È possibile utilizzare queste istruzioni su cluster

database in cui l'inoltro di scrittura non è abilitato o all'interno di sessioni in cui l'impostazione `aurora_replica_read_consistency` è vuota. Prima di abilitare l'inoltro di scrittura all'interno di una sessione, occorre verificare se il codice utilizza queste istruzioni.

```
SAVEPOINT t1_save;
ROLLBACK TO SAVEPOINT t1_save;
RELEASE SAVEPOINT t1_save;
```

Livelli di isolamento per l'inoltro di scrittura

Nelle sessioni che utilizzano l'inoltro di scrittura, è possibile utilizzare solo il livello di isolamento `REPEATABLE READ`. Sebbene sia possibile utilizzare il livello di isolamento `READ COMMITTED` anche con repliche di Aurora, questa operazione non funziona con l'inoltro di scrittura. Per informazioni sui livelli di isolamento `REPEATABLE READ` e `READ COMMITTED`, consulta Livelli di isolamento di Aurora MySQL.

Coerenza di lettura per l'inoltro di scrittura

È possibile controllare il grado di coerenza di lettura in un cluster database. Il livello di coerenza di lettura determina il tempo di attesa di un cluster database prima di ogni operazione di lettura per garantire che alcune o tutte le modifiche vengano replicate dall'istanza di scrittura. È possibile regolare il livello di coerenza di lettura per garantire che tutte le operazioni di scrittura inoltrate dalla sessione siano visibili nel cluster database prima di qualsiasi query successiva. Inoltre, è possibile utilizzare questa impostazione per garantire che le query sul cluster database visualizzino sempre gli aggiornamenti più recenti dall'istanza di scrittura. Questa impostazione si applica anche alle query inviate da altre sessioni o da altri cluster. Per specificare questo tipo di comportamento per l'applicazione, scegli un valore per il parametro database `aurora_replica_read_consistency` o il parametro cluster database.

> **Important**
>
> Imposta sempre il parametro database `aurora_replica_read_consistency` o il parametro cluster database quando desideri inoltrare le scritture. In caso contrario, Aurora non inoltra le scritture. Questo parametro ha un valore vuoto per impostazione predefinita, quindi scegli un valore specifico quando utilizzi questo parametro. Il parametro `aurora_replica_read_consistency` influisce solo sui cluster o sulle istanze database in cui l'inoltro di scrittura è abilitato.

All'aumentare del livello di coerenza, l'applicazione rimane più tempo in attesa della propagazione delle modifiche tra le istanze database. È possibile scegliere il giusto equilibrio tra tempi di risposta rapidi e garanzia che le modifiche apportate in altre istanze database siano completamente disponibili prima dell'esecuzione delle query.

È possibile specificare i valori seguenti per il parametro `aurora_replica_read_consistency`:

- `EVENTUAL`: i risultati delle operazioni di scrittura nella stessa sessione non sono visibili finché l'operazione di scrittura non viene eseguita sull'istanza database di scrittura. La query non attende la disponibilità dei risultati aggiornati. Pertanto, potrebbe recuperare i dati più vecchi o i dati aggiornati, a seconda della tempistica delle istruzioni e della quantità di ritardo di replica. Questa è la stessa coerenza dei cluster database Aurora MySQL che non utilizzano l'inoltro di scrittura.
- `SESSION`: tutte le query che utilizzano l'inoltro di scrittura visualizzano i risultati di tutte le modifiche apportate in tale sessione. Le modifiche sono visibili indipendentemente dal fatto che la transazione sia stata impegnata. Se necessario, la query attende che i risultati delle operazioni di scrittura inoltrate vengano replicati.
- `GLOBAL`: una sessione visualizza tutte le modifiche confermate in tutte le sessioni e le istanze nel cluster database. Ogni query potrebbe attendere un periodo che varia a seconda della quantità di ritardo della sessione. La query procede quando il cluster database viene aggiornato con tutti i dati confermati dall'istanza di scrittura, a partire dal momento in cui la query è iniziata.

Per informazioni su tutti i parametri di configurazione coinvolti nell'inoltro di scrittura, consulta Parametri di configurazione per l'inoltro di scrittura.

> **Note**
>
> È anche possibile usare `aurora_replica_read_consistency` come una variabile di sessione, ad esempio:
>
> ```
> mysql> set aurora_replica_read_consistency = 'session';
> ```

Esempi di utilizzo dell'inoltro di scrittura

Negli esempi seguenti vengono illustrati gli effetti del parametro `aurora_replica_read_consistency` sull'esecuzione delle

istruzioni `INSERT` seguite dalle istruzioni `SELECT`. I risultati possono variare, a seconda del valore di `aurora_replica_read_consistency` e della tempistica delle istruzioni.

Per ottenere una maggiore coerenza, è possibile attendere brevemente prima di rilasciare l'istruzione `SELECT`. Oppure Aurora può attendere automaticamente il completamento della replica dei risultati prima di procedere con `SELECT`.

Per informazioni sull'impostazione dei parametri database, consulta Gruppi di parametri per Amazon Aurora.

Example con **`aurora_replica_read_consistency`** impostato su **`EVENTUAL`**

L'esecuzione di un'istruzione `INSERT`, immediatamente seguita da un'istruzione `SELECT`, restituisce un valore per `COUNT(*)` con il numero di righe prima dell'inserimento della nuova riga. Se poco dopo si esegue nuovamente `SELECT`, viene restituito il conteggio delle righe aggiornato. Le istruzioni `SELECT` non attendono.

```
mysql> select count(*) from t1;
+----------+
| count(*) |
+----------+
|        5 |
+----------+
1 row in set (0.00 sec)

mysql> insert into t1 values (6); select count(*) from t1;
+----------+
| count(*) |
+----------+
|        5 |
+----------+
1 row in set (0.00 sec)

mysql> select count(*) from t1;
+----------+
| count(*) |
+----------+
|        6 |
+----------+
1 row in set (0.00 sec)
```

Example con **aurora_replica_read_consistency** impostato su **SESSION**

Un'istruzione SELECT subito dopo l'istruzione INSERT attende fino a quando le modifiche apportate dall'istruzione INSERT diventano visibili. Le istruzioni SELECT successive non attendono.

```
mysql> select count(*) from t1;
+----------+
| count(*) |
+----------+
|        6 |
+----------+
1 row in set (0.01 sec)

mysql> insert into t1 values (6); select count(*) from t1; select count(*) from t1;
Query OK, 1 row affected (0.08 sec)
+----------+
| count(*) |
+----------+
|        7 |
+----------+
1 row in set (0.37 sec)
+----------+
| count(*) |
+----------+
|        7 |
+----------+
1 row in set (0.00 sec)
```

Con l'impostazione di coerenza di lettura ancora impostata su SESSION, l'introduzione di una breve attesa dopo l'esecuzione di un'istruzione INSERT rende disponibile il conteggio delle righe aggiornate dal momento dell'esecuzione dell'istruzione SELECT successiva.

```
mysql> insert into t1 values (6); select sleep(2); select count(*) from t1;
Query OK, 1 row affected (0.07 sec)
+----------+
| sleep(2) |
+----------+
|        0 |
+----------+
1 row in set (2.01 sec)
+----------+
| count(*) |
```

```
+----------+
|        8 |
+----------+
1 row in set (0.00 sec)
```

Example con **`aurora_replica_read_consistency`** impostato su **`GLOBAL`**

Ogni istruzione `SELECT` attende che tutte le modifiche ai dati, a partire dall'ora di inizio dell'istruzione, diventino visibili prima di eseguire la query. Il tempo di attesa per ciascuna istruzione `SELECT` varia, a seconda dell'entità di ritardo di replica.

```
mysql> select count(*) from t1;
+----------+
| count(*) |
+----------+
|        8 |
+----------+
1 row in set (0.75 sec)

mysql> select count(*) from t1;
+----------+
| count(*) |
+----------+
|        8 |
+----------+
1 row in set (0.37 sec)

mysql> select count(*) from t1;
+----------+
| count(*) |
+----------+
|        8 |
+----------+
1 row in set (0.66 sec)
```

Esecuzione di istruzioni a più parti con inoltro scrittura

Un'istruzione DML può essere costituita da più parti, ad esempio un'istruzione `INSERT ... SELECT` o un'istruzione `DELETE ... WHERE`. In questo caso, l'intera istruzione viene inoltrata all'istanza database di scrittura ed eseguita lì.

Transazioni con inoltro di scrittura

Se la modalità di accesso alle transazioni è impostata su sola lettura, l'inoltro di scrittura non viene utilizzato. È possibile specificare la modalità di accesso per la transazione utilizzando l'istruzione `SET TRANSACTION` o `START TRANSACTION`. È anche possibile specificare la modalità di accesso alle transazioni modificando il valore della variabile di sessione transaction_read_only. È possibile modificare questo valore di sessione solo quando si è connessi a un cluster database in cui l'inoltro di scrittura è abilitato.

Se una transazione di lunga durata non emette alcuna dichiarazione per un periodo di tempo considerevole, potrebbe superare il periodo di timeout inattivo. Questo periodo ha un valore predefinito di un minuto. È possibile impostare il parametro `aurora_fwd_writer_idle_timeout` per aumentarlo fino a un giorno. Una transazione che supera il timeout di inattività viene annullata dall'istanza di scrittura. L'istruzione successiva inviata riceve un errore di timeout. Quindi Aurora esegue il rollback della transazione.

Questo tipo di errore può verificarsi in altri casi in cui l'inoltro di scrittura non diventa disponibile. Ad esempio, Aurora annulla le eventuali transazioni che utilizzano l'inoltro di scrittura se si riavvia il cluster database o si disabilita l'inoltro di scrittura.

Quando un'istanza di scrittura in un cluster che utilizza l'inoltro di scrittura locale viene riavviata, le eventuali transazioni e query attive, inoltrate sulle istanze di lettura che utilizzano l'inoltro di scrittura locale vengono chiuse automaticamente. Dopo che l'istanza di scrittura sarà nuovamente disponibile, è possibile riprovare a eseguire queste transazioni.

Parametri di configurazione per l'inoltro di scrittura

I gruppi di parametri database Aurora contengono impostazioni per la funzionalità di inoltro di scrittura. I dettagli su questi parametri sono riepilogati nella tabella seguente, con note di utilizzo dopo la tabella.

Parametro	Ambito	Tipo	Valore predefinito	Valori validi
`aurora_fwd_writer_idle_timeout`	Cluster	Intero senza segno	60	1–86.400

Parametro	Ambito	Tipo	Valore predefinito	Valori validi
`aurora_fwd_writer_max_connections_pct`	Cluster	Intero lungo senza segno	10	0–90
`aurora_replica_read_consistency`	Cluster o istanza	Enum	" (null)	`EVENTUAL`, `SESSION`, `GLOBAL`

Per controllare le richieste di scrittura in entrata, usare queste impostazioni:

- `aurora_fwd_writer_idle_timeout`: il numero di secondi per cui l'istanza database di scrittura rimane in attesa su una connessione inoltrata da un'istanza di lettura prima di chiuderla. Se la sessione rimane inattiva oltre questo periodo, la sessione viene annullata da Aurora.
- `aurora_fwd_writer_max_connections_pct`: il limite superiore delle connessioni al database che può essere utilizzato su un'istanza database di scrittura per gestire le query inoltrate da istanze di lettura. È espresso come una percentuale dell'impostazione `max_connections` per l'istanza di scrittura. Ad esempio, se `max_connections` è 800 e `aurora_fwd_master_max_connections_pct` o `aurora_fwd_writer_max_connections_pct` è 10, allora lo scrittore consente un massimo di 80 sessioni inoltrate simultanee. Queste connessioni provengono dallo stesso pool di connessioni gestito dall'impostazione `max_connections`.

  Questa impostazione si applica solo all'istanza di scrittura quando l'inoltro di scrittura è abilitato. Se si riduce il valore, le connessioni esistenti non vengono influenzate. Aurora tiene in considerazione il nuovo valore dell'impostazione durante il tentativo di creazione di una nuova connessione da un cluster database. Il valore predefinito è 10, che rappresenta il 10% del valore `max_connections`.

> **Note**
>
> Poiché `aurora_fwd_writer_idle_timeout` e `aurora_fwd_writer_max_connections_pct` s
> parametri del cluster database, tutte le istanze database in ogni cluster hanno gli stessi valori per questi parametri.

Per ulteriori informazioni su `aurora_replica_read_consistency`, consulta Coerenza di lettura per l'inoltro di scrittura.

Per ulteriori informazioni sui gruppi di parametri database, consulta Gruppi di parametri per Amazon Aurora.

Parametri di Amazon CloudWatch e variabili di stato di Aurora MySQL per l'inoltro di scrittura

Le metriche Amazon CloudWatch e le variabili di stato Aurora MySQL seguenti si applicano quando si utilizza l'inoltro di scrittura per Aurora MySQL. Per ulteriori informazioni sulle metriche per le istanze database di lettura e scrittura Aurora MySQL, vedere i seguenti argomenti.

Argomenti

- Metriche per l'inoltro della scrittura per le istanze database di scrittura di Aurora MySQL
- Metriche per l'inoltro della scrittura per le istanze database di lettura MySQL Aurora

Metriche per l'inoltro della scrittura per le istanze database di scrittura di Aurora MySQL

Le seguenti metriche Amazon CloudWatch si applicano quando si utilizza l'inoltro di scrittura su uno o più cluster di database. Queste metriche sono tutte misurati sull'istanza database di scrittura.

Parametro CloudWatch	Unità	Descrizione
`ForwardingWriterDMLLatency`	Millisecondi	Tempo medio per elaborare ogni istruzione DML inoltrata sull'istanza database writer. Non include il tempo impiegato dal cluster database per

Parametro CloudWatch	Unità	Descrizione
		inoltrare la richiesta di scrittura o il tempo necessario per replicare le modifiche all'istanza di scrittura.
`ForwardingWriterDMLThroughput`	Conteggio al secondo	Numero di istruzioni DML inoltrate elaborate ogni secondo da questa istanza database writer.
`ForwardingWriterOpenSessions`	Conteggio	Numero di sessioni inoltrate sull'istanza database writer.

Le variabili di stato Aurora MySQL seguenti si applicano quando si utilizza l'inoltro di scrittura su uno o più cluster di database. Queste variabili di stato vengono tutte misurati sull'istanza database di scrittura.

Variabile di stato Aurora MySQL	Unità	Descrizione
`Aurora_fwd_writer_dml_stmt_count`	Conteggio	Numero totale di istruzioni DML inoltrate a questa istanza database writer.
`Aurora_fwd_writer_dml_stmt_duration`	Microsecondi	Durata totale delle istruzioni DML inoltrate a questa istanza database writer.
`Aurora_fwd_writer_open_sessions`	Conteggio	Numero di sessioni inoltrate sull'istanza database writer.
`Aurora_fwd_writer_select_stmt_count`	Conteggio	Numero totale di istruzioni `SELECT` inoltrate a questa istanza database writer.

Variabile di stato Aurora MySQL	Unità	Descrizione
`Aurora_fwd_writer_select_stmt_duration`	Microsecondi	Durata totale delle istruzioni `SELECT` inoltrate a questa istanza database writer.

Metriche per l’inoltro della scrittura per le istanze database di lettura MySQL Aurora

I seguenti parametri CloudWatch vengono misurati su ciascuna istanza database di lettura in un cluster di database con l’inoltro di scrittura abilitato.

Parametro CloudWatch	Unità	Descrizione
`ForwardingReplicaDMLLatency`	Millisecondi	Tempo medio di risposta di DML inoltrati su replica.
`ForwardingReplicaDMLThroughput`	Conteggio al secondo	Numero di istruzioni DML inoltrate elaborate al secondo.
`ForwardingReplicaOpenSessions`	Conteggio	Numero di sessioni che utilizzano l'inoltro di scrittura su un'istanza database di lettura.
`ForwardingReplicaReadWaitLatency`	Millisecondi	Tempo medio di attesa di un'istruzione `SELECT` su un'istanza database di lettura prima di raggiungere l'istanza di scrittura. Il grado di attesa dell'istanza database del lettore prima di elaborare una query dipende dall'impostazione `aurora_replica_read_consistency` .

Parametro CloudWatch	Unità	Descrizione
`ForwardingReplicaReadWaitThroughput`	Conteggio al secondo	Numero totale di istruzioni `SELECT` elaborate ogni secondo in tutte le sessioni che inoltrano scritture.
`ForwardingReplicaSelectLatency`	Millisecondi	Latenza `SELECT` inoltrata, mediata su tutte le istruzioni `SELECT` inoltrate all'interno del periodo di monitoraggio.
`ForwardingReplicaSelectThroughput`	Conteggio al secondo	Throughput inoltrato medio per secondo `SELECT` all'interno del periodo di monitoraggio.

Le seguenti variabili di stato Aurora MySQL vengono misurate su ciascuna istanza database di lettura in un cluster di database con l'inoltro di scrittura abilitato.

Variabile di stato Aurora MySQL	Unità	Descrizione
`Aurora_fwd_replica_dml_stmt_count`	Conteggio	Numero totale di istruzioni DML inoltrate da questa istanza database del lettore.
`Aurora_fwd_replica_dml_stmt_duration`	Microsecondi	Durata totale delle istruzioni DML inoltrate da questa istanza database del lettore.
`Aurora_fwd_replica_errors_session_limit`	Conteggio	Numero di sessioni rifiutate dal cluster primario a causa di una delle seguenti condizioni di errore: • istanza di scrittura piena

Variabile di stato Aurora MySQL	Unità	Descrizione
		• Troppe istruzioni inoltrate in corso.
`Aurora_fwd_replica_open_sessions`	Conteggio	Numero di sessioni che utilizzano l'inoltro di scrittura su un'istanza database di lettura.
`Aurora_fwd_replica_read_wait_count`	Conteggio	Numero totale di attese di lettura-post-scrittura su questa istanza database del lettore.
`Aurora_fwd_replica_read_wait_duration`	Microsecondi	Durata totale delle attese a causa dell'impostazione di coerenza di lettura su questa istanza database del lettore.
`Aurora_fwd_replica_select_stmt_count`	Conteggio	Numero totale di istruzioni `SELECT` inoltrate dall'istanza database del lettore.
`Aurora_fwd_replica_select_stmt_duration`	Microsecondi	Durata totale delle istruzioni `SELECT` inoltrate da questa istanza database del lettore.

Identificazione delle transazioni e delle query inoltrate

È possibile usare la tabella `information_schema.aurora_forwarding_processlist` per identificare le transazioni e le query inoltrate. Per ulteriori informazioni su questa tabella, consultare information_schema.aurora_forwarding_processlist.

L'esempio seguente mostra tutte le connessioni inoltrate su un'istanza database di scrittura.

```
mysql> select * from information_schema.AURORA_FORWARDING_PROCESSLIST where
 IS_FORWARDED=1 order by REPLICA_SESSION_ID;
```

```
+-----+----------+--------------------+----------+---------+------+-------------
+-----------------------------------------+-------------+-------------------
+---------------------------------+---------------------+----------------+
| ID  | USER     | HOST               | DB       | COMMAND | TIME | STATE        |
 INFO                                     | IS_FORWARDED | REPLICA_SESSION_ID |
 REPLICA_INSTANCE_IDENTIFIER     | REPLICA_CLUSTER_NAME | REPLICA_REGION |
+-----+----------+--------------------+----------+---------+------+-------------
+-----------------------------------------+-------------+-------------------
+---------------------------------+-------------------------------------+
| 648 | myuser   | IP_address:port1   | sysbench | Query   |    0 | async commit |
 UPDATE sbtest58 SET k=k+1 WHERE id=4802579 |            1 |                637 | my-
db-cluster-instance-2        | my-db-cluster        | us-west-2      |
| 650 | myuser   | IP_address:port2   | sysbench | Query   |    0 | async commit |
 UPDATE sbtest54 SET k=k+1 WHERE id=2503953 |            1 |                639 | my-
db-cluster-instance-2        | my-db-cluster        | us-west-2      |
+-----+----------+--------------------+----------+---------+------+-------------
+-----------------------------------------+-------------+-------------------
+---------------------------------+---------------------+----------------+
```

Nell'istanza database di lettura di inoltro, è possibile visualizzare i thread associati a queste connessioni database di scrittura eseguendo`SHOW PROCESSLIST`. Il valori `REPLICA_SESSION_ID` sull'istanza di scrittura, 637 e 639, sono identici ai valori`Id` sull'istanza di lettura.

```
mysql> select @@aurora_server_id;

+---------------------------------+
| @@aurora_server_id              |
+---------------------------------+
| my-db-cluster-instance-2        |
+---------------------------------+
1 row in set (0.00 sec)

mysql> show processlist;

+-----+----------+--------------------+----------+---------+------+-------------
+-----------------------------------------+
| Id  | User     | Host               | db       | Command | Time | State        | Info
                                        |
+-----+----------+--------------------+----------+---------+------+-------------
+-----------------------------------------+
| 637 | myuser   | IP_address:port1   | sysbench | Query   |    0 | async commit |
 UPDATE sbtest12 SET k=k+1 WHERE id=4802579  |
```

```
| 639 | myuser   | IP_address:port2   | sysbench | Query   |    0 | async commit |
 UPDATE sbtest61 SET k=k+1 WHERE id=2503953  |
+-----+----------+--------------------+----------+---------+------+--------------
+-------------------------------------------+
12 rows in set (0.00 sec)
```

Integrazione di Amazon Aurora MySQL con altri servizi AWS

Amazon Aurora MySQL si integra con altri servizi AWS per permettere di estendere il cluster database Aurora MySQL allo scopo di utilizzare capacità aggiuntive in AWS Cloud. Il cluster database Aurora MySQL può utilizzare i servizi AWSper gli scopi seguenti:

- Chiamata sincrona o asincrona di una funzione AWS Lambda utilizzando le funzioni native `lambda_sync` o `lambda_async`. Per ulteriori informazioni, consulta Chiamare una funzione Lambda da un cluster DB Amazon Aurora MySQL.
- Caricamento dei dati da file di testo o XML archiviati in un bucket Amazon Simple Storage Service (Amazon S3) nel cluster DB utilizzando il comando `LOAD DATA FROM S3` o `LOAD XML FROM S3`. Per ulteriori informazioni, consulta Caricamento dei dati in un cluster DB Amazon Aurora MySQL da file di testo in un bucket Amazon S3.
- Salvataggio dei dati nei file di testo archiviati in un bucket Amazon S3 dal cluster DB utilizzando il comando `SELECT INTO OUTFILE S3`. Per ulteriori informazioni, consulta Salvataggio dei dati da un cluster DB Amazon Aurora MySQL nei file di testo in un bucket Amazon S3.
- Aggiunta o rimozione automatica di repliche Aurora con Application Auto Scaling. Per ulteriori informazioni, consulta Dimensionamento automatico di Amazon Aurora con le repliche Aurora.
- Esegui l'analisi del sentiment con Amazon Comprehend o un'ampia varietà di algoritmi di machine learning con SageMaker AI. Per ulteriori informazioni, consulta Utilizzo di machine learning di Amazon Aurora.

Aurora protegge la possibilità di accedere ad altri servizi AWS tramite AWS Identity and Access Management (IAM). Puoi concedere l'accesso ad altri servizi AWS creando un ruolo IAM con le autorizzazioni necessarie e quindi associando il ruolo al cluster database. Per dettagli e istruzioni su come permettere al cluster DB Aurora MySQL per accedere ad altri servizi AWS per tuo conto, consulta Autorizzazione di Amazon Aurora MySQL ad accedere ad altri servizi AWS per tuo conto.

Autorizzazione di Amazon Aurora MySQL ad accedere ad altri servizi AWS per tuo conto

Perché il tuo cluster database Aurora MySQL possa accedere ad altri servizi per tuo conto, crea e configura un ruolo AWS Identity and Access Management (IAM). Questo ruolo autorizza gli utenti del database nel cluster DB ad accedere ad altri servizi AWS. Per ulteriori informazioni, consulta Configurazione dei ruoli IAM per accedere ai servizi AWS.

È anche necessario configurare il cluster database Aurora per consentire le connessioni in uscita al servizio AWS di destinazione. Per ulteriori informazioni, consulta Abilitazione delle comunicazioni di rete da Amazon Aurora ad altri servizi AWS.

In questo caso, gli utenti del database possono eseguire le operazioni utilizzando gli altri servizi AWS:

- Chiamata sincrona o asincrona di una funzione AWS Lambda utilizzando le funzioni native `lambda_sync` o `lambda_async`. In alternativa, chiamata asincrona di una funzione AWS Lambda utilizzando la procedura `mysql.lambda_async`. Per ulteriori informazioni, consulta Chiamare una funzione Lambda con una funzione nativa Aurora MySQL.
- Caricamento dei dati da file di testo o XML memorizzati in un bucket Amazon S3 nel cluster DB utilizzando l'istruzione `LOAD DATA FROM S3` o `LOAD XML FROM S3`. Per ulteriori informazioni, consulta Caricamento dei dati in un cluster DB Amazon Aurora MySQL da file di testo in un bucket Amazon S3.
- Salvataggio dei dati dal cluster DB nei file di testo memorizzati in un bucket Amazon S3 utilizzando l'istruzione `SELECT INTO OUTFILE S3`. Per ulteriori informazioni, consulta Salvataggio dei dati da un cluster DB Amazon Aurora MySQL nei file di testo in un bucket Amazon S3.
- Esportazione dei dati di log in Amazon CloudWatch Logs MySQL. Per ulteriori informazioni, consulta Pubblicazione di log Amazon Aurora MySQL in Amazon CloudWatch Logs.
- Aggiunta o rimozione automatica di repliche Aurora con Application Auto Scaling. Per ulteriori informazioni, consulta Dimensionamento automatico di Amazon Aurora con le repliche Aurora.

Configurazione dei ruoli IAM per accedere ai servizi AWS

Per consentire l'accesso di un cluster database Aurora a un altro servizio AWS, effettua le seguenti operazioni:

1. Crea una policy IAM che conceda l'autorizzazione al servizio AWS. Per ulteriori informazioni, consulta i seguenti argomenti.
   - Creazione di una policy IAM per l'accesso alle risorse Amazon S3
   - Creazione di una policy IAM per accedere alle AWS Lambda risorse
   - Creazione di una policy IAM per accedere alle CloudWatch risorse Logs
   - Creazione di una policy IAM per accedere alle AWS KMS risorse
2. Crea un ruolo IAM e collega la policy che hai creato. Per ulteriori informazioni, consulta Creazione di un ruolo IAM per consentire ad Amazon Aurora di accedere ai servizi AWS.

3. Associa il ruolo IAM al cluster DB Aurora. Per ulteriori informazioni, consulta Associazione di un ruolo IAM a un cluster DB Amazon Aurora MySQL.

Creazione di una policy IAM per l'accesso alle risorse Amazon S3

Aurora può accedere alle risorse di Amazon S3 per caricare i dati o per salvare i dati da un cluster DB Aurora. Tuttavia, è necessario prima creare una policy IAM che assegni le autorizzazioni del bucket e dell'oggetto che rendono possibile l'accesso di Aurora ad Amazon S3.

La tabella seguente indica le funzionalità di Aurora che possono accedere a un bucket Amazon S3 per tuo conto e le autorizzazioni minime del bucket e dell'oggetto richieste da ciascuna funzionalità.

Caratteristica	Autorizzazioni del bucket	Autorizzazioni dell'oggetto
`LOAD DATA FROM S3`	`ListBucket`	`GetObject` `GetObjectVersion`
`LOAD XML FROM S3`	`ListBucket`	`GetObject` `GetObjectVersion`

`SELECT INTO OUTFILE S3`	`ListBucket`	`AbortMultipartUpload`
		`DeleteObject`
		`GetObject`
		`ListMultipartUploadParts`
		`PutObject`

La seguente policy aggiunge le autorizzazioni che Amazon S3 potrebbe richiedere da Aurora per accedere al bucket per tuo conto.

JSON

```
{
```

```
    "Version":"2012-10-17",
    "Statement": [
        {
            "Sid": "AllowAuroraToExampleBucket",
            "Effect": "Allow",
            "Action": [
                "s3:PutObject",
                "s3:GetObject",
                "s3:AbortMultipartUpload",
                "s3:ListBucket",
                "s3:DeleteObject",
                "s3:GetObjectVersion",
                "s3:ListMultipartUploadParts"
            ],
            "Resource": [
                "arn:aws:s3:::amzn-s3-demo-bucket/*",
                "arn:aws:s3:::amzn-s3-demo-bucket"
            ]
        }
    ]
}
```

> **Note**
>
> Assicurati di includere entrambe le voci per il valore `Resource`. Aurora ha bisogno delle autorizzazioni sia sul bucket stesso che su tutti gli oggetti all'interno del bucket.
> In base al caso d'uso, potrebbe non essere necessario aggiungere tutte le autorizzazioni presenti nella policy di esempio. Potrebbero inoltre essere richieste altre autorizzazioni. Ad esempio, se il bucket Amazon S3 è crittografato, occorre aggiungere autorizzazioni `kms:Decrypt`.

È possibile seguire i passaggi seguenti per creare una policy IAM che conceda le autorizzazioni minime necessarie affinché Aurora possa accedere a un bucket Amazon S3 per tuo conto. Per consentire ad Aurora l'accesso a tutti i bucket Amazon S3, puoi saltare questi passaggi e utilizzare la policy IAM predefinita `AmazonS3ReadOnlyAccess` o `AmazonS3FullAccess` invece di crearne una nuova.

Per creare una policy IAM per consentire l'accesso alle risorse Amazon S3

1. Aprire la console di gestione IAM.
2. Nel pannello di navigazione, selezionare Policies (Policy).
3. Scegli Create Policy (Crea policy).
4. Nella scheda Visual editor (Editor visivo) selezionare Choose a service (Scegli un servizio) e quindi S3.
5. Per Actions (Operazioni), scegliere Expand all (Espandi tutto), quindi scegliere le autorizzazioni del bucket e dell'oggetto necessarie per la policy IAM.

   Le autorizzazioni dell'oggetto si riferiscono alle operazioni sugli oggetti in Amazon S3 e devono essere concesse per gli oggetti presenti nel bucket e non per il bucket stesso. Per ulteriori informazioni sulle autorizzazioni per le operazioni degli oggetti in Amazon S3, consulta Autorizzazioni per operazioni relative agli oggetti.
6. Scegliere Resources (Risorse) e selezionare Add ARN (Aggiungi ARN) per bucket.
7. Nella finestra di dialogo Add ARN(s) (Aggiungi ARN), fornire i dettagli sulla risorsa e scegliere Add (Aggiungi).

   Specificare il bucket Amazon S3 a cui consentire l'accesso. Ad esempio, per consentire ad Aurora di accedere al bucket Amazon S3 denominato *`amzn-s3-demo-bucket`*, impostare il valore Amazon Resource Name (ARN) su `arn:aws:s3:::`*`amzn-s3-demo-bucket`*.
8. Se la risorsa object (oggetto) è elencata, scegliere Add ARN (Aggiungi ARN) per object (oggetto).
9. Nella finestra di dialogo Add ARN(s) (Aggiungi ARN), fornire i dettagli sulla risorsa.

   Per il bucket Amazon S3 specificare il bucket Amazon S3 a cui consentire l'accesso. Per l'oggetto, è anche possibile scegliere Any (Qualsiasi) per concedere le autorizzazioni per qualsiasi oggetto nel bucket.

> **Note**
>
> È possibile impostare Amazon Resource Name (ARN) su un valore dell'ARN più specifico in modo da consentire ad Aurora di accedere solo a specifici file o cartelle presenti in un bucket Amazon S3. Per ulteriori informazioni su come definire una policy di accesso per Amazon S3, consulta Gestione delle autorizzazioni di accesso alle risorse Amazon S3.

10. (Facoltativo) Scegliere Add ARN (Aggiungi ARN) affinché il bucket aggiunga un altro bucket Amazon S3 alla policy e ripetere i passaggi precedenti per il bucket.

    > **Note**
    >
    > È possibile ripetere questa operazione per aggiungere le istruzioni di autorizzazione del bucket corrispondenti alla policy per ciascun bucket Amazon S3 a cui Aurora deve accedere. Facoltativamente, è anche possibile concedere l'accesso a tutti i bucket e gli oggetti in Amazon S3.

11. Scegliere Review policy (Esamina policy).
12. Per Name (Nome), immettere un nome per la policy IAM, ad esempio `AllowAuroraToExampleBucket`. Questo nome viene utilizzato quando si crea un ruolo IAM e lo si associa al cluster DB Aurora. È anche possibile aggiungere un valore Description (Descrizione) facoltativo.
13. Seleziona Create Policy (Crea policy).
14. Completa le fasi descritte in Creazione di un ruolo IAM per consentire ad Amazon Aurora di accedere ai servizi AWS.

Creazione di una policy IAM per accedere alle AWS Lambda risorse

Puoi creare una policy IAM che fornisca le autorizzazioni minime richieste ad Aurora per richiamare AWS Lambda una funzione per tuo conto.

La seguente policy aggiunge le autorizzazioni richieste da Aurora per richiamare la funzione AWS Lambda per tuo conto.

JSON

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Sid": "AllowAuroraToExampleFunction",
      "Effect": "Allow",
      "Action": "lambda:InvokeFunction",
```

```
      "Resource": "arn:aws:lambda:us-
east-1:123456789012:function:example_function"
    }
  ]
}
```

Puoi utilizzare i seguenti passaggi per creare una policy IAM che fornisca le autorizzazioni minime richieste ad Aurora per richiamare AWS Lambda una funzione per tuo conto. Per consentire ad Aurora di richiamare tutte le tue AWS Lambda funzioni, puoi saltare questi passaggi e utilizzare la `AWSLambdaRole` policy predefinita invece di crearne una tua.

Per creare una policy IAM per consentire la chiamata delle funzioni AWS Lambda

1. Aprire la console IAM.
2. Nel riquadro di navigazione, scegli Policy.
3. Scegli Create Policy (Crea policy).
4. Nella scheda Visual editor (Editor visivo), selezionare Choose a service (Scegli un servizio) e scegliere Lambda.
5. Per Actions (Operazioni), scegliere Expand all (Espandi tutto), quindi scegliere le autorizzazioni AWS Lambda necessarie per la policy IAM.

   Verificare che `InvokeFunction` sia selezionato. È l'autorizzazione minima richiesta per consentire ad Amazon Aurora di richiamare una funzione. AWS Lambda
6. Scegliere Resources (Risorse) e selezionare Add ARN (Aggiungi ARN) per function (funzione).
7. Nella finestra di dialogo Add ARN(s) (Aggiungi ARN), fornire i dettagli sulla risorsa.

   Specificare la funzione Lambda a cui consentire l'accesso. Ad esempio, per consentire ad Aurora l'accesso a una funzione Lambda denominata `example_function`, impostare il valore dell'ARN su `arn:aws:lambda:::function:example_function`.

   Per ulteriori informazioni su come definire una politica di accesso perAWS Lambda, consulta Autenticazione e controllo degli accessi per. AWS Lambda
8. Facoltativamente, scegli Aggiungi autorizzazioni aggiuntive per aggiungere un'altra AWS Lambda funzione alla politica e ripeti i passaggi precedenti per la funzione.

> **Note**
>
> Puoi ripetere l'operazione per aggiungere le istruzioni di autorizzazione alla funzione corrispondenti alla tua politica per ogni AWS Lambda funzione a cui desideri che Aurora acceda.

9. Scegliere Review policy (Esamina policy).
10. Impostare Name (Nome) su un nome per la policy IAM, ad esempio `AllowAuroraToExampleFunction`. Questo nome viene utilizzato quando si crea un ruolo IAM e lo si associa al cluster DB Aurora. È anche possibile aggiungere un valore Description (Descrizione) facoltativo.
11. Seleziona Create Policy (Crea policy).
12. Completa le fasi descritte in Creazione di un ruolo IAM per consentire ad Amazon Aurora di accedere ai servizi AWS.

Creazione di una policy IAM per accedere alle CloudWatch risorse Logs

Aurora può accedere ai CloudWatch log per esportare i dati dei log di controllo da un cluster Aurora DB. Tuttavia, devi prima creare una policy IAM che fornisca le autorizzazioni per il gruppo di log e il flusso di log che consentono ad Aurora di accedere ai CloudWatch log.

La seguente policy aggiunge le autorizzazioni richieste da Aurora per accedere ad CloudWatch Amazon Logs per tuo conto e le autorizzazioni minime richieste per creare gruppi di log ed esportare dati.

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Sid": "EnableCreationAndManagementOfRDSCloudwatchLogEvents",
            "Effect": "Allow",
            "Action": [
                "logs:GetLogEvents",
                "logs:PutLogEvents"
            ],
            "Resource": "arn:aws:logs:*:*:log-group:/aws/rds/*:log-stream:*"
```

```
        },
        {
            "Sid":
 "EnableCreationAndManagementOfRDSCloudwatchLogGroupsAndStreams",
            "Effect": "Allow",
            "Action": [
                "logs:CreateLogStream",
                "logs:DescribeLogStreams",
                "logs:PutRetentionPolicy",
                "logs:CreateLogGroup"
            ],
            "Resource": "arn:aws:logs:*:*:log-group:/aws/rds/*"
        }
    ]
}
```

Puoi modificare la ARNs policy per limitare l'accesso a una regione e a un account specificiAWS.

Puoi utilizzare i seguenti passaggi per creare una policy IAM che fornisca le autorizzazioni minime richieste ad Aurora per CloudWatch accedere ai log per tuo conto. Per consentire ad Aurora l'accesso completo ai CloudWatch log, puoi saltare questi passaggi e utilizzare la policy IAM `CloudWatchLogsFullAccess` predefinita invece di crearne una tua. Per ulteriori informazioni, consulta Using Identity-Based Policy (IAM policies) for CloudWatch Logs nella Amazon User Guide. CloudWatch

Per creare una policy IAM per concedere l'accesso alle tue risorse Logs CloudWatch

1. Aprire la console IAM.
2. Nel riquadro di navigazione, scegli Policy.
3. Scegli Create Policy (Crea policy).
4. Nella scheda Visual Editor, scegli Scegli un servizio, quindi scegli CloudWatchLogs.
5. Per Actions, scegli Espandi tutto (a destra), quindi scegli le autorizzazioni Amazon CloudWatch Logs necessarie per la policy IAM.

   Verificare che le seguenti autorizzazioni siano selezionate:

   - `CreateLogGroup`
   - `CreateLogStream`
   - `DescribeLogStreams`

- `GetLogEvents`
- `PutLogEvents`
- `PutRetentionPolicy`

6. Scegliere Resources (Risorse) e selezionare Add ARN (Aggiungi ARN) per log-group (gruppo di log).
7. Nella finestra di dialogo Add ARN(s) (Aggiungi ARN), immettere i seguenti valori:
   - Regione: una AWS regione o *
   - Account – Un numero di account o *
   - Log Group Name (Nome gruppo di log – `/aws/rds/*`
8. Nella finestra di dialogo Add ARN(s) (Aggiungi ARN), scegliere Add (Aggiungi).
9. Scegliere Add ARN (Aggiungi ARN) per log-stream (flusso di log).
10. Nella finestra di dialogo Add ARN(s) (Aggiungi ARN), immettere i seguenti valori:
    - Regione: una AWS regione o *
    - Account – Un numero di account o *
    - Log Group Name (Nome gruppo di log – `/aws/rds/*`
    - Log Stream Name (Nome flusso di log – *
11. Nella finestra di dialogo Add ARN(s) (Aggiungi ARN), scegliere Add (Aggiungi).
12. Scegliere Review policy (Esamina policy).
13. Impostare Name (Nome) su un nome per la policy IAM, ad esempio `AmazonRDSCloudWatchLogs`. Questo nome viene utilizzato quando si crea un ruolo IAM e lo si associa al cluster DB Aurora. È anche possibile aggiungere un valore Description (Descrizione) facoltativo.
14. Seleziona Create Policy (Crea policy).
15. Completa le fasi descritte in Creazione di un ruolo IAM per consentire ad Amazon Aurora di accedere ai servizi AWS.

Creazione di una policy IAM per accedere alle AWS KMS risorse

Aurora può accedere alle AWS KMS keys utilizzate per la crittografia dei backup dei database. Tuttavia, è necessario innanzitutto creare una policy IAM che assegni le autorizzazioni che consentano ad Aurora di accedere alle chiavi KMS.

La seguente policy aggiunge le autorizzazioni che Aurora richiede per accedere alle chiavi KMS per tuo conto.

JSON

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Sid": "AllowAuroraToAccessKey",
      "Effect": "Allow",
      "Action": [
        "kms:Decrypt"
      ],
      "Resource": "arn:aws:kms:us-east-1:123456789012:key/key-ID"
    }
  ]
}
```

È possibile seguire i passaggi seguenti per creare una policy IAM che concede le autorizzazioni minime necessarie affinché Aurora possa accedere alle chiavi KMS per tuo conto.

Per creare una policy IAM per consentire l'accesso alle chiavi KMS

1. Aprire la console IAM.
2. Nel riquadro di navigazione, scegli Policy.
3. Scegli Create Policy (Crea policy).
4. Nella scheda Visual editor (Editor visivo), selezionare Choose a service (Scegli un servizio) e scegliere KMS.
5. Per Actions (Operazioni), scegliere Write (Scrivi) e selezionare Decrypt (Decrittografa).
6. Scegliere Resources (Risorse) e selezionare Add ARN (Aggiungi ARN).
7. Nella finestra di dialogo Add ARN(s) (Aggiungi ARN), immettere i seguenti valori:
   - Regione: digita la AWS regione, ad esempio`us-west-2`.
   - Account – Specificare il numero dell'account utente.

- Nome flusso di registro: digitare l'identificatore della chiave KMS.

8. Nella finestra di dialogo Add ARN(s) (Aggiungi ARN), scegliere Add (Aggiungi).
9. Scegliere Review policy (Esamina policy).
10. Impostare Name (Nome) su un nome per la policy IAM, ad esempio `AmazonRDSKMSKey`. Questo nome viene utilizzato quando si crea un ruolo IAM e lo si associa al cluster DB Aurora. È anche possibile aggiungere un valore Description (Descrizione) facoltativo.
11. Seleziona Create Policy (Crea policy).
12. Completa le fasi descritte in Creazione di un ruolo IAM per consentire ad Amazon Aurora di accedere ai servizi AWS.

Creazione di un ruolo IAM per consentire ad Amazon Aurora di accedere ai servizi AWS

Dopo aver creato una policy IAM per consentire ad Aurora di accedere alle risorse AWS, è necessario creare un ruolo IAM e collegare la policy IAM al nuovo ruolo IAM.

Per creare un ruolo IAM che consenta al cluster Amazon RDS di comunicare con altri servizi AWS per tuo conto, completa la procedura riportata di seguito.

Come creare un ruolo IAM per consentire ad Amazon RDS di accedere ai servizi AWS

1. Aprire la console IAM.
2. Nel pannello di navigazione, selezionare Ruoli.
3. Selezionare Create role (Crea ruolo).
4. In Servizio AWS, scegli RDS.
5. In Select your use case (Seleziona caso di utilizzo), selezionare RDS – Add Role to Database (RDS – Aggiungi ruolo al database).
6. Scegli Next (Successivo).
7. Nella pagina Permissions policies (Policy di autorizzazione), immetti il nome della policy nel campo Search (Cerca).
8. Quando viene visualizzata nell'elenco, selezionare la policy definita in precedenza utilizzando le istruzioni presenti in una delle seguenti sezioni:

   - Creazione di una policy IAM per l'accesso alle risorse Amazon S3
   - Creazione di una policy IAM per accedere alle AWS Lambda risorse
   - Creazione di una policy IAM per accedere alle CloudWatch risorse Logs

- Creazione di una policy IAM per accedere alle AWS KMS risorse

9. Scegli Next (Successivo).
10. IN Role name (Nome ruolo), digitare un nome per il ruolo IAM, ad esempio `RDSLoadFromS3`. È anche possibile aggiungere un valore Description (Descrizione) facoltativo.
11. Selezionare Create Role (Crea ruolo).
12. Completa le fasi descritte in Associazione di un ruolo IAM a un cluster DB Amazon Aurora MySQL.

Associazione di un ruolo IAM a un cluster DB Amazon Aurora MySQL

Per consentire agli utenti del database in un cluster database Amazon Aurora di accedere ad altri servizi AWS, è necessario associare il ruolo IAM creato in Creazione di un ruolo IAM per consentire ad Amazon Aurora di accedere ai servizi AWS al cluster database. È anche possibile configurare AWS per creare un nuovo ruolo IAM associando direttamente il servizio.

> **Note**
>
> Non è possibile associare un ruolo IAM a un cluster DB Aurora Serverless v1. Per ulteriori informazioni, consulta Utilizzo di Amazon Aurora Serverless v1.
> Puoi associare un ruolo IAM a un cluster database Aurora Serverless v2.

Per associare un ruolo IAM a un cluster DB è necessario effettuare due operazioni:

1. Aggiungere il ruolo all'elenco dei ruoli associati del cluster di database tramite la console RDS, il comando add-role-to-db-cluster della AWS CLI o l'operazione API RDS AddRoleToDBCluster.

   Puoi aggiungere un massimo di cinque ruoli IAM per ogni cluster DB Aurora.
2. Impostare l'ARN del ruolo IAM associato nel parametro a livello di cluster per il servizio AWS interessato.

   La seguente tabella elenca i nomi dei parametri a livello di cluster per i ruoli IAM utilizzati per accedere ad altri servizi AWS.

Parametro a livello di cluster	Descrizione	
`aws_default_lambda_role`	Viene utilizzato quando si richiama una funzione Lambda dal cluster DB.	
`aws_default_logs_role`	Questo parametro non è più richiesto per esportare i dati del log dal cluster database ad Amazon CloudWatch Logs. Aurora MySQL ora utilizza un ruolo collegato ai servizi per le autorizzazioni necessarie. Per ulteriori informazioni sui ruoli collegati al servizio, consulta Utilizzo di ruoli collegati ai servizi per Amazon Aurora.	
`aws_default_s3_role`	Viene utilizzato quando si richiama l'istruzione `LOAD DATA FROM S3`, `LOAD XML FROM S3` o `SELECT INTO OUTFILE S3` dal cluster DB. In Aurora MySQL versione 2, il ruolo IAM specificato in questo parametro viene utilizzato se un ruolo IAM non è specificato per `aurora_load_from_s3_role` o `aurora_select_into_s3_role` per l'istruzione appropriata. In Aurora MySQL versione 3, il ruolo IAM specificato per questo parametro è sempre utilizzato.	

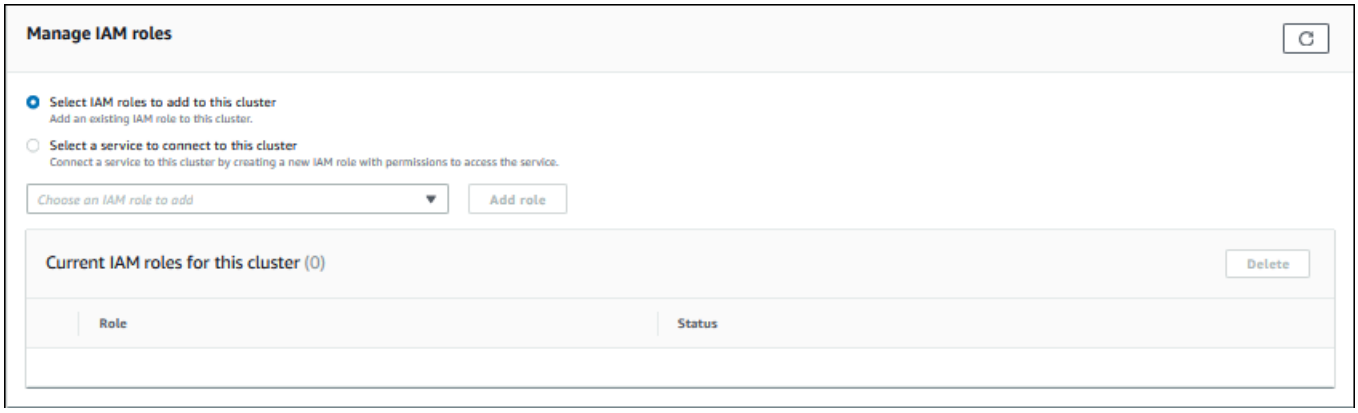
Parametro a livello di cluster	Descrizione	
`aurora_load_from_s3_role`	Viene utilizzato quando si richiama l'istruzione `LOAD DATA FROM S3` o `LOAD XML FROM S3` dal cluster DB. Se un ruolo IAM non è specificato per questo parametro viene usato il ruolo IAM specificato in `aws_default_s3_role` . In Aurora MySQL versione 3, questo parametro non è disponibile.	
`aurora_select_into_s3_role`	Viene utilizzato quando si richiama l'istruzione `SELECT INTO OUTFILE S3` dal cluster DB. Se un ruolo IAM non è specificato per questo parametro viene usato il ruolo IAM specificato in `aws_default_s3_role` . In Aurora MySQL versione 3, questo parametro non è disponibile.	

Per associare un ruolo IAM che consenta al cluster Amazon RDS di comunicare con altri servizi AWS per tuo conto, completa la procedura riportata di seguito.

Console

Per associare un ruolo IAM a un cluster DB Aurora utilizzando la console

1. Apri la console RDS all'indirizzo https://console.aws.amazon.com/rds/.
2. Scegliere Databases (Database).
3. Scegliere il nome del cluster DB Aurora da associare a un ruolo IAM per mostrarne i dettagli.
4. Nella scheda Connettività e sicurezza, nella sezione Gestisci ruoli IAM, esegui una delle seguenti operazioni:
   - Seleziona ruoli IAM da aggiungere a questo cluster (impostazione predefinita)
   - Seleziona un servizio per connetterti a questo cluster



5. Per utilizzare un ruolo IAM esistente, selezionalo dal menu, quindi scegli Aggiungi ruolo.

   Se l'aggiunta del ruolo ha esito positivo, il suo stato viene visualizzato come `Pending`, quindi `Available`.

6. Per connettere direttamente un servizio:

   a. Scegli Seleziona un servizio per connetterti a questo cluster.

   b. Scegli il servizio dal menu, quindi seleziona Connetti un servizio.

   c. Per Connetti il cluster al ***`Service Name`***, immetti il nome della risorsa Amazon (ARN) da utilizzare per connettersi al servizio, quindi scegli Connetti un servizio.

   AWS crea un nuovo ruolo IAM per la connessione al servizio. Il suo stato viene visualizzato come `Pending`, quindi `Available`.

7. (Facoltativo) Per arrestare l'associazione di un ruolo IAM a un cluster database e rimuovere l'autorizzazione correlata, scegli il ruolo e quindi seleziona Elimina.

Per impostare il parametro a livello di cluster per il ruolo IAM associato

1. Nel riquadro di navigazione della console di RDS selezionare Parameter groups (Gruppi di parametri).
2. Se si sta già utilizzando un gruppo di parametri database personalizzato, è possibile selezionare quel gruppo per usarlo invece di creare un nuovo gruppo di parametri del cluster DB. Se si sta utilizzando il gruppo di parametri del cluster DB predefinito, creare un nuovo gruppo di parametri del cluster DB, come descritto nei passaggi seguenti:

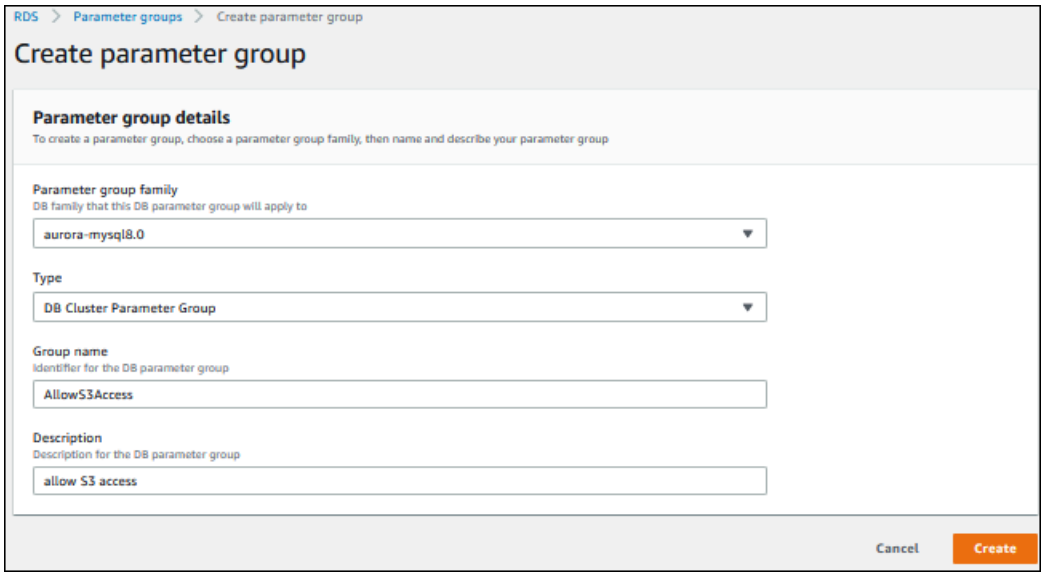
   a. Scegliere Create parameter group (Crea gruppo di parametri).

b. Per Famiglia del gruppo di parametri, scegli `aurora-mysql8.0` per un cluster di database compatibile con Aurora MySQL 8.0 o scegli `aurora-mysql5.7` per un cluster di database compatibile con Aurora MySQL 5.7.

c. Per Type (Tipo), scegliere DB Cluster Parameter Group (Gruppo di parametri del cluster DB).

d. Per Group name (Nome gruppo), digitare il nome del nuovo gruppo di parametri del cluster DB.

e. Per Description (Descrizione), digitare una descrizione per il nuovo gruppo di parametri del cluster DB.



f. Scegliere Create (Crea).

3. Nella pagina Parameter groups (Gruppi di parametri), selezionare il gruppo di parametri del cluster DB e scegliere Edit (Modifica) per Parameter group actions (Operazioni del gruppo di parametri).

4. Imposta i parametri appropriati a livello di cluster sui valori ARN del ruolo IAM correlato.

   Ad esempio, è possibile impostare il parametro `aws_default_s3_role` su `arn:aws:iam::123456789012:role/AllowS3Access`.

5. Seleziona Save changes (Salva modifiche).

6. Per modificare il gruppo di parametri del cluster di database per il cluster di database, completare le seguenti fasi:

   a. Scegliere Databases (Database), quindi scegliere il cluster di database Aurora.

   b. Scegliere Modify (Modifica).

c. Scorrere fino a Database options (Opzioni database) e impostare DB cluster parameter group (Gruppo di parametri del cluster di database) sul gruppo di parametri del cluster di database.

d. Scegli Continue (Continua).

e. Controllare le modifiche e scegliere Apply immediately (Applica immediatamente).

f. Scegliere Modify cluster (Modifica cluster).

g. Scegliere Databases (Database) e scegliere l'istanza primaria per il cluster di database.

h. In Actions (Operazioni), scegliere Reboot (Riavvia).

   Al riavvio dell'istanza, il ruolo IAM viene associato al cluster DB.

   Per ulteriori informazioni sui gruppi di parametri del cluster, consulta Parametri di configurazione Aurora MySQL.

CLI

Per associare un ruolo IAM a un cluster DB utilizzando l AWS CLI

1. Chiamare il comando `add-role-to-db-cluster` dell'AWS CLI per aggiungere gli ARN per i ruoli IAM al cluster DB, come mostrato di seguito.

```
PROMPT> aws rds add-role-to-db-cluster --db-cluster-identifier my-cluster --role-
arn arn:aws:iam::123456789012:role/AllowAuroraS3Role
PROMPT> aws rds add-role-to-db-cluster --db-cluster-identifier my-cluster --role-
arn arn:aws:iam::123456789012:role/AllowAuroraLambdaRole
```

2. Se si sta utilizzando il gruppo di parametri del cluster DB predefinito, creare un nuovo gruppo di parametri del cluster DB. Se si sta già utilizzando un gruppo di parametri database personalizzato, è possibile usare quel gruppo invece di creare un nuovo gruppo di parametri del cluster DB.

   Per creare un nuovo gruppo di parametri del cluster DB, chiamare il comando `create-db-cluster-parameter-group` dell'AWS CLI, come mostrato di seguito.

```
PROMPT> aws rds create-db-cluster-parameter-group  --db-cluster-parameter-group-
name AllowAWSAccess \
     --db-parameter-group-family aurora5.7 --description "Allow access to Amazon S3
 and AWS Lambda"
```

Per un cluster DB compatibile con Aurora MySQL 5.7, specificare `aurora-mysql5.7` per `--db-parameter-group-family`. Per un cluster database compatibile con Aurora MySQL 8.0, specificare `aurora-mysql8.0` per `--db-parameter-group-family`.

3. Impostare i parametri o i parametri appropriati a livello di cluster e i relativi valori ARN del ruolo IAM nel gruppo di parametri del cluster DB, come illustrato di seguito.

```
PROMPT> aws rds modify-db-cluster-parameter-group --db-cluster-parameter-group-name
 AllowAWSAccess \
    --parameters
 "ParameterName=aws_default_s3_role,ParameterValue=arn:aws:iam::123456789012:role/
AllowAuroraS3Role,method=pending-reboot" \
    --parameters
 "ParameterName=aws_default_lambda_role,ParameterValue=arn:aws:iam::123456789012:role/
AllowAuroraLambdaRole,method=pending-reboot"
```

4. Modificare il cluster DB per usare il nuovo gruppo di parametri del cluster DB e riavviare il cluster, come mostrato di seguito.

```
PROMPT> aws rds modify-db-cluster --db-cluster-identifier my-cluster --db-cluster-
parameter-group-name AllowAWSAccess
PROMPT> aws rds reboot-db-instance --db-instance-identifier my-cluster-primary
```

Al riavvio dell'istanza, i ruoli IAM vengono associati al cluster DB.

Per ulteriori informazioni sui gruppi di parametri del cluster, consulta Parametri di configurazione Aurora MySQL.

Abilitazione delle comunicazioni di rete da Amazon Aurora ad altri servizi AWS

Per utilizzare alcuni altri servizi AWS con Amazon Aurora, la configurazione di rete del cluster database Aurora deve consentire connessioni in uscita agli endpoint per tali servizi. Le seguenti operazioni richiedono questa configurazione di rete.

- Richiamo di funzioni AWS Lambda Per informazioni su questa funzionalità, vedere Chiamare una funzione Lambda con una funzione nativa Aurora MySQL.
- Accesso ai file da Amazon S3. Per informazioni su questa funzionalità, vedere Caricamento dei dati in un cluster DB Amazon Aurora MySQL da file di testo in un bucket Amazon S3 e Salvataggio dei dati da un cluster DB Amazon Aurora MySQL nei file di testo in un bucket Amazon S3.

- Accesso agli endpoint AWS KMS. L'accesso a AWS KMS è richiesto per utilizzare i flussi di attività del database con Aurora MySQL. Per informazioni su questa funzionalità, vedere Monitoraggio di Amazon Aurora tramite i flussi di attività del database.
- Accesso agli endpoint SageMaker AI. L'accesso a SageMaker AI è necessario per utilizzare il machine learning di SageMaker AI con Aurora MySQL. Per informazioni su questa funzionalità, vedere Utilizzo di machine learning di Amazon Aurora con Aurora MySQL.

Quando non è possibile connettersi a un endpoint del servizio, Aurora restituisce il seguente messaggio di errore:

```
ERROR 1871 (HY000): S3 API returned error: Network Connection
```

```
ERROR 1873 (HY000): Lambda API returned error: Network Connection. Unable to connect to
 endpoint
```

```
ERROR 1815 (HY000): Internal error: Unable to initialize S3Stream
```

Per i flussi di attività del database che utilizzano Aurora MySQL, il flusso di attività smette di funzionare se il cluster database non è in grado di accedere all'endpoint AWS KMS. In tal caso, Aurora notifica questo problema utilizzando eventi RDS.

Se si verificano questi messaggi durante l'utilizzo dei servizi AWS corrispondenti, verificare se il cluster database Aurora è pubblico o privato. Se il cluster DB Aurora è privato, devi configurarlo per abilitare le connessioni.

Affinché un cluster DB Aurora sia pubblico, deve essere contrassegnato come accessibile pubblicamente. Se osservi i dettagli del cluster DB nell'Console di gestione AWS, Publicly Accessible (Accessibile pubblicamente) sarà Yes (Sì) in tal caso. Inoltre, il cluster DB deve trovarsi in una sottorete pubblica di Amazon VPC. Per ulteriori informazioni sulle istanze database accessibili pubblicamente, consulta Uso di un cluster database in un VPC. Per ulteriori informazioni sulle sottoreti pubbliche Amazon VPC, consulta VPC e sottoreti.

Se il cluster DB Aurora non è accessibile pubblicamente e si trova in una sottorete pubblica VPC significa che è privato. È possibile che si disponga di un cluster DB privato e si desideri utilizzare una delle funzionalità che richiedono questa configurazione di rete. In tal caso, configura il cluster in modo che possa connettersi agli indirizzi Internet tramite Network Address Translation (NAT). Come alternativa ad Amazon S3, Amazon SageMaker AI e AWS Lambda, puoi configurare il VPC in modo

che abbia un endpoint VPC per l'altro servizio associato alla tabella di routing del cluster di database (consulta Uso di un cluster database in un VPC). Per ulteriori informazioni sulla configurazione di NAT in VPC, consulta Gateway NAT. Per ulteriori informazioni sulla configurazione degli endpoint VPC, consulta Endpoint VPC. Puoi anche creare un endpoint del gateway S3 per accedere al bucket S3. Per ulteriori informazioni, consulta Endpoint gateway per Amazon S3.

Potrebbe anche essere necessario aprire le porte effimere per le liste di controllo degli accessi (ACL) nelle regole in uscita per il gruppo di sicurezza VPC. Per ulteriori informazioni sulle porte effimere per le liste di controllo degli accessi di rete, consulta Porte Effimerenella Guida per l'utente di Amazon Virtual Private Cloud.

Argomenti correlati

- Integrazione di Aurora con altri servizi AWS
- Gestione di un cluster DB Amazon Aurora

Caricamento dei dati in un cluster DB Amazon Aurora MySQL da file di testo in un bucket Amazon S3

Puoi usare l'istruzione `LOAD DATA FROM S3` o `LOAD XML FROM S3` per caricare i dati dai file memorizzati in un bucket Amazon S3. In Aurora MySQL, i file vengono prima archiviati sul disco locale e quindi importati nel database. Al termine delle importazioni nel database, i file locali vengono eliminati.

> **Note**
>
> Il caricamento dei dati in una tabella dai file di testo non è supportato in Aurora Serverless v1. È supportata per Aurora Serverless v2.

Indice

- Accesso di Aurora a Amazon S3;
- Concessione dei privilegi per caricare dati in Amazon Aurora MySQL
- Specifica del percorso (URI) in un bucket Amazon S3
- LOAD DATA FROM S3
  - Sintassi

- Parametri
- Utilizzo di un manifest per specificare i file di dati da caricare
  - Verifica dei file caricati utilizzando la tabella aurora_s3_load_history
- Esempi
- LOAD XML FROM S3
  - Sintassi
  - Parametri

Accesso di Aurora a Amazon S3;

Prima di poter caricare i dati da un bucket Amazon S3, è necessario innanzitutto concedere l'autorizzazione al cluster DB Aurora MySQL per accedere ad Amazon S3.

Per concedere ad Aurora MySQL l'accesso ad Amazon S3

1. Creare una policy AWS Identity and Access Management (IAM) che assegni le autorizzazioni bucket e di oggetto che rendono possibile l'accesso del cluster di database Aurora MySQL ad Amazon S3. Per istruzioni, consultare Creazione di una policy IAM per l'accesso alle risorse Amazon S3.

   > **Note**
   >
   > In Aurora MySQL 3.05 e versioni successive, è possibile caricare oggetti crittografati utilizzando AWS KMS keys gestite dal cliente. A tale scopo, includi l'autorizzazione `kms:Decrypt` nella policy IAM. Per ulteriori informazioni, consulta Creazione di una policy IAM per accedere alle AWS KMS risorse.
   > Non è necessaria questa autorizzazione per caricare oggetti crittografati utilizzando Chiavi gestite da AWS o chiavi gestite da Amazon S3 (SSE-S3).

2. Creare un ruolo IAM e collegare la policy IAM creata in Creazione di una policy IAM per l'accesso alle risorse Amazon S3 al nuovo ruolo IAM. Per istruzioni, consulta Creazione di un ruolo IAM per consentire ad Amazon Aurora di accedere ai servizi AWS.
3. Assicurati che il cluster DB stia utilizzando un gruppo di parametri del cluster DB personalizzato.

   Per ulteriori informazioni sulla creazione di un gruppo di parametri del cluster DB, consulta Creazione di un gruppo di parametri del cluster di database in Amazon Aurora.

4. Per Aurora MySQL versione 2, imposta il parametro del cluster di database `aurora_load_from_s3_role` o `aws_default_s3_role` sul nome della risorsa Amazon (ARN) del nuovo ruolo IAM. Se un ruolo IAM non è specificato per `aurora_load_from_s3_role`, Aurora utilizza il ruolo IAM specificato in `aws_default_s3_role`.

   Per Aurora MySQL versione 3, utilizza `aws_default_s3_role`.

   Se il cluster fa parte di un database globale Aurora, imposta questo parametro per ogni cluster Aurora nel database globale. Sebbene solo il cluster primario in un database globale Aurora può caricare i dati, un altro cluster potrebbe essere promosso dal meccanismo di failover e diventare il cluster primario.

   Per ulteriori informazioni sui parametri del cluster DB, vedi Parametri dell'istanza database e del cluster database di Amazon Aurora.

5. Per consentire l'accesso ad Aurora MySQL agli utenti del database in un cluster DB Amazon S3, è necessario associare il ruolo creato in Creazione di un ruolo IAM per consentire ad Amazon Aurora di accedere ai servizi AWS al cluster DB. Per un database globale Aurora, associa il ruolo a ogni cluster Aurora nel database globale. Per informazioni su come associare un ruolo IAM a un cluster DB, vedi Associazione di un ruolo IAM a un cluster DB Amazon Aurora MySQL.

6. Configura il cluster DB Aurora MySQL per consentire le connessioni in uscita ad Amazon S3. Per istruzioni, consultare Abilitazione delle comunicazioni di rete da Amazon Aurora ad altri servizi AWS.

   Se il cluster DB non è accessibile pubblicamente e si trova in una sottorete pubblica VPC significa che è privato. Puoi creare un endpoint del gateway S3 per accedere al bucket S3. Per ulteriori informazioni, consulta Endpoint gateway per Amazon S3.

   Per un database globale Aurora, abilita le connessioni in uscita per ogni cluster Aurora nel database globale.

Concessione dei privilegi per caricare dati in Amazon Aurora MySQL

L'utente del database che invia l'istruzione `LOAD DATA FROM S3` o `LOAD XML FROM S3` deve avere un ruolo o privilegio specifico per rilasciare una delle due istruzioni. In Aurora MySQL versione 3, concedi il ruolo `AWS_LOAD_S3_ACCESS`. In Aurora MySQL versione 2, concedi il privilegio `LOAD FROM S3`. All'utente amministrativo per un cluster di database è concesso il ruolo o il privilegio

appropriato per impostazione predefinita. Puoi concedere il privilegio a un altro utente usando le seguenti istruzioni.

Utilizzare la seguente istruzione per Aurora MySQL versione 3:

```
GRANT AWS_LOAD_S3_ACCESS TO 'user'@'domain-or-ip-address'
```

> **Tip**
>
> Quando utilizzi la tecnica basata sul ruolo in Aurora MySQL versione 3, puoi anche attivare il ruolo utilizzando l'istruzione `SET ROLE role_name` o `SET ROLE ALL`. Se non si ha familiarità con il sistema dei ruoli MySQL 8.0, è possibile ottenere ulteriori informazioni in Privilegio basato sui ruoli. Per maggiori dettagli, consulta la pagina Using roles nel Manuale di riferimento di MySQL.
> Questo vale solo per la sessione attiva corrente. Quando ti riconnetti, devi eseguire di nuovo l'istruzione `SET ROLE` per concedere i privilegi. Per ulteriori informazioni, consulta Istruzione SET ROLE nel Manuale di riferimento di MySQL.
> Puoi utilizzare il parametro `activate_all_roles_on_login` del cluster di database per attivare automaticamente tutti i ruoli quando un utente si connette a un'istanza database. Quando questo parametro è impostato, non è necessario chiamare esplicitamente l'istruzione `SET ROLE` per attivare un ruolo. Per ulteriori informazioni, consulta activate_all_roles_on_login nel Manuale di riferimento di MySQL.
> Tuttavia, è necessario chiamare esplicitamente `SET ROLE ALL` all'inizio di una stored procedure per attivare il ruolo, quando la stored procedure viene chiamata da un altro utente.

Utilizza la seguente istruzione per Aurora MySQL versione 2:

```
GRANT LOAD FROM S3 ON *.* TO 'user'@'domain-or-ip-address'
```

Il ruolo `AWS_LOAD_S3_ACCESS` e il privilegio `LOAD FROM S3` sono specifici di Amazon Aurora e non sono disponibili per i database MySQL esterni o le istanze database RDS per MySQL. Se è stata impostata la replica tra un cluster di database Aurora come origine di replica e un database MySQL come client di replica, l'istruzione `GRANT` per il ruolo o il privilegio causa l'arresto della replica con un errore. Puoi ignorare l'errore in modo sicuro per riprendere la replica. Per ignorare l'errore su un'istanza RDS per MySQL, utilizza la procedura mysql_rds_skip_repl_error. Per ignorare l'errore su un database MySQL esterno, usa la variabile di sistema slave_skip_errors (Aurora MySQL versione 2) o replica_skip_errors (Aurora MySQL versione 3).

> **Note**
>
> L'utente del database deve disporre di privilegi `INSERT` per il database in cui carica i dati.

Specifica del percorso (URI) in un bucket Amazon S3

La sintassi per specificare il percorso (URI) dei file archiviati su un bucket Amazon S3 è la seguente.

```
s3-region://amzn-s3-demo-bucket/file-name-or-prefix
```

Il percorso include i seguenti valori:

- `region` (facoltativo): la regione AWS che contiene il bucket Amazon S3 da cui effettuare il caricamento. Questo valore è facoltativo. Se non specifichi un valore per `region`, Aurora carica il file da Amazon S3 nella stessa regione del cluster DB.
- `bucket-name` – Il nome del bucket Amazon S3 che contiene i dati da caricare. Sono supportati i prefissi degli oggetti che identificano un percorso di cartella virtuale.
- `file-name-or-prefix` – Il nome del file di testo Amazon S3 o XML o un prefisso che identifica uno o più file di testo o XML da caricare. È anche possibile specificare un file manifest che identifica uno o più file di testo da caricare. Per ulteriori informazioni sull'utilizzo di un file manifest per caricare file di testo da Amazon S3, consulta Utilizzo di un manifest per specificare i file di dati da caricare.

Copia dell'URI dei file in un bucket S3

1. Accedi alla Console di gestione AWS e apri la console di Amazon S3 all'indirizzo https://console.aws.amazon.com/s3/.
2. Nel riquadro di navigazione, scegli Bucket, quindi scegli il bucket di cui desideri copiare l'URI.
3. Seleziona il prefisso o il file che desideri caricare da S3.
4. Scegli Copia URI S3.

LOAD DATA FROM S3

Puoi usare l'istruzione `LOAD DATA FROM S3` per caricare i dati da un qualsiasi formato di file di testo supportato dall'istruzione MySQL LOAD DATA INFILE, ad esempio i dati di testo delimitati da virgola. I file compressi non sono supportati.

> **Note**
>
> Assicurati che il cluster database Aurora MySQL consenta le connessioni in uscita a S3. Per ulteriori informazioni, consulta Abilitazione delle comunicazioni di rete da Amazon Aurora ad altri servizi AWS.

Sintassi

```
LOAD DATA [FROM] S3 [FILE | PREFIX | MANIFEST] 'S3-URI'
    [REPLACE | IGNORE]
    INTO TABLE tbl_name
    [PARTITION (partition_name,...)]
    [CHARACTER SET charset_name]
    [{FIELDS | COLUMNS}
        [TERMINATED BY 'string']
        [[OPTIONALLY] ENCLOSED BY 'char']
        [ESCAPED BY 'char']
    ]
    [LINES
        [STARTING BY 'string']
        [TERMINATED BY 'string']
    ]
    [IGNORE number {LINES | ROWS}]
    [(col_name_or_user_var,...)]
    [SET col_name = expr,...]
```

> **Note**
>
> In Aurora MySQL 3.05 e versioni successive, la parola chiave `FROM` è facoltativa.

Parametri

L'istruzione `LOAD DATA FROM S3` utilizza i seguenti parametri obbligatori e facoltativi. Altri dettagli su questi parametri sono disponibili in Istruzione LOAD DATA nella documentazione di MySQL.

FILE | PREFIX | MANIFEST

Indica se caricare i dati da un singolo file, da tutti i file che corrispondono a un determinato prefisso o da tutti i file in un manifesto specificato. `FILE` è il valore predefinito.

S3-URI

Specifica l'URI per un file di testo o manifesto da caricare o un prefisso Amazon S3 da utilizzare. Specificare l'URI usando la sintassi descritta in Specifica del percorso (URI) in un bucket Amazon S3.

REPLACE | IGNORE

Determina quale azione intraprendere se una riga di input ha gli stessi valori chiave univoci di una riga esistente nella tabella del database.

- Specifica `REPLACE` se la riga di input deve sostituire la riga esistente nella tabella.
- Specifica `IGNORE` se la riga di input deve essere scartata.

INTO TABLE

Identifica il nome della tabella del database in cui caricare le righe di input.

PARTITION

Richiede che tutte le righe di input vengano inserite nelle partizioni identificate dall'elenco specificato di nomi di partizione separati da virgola. Se una riga di input non può essere inserita in una delle partizioni specificate, l'istruzione non riesce e viene restituito un errore.

CHARACTER SET

Identifica il set di caratteri dei dati nel file di input.

FIELDS | COLUMNS

Identifica il modo in cui i campi o le colonne nel file di input sono delimitati. I campi sono delimitati da tabulazioni per impostazione predefinita.

LINES

Identifica il modo in cui le righe nel file di input sono delimitati. Le linee sono delimitate da un carattere di nuova riga (`'\n'`) per impostazione predefinita.

IGNORE *numero* LINES | ROWS

Specifica di ignorare un determinato numero di righe all'inizio del file di input. Ad esempio, è possibile utilizzare `IGNORE 1 LINES` per ignorare la riga di intestazione iniziale contenente i nomi di colonna o `IGNORE 2 ROWS` per ignorare le prime due righe di dati nel file di input. Se si utilizza anche `PREFIX`, `IGNORE` ignora un numero specifico di righe all'inizio del primo file di input.

col_name_or_user_var, ...

Specifica un elenco separato da virgola di uno o più nomi di colonne o variabili utente che identificano quali colonne caricare in base al nome. Il nome di una variabile utente utilizzata per questo scopo deve corrispondere al nome di un elemento del file di testo, preceduto da @. Puoi utilizzare le variabili utente per memorizzare i valori dei campi corrispondenti per un successivo riutilizzo.

Ad esempio, la seguente istruzione carica la prima colonna dal file di input nella prima colonna di `table1` e imposta il valore della colonna `table_column2` in `table1` sul valore di input della seconda colonna diviso per 100.

```
LOAD DATA FROM S3 's3://amzn-s3-demo-bucket/data.txt'
    INTO TABLE table1
    (column1, @var1)
    SET table_column2 = @var1/100;
```

SET

Specifica l'elenco delle operazioni di assegnazione separate da virgole che imposta i valori delle colonne della tabella sui valori non inclusi nel file di input.

Ad esempio, la seguente istruzione imposta le prime due colonne di `table1` sui valori delle prime due colonne del file di input e imposta il valore della colonna `column3` in `table1` sul timestamp corrente.

```
LOAD DATA FROM S3  's3://amzn-s3-demo-bucket/data.txt'
    INTO TABLE table1
    (column1, column2)
    SET column3 = CURRENT_TIMESTAMP;
```

È possibile utilizzare le sottoquery nella parte destra delle assegnazioni `SET`. Per una sottoquery che restituisce un valore da assegnare a una colonna, puoi utilizzare solo una sottoquery scalare. Inoltre, non è possibile utilizzare una sottoquery per selezionare dalla tabella che viene caricata.

Non puoi usare la parola chiave `LOCAL` dell'istruzione `LOAD DATA FROM S3` se stai caricando i dati da un bucket Amazon S3.

Utilizzo di un manifest per specificare i file di dati da caricare

È possibile utilizzare l'istruzione `LOAD DATA FROM S3` con la parola chiave `MANIFEST` per specificare un file manifest in formato JSON che elenca i file di testo da caricare in una tabella nel cluster DB.

Il seguente schema JSON descrive il formato e il contenuto di un file manifest.

```
{
    "$schema": "http://json-schema.org/draft-04/schema#",
    "additionalProperties": false,
    "definitions": {},
    "id": "Aurora_LoadFromS3_Manifest",
    "properties": {
        "entries": {
            "additionalItems": false,
            "id": "/properties/entries",
            "items": {
                "additionalProperties": false,
                "id": "/properties/entries/items",
                "properties": {
                    "mandatory": {
                        "default": "false",
                        "id": "/properties/entries/items/properties/mandatory",
                        "type": "boolean"
                    },
                    "url": {
                        "id": "/properties/entries/items/properties/url",
                        "maxLength": 1024,
                        "minLength": 1,
                        "type": "string"
                    }
                },
                "required": [
                    "url"
                ],
                "type": "object"
            },
            "type": "array",
            "uniqueItems": true
```

```
        }
    },
    "required": [
        "entries"
    ],
    "type": "object"
}
```

Ogni `url` nel manifest deve specificare un URL con il nome bucket e il percorso completo dell'oggetto per il file, non solo un prefisso. Puoi utilizzare un manifest per caricare file da diversi bucket, regioni o file che non condividono lo stesso prefisso. Se una regione non è specificata nell'URL, viene utilizzata la regione del cluster DB Aurora di destinazione. L'esempio seguente mostra un file manifest che carica quattro file da diversi bucket.

```
{
  "entries": [
    {
      "url":"s3://aurora-bucket/2013-10-04-customerdata",
      "mandatory":true
    },
    {
      "url":"s3-us-west-2://aurora-bucket-usw2/2013-10-05-customerdata",
      "mandatory":true
    },
    {
      "url":"s3://aurora-bucket/2013-10-04-customerdata",
      "mandatory":false
    },
    {
      "url":"s3://aurora-bucket/2013-10-05-customerdata"
    }
  ]
}
```

Il flag facoltativo `mandatory` specifica se `LOAD DATA FROM S3` deve restituire un errore qualora il file non venga trovato. L'impostazione predefinita del flag `mandatory` è `false`. Indipendentemente dall'impostazione di `mandatory`, `LOAD DATA FROM S3` termina se non viene trovato alcun file.

I file manifest possono avere qualsiasi estensione. L'esempio seguente esegue l'istruzione `LOAD DATA FROM S3` con manifest nell'esempio precedente, che viene denominato **`customer.manifest`**.

```
LOAD DATA FROM S3 MANIFEST 's3-us-west-2://aurora-bucket/customer.manifest'
    INTO TABLE CUSTOMER
    FIELDS TERMINATED BY ','
    LINES TERMINATED BY '\n'
    (ID, FIRSTNAME, LASTNAME, EMAIL);
```

Al termine dell'istruzione, per ogni file caricato viene scritta una voce nella tabella `aurora_s3_load_history`.

Verifica dei file caricati utilizzando la tabella aurora_s3_load_history

Ogni istruzione `LOAD DATA FROM S3` con esito positivo aggiorna la tabella `aurora_s3_load_history` nello schema `mysql` con una voce per ogni file che è stato caricato.

Dopo aver eseguito l'istruzione `LOAD DATA FROM S3`, puoi verificare quali file sono stati caricati eseguendo una query sulla tabella `aurora_s3_load_history`. Per visualizzare i file caricati da un'iterazione dell'istruzione, utilizza la clausola `WHERE` per filtrare i record sull'URI Amazon S3 per il file manifest utilizzato nell'istruzione. Se hai utilizzato lo stesso file manifest di prima, filtra i risultati utilizzando il campo `timestamp`.

```
select * from mysql.aurora_s3_load_history where load_prefix = 'S3_URI';
```

La tabella seguente descrive i campi della tabella `aurora_s3_load_history`.

Campo	Descrizione
`load_prefix`	L'URI specificato nell'istruzione di caricamento. L'URI può mappare a uno dei seguenti elementi: • Un singolo file di dati per un'istruzione `LOAD DATA FROM S3 FILE` • Un prefisso Amazon S3 che mappa a più file di dati per un'istruzione `LOAD DATA FROM S3 PREFIX` • Un singolo file manifest contenente i nomi dei file da caricare per un'istruzione `LOAD DATA FROM S3 MANIFEST`
`file_name`	Il nome di un file che è stato caricato in Aurora da Amazon S3 utilizzando l'URI identificato nel campo `load_prefix` .

Campo	Descrizione
`version_number`	Il numero di versione del file identificato dal campo `file_name` che è stato caricato se il bucket Amazon S3 ha un numero di versione.
`bytes_loaded`	Le dimensioni del file caricato in byte.
`load_timestamp`	Timestamp relativo al momento del completamento dell'istruzione `LOAD DATA FROM S3`.

Esempi

La seguente istruzione carica i dati da un bucket Amazon S3 che si trova nella stessa regione del cluster DB Aurora. L'istruzione legge i dati delimitati da virgole nel file `customerdata.txt` che si trova nel bucket Amazon S3 *`amzn-s3-demo-bucket`* e poi carica i dati nella tabella `store-schema.customer-table`.

```
LOAD DATA FROM S3 's3://amzn-s3-demo-bucket/customerdata.csv'
    INTO TABLE store-schema.customer-table
    FIELDS TERMINATED BY ','
    LINES TERMINATED BY '\n'
    (ID, FIRSTNAME, LASTNAME, ADDRESS, EMAIL, PHONE);
```

La seguente istruzione carica i dati da un bucket Amazon S3 che si trova in una regione diversa rispetto al cluster DB Aurora. L'istruzione legge i dati delimitati da virgole da tutti i file che corrispondono al prefisso di oggetto `employee-data` nel bucket Amazon S3 *`amzn-s3-demo-bucket`* nella Regione `us-west-2` e poi carica i dati nella tabella `employees`.

```
LOAD DATA FROM S3 PREFIX 's3-us-west-2://amzn-s3-demo-bucket/employee_data'
    INTO TABLE employees
    FIELDS TERMINATED BY ','
    LINES TERMINATED BY '\n'
    (ID, FIRSTNAME, LASTNAME, EMAIL, SALARY);
```

La seguente istruzione carica i dati dai file specificati in un file manifest JSON denominato q1_sales.json nella tabella `sales`.

```
LOAD DATA FROM S3 MANIFEST 's3-us-west-2://amzn-s3-demo-bucket1/q1_sales.json'
```

```
    INTO TABLE sales
    FIELDS TERMINATED BY ','
    LINES TERMINATED BY '\n'
    (MONTH, STORE, GROSS, NET);
```

LOAD XML FROM S3

Puoi usare l'istruzione `LOAD XML FROM S3` per caricare i dati dai file XML memorizzati in un bucket Amazon S3 in uno dei seguenti tre diversi formati XML:

- I nomi di colonna come attributi di un elemento `<row>`. Il valore dell'attributo identifica il contenuto del campo della tabella.

  ```
  <row column1="value1" column2="value2" .../>
  ```

- I nomi di colonna come elementi figlio di un elemento `<row>`. Il valore dell'elemento figlio identifica il contenuto del campo della tabella.

  ```
  <row>
    <column1>value1</column1>
    <column2>value2</column2>
  </row>
  ```

- I nomi di colonna nell'attributo `name` dell'elemento `<field>` in un elemento `<row>`. Il valore dell'elemento `<field>` identifica il contenuto del campo della tabella.

  ```
  <row>
    <field name='column1'>value1</field>
    <field name='column2'>value2</field>
  </row>
  ```

Sintassi

```
LOAD XML FROM S3 'S3-URI'
    [REPLACE | IGNORE]
    INTO TABLE tbl_name
    [PARTITION (partition_name,...)]
    [CHARACTER SET charset_name]
    [ROWS IDENTIFIED BY '<element-name>']
    [IGNORE number {LINES | ROWS}]
```

```
    [(field_name_or_user_var,...)]
    [SET col_name = expr,...]
```

Parametri

L'istruzione `LOAD XML FROM S3` utilizza i seguenti parametri obbligatori e facoltativi. Altri dettagli su questi parametri sono disponibili in Istruzione LOAD XML nella documentazione di MySQL.

FILE | PREFIX

Indica se caricare i dati da un singolo file o da tutti i file che corrispondono a un determinato prefisso. `FILE` è il valore predefinito.

REPLACE | IGNORE

Determina quale azione intraprendere se una riga di input ha gli stessi valori chiave univoci di una riga esistente nella tabella del database.

- Specifica `REPLACE` se la riga di input deve sostituire la riga esistente nella tabella.
- Specifica `IGNORE` per ignorare la riga di input. `IGNORE` è il valore predefinito.

INTO TABLE

Identifica il nome della tabella del database in cui caricare le righe di input.

PARTITION

Richiede che tutte le righe di input vengano inserite nelle partizioni identificate dall'elenco specificato di nomi di partizione separati da virgola. Se una riga di input non può essere inserita in una delle partizioni specificate, l'istruzione non riesce e viene restituito un errore.

CHARACTER SET

Identifica il set di caratteri dei dati nel file di input.

ROWS IDENTIFIED BY

Identifica il nome dell'elemento che identifica una riga nel file di input. Il valore predefinito è `<row>`.

IGNORE ***numero*** LINES | ROWS

Specifica di ignorare un determinato numero di righe all'inizio del file di input. Ad esempio, è possibile utilizzare `IGNORE 1 LINES` per ignorare la prima riga del file di testo o `IGNORE 2 ROWS` per ignorare le prime due righe di dati nell'XML di input.

field_name_or_user_var, ...

Specifica un elenco separato da virgola di uno o più nomi di elementi XML o variabili utente che identificano quali elementi caricare in base al nome. Il nome di una variabile utente utilizzata per questo scopo deve corrispondere al nome di un elemento del file XML, preceduto da @. Puoi utilizzare le variabili utente per memorizzare i valori dei campi corrispondenti per un successivo riutilizzo.

Ad esempio, la seguente istruzione carica la prima colonna dal file di input nella prima colonna di `table1` e imposta il valore della colonna `table_column2` in `table1` sul valore di input della seconda colonna diviso per 100.

```
LOAD XML FROM S3 's3://amzn-s3-demo-bucket/data.xml'
   INTO TABLE table1
   (column1, @var1)
   SET table_column2 = @var1/100;
```

SET

Specifica l'elenco delle operazioni di assegnazione separate da virgole che imposta i valori delle colonne della tabella sui valori non inclusi nel file di input.

Ad esempio, la seguente istruzione imposta le prime due colonne di `table1` sui valori delle prime due colonne del file di input e imposta il valore della colonna `column3` in `table1` sul timestamp corrente.

```
LOAD XML FROM S3 's3://amzn-s3-demo-bucket/data.xml'
   INTO TABLE table1
   (column1, column2)
   SET column3 = CURRENT_TIMESTAMP;
```

È possibile utilizzare le sottoquery nella parte destra delle assegnazioni `SET`. Per una sottoquery che restituisce un valore da assegnare a una colonna, puoi utilizzare solo una sottoquery scalare. Inoltre, non è possibile utilizzare una sottoquery per eseguire selezioni nella tabella caricata.

Salvataggio dei dati da un cluster DB Amazon Aurora MySQL nei file di testo in un bucket Amazon S3

Puoi utilizzare l'istruzione `SELECT INTO OUTFILE S3` per eseguire una query sui dati da un cluster di database Amazon Aurora MySQL ed eseguirne il salvataggio nei file di testo archiviati in un bucket Amazon S3. In Aurora MySQL, i file vengono prima archiviati sul disco locale e quindi esportati in S3. Al termine delle esportazioni, i file locali vengono eliminati.

Puoi crittografare il bucket Amazon S3 utilizzando una chiave gestita da Amazon S3 (SSE-S3) o una AWS KMS key (SSE-KMS: Chiave gestita da AWS o chiave gestita dal cliente).

L'istruzione `LOAD DATA FROM S3` può utilizzare i file creati dall'istruzione `SELECT INTO OUTFILE S3` per caricare i dati in un cluster di database Aurora. Per ulteriori informazioni, consulta Caricamento dei dati in un cluster DB Amazon Aurora MySQL da file di testo in un bucket Amazon S3.

> **Note**
>
> Questa funzionalità non è supportata per i cluster di database Aurora Serverless v1. È supportata per i cluster di database Aurora Serverless v2.
> È anche possibile salvare i dati del cluster di database e dello snapshot del cluster di database in Amazon S3 utilizzando la Console di gestione AWS, l'interfaccia AWS CLI o l'API di Amazon RDS. Per ulteriori informazioni, consultare Esportazione dei dati del cluster database in Amazon S3 e Esportazione dei dati dello snapshot del cluster di database in Amazon S3.

Indice

- Accesso di Aurora MySQL a Amazon S3;
- Concessione dei privilegi per salvare dati in Aurora MySQL
- Specifica di un percorso in un bucket Amazon S3
- Creazione di un manifest per elencare i fili di dati
- SELECT INTO OUTFILE S3
  - Sintassi
  - Parametri
  - Considerazioni
  - Esempi

Accesso di Aurora MySQL a Amazon S3;

Prima di poter salvare i dati in un bucket Amazon S3, è necessario innanzitutto concedere l'autorizzazione al cluster DB Aurora MySQL per accedere ad Amazon S3.

Per concedere ad Aurora MySQL l'accesso ad Amazon S3

1. Creare una policy AWS Identity and Access Management (IAM) che assegni le autorizzazioni bucket e di oggetto che rendono possibile l'accesso del cluster di database Aurora MySQL ad Amazon S3. Per istruzioni, consultare Creazione di una policy IAM per l'accesso alle risorse Amazon S3.

   > **Note**
   >
   > In Aurora MySQL 3.05 e versioni successive, è possibile crittografare oggetti utilizzando chiavi AWS KMS gestite dal cliente. A tale scopo, includi l'autorizzazione `kms:GenerateDataKey` nella policy IAM. Per ulteriori informazioni, consulta Creazione di una policy IAM per accedere alle AWS KMS risorse.
   > Non è necessaria questa autorizzazione per caricare oggetti crittografati utilizzando Chiavi gestite da AWS o chiavi gestite da Amazon S3 (SSE-S3).

2. Creare un ruolo IAM e collegare la policy IAM creata in Creazione di una policy IAM per l'accesso alle risorse Amazon S3 al nuovo ruolo IAM. Per istruzioni, consultare Creazione di un ruolo IAM per consentire ad Amazon Aurora di accedere ai servizi AWS.

3. Per Aurora MySQL versione 2, imposta il parametro del cluster di database `aurora_select_into_s3_role` o `aws_default_s3_role` sul nome della risorsa Amazon (ARN) del nuovo ruolo IAM. Se un ruolo IAM non è specificato per `aurora_select_into_s3_role`, Aurora utilizza il ruolo IAM specificato in `aws_default_s3_role`.

   Per Aurora MySQL versione 3, utilizza `aws_default_s3_role`.

   Se il cluster fa parte di un database globale Aurora, imposta questo parametro per ogni cluster Aurora nel database globale.

   Per ulteriori informazioni sui parametri del cluster DB, vedi Parametri dell'istanza database e del cluster database di Amazon Aurora.

4. Per consentire l'accesso ad Aurora MySQL agli utenti del database in un cluster DB Amazon S3, è necessario associare il ruolo creato in Creazione di un ruolo IAM per consentire ad Amazon Aurora di accedere ai servizi AWS al cluster DB.

   Per un database globale Aurora, associa il ruolo a ogni cluster Aurora nel database globale.

   Per informazioni su come associare un ruolo IAM a un cluster DB, vedi Associazione di un ruolo IAM a un cluster DB Amazon Aurora MySQL.

5. Configura il cluster DB Aurora MySQL per consentire le connessioni in uscita ad Amazon S3. Per istruzioni, consulta Abilitazione delle comunicazioni di rete da Amazon Aurora ad altri servizi AWS.

   Per un database globale Aurora, abilita le connessioni in uscita per ogni cluster Aurora nel database globale.

Concessione dei privilegi per salvare dati in Aurora MySQL

L'utente del database che emette l'istruzione `SELECT INTO OUTFILE S3` deve avere un ruolo o un privilegio specifico. In Aurora MySQL versione 3, concedi il ruolo `AWS_SELECT_S3_ACCESS`. In Aurora MySQL versione 2, concedi il privilegio `SELECT INTO S3`. All'utente amministrativo per un cluster di database è concesso il ruolo o il privilegio appropriato per impostazione predefinita. Puoi concedere il privilegio a un altro utente usando le seguenti istruzioni.

Utilizzare la seguente istruzione per Aurora MySQL versione 3:

```
GRANT AWS_SELECT_S3_ACCESS TO 'user'@'domain-or-ip-address'
```

> **Tip**
>
> Quando utilizzi la tecnica basata sul ruolo in Aurora MySQL versione 3, puoi anche attivare il ruolo utilizzando l'istruzione `SET ROLE role_name` o `SET ROLE ALL`. Se non si ha familiarità con il sistema dei ruoli MySQL 8.0, è possibile ottenere ulteriori informazioni in Privilegio basato sui ruoli. Per maggiori dettagli, consulta la pagina Using roles nel Manuale di riferimento di MySQL.
> Questo vale solo per la sessione attiva corrente. Quando ti riconnetti, devi eseguire di nuovo l'istruzione `SET ROLE` per concedere i privilegi. Per ulteriori informazioni, consulta Istruzione SET ROLE nel Manuale di riferimento di MySQL.

Puoi utilizzare il parametro `activate_all_roles_on_login` del cluster di database per attivare automaticamente tutti i ruoli quando un utente si connette a un'istanza database. Quando questo parametro è impostato, non è necessario chiamare esplicitamente l'istruzione `SET ROLE` per attivare un ruolo. Per ulteriori informazioni, consulta activate_all_roles_on_login nel Manuale di riferimento di MySQL.
Tuttavia, è necessario chiamare esplicitamente `SET ROLE ALL` all'inizio di una stored procedure per attivare il ruolo, quando la stored procedure viene chiamata da un altro utente.

Utilizza la seguente istruzione per Aurora MySQL versione 2:

```
GRANT SELECT INTO S3 ON *.* TO 'user'@'domain-or-ip-address'
```

Il ruolo `AWS_SELECT_S3_ACCESS` e privilegio `SELECT INTO S3` sono specifici di Amazon Aurora MySQL e non sono disponibili per i database MySQL o le istanze database RDS per MySQL. Se è stata configurata la replica tra un cluster di database Aurora MySQL come origine di replica e un database MySQL come client di replica, l'istruzione `GRANT` per il ruolo o il privilegio causa l'arresto della replica con un errore. Puoi ignorare l'errore in modo sicuro per riprendere la replica. Per ignorare l'errore su un'istanza database RDS for MySQL, utilizzare la procedura mysql_rds_skip_repl_error. Per ignorare l'errore su un database MySQL esterno, usa la variabile di sistema slave_skip_errors (Aurora MySQL versione 2) o replica_skip_errors (Aurora MySQL versione 3).

Specifica di un percorso in un bucket Amazon S3

La sintassi per specificare un percorso per archiviare i file manifest e di dati in un bucket Amazon S3 è simile a quella utilizzata nell'istruzione `LOAD DATA FROM S3 PREFIX`, come mostrato di seguito.

```
s3-region://bucket-name/file-prefix
```

Il percorso include i seguenti valori:

- `region` (facoltativo): la regione AWS che contiene il bucket Amazon S3 in cui salvare i dati. Questo valore è facoltativo. Se non specifichi un valore per `region`, Aurora salva i file in Amazon S3 nella stessa regione del cluster DB.
- `bucket-name` – Il nome del bucket Amazon S3 in cui salvare i dati. Sono supportati i prefissi degli oggetti che identificano un percorso di cartella virtuale.
- `file-prefix` – Il prefisso dell'oggetto Amazon S3 che identifica i file da salvare in Amazon S3.

I file di dati creati dall'istruzione `SELECT INTO OUTFILE S3` utilizzano il seguente percorso, in cui *`00000`* rappresenta un numero intero a 5 cifre, a base zero.

```
s3-region://bucket-name/file-prefix.part_00000
```

Ad esempio, supponiamo che un'istruzione `SELECT INTO OUTFILE S3` specifichi `s3-us-west-2://bucket/prefix` come percorso in cui archiviare i file di dati e crea tre file di dati. Il bucket Amazon S3 specificato contiene i seguenti file di dati.

- s3-us-west-2://bucket/prefix.part_00000
- s3-us-west-2://bucket/prefix.part_00001
- s3-us-west-2://bucket/prefix.part_00002

Creazione di un manifest per elencare i fili di dati

È possibile utilizzare l'istruzione `SELECT INTO OUTFILE S3` con l'opzione `MANIFEST ON` per creare un file manifest in formato JSON che elenca i file di testo creati dall'istruzione. L'istruzione `LOAD DATA FROM S3` può utilizzare il file manifest per caricare i file di dati di nuovo in un cluster DB Aurora MySQL. Per ulteriori informazioni sull'utilizzo di un file manifest per caricare file di dati da Amazon S3 in un cluster DB Aurora MySQL, consulta Utilizzo di un manifest per specificare i file di dati da caricare.

I file di dati inclusi nel manifest creati dall'istruzione `SELECT INTO OUTFILE S3` sono elencati nell'ordine in cui vengono creati dall'istruzione. Ad esempio, supponiamo che un'istruzione `SELECT INTO OUTFILE S3` specifichi `s3-us-west-2://bucket/prefix` come percorso in cui archiviare i file di dati e crea tre file di dati e un file manifest. Il bucket Amazon S3 specificato contiene un file manifest denominato `s3-us-west-2://bucket/prefix.manifest` che contiene le seguenti informazioni.

```
{
  "entries": [
    {
      "url":"s3-us-west-2://bucket/prefix.part_00000"
    },
    {
      "url":"s3-us-west-2://bucket/prefix.part_00001"
    },
    {
```

```
      "url":"s3-us-west-2://bucket/prefix.part_00002"
    }
  ]
}
```

SELECT INTO OUTFILE S3

Puoi usare l'istruzione `SELECT INTO OUTFILE S3` per eseguire una query sui dati da un cluster DB e salvarlo direttamente nei file di testo delimitato archiviati in un bucket Amazon S3.

I file compressi non sono supportati. I file crittografati sono supportati a partire da Aurora MySQL versione 2.09.0.

Sintassi

```
SELECT
    [ALL | DISTINCT | DISTINCTROW ]
        [HIGH_PRIORITY]
        [STRAIGHT_JOIN]
        [SQL_SMALL_RESULT] [SQL_BIG_RESULT] [SQL_BUFFER_RESULT]
        [SQL_CACHE | SQL_NO_CACHE] [SQL_CALC_FOUND_ROWS]
    select_expr [, select_expr ...]
    [FROM table_references
        [PARTITION partition_list]
    [WHERE where_condition]
    [GROUP BY {col_name | expr | position}
        [ASC | DESC], ... [WITH ROLLUP]]
    [HAVING where_condition]
    [ORDER BY {col_name | expr | position}
         [ASC | DESC], ...]
    [LIMIT {[offset,] row_count | row_count OFFSET offset}]
INTO OUTFILE S3 's3_uri'
[CHARACTER SET charset_name]
    [export_options]
    [MANIFEST {ON | OFF}]
    [OVERWRITE {ON | OFF}]
    [ENCRYPTION {ON | OFF | SSE_S3 | SSE_KMS ['cmk_id']}]

export_options:
    [FORMAT {CSV|TEXT} [HEADER]]
    [{FIELDS | COLUMNS}
        [TERMINATED BY 'string']
        [[OPTIONALLY] ENCLOSED BY 'char']
```

```
        [ESCAPED BY 'char']
    ]
    [LINES
        [STARTING BY 'string']
        [TERMINATED BY 'string']
]
```

Parametri

L'istruzione `SELECT INTO OUTFILE S3` utilizza i seguenti parametri obbligatori e facoltativi specifici per Aurora.

s3-uri

Specifica l'URI per un prefisso Amazon S3 da utilizzare. Utilizza la sintassi descritta in Specifica di un percorso in un bucket Amazon S3.

FORMAT {CSV|TEXT} [HEADER]

Facoltativamente salva i dati in formato CSV.

L'opzione `TEXT` è quella predefinita e produce il formato di esportazione MySQL esistente.

L'opzione `CSV` genera valori di dati separati da virgole. Il formato CSV segue le specifiche nella RFC-4180. Se si specifica la parola chiave facoltativa `HEADER`, il file di output contiene una riga di intestazione. Le etichette nella riga di intestazione corrispondono ai nomi di colonna dell'istruzione `SELECT`. È possibile utilizzare i file CSV per modelli di dati di training da utilizzare con i servizi AWS ML. Per ulteriori informazioni sull'uso dei dati Aurora esportati con i servizi ML di AWS, consulta Esportazione dei dati in Amazon S3 per l'addestramento dei modelli SageMaker AI (avanzato).

MANIFEST {ON | OFF}

Indica se un file manifesto viene creato in Amazon S3. Il file manifest è un file JSON (JavaScript Object Notation) che può essere utilizzato per caricare dati in un cluster DB Aurora con l'istruzione `LOAD DATA FROM S3 MANIFEST`. Per ulteriori informazioni su `LOAD DATA FROM S3 MANIFEST`, consulta Caricamento dei dati in un cluster DB Amazon Aurora MySQL da file di testo in un bucket Amazon S3.

Se `MANIFEST ON` è specificato nella query, il file manifest viene creato in Amazon S3 dopo che tutti i file di dati sono stati creati e caricati. Il file manifest viene creato utilizzando il seguente percorso:

```
s3-region://bucket-name/file-prefix.manifest
```

Per ulteriori informazioni sul formato del contenuto del file manifest, consulta Creazione di un manifest per elencare i fili di dati.

OVERWRITE {ON | OFF}

Indica se i file esistenti nel bucket Amazon S3 specificato vengono sovrascritti. Se `OVERWRITE ON` è specificato, i file esistenti che corrispondono al prefisso del file nell'URI specificato in `s3-uri` vengono sovrascritti. In caso contrario, si verifica un errore.

ENCRYPTION {ON | OFF | SSE_S3 | SSE_KMS ['*cmk_id*']}

Indica se utilizzare la crittografia lato server con chiavi gestite da Amazon S3 (SSE-S3) o AWS KMS keys (SSE-KMS: incluse Chiavi gestite da AWS o chiavi gestite dal cliente). Le impostazioni `SSE_S3` e `SSE_KMS` sono disponibili in Aurora MySQL 3.05 e versioni successive.

Puoi anche utilizzare la variabile di sessione `aurora_select_into_s3_encryption_default` anziché la clausola `ENCRYPTION`, come illustrato nell'esempio seguente. Usa la clausola SQL o la variabile di sessione, ma non entrambe.

```
set session set session aurora_select_into_s3_encryption_default={ON | OFF | SSE_S3
 | SSE_KMS};
```

Le impostazioni `SSE_S3` e `SSE_KMS` sono disponibili in Aurora MySQL 3.05 e versioni successive.

Impostando `aurora_select_into_s3_encryption_default` sul seguente valore:

- `OFF`: viene seguita la policy di crittografia predefinita del bucket S3. Il valore predefinito di `aurora_select_into_s3_encryption_default` è `OFF`.
- `ON` oppure `SSE_S3`: l'oggetto S3 viene crittografato tramite chiavi gestite da Amazon S3 (SSE-S3).
- `SSE_KMS`: l'oggetto S3 viene crittografato utilizzando una AWS KMS key.

  In questo caso, includi anche la variabile di sessione `aurora_s3_default_cmk_id`, ad esempio:

  ```
  set session aurora_select_into_s3_encryption_default={SSE_KMS};
  set session aurora_s3_default_cmk_id={NULL | 'cmk_id'};
  ```

- Quando `aurora_s3_default_cmk_id` è `NULL`, l'oggetto S3 viene crittografato utilizzando una Chiave gestita da AWS.
- Quando `aurora_s3_default_cmk_id` è una stringa `cmk_id` non vuota, l'oggetto S3 viene crittografato utilizzando una chiave gestita dal cliente.

  Il valore di `cmk_id` non può essere una stringa vuota.

Quando utilizzi il comando `SELECT INTO OUTFILE S3`, Aurora determina la crittografia come segue:

- Se nel comando SQL è inclusa la clausola `ENCRYPTION`, Aurora si basa solo sul valore di `ENCRYPTION` e non utilizza una variabile di sessione.
- Se la clausola `ENCRYPTION` non è presente, Aurora si basa sul valore della variabile di sessione.

Per ulteriori informazioni, consulta Uso della crittografia lato server con chiavi gestite da Amazon S3 (SSE-S3) e Utilizzo della crittografia lato server con chiavi AWS KMS (SSE-KMS) nella Guida per l'utente di Amazon Simple Storage Service.

Altri dettagli su questi parametri sono disponibili in Istruzione SELECT e Istruzione LOAD DATA nella documentazione di MySQL.

Considerazioni

Il numero di file scritti nel bucket Amazon S3 dipende dalla quantità di dati selezionati dall'istruzione `SELECT INTO OUTFILE S3` e dalla soglia della dimensione del file per Aurora MySQL. La soglia della dimensione del file predefinita è 6 gigabyte (GB). Se i dati selezionati dall'istruzione sono inferiori alla soglia della dimensione del file, viene creato un singolo file; in caso contrario, vengono creati più file. Altre considerazioni per i file creati da questa dichiarazione includono quanto segue:

- Aurora MySQL garantisce che le righe nei file di dati non siano suddivise tra i limiti dei file. Nel caso di più file, la dimensione di ogni file di dati tranne l'ultimo è in genere vicino alla soglia della dimensione del file. Tuttavia, ogni tanto poiché restano sotto la soglia della dimensione del file, una riga viene suddivisa su due file di dati. In questo caso, Aurora MySQL crea un file di dati che mantiene intatta la riga, ma potrebbe essere più grande della soglia della dimensione del file.
- Perché ogni istruzione `SELECT` in Aurora MySQL viene eseguita come una transazione atomica, un'istruzione `SELECT INTO OUTFILE S3` che seleziona un set di dati di grandi dimensioni potrebbe essere eseguita per qualche tempo. Se l'istruzione non riesce per qualsiasi motivo,

potrebbe essere necessario ricominciare ed emettere nuovamente l'istruzione. Se l'istruzione non riesce, tuttavia, i file che sono già stati caricati in Amazon S3 rimangono nel bucket Amazon S3 specifico. Puoi utilizzare un'altra istruzione per caricare i dati rimanenti anziché ricominciare da capo.

- Se la quantità di dati da selezionare è elevata (oltre 25 GB), ti consigliamo di utilizzare più istruzioni `SELECT INTO OUTFILE S3` per salvare i dati in Amazon S3. Ogni istruzione dovrebbe selezionare una parte diversa dei dati da salvare e anche specificare un diverso `file_prefix` nel parametro `s3-uri` da utilizzare quando si salvano i file di dati. Il partizionamento dei dati da selezionare con più istruzioni facilita il recupero da un errore in un'istruzione. Se si verifica un errore per un'istruzione, solo una parte dei dati deve essere riselezionata e caricata in Amazon S3. L'utilizzo di più istruzioni aiuta anche a evitare una singola transazione di lunga durata, che può migliorare le prestazioni.
- Se più istruzioni `SELECT INTO OUTFILE S3` che utilizzano lo stesso `file_prefix` nel parametro `s3-uri` vengono eseguite in parallelo per selezionare i dati in Amazon S3, il comportamento non è definito.
- I metadati, come lo schema della tabella o i metadati dei file, non vengono caricati da Aurora MySQL in Amazon S3.
- In alcuni casi, potresti rieseguire una query `SELECT INTO OUTFILE S3`, ad esempio per ripristinare a causa di un errore. In questi casi, devi rimuovere tutti i file di dati esistenti nel bucket Amazon S3 con lo stesso prefisso di file specificato in `s3-uri` o includere `OVERWRITE ON` nella query `SELECT INTO OUTFILE S3`.

L'istruzione `SELECT INTO OUTFILE S3` restituisce un tipico numero di errore MySQL e la risposta in caso di esito positivo o negativo. Se non hai accesso al numero di errore e alla risposta di MySQL, il modo più semplice per determinare quando termina l'esecuzione è specificare `MANIFEST ON` nell'istruzione. Il file manifest è l'ultimo file scritto dall'istruzione. In altre parole, se hai un file manifest, l'istruzione ha completato.

Al momento non è possibile monitorare direttamente l'avanzamento dell'istruzione `SELECT INTO OUTFILE S3` durante l'esecuzione. Tuttavia, supponiamo di scrivere una grande quantità di dati da Aurora MySQL in Amazon S3 usando questa istruzione e di conoscere la dimensione dei dati selezionati dall'istruzione. In questo caso, puoi stimare l'avanzamento monitorando la creazione di file di dati in Amazon S3.

Per fare ciò, è possibile utilizzare il fatto che un file di dati viene creato nel bucket Amazon S3 specificato ogni circa 6 GB di dati selezionati dall'istruzione. Dividi la dimensione dei dati

selezionati da 6 GB per ottenere il numero stimato di file di dati da creare. È quindi possibile stimare l'avanzamento dell'istruzione monitorando il numero di file caricati in Amazon S3 durante l'esecuzione.

Esempi

La seguente istruzione seleziona tutti i dati nella tabella `employees` e li salva in un bucket Amazon S3 che si trova in una regione diversa dal cluster DB Aurora MySQL. L'istruzione crea file di dati in cui ogni campo è terminato da una virgola (`,`) e ogni riga è terminata da un carattere di nuova riga (`\n`). L'istruzione restituisce un errore se i file che corrispondono al prefisso del file `sample_employee_data` esistono nel bucket Amazon S3 specificato.

```
SELECT * FROM employees INTO OUTFILE S3 's3-us-west-2://aurora-select-into-s3-pdx/
sample_employee_data'
    FIELDS TERMINATED BY ','
    LINES TERMINATED BY '\n';
```

La seguente istruzione seleziona tutti i dati nella tabella `employees` e li salva in un bucket Amazon S3 che si trova nella stessa regione del cluster DB Aurora MySQL. L'istruzione crea file di dati in cui ogni campo è terminato da una virgola (`,`) e ogni riga è terminata da un carattere di nuova riga (`\n`) e un file manifest. L'istruzione restituisce un errore se i file che corrispondono al prefisso del file `sample_employee_data` esistono nel bucket Amazon S3 specificato.

```
SELECT * FROM employees INTO OUTFILE S3 's3://aurora-select-into-s3-pdx/
sample_employee_data'
    FIELDS TERMINATED BY ','
    LINES TERMINATED BY '\n'
    MANIFEST ON;
```

La seguente istruzione seleziona tutti i dati nella tabella `employees` e li salva in un bucket Amazon S3 che si trova in una regione diversa dal cluster DB Aurora. L'istruzione crea file di dati in cui ogni campo è terminato da una virgola (`,`) e ogni riga è terminata da un carattere di nuova riga (`\n`). L'istruzione sostituisce ogni file che corrisponde al prefisso del file `sample_employee_data` esistono nel bucket Amazon S3 specificato.

```
SELECT * FROM employees INTO OUTFILE S3 's3-us-west-2://aurora-select-into-s3-pdx/
sample_employee_data'
    FIELDS TERMINATED BY ','
    LINES TERMINATED BY '\n'
```

```
    OVERWRITE ON;
```

La seguente istruzione seleziona tutti i dati nella tabella `employees` e li salva in un bucket Amazon S3 che si trova nella stessa regione del cluster DB Aurora MySQL. L'istruzione crea file di dati in cui ogni campo è terminato da una virgola (`,`) e ogni riga è terminata da un carattere di nuova riga (`\n`) e un file manifest. L'istruzione sostituisce ogni file che corrisponde al prefisso del file `sample_employee_data` esistono nel bucket Amazon S3 specificato.

```
SELECT * FROM employees INTO OUTFILE S3 's3://aurora-select-into-s3-pdx/
sample_employee_data'
    FIELDS TERMINATED BY ','
    LINES TERMINATED BY '\n'
    MANIFEST ON
    OVERWRITE ON;
```

Chiamare una funzione Lambda da un cluster DB Amazon Aurora MySQL

Puoi chiamare una funzione AWS Lambda da un cluster database Amazon Aurora edizione compatibile con MySQL utilizzando una funzione nativa `lambda_sync` o `lambda_async`. Prima di chiamare una funzione Lambda da Aurora MySQL, il un cluster DB Aurora deve avere l'accesso a Lambda. Per dettagli sulla concessione dell'accesso ad Aurora MySQL, consultare Accesso di Aurora a Lambda;. Per informazioni sulla `lambda_sync` e su funzioni archiviate `lambda_async`, consultare Chiamare una funzione Lambda con una funzione nativa Aurora MySQL.

Puoi anche chiamare una funzione AWS Lambda utilizzando una procedura archiviata. Tuttavia, l'utilizzo di una procedura archiviate è obsoleto. Ti consigliamo vivamente di usare una funzione nativa Aurora MySQL se utilizzi una delle seguenti versioni di Aurora MySQL:

- Aurora MySQL versione 2, per cluster compatibili con MySQL 5.7.
- Aurora MySQL versione 3.01 e successive, per cluster compatibili con MySQL 8.0. La stored procedure non è disponibile in Aurora MySQL versione 3.

Per informazioni su come concedere ad Aurora l’accesso a Lambda e come invocare una funzione Lambda, consulta i seguenti argomenti.

Argomenti

- Accesso di Aurora a Lambda;
- Chiamare una funzione Lambda con una funzione nativa Aurora MySQL

- Chiamare una funzione Lambda con una procedura archiviata Aurora MySQL (obsoleto)

Accesso di Aurora a Lambda;

Prima di poter chiamare le funzioni Lambda da un cluster database Aurora MySQL, assicurarsi innanzitutto di concedere l'autorizzazione al cluster per accedere a Lambda.

Per concedere ad Aurora MySQL l'accesso ad Lambda

1. Creare una policy AWS Identity and Access Management (IAM) che assegni le autorizzazioni che permettono al cluster database Aurora MySQL di richiamare le funzioni Lambda. Per istruzioni, consulta Creazione di una policy IAM per accedere alle AWS Lambda risorse.
2. Creare un ruolo IAM e collegare la policy IAM creata in Creazione di una policy IAM per accedere alle AWS Lambda risorse al nuovo ruolo IAM. Per istruzioni, consulta Creazione di un ruolo IAM per consentire ad Amazon Aurora di accedere ai servizi AWS.
3. Impostare il parametro `aws_default_lambda_role` del cluster DB sull'Amazon Resource Name (ARN) del nuovo ruolo IAM.

   Se il cluster fa parte di un database globale Aurora, applica la stessa impostazione per ogni cluster Aurora nel database globale.

   Per ulteriori informazioni sui parametri del cluster DB, vedi Parametri dell'istanza database e del cluster database di Amazon Aurora.
4. Per consentire di chiamare le funzioni Lambda agli utenti del database in un cluster DB Aurora MySQL, è necessario associare il ruolo creato in Creazione di un ruolo IAM per consentire ad Amazon Aurora di accedere ai servizi AWS con il cluster DB. Per informazioni su come associare un ruolo IAM a un cluster DB, vedi Associazione di un ruolo IAM a un cluster DB Amazon Aurora MySQL.

   Se il cluster fa parte di un database globale Aurora, associa il ruolo a ogni cluster Aurora nel database globale.
5. Configura il cluster DB Aurora MySQL per consentire le connessioni in uscita ad Lambda. Per istruzioni, consulta Abilitazione delle comunicazioni di rete da Amazon Aurora ad altri servizi AWS.

   Se il cluster fa parte di un database globale Aurora, abilita le connessioni in uscita per ogni cluster Aurora nel database globale.

Chiamare una funzione Lambda con una funzione nativa Aurora MySQL

> **Note**
>
> È possibile chiamare le funzioni native `lambda_sync` e `lambda_async` con Aurora MySQL versione 2 o Aurora MySQL versione 3.01 e versioni successive. Per ulteriori informazioni sulle versioni di Aurora MySQL, consulta Aggiornamenti del motore del database per Amazon Aurora MySQL.

Puoi chiamare una funzione AWS Lambda da un cluster DB Aurora MySQL utilizzando le funzioni native `lambda_sync` e `lambda_async`. Questo approccio può essere utile quando si desidera integrare il database in esecuzione su Aurora MySQL con altri servizi AWS. Ad esempio, potresti voler inviare una notifica usando Amazon Simple Notification Service (Amazon SNS) ogni volta che una riga viene inserita in una tabella specifica nel database.

Indice

- Utilizzo delle funzioni native per chiamare una funzione Lambda
  - Concessione del ruolo in Aurora MySQL versione 3
  - Concessione del privilegio in Aurora MySQL versione 2
  - Sintassi per la funzione lambda_sync
  - Parametri per la funzione lambda_sync
  - Esempio per la funzione lambda_sync
  - Sintassi per la funzione lambda_async
  - Parametri per la funzione lambda_async
  - Esempio per la funzione lambda_async
  - Invocazione di una funzione Lambda all'interno di un trigger

Utilizzo delle funzioni native per chiamare una funzione Lambda

Le funzioni `lambda_sync` e `lambda_async` sono funzioni native predefinite che chiamano una funzione Lambda in modo sincrono o asincrono. Quando è necessario conoscere il risultato dell'esecuzione della funzione Lambda prima di passare a un'altra operazione, utilizza la funzione sincrona `lambda_sync`. Quando non è necessario conoscere il risultato della funzione Lambda prima di passare a un'altra operazione, utilizza la funzione asincrona `lambda_async`.

Concessione del ruolo in Aurora MySQL versione 3

In Aurora MySQL versione 3, l'utente che invoca una funzione nativa deve avere il ruolo `AWS_LAMBDA_ACCESS`. Per concedere questo privilegio a un utente, connettersi all'istanza database come utente amministratore ed eseguire la seguente istruzione.

```
GRANT AWS_LAMBDA_ACCESS TO user@domain-or-ip-address
```

Si può revocare questo ruolo eseguendo la seguente istruzione.

```
REVOKE AWS_LAMBDA_ACCESS FROM user@domain-or-ip-address
```

> **Tip**
>
> Quando utilizzi la tecnica basata sul ruolo in Aurora MySQL versione 3, puoi anche attivare il ruolo utilizzando l'istruzione `SET ROLE role_name` o `SET ROLE ALL`. Se non si ha familiarità con il sistema dei ruoli MySQL 8.0, è possibile ottenere ulteriori informazioni in Privilegio basato sui ruoli. Per maggiori dettagli, consulta la pagina Using roles nel Manuale di riferimento di MySQL.
> Questo vale solo per la sessione attiva corrente. Quando ti riconnetti, devi eseguire di nuovo l'istruzione `SET ROLE` per concedere i privilegi. Per ulteriori informazioni, consulta Istruzione SET ROLE nel Manuale di riferimento di MySQL.
> Puoi utilizzare il parametro `activate_all_roles_on_login` del cluster di database per attivare automaticamente tutti i ruoli quando un utente si connette a un'istanza database. Quando questo parametro è impostato, non è necessario chiamare esplicitamente l'istruzione `SET ROLE` per attivare un ruolo. Per ulteriori informazioni, consulta activate_all_roles_on_login nel Manuale di riferimento di MySQL.
> Tuttavia, è necessario chiamare esplicitamente `SET ROLE ALL` all'inizio di una stored procedure per attivare il ruolo, quando la stored procedure viene chiamata da un altro utente.

Se ricevi un errore come il seguente quando provi a richiamare una funzione Lambda, esegui un'istruzione `SET ROLE`.

```
SQL Error [1227] [42000]: Access denied; you need (at least one of) the Invoke Lambda
 privilege(s) for this operation
```

Assicurati di concedere il ruolo all'utente corretto, come mostrato nelle voci della tabella `mysql.users`. Potrebbero esserci più utenti con lo stesso nome, ma su host diversi. A seconda dell'applicazione o dell'host che invoca la funzione `lambda_sync`, MySQL seleziona l'utente con la corrispondenza migliore in base alle voci della colonna `host`.

Concessione del privilegio in Aurora MySQL versione 2

In Aurora MySQL versione 2, l'utente che richiama una funzione nativa deve avere il privilegio `INVOKE LAMBDA`. Per concedere questo privilegio a un utente, connettersi all'istanza database come utente amministratore ed eseguire la seguente istruzione.

```
GRANT INVOKE LAMBDA ON *.* TO user@domain-or-ip-address
```

Puoi revocare questo privilegio eseguendo la seguente istruzione.

```
REVOKE INVOKE LAMBDA ON *.* FROM user@domain-or-ip-address
```

Sintassi per la funzione lambda_sync

Invochi la funzione `lambda_sync` in modo sincrono con il tipo di chiamata `RequestResponse`. La funzione restituisce il risultato della chiamata Lambda in un payload JSON. La funzione ha la seguente sintassi.

```
lambda_sync (
  lambda_function_ARN,
  JSON_payload
)
```

Parametri per la funzione lambda_sync

La funzione `lambda_sync` include i seguenti parametri.

lambda_function_ARN

Amazon Resource Name (ARN) della funzione Lambda da chiamare.

JSON_payload

Il payload per la funzione Lambda chiamata in formato JSON.

> **Note**
>
> Aurora MySQL versione 3 supporta le funzioni di analisi JSON di MySQL 8.0. Tuttavia, Aurora MySQL versione 2 non include queste funzioni. L'analisi JSON non è richiesta quando una funzione Lambda restituisce un valore atomico, come un numero o una stringa.

Esempio per la funzione lambda_sync

La seguente query basata su `lambda_sync` chiama la funzione Lambda `BasicTestLambda` in modo sincrono usando l'ARN della funzione. Il payload per la funzione è `{"operation": "ping"}`.

```
SELECT lambda_sync(
    'arn:aws:lambda:us-east-1:123456789012:function:BasicTestLambda',
    '{"operation": "ping"}');
```

Sintassi per la funzione lambda_async

Invochi la funzione `lambda_async` in modo asincrono con il tipo di chiamata `Event`. La funzione restituisce il risultato della chiamata Lambda in un payload JSON. La funzione ha la seguente sintassi.

```
lambda_async (
  lambda_function_ARN,
  JSON_payload
)
```

Parametri per la funzione lambda_async

La funzione `lambda_async` include i seguenti parametri.

lambda_function_ARN

Amazon Resource Name (ARN) della funzione Lambda da chiamare.

JSON_payload

Il payload per la funzione Lambda chiamata in formato JSON.

> **Note**
>
> Aurora MySQL versione 3 supporta le funzioni di analisi JSON di MySQL 8.0. Tuttavia, Aurora MySQL versione 2 non include queste funzioni. L'analisi JSON non è richiesta quando una funzione Lambda restituisce un valore atomico, come un numero o una stringa.

Esempio per la funzione lambda_async

La seguente query basata su `lambda_async` chiama la funzione Lambda `BasicTestLambda` in modo asincrono usando l'ARN della funzione. Il payload per la funzione è `{"operation": "ping"}`.

```
SELECT lambda_async(
    'arn:aws:lambda:us-east-1:123456789012:function:BasicTestLambda',
    '{"operation": "ping"}');
```

Invocazione di una funzione Lambda all'interno di un trigger

Puoi utilizzare i trigger per chiamare Lambda sulle istruzioni di modifica dei dati. L'esempio seguente illustra come utilizzare la funzione `lambda_async` nativa e memorizzare il risultato in una variabile.

```
mysql>SET @result=0;
mysql>DELIMITER //
mysql>CREATE TRIGGER myFirstTrigger
      AFTER INSERT
          ON Test_trigger FOR EACH ROW
      BEGIN
      SELECT lambda_async(
          'arn:aws:lambda:us-east-1:123456789012:function:BasicTestLambda',
          '{"operation": "ping"}')
          INTO @result;
      END; //
mysql>DELIMITER ;
```

> **Note**
>
> I trigger non vengono eseguiti una volta per istruzione SQL, ma una volta per riga modificata, una riga alla volta. Quando viene eseguito un trigger, il processo è sincrono. L'istruzione di modifica dei dati viene restituita solo al completamento del trigger.

Prestare attenzione quando si richiama una funzione AWS Lambda dai trigger sulle tabelle che hanno un elevato traffico di scrittura. Per ciascuna riga vengono attivati i trigger `INSERT`, `UPDATE` e `DELETE`. Un carico di lavoro pesante in scrittura su una tabella con trigger `INSERT`, `UPDATE` e `DELETE` genera un numero elevato di chiamate alla funzione AWS Lambda.

Chiamare una funzione Lambda con una procedura archiviata Aurora MySQL (obsoleto)

Puoi chiamare una funzione AWS Lambda da un cluster database Aurora MySQL utilizzando la procedura `mysql.lambda_async`. Questo approccio può essere utile quando si desidera integrare il database in esecuzione su Aurora MySQL con altri servizi AWS. Ad esempio, potresti voler inviare una notifica usando Amazon Simple Notification Service (Amazon SNS) ogni volta che una riga viene inserita in una tabella specifica nel database.

Indice

- Considerazioni sulle versioni Aurora MySQL
- Utilizzo della procedura mysql.lambda_async per chiamare una funzione Lambda (obsoleto)
  - Sintassi
  - Parametri
  - Esempi

Considerazioni sulle versioni Aurora MySQL

In Aurora MySQL versione 2 puoi utilizzare il metodo delle funzioni native anziché queste stored procedure per richiamare una funzione Lambda. Per ulteriori informazioni sulle funzioni native, consulta Utilizzo delle funzioni native per chiamare una funzione Lambda.

In Aurora MySQL versione 2, la stored procedure `mysql.lambda_async` non è più supportata. Ti consigliamo di utilizzare le funzioni Lambda native.

In Aurora MySQL versione 3, la procedura archiviata non è disponibile.

Utilizzo della procedura mysql.lambda_async per chiamare una funzione Lambda (obsoleto)

La procedura `mysql.lambda_async` è una stored procedure predefinita che chiama una funzione Lambda in modo asincrono. Per utilizzare questa procedura, l'utente del database deve avere il privilegio `EXECUTE` sulla stored procedure `mysql.lambda_async`.

Sintassi

La procedura `mysql.lambda_async` ha la seguente sintassi.

```
CALL mysql.lambda_async (
  lambda_function_ARN,
  lambda_function_input
)
```

Parametri

La procedura `mysql.lambda_async` include i seguenti parametri.

lambda_function_ARN

Amazon Resource Name (ARN) della funzione Lambda da chiamare.

lambda_function_input

La stringa di input in formato JSON per chiamare la funzione Lambda.

Esempi

Come best practice, ti consigliamo di inserire le chiamate alla procedura `mysql.lambda_async` in una procedura archiviata che può essere richiamata da origini diverse come trigger o codice client. Questo approccio può evitare problemi di resistenza non corrispondente e rendere più semplice per richiamare le funzioni Lambda.

> **Note**
>
> Prestare attenzione quando si richiama una funzione AWS Lambda dai trigger sulle tabelle che hanno un elevato traffico di scrittura. Per ciascuna riga vengono attivati i trigger `INSERT`, `UPDATE` e `DELETE`. Un carico di lavoro pesante in scrittura su una tabella con trigger `INSERT`, `UPDATE` e `DELETE` genera un numero elevato di chiamate alla funzione AWS Lambda.

Sebbene le chiamate alla procedura `mysql.lambda_async` siano asincrone, i trigger sono sincroni. Un'istruzione che genera un numero elevato di attivazioni di trigger non attende il completamento della chiamata alla funzione AWS Lambda, ma attende il completamento dei trigger prima di restituire il controllo al client.

Example Esempio: chiama una funzione AWS Lambda per inviare un'e-mail

Nell'esempio seguente viene creata una stored procedure che è possibile chiamare nel codice del database per inviare un'e-mail utilizzando una funzione Lambda.

Funzione AWS Lambda

```
import boto3

ses = boto3.client('ses')

def SES_send_email(event, context):

    return ses.send_email(
        Source=event['email_from'],
        Destination={
            'ToAddresses': [
            event['email_to'],
            ]
        },

        Message={
            'Subject': {
            'Data': event['email_subject']
            },
            'Body': {
                'Text': {
                    'Data': event['email_body']
                }
            }
        }
    )
```

Stored procedure

```
DROP PROCEDURE IF EXISTS SES_send_email;
```

```
DELIMITER ;;
  CREATE PROCEDURE SES_send_email(IN email_from VARCHAR(255),
                                  IN email_to VARCHAR(255),
                                  IN subject VARCHAR(255),
                                  IN body TEXT) LANGUAGE SQL
  BEGIN
    CALL mysql.lambda_async(
         'arn:aws:lambda:us-west-2:123456789012:function:SES_send_email',
         CONCAT('{"email_to" : "', email_to,
             '", "email_from" : "', email_from,
             '", "email_subject" : "', subject,
             '", "email_body" : "', body, '"}')
     );
  END
  ;;
DELIMITER ;
```

Chiama la stored procedure per chiamare la funzione AWS Lambda

```
mysql> call SES_send_email('example_from@amazon.com', 'example_to@amazon.com', 'Email
 subject', 'Email content');
```

Example Esempio: chiama una funzione AWS Lambda per pubblicare un evento di un trigger

Nell'esempio seguente viene creata una stored procedure che pubblica un evento utilizzando Amazon SNS. Il codice chiama la procedura da un trigger quando una riga viene aggiunta a una tabella.

Funzione AWS Lambda

```
import boto3

sns = boto3.client('sns')

def SNS_publish_message(event, context):

    return sns.publish(
        TopicArn='arn:aws:sns:us-west-2:123456789012:Sample_Topic',
        Message=event['message'],
        Subject=event['subject'],
        MessageStructure='string'
    )
```

Stored procedure

```
DROP PROCEDURE IF EXISTS SNS_Publish_Message;
DELIMITER ;;
CREATE PROCEDURE SNS_Publish_Message (IN subject VARCHAR(255),
                                      IN message TEXT) LANGUAGE SQL
BEGIN
  CALL mysql.lambda_async('arn:aws:lambda:us-
west-2:123456789012:function:SNS_publish_message',
     CONCAT('{ "subject" : "', subject,
            '", "message" : "', message, '" }')
     );
END
;;
DELIMITER ;
```

Tabella

```
CREATE TABLE 'Customer_Feedback' (
  'id' int(11) NOT NULL AUTO_INCREMENT,
  'customer_name' varchar(255) NOT NULL,
  'customer_feedback' varchar(1024) NOT NULL,
  PRIMARY KEY ('id')
) ENGINE=InnoDB DEFAULT CHARSET=utf8;
```

Trigger

```
DELIMITER ;;
CREATE TRIGGER TR_Customer_Feedback_AI
  AFTER INSERT ON Customer_Feedback
  FOR EACH ROW
BEGIN
  SELECT CONCAT('New customer feedback from ', NEW.customer_name),
 NEW.customer_feedback INTO @subject, @feedback;
  CALL SNS_Publish_Message(@subject, @feedback);
END
;;
DELIMITER ;
```

Inserisci una riga nella tabella per attivare la notifica

```
mysql> insert into Customer_Feedback (customer_name, customer_feedback) VALUES ('Sample
 Customer', 'Good job guys!');
```

Pubblicazione di log Amazon Aurora MySQL in Amazon CloudWatch Logs

Puoi configurare il cluster DB Aurora MySQL per pubblicare log dei dati di errore, generici, dei rallentamenti e delle revisioni in un gruppo di log in Amazon CloudWatch Logs. Con CloudWatch Logs puoi eseguire analisi in tempo reale dei dati di log e usare CloudWatch per creare allarmi e visualizzare parametri. Puoi utilizzare CloudWatch Logs per archiviare i record dei log in uno storage estremamente durevole.

Per pubblicare i log in CloudWatch Logs i rispettivi log devono essere abilitati. I registri dei dati di errore sono abilitati per impostazione predefinita, ma è necessario abilitare esplicitamente gli altri tipi di log. Per informazioni sull'abilitazione dei log in MySQL, consulta la sezione relativa alla selezione delle destinazioni di output del log delle query generiche e lente nella documentazione di MySQL. Per ulteriori informazioni sull'abilitazione dei log di audit di Aurora MySQL, consulta Abilitazione dell'audit avanzato.

> **Note**
>
> - Se l'esportazione dei dati del registro è disabilitata, Aurora non elimina i gruppi di log esistenti o i flussi di log. Se l'esportazione dei dati del log è disabilitata, i dati del log esistenti rimangono disponibili in CloudWatch Logs in base al valore di conservazione di log e si continuano a sostenere i costi per i dati del log di revisione archiviati. Puoi eliminare i flussi di log e i gruppo di log usando la console CloudWatch Logs, la AWS CLI o l'API CloudWatch Logs.
> - Un modo alternativo per pubblicare i log di audit in CloudWatch Logs consiste nell'abilitare l'auditing avanzato, quindi nel creare un gruppo di parametri del cluster di database personalizzato e nell'impostare il parametro `server_audit_logs_upload` su `1`. Il valore predefinito per il parametro del cluster di database `server_audit_logs_upload` è `0`. Per informazioni sull'abilitazione dell'auditing avanzato, consulta Utilizzo dell'audit avanzato con un cluster di database Amazon Aurora MySQL.
>
>   Se utilizzi questo metodo alternativo, devi avere un ruolo IAM per accedere a CloudWatch Logs e impostare il parametro a livello di cluster `aws_default_logs_role` sull'ARN di questo ruolo. Per ulteriori informazioni sulla creazione del ruolo, consulta Configurazione dei ruoli IAM per accedere ai servizi AWS. Tuttavia, se disponi del ruolo

`AWSServiceRoleForRDS` collegato ai servizi, tale ruolo fornisce accesso a CloudWatch Logs e sovrascrive qualsiasi ruolo personalizzato. Per informazioni sui ruoli collegati ai servizi per Amazon RDS, consultare Utilizzo di ruoli collegati ai servizi per Amazon Aurora.

- Se non desideri esportare i log di revisione in CloudWatch Logs assicurati che tutti i metodi di esportazione dei log di revisione siano disabilitati. Questi metodi sono l'Console di gestione AWS, l'AWS CLI, l'API RDS e il parametro `server_audit_logs_upload`.
- La procedura per i cluster di database Aurora Serverless v1 è leggermente diversa rispetto a quella dei cluster di database con istanze database allocate o Aurora Serverless v2. I cluster Aurora Serverless v1 caricano automaticamente tutti i log che abiliti mediante i parametri di configurazione.

  Pertanto, puoi attivare o disattivare il caricamento dei log per i cluster di database Aurora Serverless v1 attivando e disattivando diversi tipi di log nel gruppo di parametri del cluster di database. Non modifichi le impostazioni del cluster stesso tramite la Console di gestione AWS, AWS CLI o l'API RDS. Per informazioni su come attivare e disattivare i registri MySQL per i cluster Aurora Serverless v1, consulta Gruppi di parametri per Aurora Serverless v1.

Console

Puoi pubblicare i log di Aurora MySQL per i cluster con provisioning in CloudWatch Logs con la console.

Per pubblicare i log Aurora MySQL dalla console

1. Apri la console Amazon RDS all'indirizzo https://console.aws.amazon.com/rds/.
2. Nel riquadro di navigazione, scegliere Databases (Database).
3. Scegliere il cluster DB Aurora MySQL per cui vuoi pubblicare i dati dei log.
4. Scegliere Modify (Modifica).
5. Nella sezione Log exports (Esportazioni log) scegliere i log che si vuole iniziare a pubblicare in CloudWatch Logs.
6. Scegliere Continue (Continua) e quindi selezionare Modify DB Cluster (Modifica cluster DB) nella pagina di riepilogo.

AWS CLI

Puoi pubblicare i log di Aurora MySQL per i cluster con provisioning con AWS CLI. Per farlo, esegui il comando AWS CLI modify-db-cluster con le opzioni seguenti:

- `--db-cluster-identifier` — – L'identificatore del cluster DB.
- `--cloudwatch-logs-export-configuration` — – L'impostazione di configurazione per i tipi di log da abilitare per l'esportazione in CloudWatch Logs per il cluster DB.

Puoi pubblicare i log Aurora MySQL anche eseguendo uno dei seguenti comandi della AWS CLI:

- create-db-cluster
- restore-db-cluster-from-s3
- restore-db-cluster-from-snapshot
- restore-db-cluster-to-point-in-time

Esegui uno di questi comandi dell'AWS CLI con le opzioni seguenti:

- `--db-cluster-identifier` — – L'identificatore del cluster DB.
- `--engine` — – Il motore di database.
- `--enable-cloudwatch-logs-exports` — – L'impostazione di configurazione per i tipi di log da abilitare per l'esportazione in CloudWatch Logs per il cluster DB.

Altre opzioni potrebbero essere richieste a seconda del comando dell'AWS CLI che esegui.

Example

Il seguente comando modifica un cluster DB Aurora MySQL esistente per pubblicare i file di log in CloudWatch Logs.

Per Linux, macOS o Unix:

```
aws rds modify-db-cluster \
    --db-cluster-identifier mydbcluster \
    --cloudwatch-logs-export-configuration '{"EnableLogTypes":
["error","general","slowquery","audit","instance"]}'
```

Per Windows:

```
aws rds modify-db-cluster ^
    --db-cluster-identifier mydbcluster ^
    --cloudwatch-logs-export-configuration '{"EnableLogTypes":
["error","general","slowquery","audit","instance"]}'
```

Example

Il seguente comando crea un cluster DB Aurora MySQL per pubblicare i file di log in CloudWatch Logs.

Per Linux, macOS o Unix:

```
aws rds create-db-cluster \
    --db-cluster-identifier mydbcluster \
    --engine aurora \
    --enable-cloudwatch-logs-exports
 '["error","general","slowquery","audit","instance"]'
```

Per Windows:

```
aws rds create-db-cluster ^
    --db-cluster-identifier mydbcluster ^
    --engine aurora ^
    --enable-cloudwatch-logs-exports
 '["error","general","slowquery","audit","instance"]'
```

API RDS

Puoi pubblicare i log di Aurora MySQL per i cluster con provisioning con l'API RDS. Per farlo, esegui l'operazione ModifyDBCluster con le seguenti opzioni:

- `DBClusterIdentifier`— – L'identificatore del cluster DB.
- `CloudwatchLogsExportConfiguration`— – L'impostazione di configurazione per i tipi di log da abilitare per l'esportazione in CloudWatch Logs per il cluster DB.

Puoi anche pubblicare i log Aurora MySQL con l'API RDS eseguendo uno delle seguenti operazioni dell'API RDS:

- CreateDBCluster
- RestoreDBClusterFromS3
- RestoreDBClusterFromSnapshot
- RestoreDBClusterToPointInTime

Utilizza l'operazione API RDS con i seguenti parametri:

- `DBClusterIdentifier`— – L'identificatore del cluster DB.
- `Engine`— – Il motore di database.
- `EnableCloudwatchLogsExports`— – L'impostazione di configurazione per i tipi di log da abilitare per l'esportazione in CloudWatch Logs per il cluster DB.

Altri parametri potrebbero essere richiesti a seconda del comando dell'AWS CLI che esegui.

Monitoraggio degli eventi di log in Amazon CloudWatch

Dopo aver abilitato gli eventi di log Aurora MySQL, puoi monitorare gli eventi in Amazon CloudWatch Logs. Un nuovo gruppo di log viene creato automaticamente per il cluster DB Aurora con il seguente prefisso, in cui *`cluster-name`* rappresenta il nome del cluster DB e *`log_type`* rappresenta il tipo di log.

```
/aws/rds/cluster/cluster-name/log_type
```

Ad esempio, se configuri la funzione di esportazione per includere il log delle query lente per un cluster DB denominato `mydbcluster`, i dati delle query lente vengono archiviati nel gruppo di log `/aws/rds/cluster/mydbcluster/slowquery`.

Gli eventi di tutte le istanze in un cluster vengono inviati a un gruppo di log utilizzando flussi di log diversi. Il comportamento dipende da quale delle seguenti condizioni è vera:

- Esiste un gruppo di registro con il nome specificato.

  Aurora utilizza il gruppo di registro esistente per esportare i dati di registro per il cluster. Per creare i gruppi di registro con periodi di conservazione di registro predefiniti, filtri di parametri e accesso cliente, puoi utilizzare la configurazione automatizzata, ad esempio AWS CloudFormation.

- Non esiste un gruppo di registro con il nome specificato.

  Quando viene rilevata una voce di log corrispondente nel file di log per l'istanza, Aurora MySQL crea automaticamente un nuovo gruppo di log in CloudWatch Logs. Il gruppo di log utilizza il periodo di conservazione predefinito dei log di Never Expire (Nessuna scadenza).

  Puoi usare la console CloudWatch Logs, la AWS CLI oppure l'API di CloudWatch Logs per cambiare il periodo di conservazione dei registri. Per ulteriori informazioni su come modificare il periodo di conservazione di log in CloudWatch Logs, consulta Modifica della conservazione dei dati di log in CloudWatch Logs.

Puoi usare la console CloudWatch Logs, la AWS CLI o l'API di CloudWatch Logs per cercare informazioni nel registro eventi per un cluster DB. Per ulteriori informazioni sulla ricerca e l'applicazione di filtri per i dati di log, consulta Ricerca e filtraggio dei dati di log.

Modalità di laboratorio per Amazon Aurora MySQL

> **Important**
>
> La modalità Lab è stata introdotta in Aurora MySQL versione 2 per abilitare l'ottimizzazione Fast DDL, che ha migliorato l'efficienza di alcune operazioni DDL. In Aurora MySQL versione 3, la modalità Lab è stata rimossa e Fast DDL è stata sostituita dalla funzionalità MySQL 8.0 chiamata Instant DDL.

La modalità di laboratorio per Aurora consente di abilitare le caratteristiche di Aurora disponibili nella versione del database Aurora corrente, ma che non vengono abilitate per impostazione predefinita. Sebbene l'utilizzo della modalità di laboratorio per Aurora non sia consigliato nei cluster di database di produzione, questa modalità Aurora può essere utilizzata per abilitare le caratteristiche per i cluster di database negli ambienti di test e sviluppo. Per ulteriori informazioni sulle caratteristiche di Aurora disponibili quando è abilitata la modalità di laboratorio di Aurora, consulta Caratteristiche della modalità di laboratorio per Aurora.

`aurora_lab_mode` è un parametro a livello di istanza impostato nel gruppo di parametri predefiniti. Il parametro è impostato su `0` (disattivato) nel gruppo di parametri predefiniti. Per abilitare la modalità di laboratorio per Aurora, crea un gruppo di parametri personalizzato, imposta il parametro `aurora_lab_mode` su `1` (abilitato) nel gruppo di parametri predefinito e modifica una o più istanze di database Aurora in modo che venga utilizzato il gruppo di parametri personalizzato. Quindi collegati all'endpoint dell'istanza pertinente per provare le funzionalità della modalità di laboratorio. Per informazioni sulla modifica di un gruppo di parametri database, consulta Modifica dei parametri in un gruppo di parametri database in Amazon Aurora. Per informazioni sui gruppi di parametri e su Amazon Aurora, consulta Parametri di configurazione Aurora MySQL.

Caratteristiche della modalità di laboratorio per Aurora

La seguente funzionalità di Aurora è attualmente disponibile quando viene abilitata la modalità Lab di Aurora. Per poter utilizzare queste caratteristiche, occorre innanzitutto attivare la modalità di laboratorio per Aurora.

Fast DDL

Questa funzione consente di eseguire un'operazione `ALTER TABLE tbl_name ADD COLUMN col_name column_definition` quasi istantaneamente. L'operazione si conclude senza che

vi sia necessità di copiare la tabella e senza alcuna conseguenza materiale sulle altre istruzioni DML. Poiché l’operazione non consuma spazio di archiviazione temporaneo per la copia della tabella, rende le istruzioni DDL pratiche da utilizzare anche nel caso di tabelle di grandi dimensioni su classi di stanza di dimensioni ridotte.

Al momento, l'operazione Fast DDL è supportata solo per l'aggiunta di una colonna nullable, senza un valore predefinito, alla fine della tabella. Per ulteriori informazioni sull'utilizzo di questa caratteristica, consulta Alterazione delle tabelle in Amazon Aurora mediante Fast DDL.

Best practice con Amazon Aurora MySQL

Questo argomento include informazioni sulle best practice e sulle opzioni per l'utilizzo o la migrazione di dati a un cluster di database Amazon Aurora MySQL. Le informazioni contenute in questo argomento riassumono e ribadiscono alcune linee guida e procedure disponibili in Gestione di un cluster DB Amazon Aurora.

Indice

- Determinazione dell'istanza database a cui si è connessi
- Best practice per le prestazioni e il dimensionamento di Aurora MySQL
  - Utilizzo delle classi di istanza T per lo sviluppo e i test
  - Ottimizzazione delle query di join indicizzate Aurora MySQL con prefetch asincrono delle chiavi
    - Abilitazione del prefetch asincrono delle chiavi
    - Ottimizzazione delle query per il prefetch asincrono delle chiavi
  - Ottimizzazione di grandi query di join Aurora MySQL con hash join
    - Abilitazione di hash join
    - Ottimizzazione delle query per gli hash join
  - Utilizzo di Amazon Aurora per dimensionare le letture per il database MySQL
  - Ottimizzazione delle operazioni di timestamp
  - Errori di overflow dell'ID di indice virtuale
- Best practice per l'elevata disponibilità di Aurora MySQL
  - Utilizzo di Amazon Aurora per il disaster recovery con i database MySQL
  - Migrazione da MySQL ad Amazon Aurora MySQL con tempi di inattività ridotti
  - Evitare il rallentamento delle prestazioni, il riavvio automatico e il failover per istanze database Aurora MySQL
- Suggerimenti per le funzionalità MySQL in Aurora MySQL
  - Utilizzo della replica multithread in Aurora MySQL
  - Invocazione delle funzioni AWS Lambda usando funzioni MySQL native
  - Evitatre le transazioni XA con Amazon Aurora MySQL
  - Mantenere le chiavi esterne attivate durante le istruzioni DML
  - Configurazione della frequenza di svuotamento del buffer dei registri
  - Contenimento e risoluzione dei problemi di deadlock di Aurora MySQL

    - Riduzione al minimo dei deadlock di InnoDB
    - Monitoraggio dei deadlock di InnoDB
- Valutazione dell'utilizzo dell'istanza database per Aurora MySQL con le metriche di Amazon CloudWatch

Determinazione dell'istanza database a cui si è connessi

Per determinare a quale istanza database in un cluster di database Aurora MySQL è collegata una connessione, controlla la variabile globale `innodb_read_only` come mostrato nell'esempio seguente.

```
SHOW GLOBAL VARIABLES LIKE 'innodb_read_only';
```

La variabile `innodb_read_only` è impostata su `0N` se si è connessi a un'istanza database di lettura. Questa impostazione è `0FF` se si è connessi a un'istanza database di scrittura, ad esempio un'istanza primaria in un cluster con provisioning.

Questo approccio può essere utile se desideri aggiungere logica al codice dell'applicazione per bilanciare il carico di lavoro o garantire che un'operazione di scrittura utilizzi la connessione corretta.

Best practice per le prestazioni e il dimensionamento di Aurora MySQL

È possibile applicare le best practice seguenti per migliorare le prestazioni e la scalabilità dei cluster Aurora MySQL.

Argomenti

- Utilizzo delle classi di istanza T per lo sviluppo e i test
- Ottimizzazione delle query di join indicizzate Aurora MySQL con prefetch asincrono delle chiavi
- Ottimizzazione di grandi query di join Aurora MySQL con hash join
- Utilizzo di Amazon Aurora per dimensionare le letture per il database MySQL
- Ottimizzazione delle operazioni di timestamp
- Errori di overflow dell'ID di indice virtuale

Utilizzo delle classi di istanza T per lo sviluppo e i test

Le istanze Amazon Aurora MySQL che utilizzano le classi di istanza database `db.t2`, `db.t3` o `db.t4g` sono ideali per applicazioni che non supportano un carico di lavoro elevato per una quantità di tempo significativa. Le istanze T sono progettate per offrire prestazioni di base moderate e garantire prestazioni notevolmente maggiori se il carico di lavoro lo richiede. Sono concepite per carichi di lavoro che non utilizzano completamente la CPU spesso o in maniera regolare, ma che occasionalmente necessitano di un incremento delle prestazioni. Consigliamo di utilizzare le classi di istanza database T solo per i server di sviluppo e test o altri server non di produzione. Per ulteriori informazioni sulle classi di istanze T, consulta Istanze espandibili.

Se il cluster Aurora è più grande di 40 TB, non utilizzare le classi di istanza T. Quando il database dispone di un volume elevato di dati, il sovraccarico di memoria per la gestione degli oggetti dello schema può superare la capacità di un'istanza T.

Non abilitare lo schema di prestazioni MySQL sulle istanze T di Amazon Aurora MySQL. In caso contrario, è possibile che la memoria disponibile per l'istanza T potrebbe esaurirsi.

> **Tip**
>
> Se il database a volte è inattivo e altre volte ha un notevole carico di lavoro, puoi utilizzare Aurora Serverless v2 come alternativa alle istanze T. Con Aurora Serverless v2 puoi definire un intervallo di capacità e Aurora dimensiona automaticamente il database in base al carico di lavoro corrente. Per informazioni dettagliate sull'utilizzo, consulta Uso di Aurora Serverless v2. Per le versioni del motore del database che puoi utilizzare con Aurora Serverless v2, consulta Requisiti e limitazioni per Aurora Serverless v2.

Quando si utilizza un'istanza T come istanza di database in un cluster di database Aurora MySQL, si consiglia quanto segue:

- Utilizza la stessa classe di istanza database per tutte le istanze nel cluster di database. Ad esempio, se si utilizza `db.t2.medium` per l'istanza di scrittura, allora si consiglia di usare `db.t2.medium` anche per le istanze di lettura.
- Non regolare le impostazioni di configurazione relative alla memoria, ad esempio `innodb_buffer_pool_size`. Aurora utilizza un insieme altamente sintonizzato di valori di default per i buffer di memoria sulle istanze T. Questi valoridi default speciali sono necessari per consentire l'esecuzione di Aurora su istanze vincolate dalla memoria. Se si modificano impostazioni

relative alla memoria in un'istanza T, è molto più probabile che si verifichino condizioni di memoria insufficiente, anche se la modifica è destinata ad aumentare le dimensioni del buffer.

- Eseguire il monitoraggio del saldo del credito CPU (`CPUCreditBalance`) per garantire che il relativo livello sia sostenibile. In altre parole, i crediti CPU vengono accumulati alla stessa velocità con cui vengono utilizzati.

  Una volta esauriti i crediti CPU per un'istanza, si assisterà a un calo immediato nella CPU disponibile e a un aumento della latenza di lettura e scrittura per l'istanza. Questa situazione provoca un'importante riduzione delle prestazioni complessive dell'istanza.

  Se il livello del saldo del credito CPU non è sostenibile, consigliamo di modificare l'istanza database in modo da utilizzare una delle classi di istanza database R supportate (dimensionamento del calcolo).

  Per ulteriori informazioni sui parametri di monitoraggio, consulta Visualizzazione delle metriche nella console Amazon RDS.

- Monitora il ritardo di replica (`AuroraReplicaLag`) tra l'istanza di scrittura e le istanze di lettura.

  Se un'istanza di lettura esaurisce i crediti della CPU prima dell'istanza si scrittura, il ritardo risultante può causare il riavvio frequente dell'istanza di lettura. Si tratta di un risultato comune nei casi in cui un'applicazione dispone di un carico elevato di operazioni di lettura distribuito tra le istanze di lettura, allo stesso tempo che l'istanza di scrittura presenta un carico minimo di operazioni di scrittura.

  Se il ritardo della replica aumenta notevolmente, sarà necessario verificare che il saldo del credito CPU per le istanze di lettura nel cluster DB non sia esaurito.

  Se il livello del saldo del credito CPU non è sostenibile, consigliamo di modificare l'istanza database in modo da utilizzare una delle classi di istanza database R supportate (dimensionamento del calcolo).

- Mantenere il numero di inserimenti per transazione al di sotto di 1 milione per i cluster di database per cui è abilitata la registrazione binaria.

  Se il parametro `binlog_format` del gruppo di parametri del cluster di database è impostato su un valore diverso da `OFF`, per il cluster di database potrebbero verificarsi condizioni di memoria esaurita qualora riceva transazioni che contengono più di 1 milione di righe da inserire. È possibile monitorare il parametro della memoria da liberare (`FreeableMemory`) per determinare se la memoria disponibile per il cluster di database sta per esaurire. Controllando quindi il

parametro delle operazioni di scrittura (`VolumeWriteIOPS`) sarà possibile verificare se l'istanza di scrittura riceve un carico elevato di operazioni di scrittura. In tal caso, consigliamo di aggiornare l'applicazione per limitare gli inserimenti in una transazione a un numero inferiore a 1 milione. In alternativa, è possibile modificare l'istanza in modo da utilizzare una delle classi di istanza database R supportate (dimensionamento del calcolo).

Ottimizzazione delle query di join indicizzate Aurora MySQL con prefetch asincrono delle chiavi

Aurora MySQL può utilizzare il prefetch asincrono delle chiavi per migliorare le prestazioni delle query che eseguono il join delle tabelle negli indici. Questa funzionalità migliora le prestazioni anticipando le righe necessarie per eseguire query in cui una query JOIN richieda l'uso dell'algoritmo di join Batched Key Access (BKA) e le funzionalità di ottimizzazione Multi-Range Read (MRR). Per ulteriori informazioni su BKA e MRR, consulta Block Nested-Loop and Batched Key Access Joins e Multi-Range Read Optimization nella documentazione di MySQL.

Per trarre vantaggio dalla funzionalità di prefetch asincrono delle chiavi, è necessario che una query utilizzi sia gli algoritmi BKA sia le funzionalità MRR. Tale query viene in genere eseguita quando la clausola JOIN di una query utilizza un indice secondario, ma richiede anche alcune colonne dell'indice principale. Puoi ad esempio utilizzare il prefetch asincrono delle chiavi quando una clausola JOIN rappresenta un equijoin sui valori di indice tra una tabella esterna di piccole dimensioni e una interna di dimensioni maggiori, in cui l'indice è altamente selettivo nella tabella più grande. Il prefetch asincrono delle chiavi interagisce con BKA e MRR per eseguire una ricerca nell'indice da secondario a principale durante la valutazione di una clausola JOIN. Il prefetch asincrono delle chiavi identifica le righe necessarie per eseguire la query durante la valutazione della clausola JOIN. Utilizza quindi un thread in background per caricare in modo asincrono le pagine contenenti le righe in memoria prima di eseguire la query.

Il prefetch asincrono delle chiavi è disponibile per Aurora MySQL versione 2.10 e successive e per la versione 3. Per ulteriori informazioni sulle versioni di Aurora MySQL, consulta Aggiornamenti del motore del database per Amazon Aurora MySQL.

Abilitazione del prefetch asincrono delle chiavi

È possibile abilitare la funzionalità di prefetch asincrono delle chiavi impostando `aurora_use_key_prefetch`, una variabile del server MySQL, su `on`. Per impostazione predefinita, questo valore è impostato su `on`. È tuttavia possibile abilitare il prefetch asincrono delle chiavi solo dopo aver abilitato anche l'algoritmo di join BKA e aver disabilitato la funzionalità MRR basata sui

costi. A tale scopo, è necessario impostare i seguenti valori per `optimizer_switch`, una variabile di server MySQL:

- Imposta `batched_key_access` su on. Questo valore controlla l'utilizzo dell'algoritmo di join BKA. Per impostazione predefinita, questo valore è impostato su `off`.
- Imposta `mrr_cost_based` su `off`. Questo valore controlla l'utilizzo della funzionalità MRR basata sui costi. Per impostazione predefinita, questo valore è impostato su on.

È attualmente possibile impostare questi valori solo a livello di sessione. Il seguente esempio illustra come impostare questi valori in modo da abilitare la funzionalità di prefetch asincrono delle chiavi per la sessione corrente eseguendo le istruzioni SET.

```
mysql> set @@session.aurora_use_key_prefetch=on;
mysql> set @@session.optimizer_switch='batched_key_access=on,mrr_cost_based=off';
```

In modo analogo, è possibile utilizzare le istruzioni SET per disabilitare la funzionalità di prefetch asincrono delle chiavi e l'algoritmo di join BKA e riabilitare la funzionalità MRR basata sui costi per la sessione corrente, come mostrato nell'esempio seguente.

```
mysql> set @@session.aurora_use_key_prefetch=off;
mysql> set @@session.optimizer_switch='batched_key_access=off,mrr_cost_based=on';
```

Per ulteriori informazioni sulle opzioni di ottimizzazione batched_key_access e mrr_cost_based, consulta Switchable Optimizations nella documentazione di MySQL.

Ottimizzazione delle query per il prefetch asincrono delle chiavi

È possibile verificare se una query può trarre vantaggio dalla funzionalità di prefetch asincrono delle chiavi. A tale scopo, utilizza l'istruzione `EXPLAIN` per profilare la query prima di eseguirla. L'istruzione `EXPLAIN` fornisce informazioni sul piano di esecuzione da utilizzare per una query specificata.

Nell'output dell'istruzione `EXPLAIN` la colonna `Extra` descrive le informazioni aggiuntive incluse nel piano di esecuzione. Se la funzionalità di prefetch asincrono delle chiavi si applica a una tabella utilizzata nella query, questa colonna include uno dei seguenti valori:

- `Using Key Prefetching`
- `Using join buffer (Batched Key Access with Key Prefetching)`

Il seguente esempio illustra l'utilizzo di `EXPLAIN` per visualizzare il piano di esecuzione per una query che può sfruttare il prefetch asincrono delle chiavi.

```
mysql> explain select sql_no_cache
    ->     ps_partkey,
    ->     sum(ps_supplycost * ps_availqty) as value
    -> from
    ->     partsupp,
    ->     supplier,
    ->     nation
    -> where
    ->     ps_suppkey = s_suppkey
    ->     and s_nationkey = n_nationkey
    ->     and n_name = 'ETHIOPIA'
    -> group by
    ->     ps_partkey having
    ->         sum(ps_supplycost * ps_availqty) > (
    ->             select
    ->                 sum(ps_supplycost * ps_availqty) * 0.0000003333
    ->             from
    ->                 partsupp,
    ->                 supplier,
    ->                 nation
    ->             where
    ->                 ps_suppkey = s_suppkey
    ->                 and s_nationkey = n_nationkey
    ->                 and n_name = 'ETHIOPIA'
    ->         )
    -> order by
    ->     value desc;
+----+-------------+----------+------+----------------------+--------------
+---------+-----------------------------------+------+----------
+-------------------------------------------------------------+
| id | select_type | table    | type | possible_keys        | key          | key_len
 | ref                              | rows | filtered | Extra
                            |
+----+-------------+----------+------+----------------------+--------------
+---------+-----------------------------------+------+----------
+-------------------------------------------------------------+
|  1 | PRIMARY     | nation   | ALL  | PRIMARY              | NULL         | NULL
 | NULL                             |   25 |   100.00 | Using where; Using temporary;
 Using filesort                |
```

```
|  1 | PRIMARY     | supplier | ref  | PRIMARY,i_s_nationkey | i_s_nationkey | 5
 | dbt3_scale_10.nation.n_nationkey | 2057 |   100.00 | Using index
                              |
|  1 | PRIMARY     | partsupp | ref  | i_ps_suppkey          | i_ps_suppkey  | 4
 | dbt3_scale_10.supplier.s_suppkey |   42 |   100.00 | Using join buffer (Batched Key
 Access with Key Prefetching) |
|  2 | SUBQUERY    | nation   | ALL  | PRIMARY               | NULL          | NULL
 | NULL                             |   25 |   100.00 | Using where
                              |
|  2 | SUBQUERY    | supplier | ref  | PRIMARY,i_s_nationkey | i_s_nationkey | 5
 | dbt3_scale_10.nation.n_nationkey | 2057 |   100.00 | Using index
                              |
|  2 | SUBQUERY    | partsupp | ref  | i_ps_suppkey          | i_ps_suppkey  | 4
 | dbt3_scale_10.supplier.s_suppkey |   42 |   100.00 | Using join buffer (Batched Key
 Access with Key Prefetching) |
+----+-------------+----------+------+-----------------------+---------------
+---------+-----------------------------------+------+----------
+-------------------------------------------------------------+
6 rows in set, 1 warning (0.00 sec)
```

Per ulteriori informazioni sul formato di output `EXPLAIN`, consulta l'argomento relativo al formato di output EXPLAIN esteso nella documentazione di MySQL.

Ottimizzazione di grandi query di join Aurora MySQL con hash join

Se devi eseguire il join di una grande quantità di dati tramite una query equijoin, l'utilizzo di un hash join può migliorare le prestazioni della query. Puoi abilitare gli hash join per Aurora MySQL.

Una colonna hash join può essere una qualsiasi espressione complessa. Di seguito sono indicati i modi per eseguire il confronto tra i tipi di dati in una colonna hash join:

- È possibile confrontare qualsiasi elemento nella categoria dei tipi di dati numerici precisi, ad esempio `int`, `bigint`, `numeric` e `bit`.
- È possibile confrontare qualsiasi elemento nella categoria dei tipi di dati numerici di approssimazione, ad esempio `float` e `double`.
- È possibile confrontare elementi nei tipi di stringa, se i tipi di stringa dispongono dello stesso set di caratteri e delle stesse regole di confronto.
- È possibile confrontare elementi con tipi di dati data e timestamp, se i tipi corrispondono.

> **Note**
>
> Non è possibile confrontare i tipi di dati in categorie differenti.

Di seguito sono indicate le limitazioni che si applicano agli hash join per Aurora MySQL:

- Gli outer join sinistri-destri non sono supportati per Aurora MySQL versione 2, ma sono supportati per la versione 3.
- I semi-join come le query secondarie non sono supportati, a meno che non vengano prima materializzate le query secondarie.
- Aggiornamenti o eliminazioni di più tabelle non sono supportati.

  > **Note**
  >
  > Aggiornamenti o eliminazioni di tabelle singole sono supportati.

- Le colonne con tipi di dati BLOB e spaziali non possono essere colonne join in un hash join.

Abilitazione di hash join

Per abilitare gli hash join:

- Aurora MySQL versione 2: imposta il parametro di database o il parametro del cluster di database `aurora_disable_hash_join` su `0`. La disattivazione di `aurora_disable_hash_join` imposta il valore di `optimizer_switch` su `hash_join=on`.
- Aurora MySQL versione 3: imposta il parametro del server MySQL `optimizer_switch` su `block_nested_loop=on`.

Gli hash join sono attivati per impostazione predefinita in Aurora MySQL Versione 3 e Aurora MySQL versione 2. Nell'esempio seguente viene illustrato come abilitare gli hash join per Aurora MySQL versione 3. È possibile rilasciare l'istruzione `select @@optimizer_switch` per prima cosa per vedere quali altre impostazioni sono presenti nella stringa dei parametri di `SET`. Aggiornamento di un'impostazione nel parametro di `optimizer_switch` non cancella o modifica le altre impostazioni.

```
mysql> SET optimizer_switch='block_nested_loop=on';
```

> **Note**
>
> Per Aurora MySQL Versione 3, il supporto hash join è disponibile in tutte le versioni secondarie ed è attivato per impostazione predefinita.
> Per Aurora MySQL versione 2, il supporto hash join è disponibile in tutte le versioni secondarie. In Aurora MySQL versione 2, la funzionalità di join hash è sempre controllata dal valore `aurora_disable_hash_join`.

Con questa impostazione, l'ottimizzatore sceglie di utilizzare un hash join in base al costo, alle caratteristiche della query e alla disponibilità delle risorse. Se la stima dei costi non è corretta, puoi imporre all'ottimizzatore di scegliere un hash join, impostando `hash_join_cost_based`, una variabile del server MySQL, su `off`. L'esempio seguente illustra come imporre all'ottimizzatore di scegliere un hash join.

```
mysql> SET optimizer_switch='hash_join_cost_based=off';
```

> **Note**
>
> Questa impostazione sostituisce le decisioni dell'ottimizzatore basato sui costi. Sebbene l'impostazione possa essere utile per il test e lo sviluppo, si consiglia di non utilizzarla in produzione.

Ottimizzazione delle query per gli hash join

Per sapere se una query può utilizzare un hash join, è possibile utilizzare in primo luogo l'istruzione `EXPLAIN` per profilare la query. L'istruzione `EXPLAIN` fornisce informazioni sul piano di esecuzione da utilizzare per una query specificata.

Nell'output dell'istruzione `EXPLAIN` la colonna `Extra` descrive le informazioni aggiuntive incluse nel piano di esecuzione. Se un hash join si applica alle tabelle utilizzate nella query, questa colonna include valori simili a quelli indicato di seguito:

- `Using where; Using join buffer (Hash Join Outer table table1_name)`
- `Using where; Using join buffer (Hash Join Inner table table2_name)`

Il seguente esempio illustra l'utilizzo di EXPLAIN per visualizzare il piano di esecuzione per una query con hash join.

```
mysql> explain SELECT sql_no_cache * FROM hj_small, hj_big, hj_big2
    ->     WHERE hj_small.col1 = hj_big.col1 and hj_big.col1=hj_big2.col1 ORDER BY 1;
+----+-------------+----------+------+---------------+------+---------+------+------
+----------------------------------------------------------------+
| id | select_type | table    | type | possible_keys | key  | key_len | ref  | rows |
 Extra                                                          |
+----+-------------+----------+------+---------------+------+---------+------+------
+----------------------------------------------------------------+
|  1 | SIMPLE      | hj_small | ALL  | NULL          | NULL | NULL    | NULL |    6 |
 Using temporary; Using filesort                                |
|  1 | SIMPLE      | hj_big   | ALL  | NULL          | NULL | NULL    | NULL |   10 |
 Using where; Using join buffer (Hash Join Outer table hj_big)  |
|  1 | SIMPLE      | hj_big2  | ALL  | NULL          | NULL | NULL    | NULL |   15 |
 Using where; Using join buffer (Hash Join Inner table hj_big2) |
+----+-------------+----------+------+---------------+------+---------+------+------
+----------------------------------------------------------------+
3 rows in set (0.04 sec)
```

Nell'output `Hash Join Inner table` indica la tabella utilizzata per creare la tabella hash e `Hash Join Outer table` indica la tabella utilizzata per il probe della tabella hash.

Per ulteriori informazioni sul formato di output `EXPLAIN` esteso, consulta Formato di output EXPLAIN esteso nella documentazione di MySQL.

In Aurora MySQL 2.08 e versioni successive, è possibile utilizzare gli hint SQL per determinare se una query utilizza o meno hash join e quali tabelle utilizzare per i lati di build e probe del join. Per informazioni dettagliate, consultare Suggerimenti di Aurora MySQL.

Utilizzo di Amazon Aurora per dimensionare le letture per il database MySQL

Puoi utilizzare Amazon Aurora con l'istanza database MySQL per usufruire delle funzionalità di dimensionamento della lettura di Amazon Aurora ed espandere il reale carico di lavoro della tua istanza database MySQL. Per utilizzare Aurora per ridimensionare la lettura dell'istanza database MySQL, crea un cluster di database Aurora MySQL e rendilo una replica di lettura per l'istanza database MySQL. Quindi connettiti al cluster Aurora MySQL per elaborare le query di lettura. Il database di origine può essere un'istanza database RDS for MySQL o un database MySQL in

esecuzione esternamente a Amazon RDS. Per ulteriori informazioni, consulta Dimensionamento delle letture per il database MySQL con Amazon Aurora.

Ottimizzazione delle operazioni di timestamp

Quando il valore della variabile di sistema `time_zone` è impostato su `SYSTEM`, ogni chiamata di funzione MySQL che richiede un calcolo del fuso orario effettua una chiamata alla libreria di sistema. Quando esegui istruzioni SQL che restituiscono o modificano tali valori `TIMESTAMP` in caso di elevata concorrenza, è possibile che si verifichi un aumento della latenza, dei conflitti di blocco e dell'utilizzo della CPU. Per ulteriori informazioni, consulta time_zone nella documentazione di MySQL.

Per evitare questo comportamento, consigliamo di modificare il valore del parametro `time_zone` del cluster database impostandolo su `UTC`. Per ulteriori informazioni, consulta Modifica dei parametri in un gruppo di parametri del cluster di database in Amazon Aurora.

Sebbene il parametro `time_zone` sia dinamico (non richiede il riavvio del server di database), il nuovo valore viene utilizzato solo per le nuove connessioni. Per assicurarti che tutte le connessioni siano aggiornate in modo che venga utilizzato il nuovo valore `time_zone`, consigliamo di riciclare le connessioni alle applicazioni dopo aver aggiornato il parametro del cluster database.

Errori di overflow dell'ID di indice virtuale

Aurora MySQL limita i valori per gli ID di indice virtuale a 8 bit per prevenire un problema causato dal formato di undo in MySQL. Se un indice supera il limite impostato per l'ID di indice virtuale, il cluster potrebbe non essere disponibile. Quando un indice si avvicina al limite impostato per l'ID di indice virtuale oppure quando tenti di creare un indice superiore al limite impostato per l'ID di indice virtuale, RDS potrebbe generare il codice di errore `63955` o il codice di avviso `63955`. Per risolvere un errore relativo al limite impostato per l'ID di indice virtuale, si consiglia di creare nuovamente il database con un dump logico e quindi di eseguire il ripristino.

Per ulteriori informazioni sul dump logico e il ripristino per Amazon Aurora MySQL, consulta Migrate very large databases to Amazon Aurora MySQL using MyDumper and MyLoader. Per ulteriori informazioni sull'accesso ai log degli errori in Amazon Aurora, consulta Monitoraggio dei file di log di Amazon Aurora.

Best practice per l'elevata disponibilità di Aurora MySQL

È possibile applicare le best practice seguenti per migliorare la disponibilità dei cluster Aurora MySQL.

Argomenti

- Utilizzo di Amazon Aurora per il disaster recovery con i database MySQL
- Migrazione da MySQL ad Amazon Aurora MySQL con tempi di inattività ridotti
- Evitare il rallentamento delle prestazioni, il riavvio automatico e il failover per istanze database Aurora MySQL

Utilizzo di Amazon Aurora per il disaster recovery con i database MySQL

Puoi utilizzare Amazon Aurora con l'istanza database MySQL per creare un backup offsite per il disaster recovery. Per utilizzare Aurora per il disaster recovery dell'istanza database MySQL è necessario creare un cluster di database Amazon Aurora e renderlo uno slave di replica dell'istanza database MySQL. Ciò si applica a un'istanza database RDS per MySQL o a un database MySQL in esecuzione esternamente a Amazon RDS.

> **Important**
>
> Quando si imposta la replica tra un'istanza database MySQL e un cluster di database Amazon Aurora MySQL è necessario monitorare la replica per assicurarsi che venga mantenuta l'integrità e se necessario eseguirne la riparazione.

Per istruzioni su come creare un cluster di database Amazon Aurora MySQL e impostarlo come slave di lettura dell'istanza database MySQL, segui la procedura in Utilizzo di Amazon Aurora per dimensionare le letture per il database MySQL.

Per ulteriori informazioni sui modelli di ripristino di emergenza, consulta How to choose the best disaster recovery option for your Amazon Aurora MySQL cluster.

Migrazione da MySQL ad Amazon Aurora MySQL con tempi di inattività ridotti

Durante l'importazione dei dati da un database MySQL che supporta un'applicazione live a un cluster di database Amazon Aurora MySQL, è possibile che voglia ridurre il periodo durante il quale il servizio risulta interrotto in fase di migrazione. Per farlo è possibile utilizzare la procedura documentata in Importing data to an Amazon RDS for MySQL DB instance with reduced downtime nella Guida per l'utente di Amazon Relational Database Service. che può essere particolarmente utile se utilizzi un database di dimensioni particolarmente elevate. La procedura ti consente di ridurre il costo dell'importazione tramite la riduzione della quantità di dati passati in rete ad AWS.

La procedura elenca i passaggi per trasferire una copia dei dati del database su un' EC2istanza Amazon e importare i dati in una nuova istanza DB RDS for MySQL. Poiché Amazon Aurora è compatibile con MySQL, puoi utilizzare un cluster di database Amazon Aurora per l'istanza database Amazon RDS MySQL.

Evitare il rallentamento delle prestazioni, il riavvio automatico e il failover per istanze database Aurora MySQL

Se si esegue uno o più carichi di lavoro pesanti che eccedono le risorse allocate dell'istanza database, è possibile che le risorse su cui si esegue l'applicazione e il database Aurora si esauriscano. Per ottenere parametri sulla tua istanza di database come l'utilizzo della CPU, l'utilizzo della memoria e il numero di connessioni al database utilizzate, puoi fare riferimento ai parametri forniti da Amazon, Performance CloudWatch Insights e Enhanced Monitoring. Per ulteriori informazioni sul monitoraggio dell'istanza database, consultare Monitoraggio dei parametri in un cluster di database Amazon Aurora.

Se il carico di lavoro esaurisce le risorse in uso, è possibile che l'istanza database rallenti, venga riavviata o esegua un failover su un'altra istanza database. Per evitare che questo si verifichi, occorre monitorare l'utilizzo delle risorse, esaminare il carico di lavoro in esecuzione sull'istanza database ed eseguire le ottimizzazioni dove necessario. Se le ottimizzazioni non migliorano le metriche dell'istanza e non mitigano l'esaurimento delle risorse, valutare la possibilità di aumentare l'istanza database prima che raggiunga i suoi limiti. Per ulteriori informazioni sulle classi di istanza database disponibili e le relative specifiche, consultare Classi di istanze database Amazon Aurora.

Suggerimenti per le funzionalità MySQL in Aurora MySQL

Le seguenti funzionalità sono disponibili in Aurora MySQL per la compatibilità con MySQL. Tuttavia, hanno problemi di prestazioni, scalabilità, stabilità o compatibilità nell'ambiente Aurora. Pertanto, suggeriamo di seguire specifiche linee guida quando si utilizzano queste caratteristiche. Ad esempio, non è consigliabile utilizzare determinate funzionalità per le distribuzioni Aurora di produzione.

Argomenti

- Utilizzo della replica multithread in Aurora MySQL
- Invocazione delle funzioni AWS Lambda usando funzioni MySQL native
- Evitatre le transazioni XA con Amazon Aurora MySQL
- Mantenere le chiavi esterne attivate durante le istruzioni DML
- Configurazione della frequenza di svuotamento del buffer dei registri

- Contenimento e risoluzione dei problemi di deadlock di Aurora MySQL

Utilizzo della replica multithread in Aurora MySQL

Con la replica del log binario multithread, un thread SQL legge gli eventi dal log di inoltro e li mette in coda per l'applicazione dei thread SQL worker. I thread di lavoro SQL sono gestiti da un thread coordinatore. Gli eventi di log binari vengono applicati in parallelo quando possibile.

La replica multithread è supportata in Aurora MySQL versione 3 e in Aurora MySQL 2.12.1 e versioni successive.

Per le versioni precedenti alla 3.04 di Aurora MySQL, Aurora utilizza la replica a thread singolo per impostazione predefinita quando un cluster di database Aurora MySQL viene utilizzato come replica di lettura per la replica dei log binari.

Le versioni precedenti di Aurora MySQL versione 2 hanno ereditato diversi problemi relativi alla replica multithread da MySQL Community Edition. Per quelle versioni, ti consigliamo di non utilizzare la replica multithread in produzione.

Qualora decidessi di utilizzarla, è consigliabile testarla accuratamente.

Per ulteriori informazioni sull'uso della replica in Amazon Aurora, consulta Replica con Amazon Aurora. Per ulteriori informazioni sulla replica multithread in Aurora MySQL, consulta Replica dei log binari multithread.

Invocazione delle funzioni AWS Lambda usando funzioni MySQL native

Consigliamo di utilizzare le funzioni native di MySQL `lambda_sync` e `lambda_async` per richiamare le funzioni Lambda.

In caso di utilizzo della procedura `mysql.lambda_async` obsoleta, ti consigliamo di eseguire il wrapping delle chiamate alla procedura `mysql.lambda_async` in una stored procedure. Questa stored procedure può essere chiamata da varie origini, ad esempio trigger o codice client. Questo approccio può evitare problemi di resistenza non corrispondente e rendere più semplice per i programmatori del database richiamare le funzioni Lambda.

Per ulteriori informazioni sul richiamo delle funzioni Lambda da Amazon Aurora, consulta Chiamare una funzione Lambda da un cluster DB Amazon Aurora MySQL.

Evitatre le transazioni XA con Amazon Aurora MySQL

Ti consigliamo di non utilizzare transazioni eXtended Architecture (XA) con Aurora MySQL, poiché possono provocare tempi di ripristino lunghi se XA si trova in stato `PREPARED`. Se devi utilizzare transazioni XA con Aurora MySQL, segui queste best practice:

- Non lasciare una transazione XA aperta nello stato `PREPARED`.
- Mantieni le dimensioni delle transazioni XA più ridotte possibile.

Per ulteriori informazioni sull'utilizzo di transazioni XA con MySQL, consulta XA Transactions nella documentazione di MySQL.

Mantenere le chiavi esterne attivate durante le istruzioni DML

Consigliamo fortemente di non eseguire alcuna istruzione DDL (Data Definition Language) quando la variabile `foreign_key_checks` è impostata su `0` (disattivata).

Per inserire o aggiornare righe che richiedono una violazione temporanea delle chiavi esterne, procedere nel seguente modo:

1. Imposta `foreign_key_checks` su `0`.
2. Apportare le modifiche alle istruzioni DML (Data Manipulation Language).
3. Assicurarsi che le modifiche apportate non violino eventuali vincoli delle chiavi esterne.
4. Impostare `foreign_key_checks` su `1` (attivato).

È inoltre necessario attenersi alle ulteriori best practice seguenti per i vincoli delle chiavi esterne:

- Assicurarsi che le applicazioni client non impostino la variabile `foreign_key_checks` su `0` come parte della variabile `init_connect`.
- Se un ripristino da un backup logico come `mysqldump` non riesce o è incompleto, assicurarsi che `foreign_key_checks` sia impostato su `1` prima di avviare eventuali altre operazioni nella stessa sessione. Un backup logico imposta `foreign_key_checks` su `0` all'avvio.

Configurazione della frequenza di svuotamento del buffer dei registri

In MySQL Community Edition, per rendere le transazioni durevoli, il buffer dei registri InnoDB deve essere svuotato in un'archiviazione durevole. Per configurare la frequenza di svuotamento del buffer dei registri su disco si utilizza il parametro `innodb_flush_log_at_trx_commit`.

Quando si imposta il parametro `innodb_flush_log_at_trx_commit` sul valore predefinito di 1, il buffer dei registri viene svuotato ad ogni commit della transazione. Questa impostazione consente di mantenere il database conforme ad ACID. Ti consigliamo di mantenere il valore predefinito di 1.

La modifica del parametro `innodb_flush_log_at_trx_commit` in un valore non predefinito può contribuire a ridurre la latenza DML (Data Manipulation Language), ma sacrifica la durabilità dei record di log. Questa mancanza di durabilità rende il database non conforme ad ACID. È consigliabile che i database siano conformi ad ACID per evitare il rischio di perdita di dati in caso di riavvio di un server. Per ulteriori informazioni su questo parametro, consultare innodb_flush_log_at_trx_commit nella documentazione di MySQL.

In Aurora MySQL, l'elaborazione dei registri redo continuerà nel livello di archiviazione, pertanto sull'istanza database non si verificherà lo svuotamento nei file di registro. Quando viene emesso un comando di scrittura, i registri redo vengono inviati direttamente dall'istanza database di scrittura al volume del cluster Aurora. Le sole scritture che attraversano la rete sono i record di registro redo. Nessuna pagina viene mai scritta dal livello database.

Per impostazione predefinita, ogni thread che esegue il commit di una transazione attende la conferma dal volume del cluster Aurora. Questa conferma indica che tale record e tutti i record di registro redo precedenti sono stati scritti e hanno raggiunto il quorum. La persistenza dei record di registro e il raggiungimento del quorum rendono la transazione durevole, tramite commit automatico o commit esplicito. Per ulteriori informazioni sull'architettura di archiviazione Aurora, consultare Amazon Aurora storage demystified.

A differenza di MySQL Community Edition, Aurora MySQL non scarica i registri nei file di dati. Tuttavia, è possibile utilizzare il parametro `innodb_flush_log_at_trx_commit` per allentare i vincoli di durabilità durante la scrittura dei record di registro redo nel volume del cluster Aurora.

Per Aurora MySQL versione 2:

- `innodb_flush_log_at_trx_commit` = 0 o 2: il database non attende la conferma che i record di log di redo vengono scritti nel volume del cluster Aurora.
- `innodb_flush_log_at_trx_commit` = 1: il database attende la conferma che i record di log di redo vengono scritti nel volume del cluster Aurora.

Per Aurora MySQL versione 3:

- `innodb_flush_log_at_trx_commit` = 0: il database non attende la conferma che i record di log di redo vengono scritti nel volume del cluster Aurora.
- `innodb_flush_log_at_trx_commit` = 1 o 2: il database attende la conferma che i record di log di redo vengono scritti nel volume del cluster Aurora.

Pertanto, per ottenere in Aurora MySQL versione 3 lo stesso comportamento non predefinito che avresti in Aurora MySQL versione 2 con il valore impostato su 0 o 2, imposta il parametro su 0.

Sebbene queste impostazioni possano ridurre la latenza DML al client, possono anche causare la perdita di dati in caso di failover o riavvio. Pertanto, ti consigliamo di mantenere il parametro `innodb_flush_log_at_trx_commit` impostato sul valore predefinito di 1.

Sebbene la perdita di dati si possa verificare sia in MySQL Community Edition sia in Aurora MySQL, il comportamento varia in ogni database a causa delle diverse architetture. Queste differenze architetturali possono portare a vari gradi di perdita di dati. Per essere certi che il database sia conforme ad ACID, imposta sempre `innodb_flush_log_at_trx_commit` su 1.

> **Note**
>
> In Aurora MySQL versione 3, prima di poter impostare `innodb_flush_log_at_trx_commit` su un valore diverso da 1, devi prima impostare il valore di `innodb_trx_commit_allow_data_loss` su 1. In questo modo, riconosci l'esistenza del rischio di perdita dei dati.

Contenimento e risoluzione dei problemi di deadlock di Aurora MySQL

Gli utenti che eseguono carichi di lavoro che restituiscono regolarmente violazioni dei vincoli su indici secondari univoci o chiavi esterne quando modificano contemporaneamente i record sulla stessa pagina di dati, possono riscontrare un aumento dei deadlock e dei timeout di attesa dei blocchi. Questi deadlock e timeout sono dovuti a una correzione di bug di MySQL Community Edition.

Questa correzione è inclusa in MySQL Community Edition 5.7.26 e versioni successive ed è stata sottoposta al backport in Aurora MySQL 2.10.3 e versioni successive. La correzione è necessaria per applicare la serializzabilità, implementando per questi tipi di operazioni del linguaggio DML (Data Manipulation Language) un blocco aggiuntivo per le modifiche apportate ai record in una tabella

InnoDB. Questo problema è stato individuato nell'ambito di un'indagine sui problemi di deadlock introdotti da una precedente correzione di bug di MySQL Community Edition.

La correzione ha modificato la gestione interna per il rollback parziale dell'aggiornamento di una tupla (riga) nel motore di archiviazione InnoDB. Le operazioni che generano violazioni dei vincoli su chiavi esterne o indici secondari univoci causano un rollback parziale. Sono incluse, a titolo esemplificativo e non esaustivo, le istruzioni `INSERT...ON DUPLICATE KEY UPDATE`, `REPLACE INTO`, e `INSERT IGNORE` simultanee (upsert).

In questo contesto, il rollback parziale non si riferisce alle transazioni a livello di applicazione, ma piuttosto a un rollback interno di InnoDB delle modifiche di un indice cluster, quando si verifica una violazione dei vincoli. Ad esempio, durante un'operazione upsert viene individuato un valore chiave duplicato.

In una normale operazione di inserimento, InnoDB crea atomicamente le voci degli indici secondari e in cluster per ciascun indice. Se InnoDB rileva un valore duplicato in un indice secondario univoco durante un'operazione di upsert, la voce inserita nell'indice in cluster deve essere ripristinata (rollback parziale) e l'aggiornamento deve quindi essere applicato alla riga duplicata esistente. Durante questa fase di rollback parziale interno, InnoDB deve bloccare ogni record considerato come parte dell'operazione. La correzione garantisce la serializzabilità delle transazioni introducendo un blocco aggiuntivo dopo il rollback parziale.

Riduzione al minimo dei deadlock di InnoDB

Puoi utilizzare i seguenti approcci per ridurre la frequenza dei deadlock nell'istanza database. Altri esempi possono sono disponibili nella documentazione di MySQL.

1. Per ridurre le possibilità di deadlock, esegui il commit delle transazioni immediatamente dopo aver apportato una serie di modifiche correlate. Puoi farlo suddividendo le transazioni di grandi dimensioni (aggiornamenti di più righe tra i commit) in transazioni più piccole. Se stai inserendo righe in batch, prova a ridurre le dimensioni degli inserimenti batch, specialmente quando usi le operazioni upsert menzionate in precedenza.

   Per ridurre il numero di possibili rollback parziali, puoi provare alcuni dei seguenti approcci:

   a. Sostituisci le operazioni di inserimento in batch con l'inserimento di una riga alla volta. In tal modo puoi ridurre il periodo di tempo in cui i blocchi vengono mantenuti dalle transazioni che possono avere conflitti.
   b. Invece di usare `REPLACE INTO`, riscrivi l'istruzione SQL come transazione a più istruzioni come la seguente:

```
BEGIN;
DELETE conflicting rows;
INSERT new rows;
COMMIT;
```

c. Invece di usare `INSERT...ON DUPLICATE KEY UPDATE`, riscrivi l'istruzione SQL come transazione a più istruzioni come la seguente:

```
BEGIN;
SELECT rows that conflict on secondary indexes;
UPDATE conflicting rows;
INSERT new rows;
COMMIT;
```

2. Evita le transazioni di lunga durata, attive o inattive, che potrebbero rimanere bloccate. Sono incluse le sessioni client MySQL interattive che potrebbero essere aperte per un periodo prolungato con una transazione non sottoposta a commit. Quando si ottimizzano le dimensioni delle transazioni o dei batch, l'impatto può variare in base a una serie di fattori come la concorrenza, il numero di duplicati e la struttura delle tabelle. Qualsiasi modifica deve essere implementata e testata in base al carico di lavoro.
3. In alcune situazioni, i deadlock possono verificarsi quando due transazioni tentano di accedere agli stessi set di dati, in una o più tabelle, in ordini diversi. Per evitare queste situazioni, è possibile modificare le transazioni per accedere ai dati nello stesso ordine, serializzando così l'accesso. Ad esempio, crea una coda di transazioni da completare. Questo approccio può aiutare a evitare deadlock quando si verificano più transazioni contemporaneamente.
4. L'aggiunta di indici scelti con attenzione alle tabelle può migliorare la selettività e ridurre la necessità di accedere alle righe, con conseguente riduzione dei blocchi.
5. Se riscontri un blocco di intervallo, puoi impostare il livello di isolamento della transazione su `READ COMMITTED` per la sessione o la transazione per evitarlo. Per ulteriori informazioni sui livelli di isolamento di InnoDB e sui relativi comportamenti, consulta Livelli di isolamento delle transazioni nella documentazione di MySQL.

> **Note**
>
> Sebbene sia possibile prendere precauzioni per ridurre la possibilità che si verifichino, i deadlock sono un comportamento previsto del database e possono comunque verificarsi. Le applicazioni devono disporre della logica necessaria per gestire i deadlock quando si

verificano. Ad esempio, implementa la logica dei tentativi e del backup nell'applicazione. È consigliabile risolvere la causa principale del problema, ma se si verifica un deadlock, l'applicazione ha la possibilità di attendere e ritentare.

Monitoraggio dei deadlock di InnoDB

I deadlock possono verificarsi in MySQL quando le transazioni delle applicazioni cercano di accettare blocchi a livello di tabella e di riga in un modo che si traduce in un'attesa circolare. Un deadlock occasionale di InnoDB non è necessariamente un problema, perché il motore di archiviazione InnoDB rileva immediatamente la condizione e ripristina automaticamente le transazioni. Se riscontri spesso dei deadlock, ti consigliamo di rivedere e modificare l'applicazione per contenere i problemi di prestazioni ed evitare i deadlock. Quando il rilevamento dei deadlock è attivato (impostazione predefinita), InnoDB rileva automaticamente i deadlock delle transazioni e ripristina una o più transazioni per interrompere il deadlock. InnoDB cerca di selezionare piccole transazioni da ripristinare, in cui la dimensione della transazione è determinata dal numero di righe inserite, aggiornate o eliminate.

- Istruzione `SHOW ENGINE`: l'istruzione `SHOW ENGINE INNODB STATUS \G` contiene i dettagli del deadlock più recente riscontrato nel database dall'ultimo riavvio.
- Log degli errori MySQL: se si verificano frequenti deadlock in cui l'output dell'istruzione `SHOW ENGINE` è inadeguato, puoi attivare il parametro del cluster di database innodb_print_all_deadlocks.

  Quando questo parametro è attivo, le informazioni su tutti i deadlock nelle transazioni utente InnoDB vengono registrate nel log degli errori di Aurora MySQL.
- Metriche di Amazon CloudWatch: ti consigliamo inoltre di monitorare in modo proattivo i deadlock utilizzando la metrica CloudWatch `Deadlocks`. Per ulteriori informazioni, consulta Parametri a livello di istanza per Amazon Aurora.
- File di log Amazon CloudWatch: con i file di log CloudWatch puoi esaminare le metriche, eseguire analisi dei dati di log e creare allarmi in tempo reale. Per ulteriori informazioni, consulta Monitorare gli errori in Amazon Aurora MySQL e Amazon RDS per MySQL con Amazon CloudWatch e inviare notifiche tramite Amazon SNS.

  Utilizzando File di log CloudWatch con `innodb_print_all_deadlocks` attivo, puoi configurare gli allarmi per avvisare quando il numero di deadlock supera una determinata soglia. Per definire una soglia, ti consigliamo di osservare le tendenze e utilizzare un valore basato sul normale carico di lavoro.

- Approfondimenti sulle prestazioni: quando utilizzi Approfondimenti sulle prestazioni, puoi monitorare le metriche `innodb_deadlocks` e `innodb_lock_wait_timeout`. Per ulteriori informazioni su tali parametri, consulta Contatori non nativi per Aurora MySQL.

Valutazione dell'utilizzo dell'istanza database per Aurora MySQL con le metriche di Amazon CloudWatch

Puoi utilizzare le metriche di CloudWatch per monitorare il throughput dell'istanza database in uso e determinare se la classe di istanza database fornisce risorse sufficienti per le tue applicazioni. Per informazioni sui limiti delle classi di istanza database, consulta Specifiche hardware per le classi di istanza database per Aurora. È possibile individuare le specifiche della classe di istanza database in uso per scoprire le prestazioni di rete.

Se l'utilizzo dell'istanza database è prossimo al limite della classe di istanza, le prestazioni potrebbero iniziare a rallentare. I parametri di CloudWatch possono verificare questa situazione in modo da poter pianificare un aumento manuale a una classe di istanza più grande.

Combina i seguenti valori dei parametri di CloudWatch per determinare se l'utilizzo è prossimo al limite della classe di istanza:

- NetworkThroughput: la velocità di trasmissione effettiva della rete in entrata e in uscita dei client per ogni istanza nel cluster database Aurora. La velocità di trasmissione effettiva non include il traffico di rete tra le istanze nel cluster database e il volume del cluster.
- StorageNetworkThroughput: la velocità di trasmissione effettiva della rete in entrata e in uscita del sottosistema di archiviazione Aurora da ogni istanza nel cluster database Aurora.

Aggiungi NetworkThroughput a StorageNetworkThroughput per trovare la velocità di trasmissione effettiva della rete in entrata e in uscita del sottosistema di archiviazione Aurora da ogni istanza nel cluster database Aurora. Il limite della classe dell'istanza deve essere maggiore della somma di questi due parametri combinati.

È possibile utilizzare i seguenti parametri per esaminare ulteriori dettagli del traffico di rete proveniente dalle applicazioni client durante l'invio e la ricezione:

- NetworkReceiveThroughput: la quantità di throughput della rete in entrata dai client per ogni istanza nel cluster di database Aurora MySQL. Questo throughput non include il traffico di rete tra le istanze nel cluster di database e il volume del cluster.

- NetworkTransmitThroughput: la velocità di trasmissione effettiva della rete in uscita dei client da ogni istanza nel cluster database Aurora. Questo throughput non include il traffico di rete tra le istanze nel cluster di database e il volume del cluster.
- StorageNetworkReceiveThroughput: la velocità di trasmissione effettiva della rete in entrata del sottosistema di archiviazione Aurora da ogni istanza nel cluster database.
- StorageNetworkTransmitThroughput: la velocità di trasmissione effettiva della rete in uscita del sottosistema di archiviazione Aurora da ogni istanza nel cluster database.

Somma tutte queste metriche per confrontare l'utilizzo della rete e il limite della classe di istanza database. Il limite della classe di istanza deve essere maggiore della somma di questi parametri combinati.

I limiti della rete e l'utilizzo della CPU per l'archiviazione sono direttamente correlati Quando il throughput di rete aumenta, cresce anche l'utilizzo della CPU. Il monitoraggio dell'utilizzo della CPU e della rete fornisce informazioni su come e perché le risorse vengono esaurite.

Per ridurre al minimo l'utilizzo della rete, puoi prendere in considerazione quanto segue:

- L'uso di una classe di istanza database più grande.
- La suddivisione delle richieste di scrittura in batch per ridurre le transazioni in generale.
- L'instradamento del carico di lavoro di sola lettura a un'istanza di sola lettura.
- L'eliminazione degli indici non utilizzati.

Risoluzione dei problemi alle prestazioni del database Amazon Aurora MySQL

Questo argomento si concentra su alcuni problemi comuni relativi alle prestazioni del database Aurora MySQL e su come risolverli o raccogliere informazioni per porvi rimedio rapidamente. Dividiamo le prestazioni del database in due categorie:

- Prestazioni del server: l'intero server di database è più lento.
- Prestazioni delle query: l'esecuzione di una o più query richiede più tempo.

Opzioni di monitoraggio AWS

Per semplificare la risoluzione dei problemi, consigliamo di utilizzare le seguenti opzioni di monitoraggio AWS:

- Amazon CloudWatch: Amazon CloudWatch monitora le risorse AWS e le applicazioni che esegui su AWS in tempo reale. Puoi utilizzare CloudWatch per raccogliere e tenere traccia delle metriche, che sono delle variabili che si possono misurare per le risorse e le applicazioni. Per ulteriori informazioni, consulta Che cos'è Amazon CloudWatch?

  È possibile visualizzare tutte le metriche di sistema e le informazioni sui processi per le tue istanze database sulla Console di gestione AWS. Puoi configurare il cluster DB Aurora MySQL per pubblicare log generali, delle query lente, di audit e degli errori in un gruppo di log in Amazon CloudWatch Logs. Ciò consente di visualizzare le tendenze, conservare i log se un host è interessato e creare una base di riferimento per prestazioni "normali" per identificare facilmente anomalie o modifiche. Per ulteriori informazioni, consulta Pubblicazione di log Amazon Aurora MySQL in Amazon CloudWatch Logs.
- Monitoraggio avanzato: per abilitare metriche Amazon CloudWatch aggiuntive per un database Aurora MySQL, attiva Monitoraggio avanzato. Quando crei o modifichi un cluster di database Aurora, seleziona Abilita il monitoraggio avanzato. Ciò consente ad Aurora di pubblicare le metriche delle prestazioni su CloudWatch. Alcune delle metriche chiave disponibili includono l'utilizzo della CPU, le connessioni al database, l'utilizzo dell'archiviazione e la latenza delle query. Queste possono aiutare a identificare i colli di bottiglia delle prestazioni.

  La quantità di informazioni trasferite per un'istanza database è direttamente proporzionale alla granularità definita per Monitoraggio avanzato. Un intervallo di monitoraggio più piccolo comporta report più frequenti sulle metriche del sistema operativo e aumenta i costi di monitoraggio. Per

gestire i costi, imposta granularità diverse per istanze diverse nei tuoi Account AWS. La granularità predefinita alla creazione di un'istanza è di 60 secondi. Per ulteriori informazioni, consulta Costo di Enhanced Monitoring (monitoraggio avanzato).

- Approfondimenti sulle prestazioni: puoi visualizzare tutte le metriche delle chiamate al database. Ciò include i blocchi del database, le attese e il numero di righe elaborate, tutti elementi che è possibile utilizzare per la risoluzione dei problemi. Quando crei o modifichi un cluster di database Aurora, seleziona Attiva Performance Insights. Per impostazione predefinita, Approfondimenti sulle prestazioni ha un periodo di conservazione dei dati di 7 giorni, ma può essere personalizzato per analizzare le tendenze delle prestazioni a lungo termine. Per una conservazione superiore a 7 giorni, è necessario passare al livello a pagamento. Per ulteriori informazioni, consulta Prezzi di Approfondimenti sulle prestazioni. È possibile impostare separatamente il periodo di conservazione dei dati per ogni istanza database Aurora. Per ulteriori informazioni, consulta Monitoraggio del carico DB con Performance Insights su Amazon Aurora.

Cause più comuni dei problemi di prestazioni del database Aurora MySQL

È possibile utilizzare i seguenti passaggi per risolvere i problemi di prestazioni nel database Aurora MySQL. Elenchiamo questi passaggi nell'ordine logico di indagine, ma non sono pensati per essere lineari. Una scoperta potrebbe passare da una fase all'altra, il che consente una serie di percorsi investigativi.

1. Carico di lavoro: comprendi il carico di lavoro del tuo database.
2. Registrazione di log: esamina tutti i log del database.
3. Connessioni al database: assicurati che le connessioni tra le applicazioni e il database siano affidabili.
4. Prestazioni delle query: esamina i piani di esecuzione delle query per vedere se sono cambiati. Le modifiche al codice possono causare la modifica dei piani.

Risoluzione dei problemi di carico di lavoro per i database Aurora MySQL

Il carico di lavoro del database può essere visualizzato come lettura e scrittura. Una volta compreso il “normale” carico di lavoro del database, è possibile ottimizzare le query e il server di database per soddisfare la domanda man mano che questa cambia. Esistono diversi motivi per cui le prestazioni possono cambiare, quindi il primo passo è capire cosa è cambiato.

- È stato effettuato un aggiornamento della versione principale o secondario?

  Un aggiornamento della versione principale include modifiche al codice del motore, in particolare nell’ottimizzatore, che possono modificare il piano di esecuzione delle query. Quando si aggiornano le versioni del database, in particolare le versioni principali, è molto importante analizzare il carico di lavoro del database e ottimizzarlo di conseguenza. L’ottimizzazione può comportare l’ottimizzazione e la riscrittura delle query o l’aggiunta e l’aggiornamento delle impostazioni dei parametri, a seconda dei risultati dei test. Capire cosa sta causando l’impatto ti consentirà di iniziare a concentrarti su quell’area specifica.

  Per ulteriori informazioni, consulta Novità di MySQL 8.0 e Server and status variables and options added, deprecated, or removed in MySQL 8.0 nella documentazione di MySQL e Confronto tra Aurora MySQL versione 2 e Aurora MySQL versione 3
- Si è verificato un aumento dei dati in fase di elaborazione (numero di righe)?
- Sono presenti più query in esecuzione contemporaneamente?
- Sono state apportate modifiche allo schema o al database?
- Sono stati rilevati difetti o correzioni del codice?

Indice

- Parametri dell’host dell’istanza
  - Utilizzo CPU
  - Utilizzo della memoria
  - Throughput di rete
- Parametri del database
- Risoluzione dei problemi di utilizzo della memoria per i database Aurora MySQL
  - Esempio 1: utilizzo continuo ed elevato della memoria
  - Esempio 2: picchi di memoria transitori
  - Esempio 3: la memoria liberabile diminuisce continuamente e non viene recuperata

- Risoluzione dei problemi di memoria insufficiente per i database Aurora MySQL

Parametri dell'host dell'istanza

Monitora le metriche dell'host dell'istanza come CPU, memoria e attività di rete per capire se è stata apportata una modifica del carico di lavoro. Esistono due concetti principali per comprendere le modifiche del carico di lavoro:

- Utilizzo: utilizzo di un dispositivo, ad esempio CPU o disco. Può essere basato sul tempo o sulla capacità.
  - Basato sul tempo: la quantità di tempo in cui una risorsa è occupata in un determinato periodo di osservazione.
  - Basato sulla capacità: la quantità di throughput che un sistema o un componente è in grado di fornire, espressa in percentuale della sua capacità.
- Saturazione: il grado in cui una risorsa richiede più lavoro di quanto ne possa elaborare. Quando l'utilizzo basato sulla capacità raggiunge il 100%, il lavoro aggiuntivo non può essere elaborato e deve essere messo in coda.

Utilizzo CPU

È possibile fare riferimento ai seguenti strumenti per identificare l'utilizzo e la saturazione della CPU:

- CloudWatch fornisce la metrica `CPUUtilization`. Se raggiunge il 100%, l'istanza è satura. Tuttavia, le metriche di CloudWatch hanno una media di 1 minuto e mancano di granularità.

  Per ulteriori informazioni sui parametri di CloudWatch, consulta Parametri a livello di istanza per Amazon Aurora.
- Monitoraggio avanzato fornisce le metriche restituite dal comando del sistema operativo `top`. Mostra le medie di carico e i seguenti stati della CPU, con una granularità di 1 secondo:
  - `Idle (%)` = Tempo di inattività
  - `IRQ (%)` = Interruzioni del software
  - `Nice (%)` = Periodo di tempo in cui i processi hanno una priorità nice.
  - `Steal (%)` = Tempo dedicato ad altri tenant (in relazione alla virtualizzazione)
  - `System (%)` = Orario sistema
  - `User (%)` = Orario utente

- `Wait (%)` = Attesa I/O

Per ulteriori informazioni sulle metriche di Monitoraggio avanzato, consulta Parametri del sistema operativo per Aurora.

Utilizzo della memoria

Se il sistema è sotto pressione in termini di memoria e il consumo di risorse sta raggiungendo la saturazione, si dovrebbe verificare un elevato grado di errori di scansione delle pagine, paginazione, scambio ed esaurimento della memoria delle pagine.

È possibile fare riferimento ai seguenti strumenti per identificare l'utilizzo e la saturazione della memoria:

CloudWatch fornisce la metrica `FreeableMemory` che mostra quanta memoria può essere recuperata svuotando alcune cache del sistema operativo e la memoria attualmente disponibile.

Per ulteriori informazioni sui parametri di CloudWatch, consulta Parametri a livello di istanza per Amazon Aurora.

Monitoraggio avanzato fornisce le seguenti metriche che possono aiutarti a identificare i problemi di utilizzo della memoria:

- `Buffers (KB)`: la quantità di memoria utilizzata per il buffering delle richieste di I/O prima della scrittura sul dispositivo di archiviazione, in kilobyte.
- `Cached (KB)`: la quantità di memoria utilizzata per la memorizzazione nella cache dell'I/O basato sul file system.
- `Free (KB)`: la quantità di memoria non assegnata, in kilobyte.
- `Swap`: memorizzato nella cache, gratuito e totale.

Ad esempio, se vedi che l'istanza database utilizza memoria `Swap`, la quantità totale di memoria per il carico di lavoro è maggiore di quella attualmente disponibile sull'istanza. Ti consigliamo di aumentare le dimensioni dell'istanza database o di ottimizzare il carico di lavoro per utilizzare meno memoria.

Per ulteriori informazioni sulle metriche di Monitoraggio avanzato, consulta Parametri del sistema operativo per Aurora.

Per informazioni più dettagliate sull'utilizzo di Schema delle prestazioni e dello schema `sys` per determinare quali connessioni e componenti stanno utilizzando la memoria, consulta Risoluzione dei problemi di utilizzo della memoria per i database Aurora MySQL

Throughput di rete

CloudWatch fornisce le seguenti metriche per il throughput totale di rete, il tutto calcolato in media su 1 minuto:

- `NetworkReceiveThroughput`: la quantità di throughput di rete ricevuto dai client da ogni istanza nel cluster di database Aurora.
- `NetworkTransmitThroughput`: la quantità di throughput di rete inviato ai client da ogni istanza nel cluster di database Aurora.
- `NetworkThroughput`: la quantità di throughput di rete ricevuto dai client e trasmesso agli stessi da ogni istanza nel cluster di database Aurora.
- `StorageNetworkReceiveThroughput`: la quantità di throughput di rete ricevuto dal sottosistema di archiviazione Aurora mediante ogni istanza nel cluster di database.
- `StorageNetworkTransmitThroughput`: la quantità di throughput di rete inviata al sottosistema di archiviazione Aurora mediante ogni istanza nel cluster di database Aurora.
- `StorageNetworkThroughput`: la quantità di throughput di rete ricevuta dal sottosistema di archiviazione Aurora e inviata allo stesso mediante ogni istanza nel cluster di database Aurora.

Per ulteriori informazioni sui parametri di CloudWatch, consulta Parametri a livello di istanza per Amazon Aurora.

Monitoraggio avanzato fornisce i grafi `network` ricevuti (RX) e trasmessi (TX), con una granularità fino a 1 secondo.

Per ulteriori informazioni sulle metriche di Monitoraggio avanzato, consulta Parametri del sistema operativo per Aurora.

Parametri del database

Esamina le seguenti metriche CloudWatch per le modifiche del carico di lavoro:

- `BlockedTransactions`: il numero medio di transazioni nel database bloccate ogni secondo.
- `BufferCacheHitRatio` la percentuale di richieste gestite dalla cache del buffer.

- `CommitThroughput`: il numero medio di operazioni di conferma al secondo.
- `DatabaseConnections`: il numero di connessioni di rete client all'istanza del database.
- `Deadlocks`: il numero medio di deadlock nel database al secondo.
- `DMLThroughput`: il numero medio delle inserzioni, degli aggiornamenti e delle eliminazioni al secondo.
- `ResultSetCacheHitRatio`: la percentuale di richieste gestite dalla cache del buffer.
- `RollbackSegmentHistoryListLength`: i log di undo che registrano le transazioni sottoposte a commit con record contrassegnati per l'eliminazione.
- `RowLockTime`: il tempo totale impiegato per l'acquisizione di blocchi di riga per le tabelle InnoDB.
- `SelectThroughput`: numero medio di query di selezione al secondo.

Per ulteriori informazioni sui parametri di CloudWatch, consulta Parametri a livello di istanza per Amazon Aurora.

Considera le seguenti domande quando esamini il carico di lavoro:

1. Sono state apportate modifiche recenti nella classe dell'istanza database, ad esempio la riduzione della dimensione dell'istanza da 8xlarge a 4xlarge o il passaggio da db.r5 a db.r6?
2. È possibile creare un clone e riprodurre il problema o si verifica solo su quell'istanza?
3. Si verifica un esaurimento delle risorse del server, un elevato esaurimento della CPU o della memoria? In caso affermativo, ciò potrebbe significare che è necessario hardware aggiuntivo.
4. Una o più query richiedono più tempo?
5. Le modifiche sono causate da un aggiornamento, in particolare da un aggiornamento della versione principale? In caso affermativo, confronta le metriche precedenti e successive all'aggiornamento.
6. Sono state apportate modifiche al numero di istanze database di lettura?
7. Hai abilitato la registrazione di log generali, di audit o binari? Per ulteriori informazioni, consulta Registrazione di log per i database Aurora MySQL.
8. Hai abilitato, disabilitato o modificato l'uso della replica dei log binari (binlog)?
9. Esistono transazioni di lunga durata che contengono un gran numero di blocchi di righe? Esamina la lunghezza dell'elenco della cronologia di InnoDB (HLL) per indicazioni di transazioni di lunga durata.

Per ulteriori informazioni, consulta La lunghezza dell'elenco della cronologia di InnoDB è aumentata in modo significativo il post sul blog Why is my SELECT query running slowly on my Amazon Aurora MySQL DB cluster?

a. Se un HLL di grandi dimensioni è causato da una transazione di scrittura, significa che i log UNDO si stanno accumulando (non vengono puliti regolarmente). In una transazione di scrittura di grandi dimensioni, questo accumulo può crescere rapidamente. In MySQL, UNDO è memorizzato nel tablespace SYSTEM. Il tablespace SYSTEM non è riducibile. Il log UNDO potrebbe far crescere il tablespace SYSTEM fino a diversi GB o addirittura TB. Dopo l'eliminazione, rilascia lo spazio allocato eseguendo un backup logico (dump) dei dati, quindi importa il dump in una nuova istanza database.

b. Se un HLL di grandi dimensioni è causato da una transazione di lettura (query a esecuzione prolungata), può significare che la query sta utilizzando una grande quantità di spazio temporaneo. Rilascia lo spazio temporaneo riavviando. Esamina le metriche del database Approfondimenti sulle prestazioni per eventuali modifiche nella sezione Temp, ad esempio `created_tmp_tables`. Per ulteriori informazioni, consulta Monitoraggio del carico DB con Performance Insights su Amazon Aurora.

10. È possibile suddividere le transazioni di lunga durata in transazioni più piccole che modificano un minor numero di righe?

11. Ci sono cambiamenti nelle transazioni bloccate o aumenti delle situazioni di stallo? Esamina le metriche del database Approfondimenti sulle prestazioni per eventuali modifiche alle variabili di stato nella sezione Locks, ad esempio `innodb_row_lock_time`, `innodb_row_lock_waits` e `innodb_dead_locks`. Usa intervalli di 1 minuto o 5 minuti.

12. I tempi di attesa sono aumentati? Esamina gli eventi e i tipi di attesa di Approfondimenti sulle prestazioni utilizzando intervalli di 1 minuto o 5 minuti. Analizza i principali eventi di attesa e verifica se sono correlati alle modifiche del carico di lavoro o ai conflitti del database. Ad esempio, `buf_pool mutex` indica la contesa del pool di buffer. Per ulteriori informazioni, consulta Regolazione di Aurora MySQL con eventi di attesa.

Risoluzione dei problemi di utilizzo della memoria per i database Aurora MySQL

Sebbene CloudWatch, Monitoraggio avanzato e Approfondimenti sulle prestazioni forniscano una buona panoramica dell'utilizzo della memoria a livello di sistema operativo, ad esempio la quantità di memoria utilizzata dal processo del database, non consentono di analizzare quali connessioni o componenti all'interno del motore potrebbero causare questo utilizzo di memoria.

Per risolvere questo problema, è possibile utilizzare lo schema delle prestazioni e lo schema `sys`. Nella versione 3 di Aurora MySQL, la strumentazione della memoria viene abilitata per impostazione predefinita quando viene abilitato lo schema delle prestazioni. Nella versione 2 di Aurora MySQL, solo la strumentazione della memoria per l'utilizzo della memoria dello schema delle prestazioni viene abilitata per impostazione predefinita. Per informazioni sulle tabelle disponibili nello schema delle prestazioni per monitorare l'utilizzo della memoria e abilitare la strumentazione della memoria dello schema delle prestazioni, consulta la pagina relativa alle tabelle di riepilogo della memoria nella documentazione di MySQL. Per ulteriori informazioni sull'utilizzo dello Schema delle prestazioni con Approfondimenti sulle prestazioni, vedi Panoramica dello schema di prestazioni per Approfondimenti sulle prestazioni su Aurora MySQL.

Sebbene nello schema delle prestazioni siano disponibili informazioni dettagliate per monitorare l'utilizzo corrente della memoria, lo schema sys di MySQL offre viste sulle tabelle dello schema delle prestazioni che è possibile utilizzare per individuare rapidamente dove viene utilizzata la memoria.

Nello schema `sys` sono disponibili le seguenti viste per monitorare l'utilizzo della memoria per connessione, componente e query.

Vista	Descrizione
memory_by_host_by_current_bytes	Fornisce informazioni sull'utilizzo della memoria del motore per host. Ciò può essere utile per identificare quali server di applicazioni o host client stanno consumando memoria.
memory_by_thread_by_current_bytes	Fornisce informazioni sull'utilizzo della memoria del motore per ID thread. L'ID thread in MySQL può essere una connessione client o un thread in background. È possibile mappare gli ID thread agli ID connessione MySQL utilizzando la vista sys.processlist o la tabella performance_schema.threads.
memory_by_user_by_current_bytes	Fornisce informazioni sull'utilizzo della memoria del motore per utente. Ciò può essere utile per identificare quali account utente o client stanno consumando memoria.

Vista	Descrizione
memory_global_by_current_bytes	Fornisce informazioni sull'utilizzo della memoria del motore per componente del motore. Ciò può essere utile per identificare l'utilizzo della memoria a livello globale da parte dei buffer o dei componenti del motore. Ad esempio, è possibile visualizzare l'evento `memory/innodb/buf_buf_pool` per il pool di buffer InnoDB o l'evento `memory/sql/Prepared_statement::main_mem_root` per le istruzioni preparate.
memory_global_total	Fornisce una panoramica dell'utilizzo totale della memoria monitorata nel motore del database.

In Aurora MySQL 3.05 e versioni successive, è anche possibile tenere traccia dell'utilizzo massimo della memoria per digest di istruzioni nelle tabelle di riepilogo delle istruzioni dello schema delle prestazioni. Le tabelle di riepilogo delle istruzioni contengono digest di istruzioni normalizzati e statistiche aggregate sull'esecuzione. La colonna `MAX_TOTAL_MEMORY` consente di identificare la quantità massima di memoria utilizzata dal digest di query dall'ultimo ripristino delle statistiche o dal riavvio dell'istanza database. Ciò può essere utile per identificare query specifiche che potrebbero consumare molta memoria.

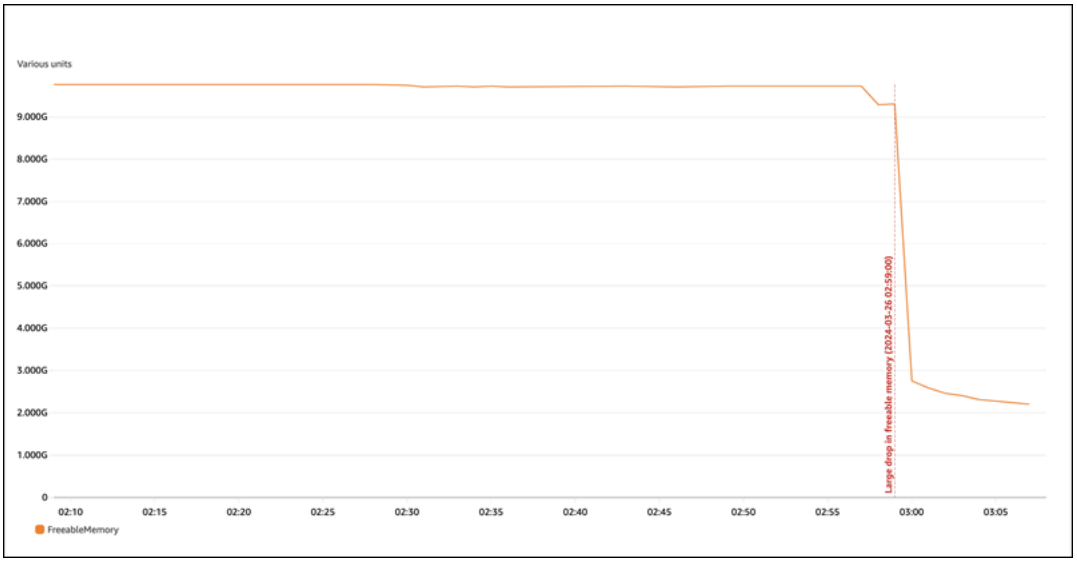
> **Note**
>
> Lo schema delle prestazioni e lo schema `sys` mostrano l'utilizzo corrente della memoria sul server e i valori massimi di memoria consumata per ogni connessione e componente del motore. Poiché lo schema delle prestazioni viene mantenuto in memoria, le informazioni vengono reimpostate al riavvio dell'istanza database. Per mantenere una cronologia nel tempo, è consigliabile configurare il recupero e l'archiviazione di questi dati al di fuori dello schema delle prestazioni.

Argomenti

- Esempio 1: utilizzo continuo ed elevato della memoria
- Esempio 2: picchi di memoria transitori
- Esempio 3: la memoria liberabile diminuisce continuamente e non viene recuperata

Esempio 1: utilizzo continuo ed elevato della memoria

Guardando a livello globale `FreeableMemory` in CloudWatch, possiamo vedere che l'utilizzo della memoria è aumentato notevolmente alle 02:59 UTC del 2024-03-26.



Questo non ci fornisce un quadro completo. Per determinare quale componente utilizza più memoria, è possibile accedere al database e vedere `sys.memory_global_by_current_bytes`. Questa tabella contiene un elenco di eventi di memoria monitorati da MySQL, insieme a informazioni sull'allocazione di memoria per evento. Ogni evento di monitoraggio della memoria inizia con `memory/%`, seguito da altre informazioni sul componente/funzionalità del motore a cui è associato l'evento.

Ad esempio, `memory/performance_schema/%` è per gli eventi di memoria relativi allo schema delle prestazioni, `memory/innodb/%` è per InnoDB e così via. Per ulteriori informazioni sulle convenzioni di denominazione degli eventi, consulta la pagina relativa alle convenzioni di denominazione per la strumentazione dello schema delle prestazioni nella documentazione di MySQL.

Dalla seguente query, possiamo trovare il probabile responsabile in base a `current_alloc`, ma possiamo anche vedere molti eventi `memory/performance_schema/%`.

```
mysql> SELECT * FROM sys.memory_global_by_current_bytes LIMIT 10;
```

```
+----------------------------------------------------------------------------
+---------------+---------------+-------------------+------------+------------
+----------------+
| event_name                                                                 |
 current_count | current_alloc | current_avg_alloc | high_count | high_alloc |
 high_avg_alloc |
+----------------------------------------------------------------------------
+---------------+---------------+-------------------+------------+------------
+----------------+
| memory/sql/Prepared_statement::main_mem_root                               |
 512817 | 4.91 GiB      | 10.04 KiB         |     512823 | 4.91 GiB   | 10.04 KiB
 |
| memory/performance_schema/prepared_statements_instances                    |
   252 | 488.25 MiB    | 1.94 MiB          |        252 | 488.25 MiB | 1.94 MiB       |
| memory/innodb/hash0hash                                                    |
     4 | 79.07 MiB     | 19.77 MiB         |          4 | 79.07 MiB  | 19.77 MiB      |
| memory/performance_schema/events_errors_summary_by_thread_by_error         |
  1028 | 52.27 MiB     | 52.06 KiB         |       1028 | 52.27 MiB  | 52.06 KiB      |
| memory/performance_schema/events_statements_summary_by_thread_by_event_name |
     4 | 47.25 MiB     | 11.81 MiB         |          4 | 47.25 MiB  | 11.81 MiB      |
| memory/performance_schema/events_statements_summary_by_digest              |
     1 | 40.28 MiB     | 40.28 MiB         |          1 | 40.28 MiB  | 40.28 MiB      |
| memory/performance_schema/memory_summary_by_thread_by_event_name           |
     4 | 31.64 MiB     | 7.91 MiB          |          4 | 31.64 MiB  | 7.91 MiB       |
| memory/innodb/memory                                                       |
 15227 | 27.44 MiB     | 1.85 KiB          |      20619 | 33.33 MiB  | 1.66 KiB       |
| memory/sql/String::value                                                   |
 74411 | 21.85 MiB     |  307 bytes        |      76867 | 25.54 MiB  |  348 bytes     |
| memory/sql/TABLE                                                           |
  8381 | 21.03 MiB     | 2.57 KiB          |       8381 | 21.03 MiB  | 2.57 KiB       |
+----------------------------------------------------------------------------
+---------------+---------------+-------------------+------------+------------
+----------------+
10 rows in set (0.02 sec)
```

Abbiamo detto in precedenza che lo schema delle prestazioni è archiviato in memoria, il che significa che viene anche monitorato nella strumentazione della memoria `performance_schema`.

> **Note**
>
> Se si riscontra che lo schema delle prestazioni utilizza molta memoria e si desidera limitarne l'utilizzo, è possibile ottimizzare i parametri del database in base alle proprie esigenze. Per

ulteriori informazioni, consulta la pagina relativa al modello di allocazione della memoria dello schema delle prestazioni nella documentazione di MySQL.

Per una maggiore leggibilità, è possibile rieseguire la stessa query escludendo gli eventi dello schema delle prestazioni. L'output mostra quanto segue:

- Il principale consumatore di memoria è `memory/sql/Prepared_statement::main_mem_root`.
- La colonna `current_alloc` indica che MySQL ha 4,91 GiB attualmente allocati a questo evento.
- `high_alloc column` indica che 4,91 GiB è il valore massimo di `current_alloc` dall'ultimo ripristino delle statistiche o dal riavvio del server. Ciò significa che `memory/sql/Prepared_statement::main_mem_root` è al suo valore massimo.

```
mysql> SELECT * FROM sys.memory_global_by_current_bytes WHERE event_name NOT LIKE
 'memory/performance_schema/%' LIMIT 10;

+-----------------------------------------------+---------------+--------------
+------------------+-----------+-----------+----------------+
| event_name                                    | current_count | current_alloc |
 current_avg_alloc | high_count | high_alloc | high_avg_alloc |
+-----------------------------------------------+---------------+--------------
+------------------+-----------+-----------+----------------+
| memory/sql/Prepared_statement::main_mem_root  |        512817 | 4.91 GiB      | 10.04
 KiB         |     512823 | 4.91 GiB   | 10.04 KiB      |
| memory/innodb/hash0hash                       |             4 | 79.07 MiB     | 19.77
 MiB         |          4 | 79.07 MiB  | 19.77 MiB      |
| memory/innodb/memory                          |         17096 | 31.68 MiB     | 1.90
 KiB         |      22498 | 37.60 MiB  | 1.71 KiB       |
| memory/sql/String::value                      |        122277 | 27.94 MiB     |  239
 bytes       |     124699 | 29.47 MiB  |  247 bytes     |
| memory/sql/TABLE                              |          9927 | 24.67 MiB     | 2.55
 KiB         |       9929 | 24.68 MiB  | 2.55 KiB       |
| memory/innodb/lock0lock                       |          8888 | 19.71 MiB     | 2.27
 KiB         |       8888 | 19.71 MiB  | 2.27 KiB       |
| memory/sql/Prepared_statement::infrastructure |        257623 | 16.24 MiB     |   66
 bytes       |     257631 | 16.24 MiB  |   66 bytes     |
| memory/mysys/KEY_CACHE                        |             3 | 16.00 MiB     | 5.33
 MiB         |          3 | 16.00 MiB  | 5.33 MiB       |
```

```
| memory/innodb/sync0arr                          |             3 | 7.03 MiB      | 2.34
 MiB          |          3 | 7.03 MiB   | 2.34 MiB       |
| memory/sql/THD::main_mem_root                   |           815 | 6.56 MiB      | 8.24
 KiB          |        849 | 7.19 MiB   | 8.67 KiB       |
+-------------------------------------------------+---------------+---------------
+------------------+-----------+-----------+----------------+
10 rows in set (0.06 sec)
```

Dal nome dell'evento, possiamo dedurre che questa memoria viene utilizzata per le istruzioni preparate. Per vedere quali connessioni utilizzano questa memoria, è possibile controllare memory_by_thread_by_current_bytes.

Nell'esempio seguente, a ogni connessione sono allocati circa 7 MiB, con un valore massimo di circa 6,29 MiB (`current_max_alloc`). Ciò è logico perché l'esempio utilizza `sysbench` con 80 tabelle e 800 connessioni con istruzioni preparate. Se si desidera ridurre l'utilizzo della memoria in questo scenario, è possibile ottimizzare l'utilizzo delle istruzioni preparate da parte dell'applicazione per ridurre il consumo di memoria.

```
mysql> SELECT * FROM sys.memory_by_thread_by_current_bytes;

+-----------+-------------------------------------------+--------------------
+------------------+------------------+------------------+-----------------+
| thread_id | user                                      | current_count_used |
 current_allocated | current_avg_alloc | current_max_alloc | total_allocated |
+-----------+-------------------------------------------+--------------------
+------------------+------------------+------------------+-----------------+
|        46 | rdsadmin@localhost                        |                405 | 8.47 MiB
          | 21.42 KiB         | 8.00 MiB          | 155.86 MiB      |
|        61 | reinvent@10.0.4.4                         |               1749 | 6.72 MiB
          | 3.93 KiB          | 6.29 MiB          | 14.24 MiB       |
|       101 | reinvent@10.0.4.4                         |               1845 | 6.71 MiB
          | 3.72 KiB          | 6.29 MiB          | 14.50 MiB       |
|        55 | reinvent@10.0.4.4                         |               1674 | 6.68 MiB
          | 4.09 KiB          | 6.29 MiB          | 14.13 MiB       |
|        57 | reinvent@10.0.4.4                         |               1416 | 6.66 MiB
          | 4.82 KiB          | 6.29 MiB          | 13.52 MiB       |
|       112 | reinvent@10.0.4.4                         |               1759 | 6.66 MiB
          | 3.88 KiB          | 6.29 MiB          | 14.17 MiB       |
|        66 | reinvent@10.0.4.4                         |               1428 | 6.64 MiB
          | 4.76 KiB          | 6.29 MiB          | 13.47 MiB       |
|        75 | reinvent@10.0.4.4                         |               1389 | 6.62 MiB
          | 4.88 KiB          | 6.29 MiB          | 13.40 MiB       |
```

```
|       116 | reinvent@10.0.4.4                       |              1333 | 6.61 MiB
          | 5.08 KiB          | 6.29 MiB          | 13.21 MiB       |
|        90 | reinvent@10.0.4.4                       |              1448 | 6.59 MiB
          | 4.66 KiB          | 6.29 MiB          | 13.58 MiB       |
|        98 | reinvent@10.0.4.4                       |              1440 | 6.57 MiB
          | 4.67 KiB          | 6.29 MiB          | 13.52 MiB       |
|        94 | reinvent@10.0.4.4                       |              1433 | 6.57 MiB
          | 4.69 KiB          | 6.29 MiB          | 13.49 MiB       |
|        62 | reinvent@10.0.4.4                       |              1323 | 6.55 MiB
          | 5.07 KiB          | 6.29 MiB          | 13.48 MiB       |
|        87 | reinvent@10.0.4.4                       |              1323 | 6.55 MiB
          | 5.07 KiB          | 6.29 MiB          | 13.25 MiB       |
|        99 | reinvent@10.0.4.4                       |              1346 | 6.54 MiB
          | 4.98 KiB          | 6.29 MiB          | 13.24 MiB       |
|       105 | reinvent@10.0.4.4                       |              1347 | 6.54 MiB
          | 4.97 KiB          | 6.29 MiB          | 13.34 MiB       |
|        73 | reinvent@10.0.4.4                       |              1335 | 6.54 MiB
          | 5.02 KiB          | 6.29 MiB          | 13.23 MiB       |
|        54 | reinvent@10.0.4.4                       |              1510 | 6.53 MiB
          | 4.43 KiB          | 6.29 MiB          | 13.49 MiB       |
.                                                                                .
.                                                                                .
.                                                                                .
|       812 | reinvent@10.0.4.4                       |              1259 | 6.38 MiB
          | 5.19 KiB          | 6.29 MiB          | 13.05 MiB       |
|       214 | reinvent@10.0.4.4                       |              1279 | 6.38 MiB
          | 5.10 KiB          | 6.29 MiB          | 12.90 MiB       |
|       325 | reinvent@10.0.4.4                       |              1254 | 6.38 MiB
          | 5.21 KiB          | 6.29 MiB          | 12.99 MiB       |
|       705 | reinvent@10.0.4.4                       |              1273 | 6.37 MiB
          | 5.13 KiB          | 6.29 MiB          | 13.03 MiB       |
|       530 | reinvent@10.0.4.4                       |              1268 | 6.37 MiB
          | 5.15 KiB          | 6.29 MiB          | 12.92 MiB       |
|       307 | reinvent@10.0.4.4                       |              1263 | 6.37 MiB
          | 5.17 KiB          | 6.29 MiB          | 12.87 MiB       |
|       738 | reinvent@10.0.4.4                       |              1260 | 6.37 MiB
          | 5.18 KiB          | 6.29 MiB          | 13.00 MiB       |
|       819 | reinvent@10.0.4.4                       |              1252 | 6.37 MiB
          | 5.21 KiB          | 6.29 MiB          | 13.01 MiB       |
|        31 | innodb/srv_purge_thread                 |             17810 | 3.14 MiB
          |  184 bytes        | 2.40 MiB          | 205.69 MiB      |
```

```
|        38 | rdsadmin@localhost                          |               599 | 1.76 MiB
         | 3.01 KiB          | 1.00 MiB          | 25.58 MiB       |
|         1 | sql/main                                    |              3756 | 1.32 MiB
         |  367 bytes        | 355.78 KiB        | 6.19 MiB        |
|       854 | rdsadmin@localhost                          |                46 | 1.08 MiB
         | 23.98 KiB         | 1.00 MiB          | 5.10 MiB        |
|        30 | innodb/clone_gtid_thread                    |              1596 | 573.14
 KiB        |  367 bytes        | 254.91 KiB        | 970.69 KiB      |
|        40 | rdsadmin@localhost                          |               235 | 245.19
 KiB        | 1.04 KiB          | 128.88 KiB        | 808.64 KiB      |
|       853 | rdsadmin@localhost                          |                96 | 94.63
 KiB         | 1009 bytes        | 29.73 KiB         | 422.45 KiB      |
|        36 | rdsadmin@localhost                          |                33 | 36.29
 KiB         | 1.10 KiB          | 16.08 KiB         | 74.15 MiB       |
|        33 | sql/event_scheduler                         |                 3 | 16.27
 KiB         | 5.42 KiB          | 16.04 KiB         | 16.27 KiB       |
|        35 | sql/compress_gtid_table                     |                 8 | 14.20
 KiB         | 1.77 KiB          | 8.05 KiB          | 18.62 KiB       |
|        25 | innodb/fts_optimize_thread                  |                12 | 1.86 KiB
         |  158 bytes        |  648 bytes        | 1.98 KiB        |
|        23 | innodb/srv_master_thread                    |                11 | 1.23 KiB
         |  114 bytes        |  361 bytes        | 24.40 KiB       |
|        24 | innodb/dict_stats_thread                    |                11 | 1.23 KiB
         |  114 bytes        |  361 bytes        | 1.35 KiB        |
|         5 | innodb/io_read_thread                       |                 1 |  144
 bytes        |  144 bytes        |  144 bytes        |  144 bytes      |
|         6 | innodb/io_read_thread                       |                 1 |  144
 bytes        |  144 bytes        |  144 bytes        |  144 bytes      |
|         2 | sql/aws_oscar_log_level_monitor             |                 0 |    0
 bytes        |    0 bytes        |    0 bytes        |    0 bytes      |
|         4 | innodb/io_ibuf_thread                       |                 0 |    0
 bytes        |    0 bytes        |    0 bytes        |    0 bytes      |
|         7 | innodb/io_write_thread                      |                 0 |    0
 bytes        |    0 bytes        |    0 bytes        |    0 bytes      |
|         8 | innodb/io_write_thread                      |                 0 |    0
 bytes        |    0 bytes        |    0 bytes        |    0 bytes      |
|         9 | innodb/io_write_thread                      |                 0 |    0
 bytes        |    0 bytes        |    0 bytes        |    0 bytes      |
|        10 | innodb/io_write_thread                      |                 0 |    0
 bytes        |    0 bytes        |    0 bytes        |    0 bytes      |
|        11 | innodb/srv_lra_thread                       |                 0 |    0
 bytes        |    0 bytes        |    0 bytes        |    0 bytes      |
|        12 | innodb/srv_akp_thread                       |                 0 |    0
 bytes        |    0 bytes        |    0 bytes        |    0 bytes      |
```

```
|        18 | innodb/srv_lock_timeout_thread            |                  0 |    0
 bytes        |    0 bytes        |    0 bytes        |  248 bytes      |
|        19 | innodb/srv_error_monitor_thread           |                  0 |    0
 bytes        |    0 bytes        |    0 bytes        |    0 bytes      |
|        20 | innodb/srv_monitor_thread                 |                  0 |    0
 bytes        |    0 bytes        |    0 bytes        |    0 bytes      |
|        21 | innodb/buf_resize_thread                  |                  0 |    0
 bytes        |    0 bytes        |    0 bytes        |    0 bytes      |
|        22 | innodb/btr_search_sys_toggle_thread       |                  0 |    0
 bytes        |    0 bytes        |    0 bytes        |    0 bytes      |
|        32 | innodb/dict_persist_metadata_table_thread |                  0 |    0
 bytes        |    0 bytes        |    0 bytes        |    0 bytes      |
|        34 | sql/signal_handler                        |                  0 |    0
 bytes        |    0 bytes        |    0 bytes        |    0 bytes      |
+-----------+-------------------------------------------+--------------------
+-------------------+------------------+------------------+-----------------+
831 rows in set (2.48 sec)
```

Come accennato in precedenza, il valore dell'ID thread (`thd_id`) può riferirsi ai thread in background del server o alle connessioni al database. Per mappare i valori dell'ID thread agli ID connessione al database, è possibile utilizzare la tabella `performance_schema.threads` o la vista `sys.processlist`, dove `conn_id` è l'ID connessione.

```
mysql> SELECT thd_id,conn_id,user,db,command,state,time,last_wait FROM sys.processlist
 WHERE user='reinvent@10.0.4.4';

+--------+---------+------------------+----------+---------+----------------+------
+-------------------------------------------------+
| thd_id | conn_id | user             | db       | command | state          | time |
 last_wait                                        |
+--------+---------+------------------+----------+---------+----------------+------
+-------------------------------------------------+
|    590 |     562 | reinvent@10.0.4.4 | sysbench | Execute | closing tables |    0 |
 wait/io/redo_log_flush                           |
|    578 |     550 | reinvent@10.0.4.4 | sysbench | Sleep   | NULL           |    0 |
 idle                                             |
|    579 |     551 | reinvent@10.0.4.4 | sysbench | Execute | closing tables |    0 |
 wait/io/redo_log_flush                           |
|    580 |     552 | reinvent@10.0.4.4 | sysbench | Execute | updating       |    0 |
 wait/io/table/sql/handler                        |
|    581 |     553 | reinvent@10.0.4.4 | sysbench | Execute | updating       |    0 |
 wait/io/table/sql/handler                        |
```

```
|    582 |     554 | reinvent@10.0.4.4 | sysbench | Sleep   | NULL           |    0 |
 idle                                          |
|    583 |     555 | reinvent@10.0.4.4 | sysbench | Sleep   | NULL           |    0 |
 idle                                          |
|    584 |     556 | reinvent@10.0.4.4 | sysbench | Execute | updating       |    0 |
 wait/io/table/sql/handler                     |
|    585 |     557 | reinvent@10.0.4.4 | sysbench | Execute | closing tables |    0 |
 wait/io/redo_log_flush                        |
|    586 |     558 | reinvent@10.0.4.4 | sysbench | Execute | updating       |    0 |
 wait/io/table/sql/handler                     |
|    587 |     559 | reinvent@10.0.4.4 | sysbench | Execute | closing tables |    0 |
 wait/io/redo_log_flush                        |
.
                                              .
.
                                              .
.
                                              .
|    323 |     295 | reinvent@10.0.4.4 | sysbench | Sleep   | NULL           |    0 |
 idle                                          |
|    324 |     296 | reinvent@10.0.4.4 | sysbench | Execute | updating       |    0 |
 wait/io/table/sql/handler                     |
|    325 |     297 | reinvent@10.0.4.4 | sysbench | Execute | closing tables |    0 |
 wait/io/redo_log_flush                        |
|    326 |     298 | reinvent@10.0.4.4 | sysbench | Execute | updating       |    0 |
 wait/io/table/sql/handler                     |
|    438 |     410 | reinvent@10.0.4.4 | sysbench | Execute | System lock    |    0 |
 wait/lock/table/sql/handler                   |
|    280 |     252 | reinvent@10.0.4.4 | sysbench | Sleep   | starting       |    0 |
 wait/io/socket/sql/client_connection          |
|     98 |      70 | reinvent@10.0.4.4 | sysbench | Query   | freeing items  |    0 |
 NULL                                          |
+--------+---------+-------------------+----------+---------+----------------+------
+-----------------------------------------------+
804 rows in set (5.51 sec)
```

Ora interrompiamo il carico di lavoro `sysbench`, che chiude le connessioni e libera la memoria. Ricontrollando gli eventi, possiamo verificare che la memoria è stata liberata, ma `high_alloc` indica ancora il limite massimo. La colonna `high_alloc` può essere molto utile per individuare brevi picchi di utilizzo della memoria, che potrebbero non essere immediatamente individuabili da `current_alloc`, che mostra solo la memoria attualmente allocata.

```
mysql> SELECT * FROM sys.memory_global_by_current_bytes WHERE event_name='memory/sql/
Prepared_statement::main_mem_root' LIMIT 10;

+-----------------------------------------------+---------------+--------------
+------------------+-----------+-----------+----------------+
| event_name                                    | current_count | current_alloc |
 current_avg_alloc | high_count | high_alloc | high_avg_alloc |
+-----------------------------------------------+---------------+--------------
+------------------+-----------+-----------+----------------+
| memory/sql/Prepared_statement::main_mem_root |            17 | 253.80 KiB    | 14.93
 KiB         |     512823 | 4.91 GiB   | 10.04 KiB      |
+-----------------------------------------------+---------------+--------------
+------------------+-----------+-----------+----------------+
1 row in set (0.00 sec)
```

Se si desidera ripristinare `high_alloc`, è possibile troncare le tabelle di riepilogo della memoria `performance_schema`, ma ciò ripristina tutta la strumentazione della memoria. Per ulteriori informazioni, consulta la pagina relativa alle caratteristiche generali delle tabelle dello schema delle prestazioni nella documentazione di MySQL.

Nell'esempio seguente, possiamo vedere che `high_alloc` viene ripristinato dopo il troncamento.

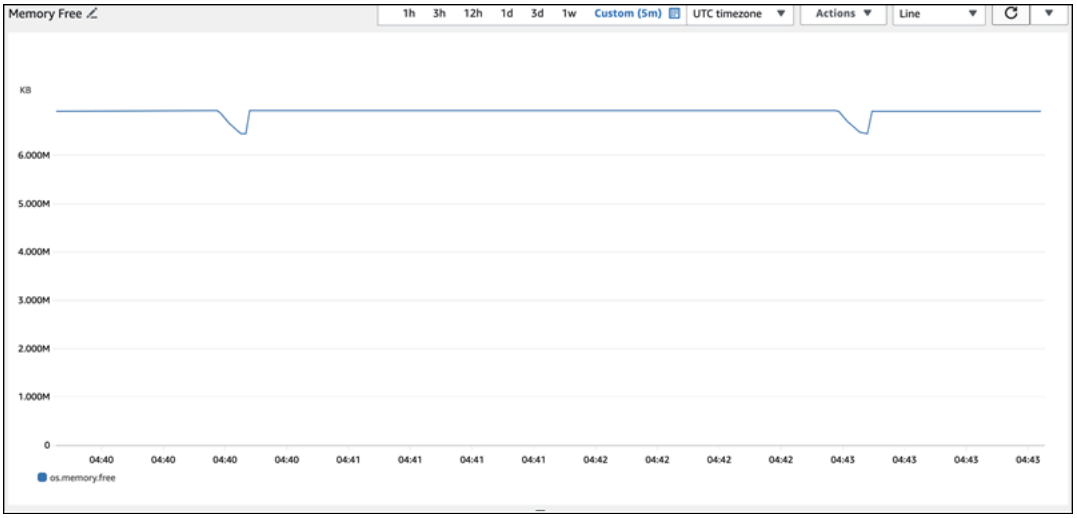
```
mysql> TRUNCATE `performance_schema`.`memory_summary_global_by_event_name`;
Query OK, 0 rows affected (0.00 sec)

mysql> SELECT * FROM sys.memory_global_by_current_bytes WHERE event_name='memory/sql/
Prepared_statement::main_mem_root' LIMIT 10;

+-----------------------------------------------+---------------+--------------
+------------------+-----------+-----------+----------------+
| event_name                                    | current_count | current_alloc |
 current_avg_alloc | high_count | high_alloc | high_avg_alloc |
+-----------------------------------------------+---------------+--------------
+------------------+-----------+-----------+----------------+
| memory/sql/Prepared_statement::main_mem_root |            17 | 253.80 KiB    | 14.93
 KiB         |         17 | 253.80 KiB | 14.93 KiB      |
+-----------------------------------------------+---------------+--------------
+------------------+-----------+-----------+----------------+
1 row in set (0.00 sec)
```

Esempio 2: picchi di memoria transitori

Un altro evento comune è rappresentato dai brevi picchi di utilizzo della memoria su un server di database. Possono trattarsi di cali periodici di memoria liberabile, difficili da risolvere con `current_alloc` e `sys.memory_global_by_current_bytes`, in quanto la memoria è già stata liberata.

> **Note**
>
> Se le statistiche dello schema delle prestazioni sono state ripristinate o l'istanza database è stata riavviata, queste informazioni non saranno disponibili in `sys` o `performance_schema`. Per mantenere queste informazioni, consigliamo di configurare la raccolta di metriche esterne.

Il seguente grafico della metrica `os.memory.free` di Monitoraggio avanzato mostra brevi picchi di utilizzo della memoria di 7 secondi. Monitoraggio avanzato consente di monitorare a intervalli di appena 1 secondo, il che è perfetto per rilevare picchi transitori come questi.



Per aiutare a diagnosticare la causa dell'utilizzo della memoria, possiamo utilizzare una combinazione di `high_alloc` nelle viste di riepilogo della memoria `sys` e nelle tabelle di riepilogo delle istruzioni dello schema delle prestazioni per cercare di identificare le sessioni e le connessioni che causano il problema.

Come previsto, poiché l'utilizzo della memoria non è attualmente elevato, non riusciamo a individuare alcun problema rilevante nella vista dello schema `sys` in `current_alloc`.

```
mysql> SELECT * FROM sys.memory_global_by_current_bytes LIMIT 10;
```

```
+-----------------------------------------------------------------------------
+---------------+---------------+-------------------+------------+------------
+----------------+
| event_name                                                                   |
 current_count | current_alloc | current_avg_alloc | high_count | high_alloc |
 high_avg_alloc |
+-----------------------------------------------------------------------------
+---------------+---------------+-------------------+------------+------------
+----------------+
| memory/innodb/hash0hash                                                      |
     4 | 79.07 MiB     | 19.77 MiB         |          4 | 79.07 MiB  | 19.77 MiB      |
| memory/innodb/os0event                                                       |
 439372 | 60.34 MiB     |  144 bytes        |     439372 | 60.34 MiB  |  144 bytes
 |
| memory/performance_schema/events_statements_summary_by_digest                |
     1 | 40.28 MiB     | 40.28 MiB         |          1 | 40.28 MiB  | 40.28 MiB      |
| memory/mysys/KEY_CACHE                                                       |
     3 | 16.00 MiB     | 5.33 MiB          |          3 | 16.00 MiB  | 5.33 MiB       |
| memory/performance_schema/events_statements_history_long                     |
     1 | 14.34 MiB     | 14.34 MiB         |          1 | 14.34 MiB  | 14.34 MiB      |
| memory/performance_schema/events_errors_summary_by_thread_by_error           |
   257 | 13.07 MiB     | 52.06 KiB         |        257 | 13.07 MiB  | 52.06 KiB      |
| memory/performance_schema/events_statements_summary_by_thread_by_event_name |
     1 | 11.81 MiB     | 11.81 MiB         |          1 | 11.81 MiB  | 11.81 MiB      |
| memory/performance_schema/events_statements_summary_by_digest.digest_text   |
     1 | 9.77 MiB      | 9.77 MiB          |          1 | 9.77 MiB   | 9.77 MiB       |
| memory/performance_schema/events_statements_history_long.digest_text        |
     1 | 9.77 MiB      | 9.77 MiB          |          1 | 9.77 MiB   | 9.77 MiB       |
| memory/performance_schema/events_statements_history_long.sql_text           |
     1 | 9.77 MiB      | 9.77 MiB          |          1 | 9.77 MiB   | 9.77 MiB       |
+-----------------------------------------------------------------------------
+---------------+---------------+-------------------+------------+------------
+----------------+
10 rows in set (0.01 sec)
```

Espandendo la vista per ordinare in base a `high_alloc`, ora possiamo vedere che il componente `memory/temptable/physical_ram` è un ottimo candidato. Al suo valore massimo, ha consumato 515,00 MiB.

Come suggerisce il nome, `memory/temptable/physical_ram` misura l'utilizzo della memoria per il motore di archiviazione `TEMP` in MySQL, introdotto in MySQL 8.0. Per ulteriori informazioni su come

MySQL utilizza le tabelle temporanee, consulta la pagina relativa all'uso delle tabelle temporanee interne in MySQL nella documentazione di MySQL.

> **Note**
>
> In questo esempio utilizziamo la vista `sys.x$memory_global_by_current_bytes`.

```
mysql> SELECT event_name, format_bytes(current_alloc) AS "currently allocated",
 sys.format_bytes(high_alloc) AS "high-water mark"
FROM sys.x$memory_global_by_current_bytes ORDER BY high_alloc DESC LIMIT 10;

+-------------------------------------------------------------------------------
+---------------------+-----------------+
| event_name                                                                    |
 currently allocated | high-water mark |
+-------------------------------------------------------------------------------
+---------------------+-----------------+
| memory/temptable/physical_ram                                                 | 4.00
 MiB            | 515.00 MiB      |
| memory/innodb/hash0hash                                                       | 79.07
 MiB           | 79.07 MiB       |
| memory/innodb/os0event                                                        | 63.95
 MiB           | 63.95 MiB       |
| memory/performance_schema/events_statements_summary_by_digest                | 40.28
 MiB           | 40.28 MiB       |
| memory/mysys/KEY_CACHE                                                        | 16.00
 MiB           | 16.00 MiB       |
| memory/performance_schema/events_statements_history_long                     | 14.34
 MiB           | 14.34 MiB       |
| memory/performance_schema/events_errors_summary_by_thread_by_error           | 13.07
 MiB           | 13.07 MiB       |
| memory/performance_schema/events_statements_summary_by_thread_by_event_name | 11.81
 MiB           | 11.81 MiB       |
| memory/performance_schema/events_statements_summary_by_digest.digest_text    | 9.77
 MiB            | 9.77 MiB        |
| memory/performance_schema/events_statements_history_long.sql_text            | 9.77
 MiB            | 9.77 MiB        |
+-------------------------------------------------------------------------------
+---------------------+-----------------+
10 rows in set (0.00 sec)
```

Nell'Esempio 1: utilizzo continuo ed elevato della memoria, abbiamo controllato l'utilizzo corrente della memoria per ogni connessione per determinare quale connessione fosse responsabile dell'utilizzo della memoria in questione. In questo esempio, la memoria è già stata liberata, quindi non è utile controllare l'utilizzo della memoria per le connessioni correnti.

Per approfondire e trovare le istruzioni, gli utenti e gli host responsabili, utilizziamo lo schema delle prestazioni. Lo schema delle prestazioni contiene più tabelle di riepilogo delle istruzioni suddivise in base a diverse dimensioni, quali nome dell'evento, digest di istruzioni, host, thread e utente. Ogni vista consentirà di approfondire dove vengono eseguite determinate istruzioni e cosa stanno facendo. Questa sezione si concentra su `MAX_TOTAL_MEMORY`, ma è possibile trovare ulteriori informazioni su tutte le colonne disponibili nella documentazione relativa alle tabelle di riepilogo delle istruzioni dello schema delle prestazioni.

```
mysql> SHOW TABLES IN performance_schema LIKE 'events_statements_summary_%';

+------------------------------------------------------------+
| Tables_in_performance_schema (events_statements_summary_%) |
+------------------------------------------------------------+
| events_statements_summary_by_account_by_event_name         |
| events_statements_summary_by_digest                        |
| events_statements_summary_by_host_by_event_name            |
| events_statements_summary_by_program                       |
| events_statements_summary_by_thread_by_event_name          |
| events_statements_summary_by_user_by_event_name            |
| events_statements_summary_global_by_event_name             |
+------------------------------------------------------------+
7 rows in set (0.00 sec)
```

Per prima cosa controlliamo `events_statements_summary_by_digest` per vedere `MAX_TOTAL_MEMORY`.

Da ciò possiamo vedere quanto segue:

- La query con digest `20676ce4a690592ff05debcffcbc26faeb76f22005e7628364d7a498769d0c4a` sembra essere un buon candidato per questo utilizzo di memoria. `MAX_TOTAL_MEMORY` è 537450710, che corrisponde al valore massimo rilevato per l'evento `memory/temptable/physical_ram` in `sys.x$memory_global_by_current_bytes`.
- È stato eseguito quattro volte (`COUNT_STAR`), la prima alle 04:08:34.943256 del 2024-03-26 e l'ultima alle 04:43:06.998310 del 2024-03-26.

```
mysql> SELECT SCHEMA_NAME,DIGEST,COUNT_STAR,MAX_TOTAL_MEMORY,FIRST_SEEN,LAST_SEEN
FROM performance_schema.events_statements_summary_by_digest ORDER BY MAX_TOTAL_MEMORY
 DESC LIMIT 5;

+-------------+------------------------------------------------------------------
+------------+-----------------+---------------------------
+----------------------------+
| SCHEMA_NAME | DIGEST                                                           |
 COUNT_STAR | MAX_TOTAL_MEMORY | FIRST_SEEN                | LAST_SEEN
  |
+-------------+------------------------------------------------------------------
+------------+-----------------+---------------------------
+----------------------------+
| sysbench    | 20676ce4a690592ff05debcffcbc26faeb76f22005e7628364d7a498769d0c4a |
     4 |        537450710 | 2024-03-26 04:08:34.943256 | 2024-03-26 04:43:06.998310 |
| NULL        | f158282ea0313fefd0a4778f6e9b92fc7d1e839af59ebd8c5eea35e12732c45d |
     4 |          3636413 | 2024-03-26 04:29:32.712348 | 2024-03-26 04:36:26.269329 |
| NULL        | 0046bc5f642c586b8a9afd6ce1ab70612dc5b1fd2408fa8677f370c1b0ca3213 |
     2 |          3459965 | 2024-03-26 04:31:37.674008 | 2024-03-26 04:32:09.410718 |
| NULL        | 8924f01bba3c55324701716c7b50071a60b9ceaf17108c71fd064c20c4ab14db |
     1 |          3290981 | 2024-03-26 04:31:49.751506 | 2024-03-26 04:31:49.751506 |
| NULL        | 90142bbcb50a744fcec03a1aa336b2169761597ea06d85c7f6ab03b5a4e1d841 |
     1 |          3131729 | 2024-03-26 04:15:09.719557 | 2024-03-26 04:15:09.719557 |
+-------------+------------------------------------------------------------------
+------------+-----------------+---------------------------
+----------------------------+
5 rows in set (0.00 sec)
```

Ora che conosciamo il digest incriminato, possiamo ottenere maggiori dettagli come il testo della query, l'utente che l'ha eseguita e dove è stata eseguita. In base al testo del digest restituito, possiamo vedere che si tratta di un'espressione di tabella comune (CTE) che crea quattro tabelle temporanee ed esegue quattro scansioni di tabelle, il che è molto inefficiente.

```
mysql> SELECT
 SCHEMA_NAME,DIGEST_TEXT,QUERY_SAMPLE_TEXT,MAX_TOTAL_MEMORY,SUM_ROWS_SENT,SUM_ROWS_EXAMINED,SUM
FROM performance_schema.events_statements_summary_by_digest
WHERE DIGEST='20676ce4a690592ff05debcffcbc26faeb76f22005e7628364d7a498769d0c4a'\G;

*************************** 1. row ***************************
           SCHEMA_NAME: sysbench
           DIGEST_TEXT: WITH RECURSIVE `cte` ( `n` ) AS ( SELECT ? FROM `sbtest1` UNION
 ALL SELECT `id` + ? FROM `sbtest1` ) SELECT * FROM `cte`
```

```
     QUERY_SAMPLE_TEXT: WITH RECURSIVE cte (n) AS (   SELECT 1  from sbtest1 UNION ALL
  SELECT id + 1 FROM sbtest1) SELECT * FROM cte
      MAX_TOTAL_MEMORY: 537450710
         SUM_ROWS_SENT: 80000000
     SUM_ROWS_EXAMINED: 80000000
SUM_CREATED_TMP_TABLES: 4
     SUM_NO_INDEX_USED: 4
1 row in set (0.01 sec)
```

Per ulteriori informazioni sulla tabella `events_statements_summary_by_digest` e su altre tabelle di riepilogo delle istruzioni dello schema delle prestazioni, consulta la pagina relativa alle tabelle di riepilogo delle istruzioni nella documentazione di MySQL.

È inoltre possibile eseguire un'istruzione EXPLAIN o EXPLAIN ANALYZE per visualizzare maggiori dettagli.

> **Note**
>
> `EXPLAIN ANALYZE` può fornire più informazioni di `EXPLAIN`, ma esegue anche la query, quindi prestare attenzione.

```
-- EXPLAIN
mysql> EXPLAIN WITH RECURSIVE cte (n) AS (SELECT 1  FROM sbtest1 UNION ALL SELECT id +
 1 FROM sbtest1) SELECT * FROM cte;

+----+-------------+------------+------------+-------+---------------+------+---------
+------+----------+----------+-------------+
| id | select_type | table      | partitions | type  | possible_keys | key  | key_len |
 ref  | rows     | filtered | Extra       |
+----+-------------+------------+------------+-------+---------------+------+---------
+------+----------+----------+-------------+
|  1 | PRIMARY     | <derived2> | NULL       | ALL   | NULL          | NULL | NULL    |
 NULL | 19221520 |   100.00 | NULL        |
|  2 | DERIVED     | sbtest1    | NULL       | index | NULL          | k_1  | 4       |
 NULL |  9610760 |   100.00 | Using index |
|  3 | UNION       | sbtest1    | NULL       | index | NULL          | k_1  | 4       |
 NULL |  9610760 |   100.00 | Using index |
+----+-------------+------------+------------+-------+---------------+------+---------
+------+----------+----------+-------------+
3 rows in set, 1 warning (0.00 sec)
```

```
-- EXPLAIN format=tree
mysql> EXPLAIN format=tree WITH RECURSIVE cte (n) AS (SELECT 1 FROM sbtest1 UNION ALL
 SELECT id + 1 FROM sbtest1) SELECT * FROM cte\G;

*************************** 1. row ***************************
EXPLAIN: -> Table scan on cte  (cost=4.11e+6..4.35e+6 rows=19.2e+6)
    -> Materialize union CTE cte  (cost=4.11e+6..4.11e+6 rows=19.2e+6)
        -> Index scan on sbtest1 using k_1  (cost=1.09e+6 rows=9.61e+6)
        -> Index scan on sbtest1 using k_1  (cost=1.09e+6 rows=9.61e+6)
1 row in set (0.00 sec)

-- EXPLAIN ANALYZE
mysql> EXPLAIN ANALYZE WITH RECURSIVE cte (n) AS (SELECT 1 from sbtest1 UNION ALL
 SELECT id + 1 FROM sbtest1) SELECT * FROM cte\G;

*************************** 1. row ***************************
EXPLAIN: -> Table scan on cte  (cost=4.11e+6..4.35e+6 rows=19.2e+6) (actual
 time=6666..9201 rows=20e+6 loops=1)
    -> Materialize union CTE cte  (cost=4.11e+6..4.11e+6 rows=19.2e+6) (actual
 time=6666..6666 rows=20e+6 loops=1)
        -> Covering index scan on sbtest1 using k_1  (cost=1.09e+6 rows=9.61e+6)
 (actual time=0.0365..2006 rows=10e+6 loops=1)
        -> Covering index scan on sbtest1 using k_1  (cost=1.09e+6 rows=9.61e+6)
 (actual time=0.0311..2494 rows=10e+6 loops=1)
1 row in set (10.53 sec)
```

Ma chi lo ha eseguito? Nello schema delle prestazioni possiamo vedere che l'utente `destructive_operator` aveva un valore `MAX_TOTAL_MEMORY` di 537450710, che corrisponde ancora una volta ai risultati precedenti.

> **Note**
>
> Lo schema delle prestazioni è archiviato in memoria, quindi non deve essere considerato l'unica fonte per il controllo. Se è necessario mantenere una cronologia delle istruzioni eseguite e degli utenti che le hanno eseguite, si consiglia di abilitare Aurora Advanced Auditing. Se è necessario conservare anche le informazioni sull'utilizzo della memoria, si consiglia di configurare il monitoraggio per esportare e archiviare questi valori.

```
mysql> SELECT USER,EVENT_NAME,COUNT_STAR,MAX_TOTAL_MEMORY FROM
 performance_schema.events_statements_summary_by_user_by_event_name
```

```
ORDER BY MAX_CONTROLLED_MEMORY DESC LIMIT 5;

+----------------------+---------------------------+------------+------------------+
| USER                 | EVENT_NAME                | COUNT_STAR | MAX_TOTAL_MEMORY |
+----------------------+---------------------------+------------+------------------+
| destructive_operator | statement/sql/select      |          4 |        537450710 |
| rdsadmin             | statement/sql/select      |       4172 |          3290981 |
| rdsadmin             | statement/sql/show_tables |          2 |          3615821 |
| rdsadmin             | statement/sql/show_fields |          2 |          3459965 |
| rdsadmin             | statement/sql/show_status |         75 |          1914976 |
+----------------------+---------------------------+------------+------------------+
5 rows in set (0.00 sec)

mysql> SELECT HOST,EVENT_NAME,COUNT_STAR,MAX_TOTAL_MEMORY FROM
 performance_schema.events_statements_summary_by_host_by_event_name
WHERE HOST != 'localhost' AND COUNT_STAR>0 ORDER BY MAX_CONTROLLED_MEMORY DESC LIMIT 5;

+------------+----------------------+------------+------------------+
| HOST       | EVENT_NAME           | COUNT_STAR | MAX_TOTAL_MEMORY |
+------------+----------------------+------------+------------------+
| 10.0.8.231 | statement/sql/select |          4 |        537450710 |
+------------+----------------------+------------+------------------+
1 row in set (0.00 sec)
```

Esempio 3: la memoria liberabile diminuisce continuamente e non viene recuperata

Il motore di database InnoDB utilizza una serie di eventi di monitoraggio della memoria specializzati per diversi componenti. Questi eventi specifici consentono un monitoraggio granulare dell'utilizzo della memoria nei sottosistemi chiave di InnoDB, ad esempio:

- `memory/innodb/buf0buf`: dedicato al monitoraggio delle allocazioni di memoria per il pool di buffer InnoDB.
- `memory/innodb/ibuf0ibuf`: monitora in modo specifico le modifiche di memoria relative al buffer di modifiche di InnoDB.

Per identificare i principali consumatori di memoria, è possibile eseguire una query su `sys.memory_global_by_current_bytes`:

```
mysql> SELECT event_name,current_alloc FROM sys.memory_global_by_current_bytes LIMIT
 10;
```

```
+-----------------------------------------------------------------+---------------+
| event_name                                                      | current_alloc |
+-----------------------------------------------------------------+---------------+
| memory/innodb/memory                                            | 5.28 GiB      |
| memory/performance_schema/table_io_waits_summary_by_index_usage | 495.00 MiB    |
| memory/performance_schema/table_shares                          | 488.00 MiB    |
| memory/sql/TABLE_SHARE::mem_root                                | 388.95 MiB    |
| memory/innodb/std                                               | 226.88 MiB    |
| memory/innodb/fil0fil                                           | 198.49 MiB    |
| memory/sql/binlog_io_cache                                      | 128.00 MiB    |
| memory/innodb/mem0mem                                           | 96.82 MiB     |
| memory/innodb/dict0dict                                         | 96.76 MiB     |
| memory/performance_schema/rwlock_instances                      | 88.00 MiB     |
+-----------------------------------------------------------------+---------------+
10 rows in set (0.00 sec)
```

I risultati mostrano che `memory/innodb/memory` è il principale consumatore, con 5,28 GiB di memoria attualmente allocata. Questo evento funge da categoria per le allocazioni di memoria tra vari componenti InnoDB non associati a eventi di attesa più specifici, come `memory/innodb/buf0buf` menzionato in precedenza.

Dopo aver stabilito che i componenti InnoDB sono i principali consumatori di memoria, possiamo approfondire le specifiche utilizzando il seguente comando MySQL:

```
SHOW ENGINE INNODB STATUS \G;
```

Il comando SHOW ENGINE INNODB STATUS fornisce un rapporto completo sullo stato del motore di archiviazione InnoDB, incluse statistiche dettagliate sull'utilizzo della memoria per diversi componenti InnoDB. Può aiutare a identificare quali strutture o operazioni specifiche di InnoDB consumano più memoria. Per ulteriori informazioni, consulta la pagina relativa alle strutture in memoria InnoDB nella documentazione di MySQL.

Analizzando la sezione `BUFFER POOL AND MEMORY` del rapporto sullo stato di InnoDB, vediamo che 5.051.647.748 byte (4,7 GiB) sono allocati alla cache degli oggetti del dizionario, che rappresenta l'89% della memoria monitorata da `memory/innodb/memory`.

```
----------------------
BUFFER POOL AND MEMORY
----------------------
Total large memory allocated 0
Dictionary memory allocated 5051647748
```

```
Buffer pool size 170512
Free buffers 142568
Database pages 27944
Old database pages 10354
Modified db pages 6
Pending reads 0
```

La cache degli oggetti del dizionario è una cache globale condivisa che archivia in memoria gli oggetti del dizionario di dati a cui si è avuto accesso in precedenza per consentire il riutilizzo degli oggetti e migliorare le prestazioni. L'elevata allocazione di memoria alla cache degli oggetti del dizionario suggerisce la presenza di un numero elevato di oggetti di database nella cache del dizionario di dati.

Ora che sappiamo che la cache del dizionario di dati è un consumatore primario, procediamo a ispezionare la cache del dizionario di dati per individuare eventuali tabelle aperte. Per trovare il numero di tabelle nella cache delle definizioni di tabella, eseguire una query sulla variabile di stato globale open_table_definition.

```
mysql> show global status like 'open_table_definitions';

+------------------------+-------+
| Variable_name          | Value |
+------------------------+-------+
| Open_table_definitions | 20000 |
+------------------------+-------+
1 row in set (0.00 sec)
```

Per ulteriori informazioni, consulta la pagina relativa ad apertura e chiusura delle tabelle in MySQL nella documentazione di MySQL.

È possibile limitare il numero di definizioni di tabella nella cache del dizionario di dati limitando il parametro `table_definition_cache` nel cluster di database o nel gruppo di parametri dell'istanza database. Per Aurora MySQL, questo valore funge da limite flessibile per il numero di tabelle nella cache delle definizioni di tabella. Il valore predefinito dipende dalla classe dell'istanza ed è impostato come segue:

```
LEAST({DBInstanceClassMemory/393040}, 20000)
```

Quando il numero di tabelle supera il limite `table_definition_cache`, un meccanismo basato sugli elementi utilizzati meno di recente (LRU) elimina e rimuove le tabelle dalla cache. Tuttavia,

le tabelle coinvolte nelle relazioni con chiavi esterne non vengono inserite nell'elenco LRU, impedendone la rimozione.

Nel nostro scenario attuale, eseguiamo FLUSH TABLES per cancellare la cache delle definizioni di tabella. Questa azione comporta un calo significativo della variabile di stato globale Open_table_definitions, da 20.000 a 12, come mostrato di seguito:

```
mysql> show global status like 'open_table_definitions';

+------------------------+-------+
| Variable_name          | Value |
+------------------------+-------+
| Open_table_definitions | 12    |
+------------------------+-------+
1 row in set (0.00 sec)
```

Nonostante questa riduzione, osserviamo che l'allocazione di memoria per `memory/innodb/memory` rimane elevata a 5,18 GiB e anche la memoria del dizionario allocata rimane invariata. Ciò è evidente dai seguenti risultati della query:

```
mysql> SELECT event_name,current_alloc FROM sys.memory_global_by_current_bytes LIMIT
 10;

+-----------------------------------------------------------------+---------------+
| event_name                                                      | current_alloc |
+-----------------------------------------------------------------+---------------+
| memory/innodb/memory                                            | 5.18 GiB      |
| memory/performance_schema/table_io_waits_summary_by_index_usage | 495.00 MiB    |
| memory/performance_schema/table_shares                          | 488.00 MiB    |
| memory/sql/TABLE_SHARE::mem_root                                | 388.95 MiB    |
| memory/innodb/std                                               | 226.88 MiB    |
| memory/innodb/fil0fil                                           | 198.49 MiB    |
| memory/sql/binlog_io_cache                                      | 128.00 MiB    |
| memory/innodb/mem0mem                                           | 96.82 MiB     |
| memory/innodb/dict0dict                                         | 96.76 MiB     |
| memory/performance_schema/rwlock_instances                      | 88.00 MiB     |
+-----------------------------------------------------------------+---------------+
10 rows in set (0.00 sec)
```

```
----------------------
BUFFER POOL AND MEMORY
```

```
----------------------
Total large memory allocated 0
Dictionary memory allocated 5001599639
Buffer pool size 170512
Free buffers 142568
Database pages 27944
Old database pages 10354
Modified db pages 6
Pending reads 0
```

Questo utilizzo persistentemente elevato della memoria può essere attribuito alle tabelle coinvolte nelle relazioni con chiavi esterne. Queste tabelle non vengono inserite nell'elenco LRU per la rimozione, il che spiega perché l'allocazione di memoria rimane elevata anche dopo aver svuotato la cache delle definizioni di tabella.

Per risolvere questo problema:

1. Esaminare e ottimizzare lo schema del database, in particolare le relazioni con chiavi esterne.
2. Valutare la possibilità di passare a una classe di istanze database più ampia con più memoria per ospitare gli oggetti del dizionario.

Seguendo questi passaggi e comprendendo i modelli di allocazione della memoria, è possibile gestire meglio l'utilizzo della memoria nell'istanza database Aurora MySQL e prevenire potenziali problemi di prestazioni dovuti alla pressione della memoria.

Risoluzione dei problemi di memoria insufficiente per i database Aurora MySQL

Il parametro a livello di istanza `aurora_oom_response` di Aurora MySQL può consentire all'istanza database di monitorare la memoria di sistema e stimare la memoria utilizzata da varie istruzioni e connessioni. Se il sistema esaurisce la memoria, può eseguire una serie di azioni per tentare di liberarla. L'obiettivo è quello di evitare il riavvio del database a causa di problemi di memoria insufficiente (OOM). Il parametro a livello di istanza acquisisce una stringa di azioni separate da virgola che un'istanza database esegue quando la memoria è insufficiente. Il parametro `aurora_oom_response` è supportato per Aurora MySQL versione 2 e 3.

Per il parametro `aurora_oom_response` è possibile utilizzare i seguenti valori e combinazioni di valori. Una stringa vuota indica che non viene intrapresa alcuna azione e di fatto disattiva la funzionalità, esponendo il database a riavvii OOM.

- `decline`: rifiuta le nuove query quando l'istanza database ha poca memoria.

- `kill_connect`: chiude le connessioni al database che consumano una grande quantità di memoria e termina le transazioni correnti e le istruzioni DDL (Data Definition Language). Questa risposta non è supportata per Aurora MySQL versione 2.

  Per ulteriori informazioni, consulta KILL statement nella documentazione di MySQL.

- `kill_query`: termina le query in ordine discendente relativamente al consumo di memoria fino a che la memoria dell'istanza non supera la soglia minima. Le istruzioni DDL non vengono terminate.

  Per ulteriori informazioni, consulta KILL statement nella documentazione di MySQL.

- `print`: stampa solo le query che consumano una grande quantità di memoria.

- `tune`: ottimizza le cache delle tabelle interne per restituire un po' di memoria al sistema. Aurora MySQL riduce la memoria utilizzata per le cache come `table_open_cache` e `table_definition_cache` in condizioni di memoria insufficiente. Eventualmente, Aurora MySQL riporta l'utilizzo della memoria alla normalità quando il sistema non è più in condizioni di memoria insufficiente.

  Per ulteriori informazioni, consulta table_open_cache e table_definition_cache nella documentazione di MySQL.

- `tune_buffer_pool`: riduce la dimensione del pool di buffer per liberare parte della memoria e renderla disponibile al server del database per l'elaborazione delle connessioni. Questa risposta è supportata per Aurora MySQL versione 3.06 e successive.

  È necessario associare `tune_buffer_pool` con `kill_query` o `kill_connect` nel valore del parametro `aurora_oom_response`. In caso contrario, il ridimensionamento del pool di buffer non avverrà, anche se si include `tune_buffer_pool` nel valore del parametro.

Nelle versioni di Aurora MySQL precedenti alla 3.06, per le classi di istanze database con memoria inferiore o uguale a 4 GiB, quando l'istanza è sotto pressione di memoria, le azioni predefinite includono `print`, `tune`, `decline` e `kill_query`. Per le classi di istanze database con memoria superiore a 4 GiB, il valore del parametro è vuoto per impostazione predefinita (disabilitato).

In Aurora MySQL versione 3.06 e successive, per le classi di istanze database con memoria inferiore o uguale a 4 GiB, Aurora MySQL chiude anche le connessioni che consumano più memoria (`kill_connect`). Per le classi di istanze database con memoria superiore a 4 GiB, il valore del parametro predefinito è `print`.

In Aurora MySQL versione 3.09 e successive, per le classi di istanze database con memoria superiore a 4 GiB, il valore del parametro predefinito è `print,decline,kill_connect`.

Se si verificano spesso problemi di memoria insufficiente, è possibile monitorare l'utilizzo della memoria tramite le tabelle di riepilogo della memoria quando `performance_schema` è abilitato.

Per le metriche di Amazon CloudWatch relative a OOM, consulta Parametri a livello di istanza per Amazon Aurora. Per le variabili di stato globali relative a OOM, consulta Variabili di stato globali di Aurora MySQL.

Registrazione di log per i database Aurora MySQL

I log di Aurora MySQL forniscono informazioni essenziali sull'attività e sugli errori del database. Abilitando questi log, è possibile identificare e risolvere i problemi, comprendere le prestazioni del database ed effettuare l'audit sull'attività del database. Consigliamo di abilitare questi log per tutte le istanze database Aurora MySQL per garantire prestazioni e disponibilità ottimali dei database. È possibile abilitare i tipi di registrazione di log seguenti. Ogni log contiene informazioni specifiche che possono portare all'individuazione degli impatti sull'elaborazione dei database.

- Errore: Aurora MySQL scrive nel log degli errori solo durante l'avvio, l'arresto e quando si verificano errori. Un'istanza database può andare avanti ore senza che ci siano nuove voci scritte nel file di log degli errori. Se non vedi voci recenti, significa che il server non ha riscontrato errori che generano una voce di registro. La registrazione del log degli errori è abilitata per impostazione predefinita. Per ulteriori informazioni, consulta Registri degli errori Aurora MySQL.
- Generale: il log generale fornisce informazioni dettagliate sull'attività del database, incluse tutte le istruzioni SQL eseguite dal motore di database. Per ulteriori informazioni sull'attivazione della registrazione di log generale e sull'impostazione dei parametri di registrazione, consulta Registri generali e delle query lente di Aurora MySQL e Log delle query generali nella documentazione di MySQL.

  > **Note**
  >
  > Le dimensioni dei log generali possono diventare molto grandi e occupare lo spazio di archiviazione. Per ulteriori informazioni, consulta Rotazione e conservazione dei registri per Autora MySQL.

- Query lente: il log di query lente è costituito dalle istruzioni SQL che impiegano più di long_query_time per l'esecuzione, oltre a richiedere il controllo di almeno min_examined_row_limit righe. È possibile utilizzare il log delle query lente per trovare le query che richiedono tempi di esecuzione prolungati e sono quindi idonee all'ottimizzazione.

  Il valore predefinito per `long_query_time` è di 10 secondi. Consigliamo di iniziare con un valore elevato per identificare le query più lente, quindi procedere verso il basso per eseguire il fine-tuning.

  È inoltre possibile utilizzare parametri correlati, ad esempio `log_slow_admin_statements` e `log_queries_not_using_indexes`. Confronta `rows_examined` con `rows_returned`. Se

`rows_examined` è molto maggiore di `rows_returned`, tali query possono potenzialmente essere bloccanti.

In Aurora MySQL versione 3, è possibile abilitare `log_slow_extra` per ottenere maggiori dettagli. Per ulteriori informazioni, consulta Log delle query lente nella documentazione di MySQL. È inoltre possibile modificare `long_query_time` a livello di sessione per eseguire il debug interattivo dell'esecuzione delle query, operazione particolarmente utile se `log_slow_extra` è abilitato a livello globale.

Per ulteriori informazioni sull'attivazione della registrazione di log generale e sull'impostazione dei parametri di registrazione di log, consulta Registri generali e delle query lente di Aurora MySQL e Log delle query lente nella documentazione di MySQL.

- Audit: il log di audit monitora e registra l'attività di database. La registrazione di controllo per Aurora MySQL è denominata Advanced Auditing. Per attivare Advanced Auditing, imposta alcuni parametri del cluster di database. Per ulteriori informazioni, consulta Utilizzo dell'audit avanzato con un cluster di database Amazon Aurora MySQL.
- Binario: il log binario (binlog) contiene eventi che descrivono le modifiche del database, come le operazioni di creazione delle tabelle e le modifiche ai dati delle tabelle. Contiene anche eventi relativi a istruzioni che potrebbero aver potuto apportare modifiche (ad esempio, un'operazione DELETE senza alcuna corrispondenza di riga), a meno che non venga utilizzata la registrazione di log basata su righe. Il log binario contiene anche informazioni sul tempo impiegato da ciascuna istruzione per l'aggiornamento dei dati.

  L'esecuzione di un server con registrazione di log binari abilitata rallenta leggermente le prestazioni. Tuttavia, i vantaggi del log binario, che consente di configurare la replica e le operazioni di ripristino, in genere superano questa lieve riduzione delle prestazioni.

  > **Note**
  >
  > Aurora MySQL non richiede la registrazione di log binari per le operazioni di ripristino.

  Per ulteriori informazioni sull'attivazione della registrazione di log binari e sull'impostazione del formato binlog, consulta Configurazione della registrazione di log binari Aurora MySQL per database Single-AZ e Log binario nella documentazione di MySQL.

È possibile pubblicare i log degli errori generali, delle query lente e di audit su Amazon CloudWatch Logs. Per ulteriori informazioni, consulta Pubblicazione di log di database su Amazon CloudWatch Logs.

Un altro strumento utile per riepilogare file di log delle query lente, generale e binario è pt-query-digest.

Risoluzione dei problemi di connessione per i database Aurora MySQL

Garantire una connettività affidabile tra le applicazioni e l'istanza database RDS è fondamentale per il corretto funzionamento dei carichi di lavoro. Tuttavia, i problemi di connettività possono insorgere a causa di vari fattori, come configurazioni di rete, problemi di autenticazione o vincoli delle risorse. Lo scopo di questa guida è fornire un approccio completo alla risoluzione dei problemi di connettività con Aurora MySQL.

Indice

- Identificazione dei problemi di connettività del database per Aurora MySQL
- Raccolta di dati sui problemi di connettività per Aurora MySQL
- Monitoraggio delle connessioni al database per Aurora MySQL
  - Monitoraggio aggiuntivo per Aurora MySQL
- Codici di errore di connettività per Aurora MySQL
- Suggerimenti per l'ottimizzazione dei parametri per Aurora MySQL
- Esempi di risoluzione dei problemi di connessione al database per Aurora MySQL
  - Esempio 1: risoluzione dei problemi relativi ai tentativi di connessione non riusciti
  - Esempio 2: risoluzione dei problemi di disconnessione anomala del client
  - Esempio 3: risoluzione dei tentativi di connessione non riusciti con IAM

Identificazione dei problemi di connettività del database per Aurora MySQL

L'identificazione della categoria specifica del problema di connettività può risultare utile per restringere le potenziali cause e guidare il processo di risoluzione dei problemi. Ogni categoria potrebbe richiedere approcci e tecniche di diagnosi e risoluzione diversi. I problemi di connettività dei database possono essere generalmente classificati nelle categorie indicate di seguito.

Errori ed eccezioni di connessione

> Gli errori e le eccezioni di connessione possono verificarsi per vari motivi, ad esempio stringhe di connessione errate, errori di autenticazione, interruzioni della rete o problemi del server di database. Le cause possono includere parametri di connessione configurati non correttamente, credenziali non valide, interruzioni della rete o arresti anomali o riavvii del server di database. Anche i gruppi di sicurezza non configurati correttamente, le impostazioni del cloud privato virtuale (VPC), le liste di controllo degli accessi alla rete ACLs () e le tabelle di routing associate alle sottoreti possono causare problemi di connessione.

Limite di connessioni raggiunto

Questo problema si verifica quando il numero di connessioni simultanee al server del database supera il limite massimo consentito. I server di database hanno in genere un limite massimo di connessioni configurabile definito dal parametro max_connections nei gruppi di parametri dei cluster e delle istanze. Stabilire un limite di connessione consente al server di database di disporre di risorse sufficienti (ad esempio memoria, CPU e handle di file) per gestire le connessioni esistenti in modo efficiente e fornire prestazioni accettabili. Le cause possono includere perdite di connessione nell'applicazione, un pool di connessioni inefficiente o un aumento imprevisto delle richieste di connessione.

Timeout di connessione.

I timeout di connessione si verificano quando l'applicazione client non è in grado di stabilire una connessione con il server del database entro un periodo di timeout specificato. Le cause più comuni includono problemi di rete, sovraccarico del server, regole del firewall e impostazioni di connessione configurate non correttamente.

Timeout per connessione inattiva

Le connessioni inattive che rimangono tali per un periodo prolungato potrebbero essere chiuse automaticamente dal server del database per risparmiare risorse. Questo timeout è in genere configurabile utilizzando `wait_timeout` e `interactive_timeout parameters` e deve essere modificato in base ai modelli di utilizzo della connessione dell'applicazione. Le cause possono includere la logica dell'applicazione che lascia le connessioni inattive per periodi prolungati o una gestione impropria delle connessioni.

Disconnessione intermittente delle connessioni esistenti

Questa classe di errori si riferisce a uno scenario in cui le connessioni stabilite tra un'applicazione client e il database vengono interrotte o disconnesse in modo imprevisto a intervalli irregolari, nonostante siano attive e in uso. Queste disconnessioni sono intermittenti, ovvero si verificano a intervalli irregolari e in modo non uniforme. Le cause possono includere:

- Problemi relativi al server di database, ad esempio riavvii o failover
- Gestione impropria della connessione dell'applicazione
- Problemi relativi al bilanciamento del carico e al proxy
- Instabilità della rete
- Problemi legati a componenti o middleware di terze parti coinvolti nel percorso di connessione
- Timeout di esecuzione delle query

- Limiti di risorse sul lato server o client

L'identificazione della causa principale attraverso il monitoraggio, la registrazione e l'analisi completi è fondamentale, mentre l'implementazione di una corretta gestione degli errori, di pool di connessioni e meccanismi di ripetizione dei tentativi può contribuire a mitigare l'impatto di queste disconnessioni intermittenti sulla funzionalità dell'applicazione e sulla relativa esperienza utente.

Raccolta di dati sui problemi di connettività per Aurora MySQL

La raccolta di dati completi relativi ai componenti dell'applicazione, del database, della rete e dell'infrastruttura è fondamentale per risolvere efficacemente i problemi di connettività tra un'applicazione e un database Aurora MySQL. Raccogliendo log, configurazioni e informazioni diagnostiche pertinenti, si ottengono informazioni preziose che possono aiutare a identificare la causa principale dei problemi di connettività e guidare verso una risoluzione appropriata.

I log e le configurazioni di rete, come le regole dei gruppi di sicurezza, le impostazioni dei VPC e le tabelle di routing, sono essenziali per identificare potenziali colli di bottiglia o configurazioni errate relativi alla rete che potrebbero impedire all'applicazione di stabilire una connessione corretta con il database. L'analisi di questi componenti di rete consente di verificare che le porte necessarie siano aperte, che gli indirizzi IP siano consentiti e che le configurazioni di routing siano impostate correttamente.

Timestamp

Registra i timestamp esatti in cui si verificano i problemi di connettività. Questo può contribuire a identificare i modelli o a correlare i problemi con altri eventi o attività.

Log del motore di database

Oltre ai log del database generali, esamina anche i log del motore di database (ad esempio, il log degli errori MySQL e il log delle query lente) per trovare eventuali informazioni o errori pertinenti che potrebbero essere correlati ai problemi di intermittenza della connettività. Per ulteriori informazioni, consulta Registrazione di log per i database Aurora MySQL.

Log delle applicazioni client

Raccogli log dettagliati dalle applicazioni client che si connettono al database. I log delle applicazioni forniscono visibilità sui tentativi di connessione, sugli errori e su qualsiasi informazione rilevante dal punto di vista dell'applicazione, favorendo così il rilevamento di problemi relativi alle stringhe di connessione, alle credenziali di autenticazione o alla gestione della connessione a livello di applicazione.

I log del database, d'altra parte, offrono informazioni sugli errori riguardanti il lato database, sulle query lente o sugli eventi che potrebbero contribuire ai problemi di connettività. Per ulteriori informazioni, consulta Registrazione di log per i database Aurora MySQL.

Variabili dell'ambiente client

Verifica se eventuali variabili di ambiente o impostazioni di configurazione sul lato client potrebbero influire sulla connessione al database, ad esempio impostazioni proxy, SSL/TLS impostazioni o altre variabili pertinenti.

Versioni della libreria client

Assicurati che il client disponga delle versioni più recenti di tutti i driver, le librerie o i framework di database utilizzati per la connettività del database. Le versioni obsolete possono presentare problemi noti o problemi di compatibilità.

Acquisizione di rete del client

Esegui un'acquisizione di rete sul lato client utilizzando uno strumento come Wireshark o `tcpdump` durante i periodi in cui si verificano problemi di connettività. Questo può aiutare a identificare eventuali problemi o anomalie relativi alla rete sul lato client.

Topologia della rete del client

Approfondisci la conoscenza della topologia di rete del client, inclusi eventuali firewall, bilanciatori del carico o altri componenti, come Server proxy per RDS o Proxy SQL, che effettuano connessioni al database al posto delle connessioni dirette del client.

Impostazioni del sistema operativo del client

Controlla le impostazioni del sistema operativo del client che potrebbero influire sulla connettività di rete, come le regole del firewall, le impostazioni degli adattatori di rete e qualsiasi altra impostazione rilevante.

Configurazione del pool di connessioni

Se nella tua applicazione utilizzi un meccanismo di pool di connessioni, esamina le impostazioni di configurazione e monitora le metriche (ad esempio, connessioni attive, connessioni inattive e timeout di connessione) per assicurarti che il pool funzioni correttamente. Controlla anche impostazioni quali la dimensione minima e massima del pool e le impostazioni di convalida della connessione, per assicurarti che siano configurate correttamente.

Stringa di connessione

La stringa di connessione include in genere parametri come il nome host o l'endpoint, il numero di porta, il nome del database e le credenziali di autenticazione. L'analisi della stringa di connessione può aiutare a identificare potenziali configurazioni o impostazioni errate che potrebbero causare problemi di connettività. Ad esempio, un nome host o un numero di porta errato può impedire al client di raggiungere l'istanza del database, mentre credenziali di autenticazione non valide possono causare errori di autenticazione e il rifiuto della connessione. Inoltre, la stringa di connessione può rivelare problemi relativi al pool di connessioni, ai timeout o ad altre impostazioni specifiche della connessione che potrebbero contribuire a creare problemi di connettività. Fornire la stringa di connessione completa utilizzata dall'applicazione client può aiutare a individuare eventuali errori di configurazione sul client.

Parametri del database

Monitora le metriche del database come l'utilizzo della CPU, l'utilizzo della memoria e del disco I/O durante i periodi in cui si verificano problemi di connettività. Questo può aiutare a comprendere se l'istanza database presenta un conflitto di risorse o problemi di prestazioni.

DB engine version (Versione motore del database

Verifica la versione del motore di database di Aurora MySQL. AWS rilascia regolarmente aggiornamenti che risolvono problemi noti e vulnerabilità della sicurezza, oltre a introdurre miglioramenti delle prestazioni. È quindi fortemente consigliato eseguire l'aggiornamento alle versioni più recenti disponibili, poiché tali aggiornamenti spesso includono correzioni di bug e miglioramenti specificamente correlati a connettività, prestazioni e stabilità. Fornire le informazioni sulla versione del database, insieme agli altri dettagli raccolti, può aiutare Supporto a diagnosticare e risolvere efficacemente i problemi di connettività.

Parametri di rete

Raccogli parametri di rete come latenza, perdita di pacchetti e throughput durante i periodi in cui si verificano problemi di connettività. Strumenti come `ping`, `traceroute` e per il monitoraggio della rete semplificano la raccolta di questi dati.

Dettagli su origine e client

Determina gli indirizzi IP dei server di applicazioni, dei bilanciatori del carico o di qualsiasi altro componente che avvia le connessioni al database. Potrebbe trattarsi di un singolo indirizzo IP o di un intervallo di indirizzi IP (notazione CIDR). Se l'origine è un' EC2 istanza Amazon, è utile anche esaminare il tipo di istanza, la zona di disponibilità, l'ID di sottorete e i gruppi di sicurezza associati all'istanza e i dettagli dell'interfaccia di rete come l'indirizzo IP privato e l'indirizzo IP pubblico.

Un’analisi approfondita dei dati raccolti consente di identificare configurazioni errate, vincoli di risorse, interruzioni della rete o altri problemi di base che causano problemi di connettività intermittenti o persistenti. Queste informazioni permettono di intraprendere azioni mirate, come la modifica delle configurazioni, la risoluzione di problemi di rete o la gestione delle connessioni a livello di applicazione.

Monitoraggio delle connessioni al database per Aurora MySQL

Per monitorare e risolvere i problemi di connettività, è possibile utilizzare le seguenti metriche e funzionalità.

CloudWatch metriche

- `CPUUtilization`: un elevato utilizzo della CPU sull’istanza database può rallentare l’esecuzione delle query, con conseguenti timeout o rifiuti di connessione.
- `DatabaseConnections`: monitora il numero di connessioni attive all’istanza database. Un numero elevato di connessioni, vicino al massimo configurato, può indicare potenziali problemi di connettività o l’esaurimento del pool di connessioni.
- `FreeableMemory`: una scarsa disponibilità di memoria può causare problemi di prestazioni e di connettività a causa di limiti di risorse.
- `NetworkReceiveThroughput` e `NetworkTransmitThroughput`: picchi o cali insoliti del throughput di rete possono indicare problemi di connettività o colli di bottiglia della rete.

Parametri Performance Insights

Per risolvere i problemi di connettività in Aurora MySQL utilizzando Approfondimenti sulle prestazioni, analizza metriche del database come le seguenti:

- Aborted_clients
- Aborted_connects
- Connessioni
- max_connections
- Threads_connected
- Threads_created
- Threads_running

Queste metriche possono aiutarti a identificare i colli di bottiglia della connessione, rilevare problemi di rete o di autenticazione, ottimizzare il pooling delle connessioni e garantire una

gestione efficiente dei thread. Per ulteriori informazioni, consulta Contatori di Performance Insights per Aurora MySQL.

Funzionalità di Approfondimenti sulle prestazioni

- Carico del database: visualizza il carico del database nel tempo e mettilo in relazione con i problemi di connettività o il degrado delle prestazioni.
- Statistiche SQL: analizza le statistiche SQL per identificare query o operazioni di database inefficienti che potrebbero contribuire a creare problemi di connettività.
- Query principali: identifica e analizza le query che richiedono più risorse per individuare potenziali rallentamenti delle prestazioni o le query di lunga durata che potrebbero causare problemi di connettività.

Il monitoraggio di queste metriche e il ricorso ad Approfondimenti sulle prestazioni consentono di ottenere visibilità sulle prestazioni dell'istanza database, sull'utilizzo delle risorse e sui potenziali colli di bottiglia che potrebbero causare problemi di connettività. Ad esempio:

- Un valore `DatabaseConnections` vicino al limite massimo può indicare l'esaurimento del pool di connessioni o una gestione impropria della connessione, con conseguenti problemi di connettività.
- Un valore di `CPUUtilization` elevato o di `FreeableMemory` ridotto può indicare vincoli delle risorse, che possono causare il rallentamento dell'esecuzione delle query e timeout o rifiuti di connessione.
- Analizzare Query principali e Statistiche SQL semplifica l'identificazione delle query inefficienti o a uso intensivo delle risorse che potrebbero contribuire a creare problemi di connettività.

Inoltre, il monitoraggio dei CloudWatch registri e l'impostazione degli allarmi possono aiutarti a identificare e rispondere in modo proattivo ai problemi di connettività prima che si aggravino.

È importante notare che, sebbene questi parametri e strumenti possano fornire informazioni preziose, devono essere utilizzati insieme ad altre fasi di risoluzione dei problemi. Inoltre, esaminando le configurazioni di rete, le regole dei gruppi di sicurezza e la gestione delle connessioni a livello di applicazione, è possibile diagnosticare e risolvere in modo completo i problemi di connettività con le istanze database Aurora MySQL.

Monitoraggio aggiuntivo per Aurora MySQL

CloudWatch metriche

- `AbortedClients`: traccia il numero di connessioni client che non sono state chiuse correttamente.
- `AuroraSlowConnectionHandleCount`: traccia del numero di operazioni di gestione delle connessioni lente, indicando potenziali problemi di connettività o colli di bottiglia delle prestazioni.
- `AuroraSlowHandshakeCount`: misura il numero di operazioni di handshake lente, un indicatore di possibili problemi di connettività.
- `ConnectionAttempts`: misura il numero di tentativi di connessione effettuati all'istanza database Aurora MySQL.

Variabili di stato globali

`Aurora_external_connection_count`: mostra il numero di connessioni database all'istanza database, ad esclusione delle connessioni al servizio RDS utilizzate per i controlli dell'integrità del database.

Monitorare questi parametri e le variabili di stato globali permette di ottenere visibilità sui modelli di connessione, sugli errori e sui potenziali colli di bottiglia che potrebbero causare problemi di connettività con l'istanza MySQL di Amazon Aurora.

Ad esempio, un numero elevato di `AbortedClients` o `AuroraSlowConnectionHandleCount` può indicare problemi di connettività.

Inoltre, la configurazione di CloudWatch allarmi e notifiche può aiutarti a identificare e rispondere in modo proattivo ai problemi di connettività prima che si aggravino e influiscano sulle prestazioni dell'applicazione.

Codici di errore di connettività per Aurora MySQL

Di seguito sono riportati alcuni errori di connettività comuni per i database Aurora MySQL, insieme ai relativi codici di errore e alle spiegazioni correlate.

Codice di errore 1040: Too many connections

Questo errore si verifica quando il client tenta di stabilire un numero di connessioni superiore al massimo consentito dal server del database. Le cause possibili sono:

- Configurazione errata del pool di connessioni: se utilizzi un meccanismo di pool di connessioni, assicurati che la dimensione massima del pool non sia troppo elevata e che le connessioni vengano rilasciate correttamente al pool.
- Configurazione dell'istanza database: verifica l'impostazione del numero massimo di connessioni consentite per l'istanza database e, se necessario, modificala impostando il parametro `max_connections`.
- Concorrenza elevata: se più client o applicazioni si connettono contemporaneamente al database, è possibile raggiungere il limite massimo di connessioni consentito.

Codice di errore 1045: Access denied for user '...'@'...' (using password: YES/NO)

Questo errore indica un problema di autenticazione durante il tentativo di connessione al database. Le cause possibili sono:

- Compatibilità del plug-in di autenticazione: verifica se il plug-in di autenticazione utilizzato dal client è compatibile con il meccanismo di autenticazione del server di database.
- Nome utente o password errati: verifica che nella stringa di connessione o nel meccanismo di autenticazione vengano utilizzati il nome utente e la password corretti.
- Autorizzazioni utente: assicurati che l'utente disponga delle autorizzazioni necessarie per connettersi all'istanza database dall'host o dalla rete specificati.

Codice di errore 1049: Unknown database '...'

Questo errore indica che il client sta tentando di connettersi a un database che non esiste sul server. Le cause possibili sono:

- Database non creato: assicurati che il database specificato sia stato creato nel server di database.
- Nome del database errato: ricontrolla il nome del database utilizzato nella stringa di connessione o nella query per verificarne la precisione.
- Autorizzazioni utente: verifica che l'utente disponga delle autorizzazioni necessarie per accedere al database specificato.

Codice di errore 1153: Got a packet bigger than 'max_allowed_packet' bytes

Questo errore si verifica quando il client tenta di inviare o ricevere dati che superano la dimensione massima dei pacchetti consentita dal server di database. Le cause possibili sono:

- Set di query o di risultati di grandi dimensioni: se le query eseguite coinvolgono grandi quantità di dati, è possibile che il limite di dimensione dei pacchetti venga superato.

- Impostazioni delle dimensioni dei pacchetti configurate non correttamente: controlla l'impostazione `max_allowed_packet`sul server del database e modificala se necessario.
- Problemi di configurazione della rete: assicurati che la configurazione di rete (ad esempio, la dimensione MTU) consenta le dimensioni dei pacchetti richieste.

Codice di errore 1226: User '...' has exceeded the 'max_user_connections' resource (current value: ...)

Questo errore indica che l'utente ha superato il numero massimo di connessioni simultanee consentite dal server del database. Le cause possibili sono:

- Configurazione errata del pool di connessioni: se utilizzi un meccanismo di pool di connessioni, assicurati che la dimensione massima del pool non sia troppo elevata per il limite di connessioni utente.
- Configurazione dell'istanza database: verifica l'impostazione `max_user_connections` per l'istanza database e, se necessario, modificala.
- Concorrenza elevata: se più client o applicazioni si connettono contemporaneamente al database con lo stesso utente, è possibile raggiungere il limite di connessioni specifiche per utente consentito.

Codice di errore 2003: Can't connect to MySQL server on '...' (10061)

Questo errore si verifica in genere quando il client non è in grado di stabilire una TCP/IP connessione con il server del database. Può essere causato da vari problemi, tra cui:

- Stato dell'istanza database: assicurati che lo stato dell'istanza database sia `available` e che non siano in corso operazioni di manutenzione o backup.
- Regole del firewall: verifica se alcuni firewall (sistema operativo, rete o gruppo di sicurezza) stanno bloccando la connessione sulla porta specificata (in genere, la 3306 per MySQL).
- Nome host o endpoint errato: assicurati che il nome host o l'endpoint utilizzato nella stringa di connessione sia corretto e corrisponda all'istanza database.
- Problemi di connettività di rete: verifica che il computer client possa raggiungere l'istanza del database attraverso la rete. Controlla se sono presenti eventuali interruzioni di rete, problemi di routing o configurazioni errate del VPC o della sottorete.

Codice di errore 2005: Unknown MySQL server host '...' (11001)

Questo errore si verifica quando il client non è in grado di risolvere il nome host o l'endpoint del server di database in un indirizzo IP. Le cause possibili sono:

- Problemi di risoluzione DNS: verifica che il computer client sia in grado di risolvere correttamente il nome host utilizzando DNS. Controlla le impostazioni DNS, la cache DNS e prova a utilizzare l'indirizzo IP anziché il nome host.
- Nome host o endpoint errato: ricontrolla il nome host o l'endpoint utilizzato nella stringa di connessione per verificare che sia corretto.
- Problemi di configurazione della rete: assicurati che la configurazione di rete del client (ad esempio, tabelle VPC, sottorete e tabelle di routing) consenta la risoluzione DNS e la connettività all'istanza del database.

Codice di errore 2026: SSL connection error

Questo errore si verifica quando si verifica un problema con la SSL/TLS configurazione o la convalida del certificato durante il tentativo di connessione. Le cause possibili sono:

- Scadenza del certificato: verifica se il SSL/TLS certificato utilizzato dal server è scaduto e deve essere rinnovato.
- Problemi di convalida del certificato: verifica che il client sia in grado di convalidare correttamente il SSL/TLS certificato del server e che il certificato sia affidabile.
- Problemi di configurazione della rete: assicurati che la configurazione di rete consenta le SSL/TLS connessioni e non blocchi o interferisca con il processo di SSL/TLS handshake.
- SSL/TLS configuration mismatch – Make sure that the SSL/TLSle impostazioni (ad esempio, suite di crittografia e versioni di protocollo) sul client e sul server sono compatibili.

Comprendere le spiegazioni dettagliate e le potenziali cause di ogni codice di errore consente di migliorare la risoluzione dei problemi di connettività quando si utilizzano i database Aurora MySQL.

Suggerimenti per l'ottimizzazione dei parametri per Aurora MySQL

Numero massimo connessioni

La modifica di questi parametri può aiutare a prevenire i problemi di connessione causati dal raggiungimento del limite massimo di connessioni consentito. Assicurati che questi valori siano impostati in modo appropriato in base ai requisiti di concorrenza e ai vincoli di risorse dell'applicazione.

- `max_connections`: questo parametro specifica il numero massimo consentito di connessioni simultanee all'istanza database.

- `max_user_connections`: questo parametro può essere specificato durante la creazione e la modifica dell'utente e imposta il numero massimo di connessioni simultanee consentite per un account utente specifico.

Dimensione del buffer di rete

L'aumento di questi valori può migliorare le prestazioni della rete, in particolare per i carichi di lavoro che comportano trasferimenti di dati o set di risultati di grandi dimensioni. È tuttavia consigliabile prestare attenzione, poiché i buffer di dimensioni maggiori possono consumare più memoria.

- `net_buffer_length`: questo parametro imposta la dimensione iniziale per la connessione del client e i buffer dei risultati, bilanciando l'utilizzo della memoria con le prestazioni delle query.
- `max_allowed_packet`: questo parametro specifica la dimensione massima di un singolo pacchetto di rete che può essere inviato o ricevuto dall'istanza database.

Compressione di rete (lato client)

L'attivazione della compressione di rete può ridurre l'utilizzo della larghezza di banda della rete, ma può aumentare il sovraccarico della CPU sia sul lato client che sul lato server.

- `compress`— Questo parametro abilita o disabilita la compressione di rete per la comunicazione. client/server
- `compress_protocol`: questo parametro specifica il protocollo di compressione da utilizzare per le comunicazioni di rete.

Ottimizzazione delle prestazioni di rete

La modifica di questi timeout può facilitare la gestione delle connessioni inattive e prevenire l'esaurimento delle risorse, ma occorre prestare attenzione, perché valori bassi possono causare interruzioni premature della connessione.

- `interactive_timeout`: questo parametro specifica il numero di secondi in cui il server attende l'attività su una connessione interattiva prima di chiuderla.
- `wait_timeout`: questo parametro determina il numero di secondi in cui il server attende l'attività su una connessione non interattiva prima di chiuderla.

Impostazioni di timeout di rete

La modifica di questi timeout può aiutare a risolvere i problemi relativi alle connessioni lente o che non rispondono. Attenzione, però, a non impostare valori troppo bassi, per evitare possibili interruzioni premature della connessione.

- `net_read_timeout`: questo parametro specifica il numero di secondi di attesa di ulteriori dati da una connessione prima di terminare l'operazione di lettura.
- `net_write_timeout`: questo parametro determina il numero di secondi di attesa per la scrittura di un blocco su una connessione prima di terminare l'operazione di scrittura.

Esempi di risoluzione dei problemi di connessione al database per Aurora MySQL

Negli esempi seguenti viene mostrato come identificare e risolvere i problemi di connessione al database per Aurora MySQL.

Esempio 1: risoluzione dei problemi relativi ai tentativi di connessione non riusciti

I tentativi di connessione possono fallire per diversi motivi, tra cui errori di autenticazione, errori di SSL/TLS handshake, `max_connections` limite raggiunto e vincoli di risorse sull'istanza DB.

È possibile tenere traccia del numero di connessioni non riuscite da Approfondimenti sulle prestazioni o utilizzando il comando seguente.

```
mysql> show global status like 'aborted_connects';
+------------------+-------+
| Variable_name    | Value |
+------------------+-------+
| Aborted_connects | 7     |
+------------------+-------+
1 row in set (0.00 sec)
```

Se il numero di `Aborted_connects` aumenta nel tempo, è possibile che l'applicazione presenti problemi di connettività intermittenti.

Puoi utilizzare le funzionalità di audit avanzato di Aurora per registrare le connessioni e le disconnessioni dalle connessioni client. A tale scopo, è possibile impostare i seguenti parametri nel gruppo di parametri del cluster di database:

- `server_audit_logging` = `1`
- `server_audit_events` = `CONNECT`

Di seguito è riportato un estratto dei log di audit relativi a un accesso non riuscito.

```
1728498527380921,auora-mysql-node1,user_1,172.31.49.222,147189,0,FAILED_CONNECT,,,1045
```

```
1728498527380940,auora-mysql-node1,user_1,172.31.49.222,147189,0,DISCONNECT,,,0
```

Dove:

- `1728498527380921` è il timestamp dell'epoch in cui si è verificato l'accesso non riuscito
- `aurora-mysql-node1` è l'identificatore dell'istanza del nodo del cluster Aurora MySQL su cui la connessione non è riuscita
- `user_1` è il nome dell'utente del database per il quale l'accesso non è riuscito
- `172.31.49.222` è l'indirizzo IP privato del client da cui è stata stabilita la connessione
- `147189` è l'ID di connessione dell'accesso non riuscito
- `FAILED_CONNECT`: indica che la connessione non è riuscita
- `1045` è il codice di restituito. Un valore diverso da zero indica un errore. In questo caso, `1045` corrisponde all'accesso negato.

Per ulteriori informazioni, consulta Server error codes e Client error codes nella documentazione di MySQL.

È inoltre possibile esaminare i log degli errori di Aurora MySQL per eventuali messaggi di errore correlati, ad esempio:

```
2024-10-09T19:26:59.310443Z 220 [Note] [MY-010926] [Server] Access denied for user
 'user_1'@'172.31.49.222' (using password: YES) (sql_authentication.cc:1502)
```

Esempio 2: risoluzione dei problemi di disconnessione anomala del client

È possibile tenere traccia del numero di disconnessioni anomale dei client da Approfondimenti sulle prestazioni o utilizzando il comando seguente.

```
mysql> show global status like 'aborted_clients';
+-----------------+-------+
| Variable_name   | Value |
+-----------------+-------+
| Aborted_clients | 9     |
+-----------------+-------+
1 row in set (0.01 sec)
```

Se il numero di `Aborted_clients` aumenta nel tempo, significa che l'applicazione non chiude correttamente le connessioni al database. Se le connessioni non vengono chiuse correttamente,

possono verificarsi perdite di risorse e potenziali problemi di prestazioni. Lasciare le connessioni aperte inutilmente può consumare risorse di sistema, come memoria e descrittori di file, con il rischio che l'applicazione o il server non risponda o si riavvii.

Per identificare gli account che non chiudono correttamente le connessioni, puoi utilizzare la seguente query, che recupera il nome dell'account utente, l'host da cui l'utente si connette, il numero di connessioni non chiuse e la percentuale di connessioni non chiuse.

```
SELECT
    ess.user,
    ess.host,
    (a.total_connections - a.current_connections) - ess.count_star AS not_closed,
    (((a.total_connections - a.current_connections) - ess.count_star) * 100) /
 (a.total_connections - a.current_connections) AS pct_not_closed
FROM
    performance_schema.events_statements_summary_by_account_by_event_name AS ess
    JOIN performance_schema.accounts AS a ON (ess.user = a.user AND ess.host = a.host)
WHERE
    ess.event_name = 'statement/com/quit'
    AND (a.total_connections - a.current_connections) > ess.count_star;

+----------+---------------+------------+----------------+
| user     | host          | not_closed | pct_not_closed |
+----------+---------------+------------+----------------+
| user1    | 172.31.49.222 |          1 |        33.3333 |
| user1    | 172.31.93.250 |       1024 |        12.1021 |
| user2    | 172.31.93.250 |         10 |        12.8551 |
+----------+---------------+------------+----------------+
3 rows in set (0.00 sec)
```

Dopo aver identificato gli account utente e gli host dai quali le connessioni non vengono chiuse, puoi procedere a controllare il codice che non chiude correttamente le connessioni.

Ad esempio, con il connettore MySQL in Python, puoi chiudere le connessioni utilizzando il metodo `close()` dell'oggetto di connessione. Ecco un esempio di funzione che stabilisce una connessione a un database, esegue una query e chiude la connessione:

```
import mysql.connector

def execute_query(query):
    # Establish a connection to the database
    connection = mysql.connector.connect(
```

```
        host="your_host",
        user="your_username",
        password="your_password",
        database="your_database"
    )

    try:
        # Create a cursor object
        cursor = connection.cursor()

        # Execute the query
        cursor.execute(query)

        # Fetch and process the results
        results = cursor.fetchall()
        for row in results:
            print(row)

    finally:
        # Close the cursor and connection
        cursor.close()
        connection.close()
```

In questo esempio, il metodo `connection.close()` viene chiamato nel blocco `finally` per assicurarsi che la connessione sia chiusa, indipendentemente dal fatto che si verifichi un'eccezione.

Esempio 3: risoluzione dei tentativi di connessione non riusciti con IAM

La connettività con gli utenti di AWS Identity and Access Management (IAM) può fallire per diversi motivi, ad esempio:

- Configurazione errata delle policy IAM
- Credenziali di sicurezza scadute
- Eventi di connettività di rete
- Mancata corrispondenza delle autorizzazioni del database

Per risolvere questi errori di autenticazione, abilita la funzionalità di esportazione dei `iam-db-auth-error` log nel tuo database Amazon Relational Database Service (RDS) o Aurora. Ciò ti consentirà di visualizzare messaggi di errore di autenticazione dettagliati nel gruppo di CloudWatch log per il tuo cluster Amazon RDS o Amazon Aurora.

Una volta abilitato, puoi esaminare questi log per identificare e risolvere la causa specifica degli errori di autenticazione IAM.

Ad esempio:

```
2025-09-22T12:02:30,806 [ERROR] Failed to authorize the connection request for
 user 'user_1' due to an internal IAM DB Auth error. (Status Code: 500, Error Code:
 InternalError)
```

e

```
2025-09-22T12:02:51,954 [ERROR] Failed to authenticate the connection request for user
 'user_2' because the provided token is malformed or otherwise invalid. (Status Code:
 400, Error Code: InvalidToken)
```

Per una guida alla risoluzione dei problemi, consulta la guida alla risoluzione dei problemi di Aurora per l'autenticazione IAM DB.

Risoluzione dei problemi delle prestazioni delle query per i database Aurora MySQL

MySQL fornisce il controllo di ottimizzazione delle query tramite variabili di sistema che influiscono sul modo in cui vengono valutati i piani di query, ottimizzazioni commutabili, suggerimenti sullo strumento di ottimizzazione e sugli indici e il modello di costo dello strumento di ottimizzazione. Questi punti dati possono essere utili non solo per confrontare diversi ambienti MySQL, ma anche per confrontare i piani di esecuzione delle query precedenti con i piani di esecuzione correnti e per comprendere l'esecuzione complessiva di una query MySQL in qualsiasi momento.

Le prestazioni delle query dipendono da molti fattori, tra cui il piano di esecuzione, lo schema e le dimensioni della tabella, le statistiche, le risorse, gli indici e la configurazione dei parametri. L'ottimizzazione delle query richiede l'identificazione dei punti critici e l'ottimizzazione del percorso di esecuzione.

- Trovare il piano di esecuzione per la query e verifica se la query utilizza gli indici appropriati. È possibile ottimizzare la query utilizzando `EXPLAIN` e rivedendo i dettagli di ciascun piano.
- Aurora MySQL versione 3 (compatibile con MySQL 8.0 Community Edition) utilizza un'istruzione `EXPLAIN ANALYZE`. L'istruzione `EXPLAIN ANALYZE` è uno strumento di profiling che mostra il punto della query in cui MySQL trascorre tempo e perché. Con `EXPLAIN ANALYZE`, Aurora MySQL pianifica, prepara ed esegue la query contando le righe e misurando il tempo impiegato in vari punti del piano di esecuzione. Al termine della query, `EXPLAIN ANALYZE` stampa il piano e le relative misurazioni anziché il risultato della query.
- Mantenere aggiornate le statistiche dello schema utilizzando l'istruzione `ANALYZE`. Lo strumento di ottimizzazione delle query a volte può scegliere piani di esecuzione scadenti a causa di statistiche obsolete. Ciò può comportare una riduzione delle prestazioni di una query a causa di stime imprecise della cardinalità sia delle tabelle che degli indici. La colonna `last_update` della tabella innodb_table_stats mostra l'ultima volta che le statistiche dello schema sono state aggiornate, il che è un buon indicatore di "stallo".
- Possono verificarsi altri problemi, come la distorsione della distribuzione dei dati, che non vengono presi in considerazione per la cardinalità della tabella. Per ulteriori informazioni, consulta Estimating ANALYZE TABLE complexity for InnoDB tables e Histogram statistics in MySQL nella documentazione MySQL.

Comprendere il tempo impiegato dalle interrogazioni

Di seguito sono riportati i modi per determinare il tempo impiegato dalle query:

- Profiling
- Schema delle prestazioni
- Strumento di ottimizzazione di query

Profiling

Per impostazione predefinita, il profiling è disabilitato. Abilita il profiling, quindi esegui la query lenta e rivedi il profilo.

```
SET profiling = 1;
Run your query.
SHOW PROFILE;
```

1. Identifica la fase in cui viene impiegato più tempo. In base agli stati generali dei thread presenti nella documentazione MySQL, la lettura e l'elaborazione delle righe di `SELECT` un'istruzione sono spesso lo stato di esecuzione più lungo per tutta la durata di una determinata query. È possibile utilizzare l'istruzione `EXPLAIN` per capire come MySQL esegue questa query.
2. Esamina il log delle query lente per valutare `rows_examined` e `rows_sent` per assicurarti che il carico di lavoro sia simile in ogni ambiente. Per ulteriori informazioni, consulta Registrazione di log per i database Aurora MySQL.
3. Esegui il comando seguente per le tabelle che fanno parte della query identificata:

   ```
   SHOW TABLE STATUS\G;
   ```

4. Acquisisci i seguenti output prima e dopo l'esecuzione della query in ogni ambiente:

   ```
   SHOW GLOBAL STATUS;
   ```

5. Esegui i seguenti comandi in ogni ambiente per vedere se ci sono altre query/sessioni che influenzano le prestazioni di questa query di esempio.

   ```
   SHOW FULL PROCESSLIST;

   SHOW ENGINE INNODB STATUS\G;
   ```

A volte, quando le risorse del server sono occupate, ciò influisce su tutte le altre operazioni sul server, comprese le query. È inoltre possibile acquisire informazioni periodicamente quando vengono eseguite delle query o impostare un processo `cron` per acquisire informazioni a intervalli utili.

Performance Schema

Lo schema delle prestazioni fornisce informazioni utili sulle prestazioni di runtime del server, con un impatto minimo su tali prestazioni. Questo è diverso da `information_schema`, che fornisce informazioni sullo schema sull'istanza database. Per ulteriori informazioni, consulta Panoramica dello schema di prestazioni per Approfondimenti sulle prestazioni su Aurora MySQL.

Traccia dell'ottimizzatore delle query

Per capire perché è stato scelto un particolare piano di query per l'esecuzione, puoi configurare `optimizer_trace` per l'accesso allo strumento di ottimizzazione delle query MySQL.

Esegui una traccia dell'ottimizzatore per mostrare informazioni dettagliate su tutti i percorsi disponibili per l'ottimizzatore e sulla sua scelta.

```
SET SESSION OPTIMIZER_TRACE="enabled=on";
SET optimizer_trace_offset=-5, optimizer_trace_limit=5;

-- Run your query.
SELECT * FROM table WHERE x = 1 AND y = 'A';

-- After the query completes:
SELECT * FROM information_schema.OPTIMIZER_TRACE;
SET SESSION OPTIMIZER_TRACE="enabled=off";
```

Revisione delle impostazioni dello strumento di ottimizzazione delle query

Aurora MySQL versione 3 (compatibile con MySQL 8.0 Community Edition) presenta molte modifiche relative allo strumento di ottimizzazione rispetto ad Aurora MySQL versione 2 (compatibile con MySQL 5.7 Community Edition). In caso di valori personalizzati per `optimizer_switch`, ti consigliamo di esaminare le differenze tra le impostazioni predefinite e di impostare i valori `optimizer_switch` più adatti al tuo carico di lavoro. Consigliamo inoltre di testare le opzioni disponibili per Aurora MySQL versione 3 per esaminare le prestazioni delle query.

> **Note**
>
> Aurora MySQL versione 3 utilizza il valore predefinito 20 della versione Community per il parametro innodb_stats_persistent_sample_pages.

Puoi utilizzare il seguente comando per visualizzare i valori `optimizer_switch`:

```
SELECT @@optimizer_switch\G;
```

La tabella seguente mostra i valori predefiniti `optimizer_switch` per Aurora MySQL versioni 2 e 3.

Impostazione	Aurora MySQL versione 2	Aurora MySQL versione 3
`batched_key_access`	off	off
`block_nested_loop`	on	on
`condition_fanout_filter`	on	on
`derived_condition_pushdown`	–	on
`derived_merge`	on	on
`duplicateweedout`	on	on
`engine_condition_pushdown`	on	on
`firstmatch`	on	on
`hash_join`	off	on
`hash_join_cost_based`	on	–
`hypergraph_optimizer`	–	off
`index_condition_pushdown`	on	on
`index_merge`	on	on
`index_merge_intersection`	on	on

Impostazione	Aurora MySQL versione 2	Aurora MySQL versione 3
index_merge_sort_union	on	on
index_merge_union	on	on
loosescan	on	on
materialization	on	on
mrr	on	on
mrr_cost_based	on	on
prefer_ordering_index	on	on
semijoin	on	on
skip_scan	–	on
subquery_materialization_cost_based	on	on
subquery_to_derived	–	off
use_index_extensions	on	on
use_invisible_indexes	–	off

Per ulteriori informazioni, consulta Switchable optimizations (MySQL 5.7) e Switchable optimizations (MySQL 8.0) nella documentazione MySQL.

Riferimento Amazon Aurora MySQL

Questo riferimento include informazioni sui parametri Aurora MySQL, le variabili di stato e le estensioni SQL generali o le differenze rispetto al motore del database MySQL della community.

Argomenti

- Parametri di configurazione Aurora MySQL
- Variabili di stato globali di Aurora MySQL
- Eventi di attesa Aurora MySQL
- Stati del thread Aurora MySQL
- Livelli di isolamento di Aurora MySQL
- Suggerimenti di Aurora MySQL
- Informazioni di riferimento sulle stored procedure Aurora MySQL
- Tabelle information_schema specifiche di Aurora MySQL

Parametri di configurazione Aurora MySQL

La gestione del cluster di database di Amazon Aurora MySQL è uguale a quella di altre istanze database Amazon RDS, ovvero utilizza i parametri di un gruppo di parametri database. La differenza tra Amazon Aurora e gli altri motori di database consiste nel fatto che un cluster di database contiene più istanze database. Di conseguenza, alcuni dei parametri utilizzati per gestire il cluster di database Aurora MySQL si applicano all'intero cluster. Altri parametri si applicano solo a una particolare istanza database del cluster di database.

Per gestire i parametri a livello di cluster, utilizza i gruppi di parametri del cluster di database. Per gestire i parametri a livello di istanza, utilizza i gruppi di parametri di database. Ogni istanza database in un cluster di database Aurora MySQL è compatibile con il motore del database MySQL. Tuttavia, si applicano alcuni dei parametri del motore del database MySQL a livello di cluster e si gestiscono questi parametri utilizzando i gruppi di parametri del cluster di DB. Non è possibile trovare parametri a livello di cluster nel gruppo di parametri DB per un'istanza in un cluster di database Aurora. Un elenco di parametri a livello di cluster è disponibile più avanti in questo argomento.

È possibile gestire sia i parametri a livello di cluster che quelli a livello di istanza con la Console di gestione AWS, la AWS CLI o l'API di Amazon RDS. Sono disponibili comandi separati per la gestione dei parametri a livello di cluster e a livello di istanza. Ad esempio, puoi usare il comando CLI modify-db-cluster-parameter-group per gestire i parametri a livello di cluster in un gruppo di parametri

del cluster di database. È possibile usare il comando CLI modify-db-parameter-group per gestire i parametri a livello di istanza in un gruppo di parametri DB per un'istanza database in un cluster di database.

Puoi visualizzare i parametri a livello di cluster e quelli a livello di istanza nella console oppure tramite l'interfaccia CLI o l'API RDS. Ad esempio, puoi usare il comando AWS CLI describe-db-cluster-parameters per visualizzare i parametri a livello di cluster in un gruppo di parametri del cluster di database. È possibile usare il comando CLI describe-db-parameters per visualizzare i parametri a livello di istanza in un gruppo di parametri DB per un'istanza database in un cluster di database.

> **Note**
>
> Ogni gruppo di parametri predefinito contiene i valori predefiniti per tutti i parametri del gruppo. Se il parametro ha come valore "engine default", consulta la documentazione MySQL o PostgreSQL specifica per la versione per il valore predefinito effettivo.
> Salvo diversa indicazione, i parametri elencati nelle tabelle seguenti sono validi per Aurora MySQL versione 2 e 3.

Per ulteriori informazioni sui gruppi di parametri database, consulta Gruppi di parametri per Amazon Aurora. Per regole e restrizioni per i cluster Aurora Serverless v1, consulta Gruppi di parametri per Aurora Serverless v1.

Argomenti

- Parametri a livello di cluster
- Parametri a livello di istanza
- I parametri MySQL seguenti non si applicano ad Aurora MySQL

Parametri a livello di cluster

La tabella seguente mostra tutti i parametri che si applicano all'intero cluster di database Aurora MySQL.

Nome del parametro	Modificabili	Note
`aurora_binlog_read_buffer_size`	Sì	Colpisce solo i cluster che utilizzano la replica binary log (binlog). Per informazi

Nome del parametro	Modificabili	Note
		oni sulla replica binlog, vedere Replica tra Aurora e MySQL o tra Aurora e un altro cluster di database Aurora (replica dei log binari). Rimosso da Aurora MySQL versione 3.
`aurora_binlog_replication_max_yield_seconds`	Sì	Colpisce solo i cluster che utilizzano la replica binary log (binlog). Per informazioni sulla replica binlog, vedere Replica tra Aurora e MySQL o tra Aurora e un altro cluster di database Aurora (replica dei log binari).
`aurora_binlog_replication_sec_index_parallel_workers`	Sì	Imposta il numero totale di thread paralleli disponibili per l'applicazione delle modifiche agli indici secondari durante la replica delle transazioni per tabelle di grandi dimensioni con più indici secondari. Per impostazione predefinita, il parametro è impostato su `0` (disabilitato). Questo parametro è disponibile in Aurora MySQL versione 306 e successive. Per ulteriori informazioni, consulta Ottimizzazione della replica dei log binari per Aurora MySQL.
`aurora_binlog_use_large_read_buffer`	Sì	Colpisce solo i cluster che utilizzano la replica binary log (binlog). Per informazioni sulla replica binlog, vedere Replica tra Aurora e MySQL o tra Aurora e un altro cluster di database Aurora (replica dei log binari). Rimosso da Aurora MySQL versione 3.

Nome del parametro	Modificabili	Note
`aurora_disable_hash_join`	Sì	Imposta questo parametro su `ON` per disabilitare l'ottimizzazione dell'hash join in Aurora MySQL versione 2.09 o successiva. Non è supportato per la versione 3. Per ulteriori informazioni, consulta Query parallela per Amazon Aurora MySQL.
`aurora_enable_replica_log_compression`	Sì	Per ulteriori informazioni, consulta Considerazioni sulle prestazioni per la replica Amazon Aurora MySQL. Non si applica ai cluster che fanno parte di un database globale Aurora. Rimosso da Aurora MySQL versione 3.
`aurora_enable_repl_bin_log_filtering`	Sì	Per ulteriori informazioni, consulta Considerazioni sulle prestazioni per la replica Amazon Aurora MySQL. Non si applica ai cluster che fanno parte di un database globale Aurora. Rimosso da Aurora MySQL versione 3.
`aurora_enable_staggered_replica_restart`	Sì	Questa impostazione è disponibile in Aurora MySQL versione 3, ma non viene utilizzata.
`aurora_enable_zdr`	Sì	Questa impostazione è attivata per impostazione predefinita in Aurora MySQL 2.10 e versioni successive.

Nome del parametro	Modificabili	Note
`aurora_in_memory_relaylog`	Sì	Imposta la modalità di log di inoltro in memoria. È possibile utilizzare questa funzionalità sulle repliche binlog per migliorare le prestazioni di replica dei log binari. Per disattivare questa funzionalità, imposta il parametro su OFF. Per attivare questa funzionalità, imposta il parametro su ON.
`aurora_enhanced_binlog`	Sì	Impostare il valore di questo parametro su 1 per attivare il file di log binario avanzato in Aurora MySQL versione 3.03.1 e successive. Per ulteriori informazioni, consulta Configurazione del file di log binario avanzato per Aurora MySQL.
`aurora_full_double_precision_in_json`	Sì	Imposta il valore di questo parametro per consentire l'analisi dei numeri in virgola mobile nei documenti JSON con la massima precisione.
`aurora_jemalloc_background_thread`	Sì	Utilizza questo parametro per abilitare un thread in background per eseguire operazioni di manutenzione della memoria. I valori consentiti sono `0` (disabilitato) e `1` (abilitato). Il valore predefinito è `0`. Questo parametro si applica solo ad Aurora MySQL 3.05 e versioni successive.

Nome del parametro	Modificabili	Note
`aurora_jemalloc_dirty_decay_ms`	Sì	Utilizza questo parametro per conservare la memoria liberata per un determinato periodo di tempo (in millisecondi). La conservazione della memoria consente un riutilizzo più rapido. I valori consentiti sono `0`–`18446744073709551615`. Il valore predefinito è `10000` (10 secondi). È possibile utilizzare un ritardo più breve per evitare problemi di esaurimento della memoria, a scapito di prestazioni più lente. Questo parametro si applica solo ad Aurora MySQL 3.05 e versioni successive.
`aurora_jemalloc_tcache_enabled`	Sì	Utilizza questo parametro per generare piccole richieste di memoria (fino a 32 KiB) in una cache locale del thread, bypassando le arene di memoria. I valori consentiti sono `0` (disabilitato) e 1 (abilitato). Il valore predefinito è 1. Questo parametro si applica solo ad Aurora MySQL 3.05 e versioni successive.
`aurora_load_from_s3_role`	Sì	Per ulteriori informazioni, consulta Caricamento dei dati in un cluster DB Amazon Aurora MySQL da file di testo in un bucket Amazon S3. Attualmente non disponibile in Aurora MySQL versione 3. Utilizza `aws_default_s3_role`.

Nome del parametro	Modificabili	Note
`aurora_mask_password_hashes_type`	Sì	Questa impostazione è attivata per impostazione predefinita in Aurora MySQL 2.11 e versioni successive. Usa questa impostazione per mascherare gli hash delle password di Aurora MySQL nei log delle query lente e di audit. I valori consentiti sono `0` e `1` (impostazione predefinita). Se è impostata su `1`, le password vengono registrate come `<secret>`. Se è impostata su `0`, le password vengono registrate come valori hash (#).
`aurora_select_into_s3_role`	Sì	Per ulteriori informazioni, consulta Salvataggio dei dati da un cluster DB Amazon Aurora MySQL nei file di testo in un bucket Amazon S3. Attualmente non disponibile in Aurora MySQL versione 3. Utilizza `aws_default_s3_role` .

Nome del parametro	Modificabili	Note
`authentication_kerberos_caseins_cmp`	Sì	Controlla il confronto dei nomi utente senza distinzione tra maiuscole e minuscole per il plugin `authentication_kerberos` . Impostalo su `true` per il confronto senza distinzione tra maiuscole e minuscole. Per impostazione predefinita, viene utilizzato il confronto con distinzione tra maiuscole e minuscole (`false`). Per ulteriori informazioni, consulta Utilizzo dell'autenticazione Kerberos per Aurora MySQL. Questo parametro è disponibile in Aurora MySQL versione 3.03 e versioni successive.
`auto_increment_increment`	Sì	Nessuno
`auto_increment_offset`	Sì	Nessuno
`aws_default_lambda_role`	Sì	Per ulteriori informazioni, consulta Chiamare una funzione Lambda da un cluster DB Amazon Aurora MySQL.

Nome del parametro	Modificabili	Note
`aws_default_s3_role`	Sì	Viene utilizzato quando si richiama l'istruzione `LOAD DATA FROM S3`, `LOAD XML FROM S3` o `SELECT INTO OUTFILE S3` dal cluster DB. In Aurora MySQL versione 2, il ruolo IAM specificato in questo parametro viene utilizzato se un ruolo IAM non è specificato per `aurora_load_from_s3_role` o `aurora_select_into_s3_role` per l'istruzione appropriata. In Aurora MySQL versione 3, il ruolo IAM specificato per questo parametro è sempre utilizzato. Per ulteriori informazioni, consulta Associazione di un ruolo IAM a un cluster DB Amazon Aurora MySQL.
`binlog_backup`	Sì	Impostare il valore di questo parametro su 0 per attivare il file di log binario avanzato in Aurora MySQL versione 3.03.1 e successive. È possibile disattivare questo parametro solo in caso di utilizzo di un file di log binario avanzato. Per ulteriori informazioni, consulta Configurazione del file di log binario avanzato per Aurora MySQL.

Nome del parametro	Modificabili	Note
`binlog_checksum`	Sì	La AWS CLI e l'API RDS segnalano un valore di `None` se questo parametro non è impostato. In tal caso, Aurora MySQL utilizza il valore predefinito del motore, ossia `CRC32`. Questo è diverso dall'impostazione esplicita di `NONE`, che disattiva il checksum.
`binlog-do-db`	Sì	Questo parametro si applica ad Aurora MySQL versione 3.
`binlog_format`	Sì	Per ulteriori informazioni, consulta Replica tra Aurora e MySQL o tra Aurora e un altro cluster di database Aurora (replica dei log binari).
`binlog_group_commit_sync_delay`	Sì	Questo parametro si applica ad Aurora MySQL versione 3.
`binlog_group_commit_sync_no_delay_count`	Sì	Questo parametro si applica ad Aurora MySQL versione 3.
`binlog-ignore-db`	Sì	Questo parametro si applica ad Aurora MySQL versione 3.
`binlog_replication_globaldb`	Sì	Impostare il valore di questo parametro su 0 per attivare il file di log binario avanzato in Aurora MySQL versione 3.03.1 e successive. È possibile disattivare questo parametro solo in caso di utilizzo di un file di log binario avanzato. Per ulteriori informazioni, consulta Configurazione del file di log binario avanzato per Aurora MySQL.
`binlog_row_image`	No	Nessuno

Nome del parametro	Modificabili	Note
binlog_row_metadata	Sì	Questo parametro si applica ad Aurora MySQL versione 3.
binlog_row_value_options	Sì	Questo parametro si applica ad Aurora MySQL versione 3.
binlog_rows_query_log_events	Sì	Nessuno
binlog_transaction_compression	Sì	Questo parametro si applica ad Aurora MySQL versione 3.
binlog_transaction_compression_level_zstd	Sì	Questo parametro si applica ad Aurora MySQL versione 3.
binlog_transaction_dependency_history_size	Sì	Questo parametro imposta un limite massimo al numero di hash di riga che vengono conservati in memoria e utilizzati per cercare la transazione dell'ultima modifica di una determinata riga. Una volta raggiunto questo numero di hash, la cronologia viene rimossa. Questo parametro si applica ad Aurora MySQL versione 2.12 e successive e versione 3.
binlog_transaction_dependency_tracking	Sì	Questo parametro si applica ad Aurora MySQL versione 3.
character-set-client-handshake	Sì	Nessuno
character_set_client	Sì	Nessuno
character_set_connection	Sì	Nessuno

Nome del parametro	Modificabili	Note
`character_set_database`	Sì	Il set di caratteri utilizzato dal database predefinito. Il valore predefinito è `utf8mb4`.
`character_set_filesystem`	Sì	Nessuno
`character_set_results`	Sì	Nessuno
`character_set_server`	Sì	Nessuno
`collation_connection`	Sì	Nessuno
`collation_server`	Sì	Nessuno
`completion_type`	Sì	Nessuno
`default_storage_engine`	No	I cluster Aurora MySQL utilizzano il motore di storage InnoDB per tutti i dati.
`enforce_gtid_consistency`	A volte	Modificabile in Aurora MySQL versione 2 e successive.
`event_scheduler`	Sì	Indica lo stato dell'utilità di pianificazione eventi. Modificabile solo a livello di cluster in Aurora MySQL versione 3.
`gtid-mode`	A volte	Modificabile in Aurora MySQL versione 2 e successive.

Nome del parametro	Modificabili	Note
`information_schema_stats_expiry`	Sì	Il numero di secondi dopo il quale il server del database MySQL recupera i dati dal motore di archiviazione e sostituisce i dati nella cache. I valori consentiti sono `0`–`31536000`. Questo parametro si applica ad Aurora MySQL versione 3.
`init_connect`	Sì	Il comando che deve essere eseguito dal server per ogni client che si connette. Utilizza le virgolette doppie (") per le impostazioni per evitare errori di connessione, ad esempio: `SET optimizer_switch="hash_join =off"` In Aurora MySQL versione 3, questo parametro non si applica agli utenti che dispongono del privilegio `CONNECTION_ADMIN` , incluso l'utente master Aurora. Per ulteriori informazioni, consulta Privilegio basato sui ruoli.
`innodb_adaptive_hash_index`	Sì	È possibile modificare questo parametro a livello di cluster database in Aurora MySQL versioni 2 e 3. L'indice adattivo Hash non è supportato nelle istanze database di lettura.

Nome del parametro	Modificabili	Note
innodb_aurora_instant_alter_column_allowed	Sì	Controlla se l'algoritmo INSTANT può essere utilizzato per operazioni ALTER COLUMN a livello globale. I valori validi consentiti sono i seguenti: • 0: l'algoritmo INSTANT non è consentito per le operazioni ALTER COLUMN (OFF). Passa all'utilizzo di altri algoritmi. • 1: l'algoritmo INSTANT è consentito per le operazioni ALTER COLUMN (ON). Si tratta del valore di default. Per ulteriori informazioni, consulta la pagina relativa alle operazioni sulle colonne nella documentazione di MySQL. Questo parametro si applica solo ad Aurora MySQL 3.05 e versioni successive.
innodb_autoinc_lock_mode	Sì	Nessuno
innodb_checksums	No	Rimosso da Aurora MySQL versione 3.
innodb_cmp_per_index_enabled	Sì	Nessuno
innodb_commit_concurrency	Sì	Nessuno
innodb_data_home_dir	No	Aurora MySQL utilizza istanze gestite in cui non si accede direttamente al filesystem.

Nome del parametro	Modificabili	Note
`innodb_deadlock_detect`	Sì	Questa opzione viene utilizzata per disabilitare il rilevamento del deadlock in Aurora MySQL versione 2.11 e successive e versione 3. Sui sistemi ad alta simultaneità, il rilevamento del deadlock può causare un rallentamento quando numerosi thread attendono lo stesso blocco. Per ulteriori informazioni su questo parametro, consulta la documentazione di MySQL.
`innodb_default_row_format`	Sì	Questo parametro definisce il formato di riga predefinito per le tabelle InnoDB (incluse le tabelle temporanee InnoDB create dall'utente). Si applica ad Aurora MySQL versioni 2 e 3. Il valore può essere `DYNAMIC`, `COMPACT` o `REDUNDANT`.
`innodb_file_per_table`	Sì	Questo parametro influisce sul modo in cui è organizzata l'archiviazione della tabella. Per ulteriori informazi oni, consulta Dimensionamento dello storage.

Nome del parametro	Modificabili	Note
innodb_flush_log_at_trx_commit	Sì	È fortemente consigliato utilizzare il valore predefinito 1. In Aurora MySQL versione 3, prima di poter impostare questo parametro su un valore diverso da 1, è necessario impostare il valore di innodb_trx_commit_allow_data_loss su 1. Per ulteriori informazioni, consulta Configurazione della frequenza di svuotamento del buffer dei registri.
innodb_ft_max_token_size	Sì	Nessuno
innodb_ft_min_token_size	Sì	Nessuno
innodb_ft_num_word_optimize	Sì	Nessuno
innodb_ft_sort_pll_degree	Sì	Nessuno
innodb_online_alter_log_max_size	Sì	Nessuno
innodb_optimize_fulltext_only	Sì	Nessuno
innodb_page_size	No	Nessuno
innodb_print_all_deadlocks	Sì	Quando è attivo, registra le informazioni su tutti i deadlock di InnoDB nel log degli errori di Aurora MySQL. Per ulteriori informazioni, consulta Contenimento e risoluzione dei problemi di deadlock di Aurora MySQL.

Nome del parametro	Modificabili	Note
innodb_purge_batch_size	Sì	Nessuno
innodb_purge_threads	Sì	Nessuno
innodb_rollback_on_timeout	Sì	Nessuno
innodb_rollback_segments	Sì	Nessuno
innodb_spin_wait_delay	Sì	Nessuno
innodb_strict_mode	Sì	Nessuno
innodb_support_xa	Sì	Rimosso da Aurora MySQL versione 3.
innodb_sync_array_size	Sì	Nessuno
innodb_sync_spin_loops	Sì	Nessuno
innodb_stats_include_delete_marked	Sì	Quando questo parametro è abilitato, InnoDB include i record contrassegnati per l'eliminazione durante il calcolo delle statistiche di ottimizzazione persistenti. Questo parametro si applica ad Aurora MySQL versione 2.12 e successive e versione 3.
innodb_table_locks	Sì	Nessuno

Nome del parametro	Modificabili	Note
innodb_trx_commit_allow_data_loss	Sì	In Aurora MySQL versione 3, imposta il valore di questo parametro su 1 in modo da poter modificare il valore di innodb_flush_log_at_trx_commit . Il valore predefinito di innodb_trx_commit_allow_data_loss è 0. Per ulteriori informazioni, consulta Configurazione della frequenza di svuotamento del buffer dei registri.
innodb_undo_directory	No	Aurora MySQL utilizza istanze gestite in cui non si accede direttamente al filesystem.
internal_tmp_disk_storage_engine	Sì	Controlla quale motore di archiviazione in memoria viene utilizzato per le tabelle temporanee interne. I valori consentiti sono INNODB e MYISAM. Questo parametro si applica ad Aurora MySQL versione 2.
internal_tmp_mem_storage_engine	Sì	Controlla quale motore di archiviazione in memoria viene utilizzato per le tabelle temporanee interne. I valori consentiti sono MEMORY e TempTable . Questo parametro si applica ad Aurora MySQL versione 3.
key_buffer_size	Sì	Cache per chiave per tabelle MyISAM. Per ulteriori informazioni, consultare keycache->cache_lock mutex.

Nome del parametro	Modificabili	Note
`lc_time_names`	Sì	Nessuno
`log_error_suppression_list`	Sì	Specifica un elenco di codici di errore che non sono registrati nel log degli errori di MySQL. In questo modo è possibile ignorare determinate condizioni di errore non critiche per mantenere puliti i log degli errori. Per ulteriori informazioni, consulta la sezione relativa a log_error_suppression_list nella documentazione di MySQL. Questo parametro si applica ad Aurora MySQL versione 3.03 e successive.
`low_priority_updates`	Sì	Le operazioni `INSERT`, `UPDATE`, `DELETE` e `LOCK TABLE WRITE` attendono fino a quando non ci sono più operazioni `SELECT` in sospeso. Questo parametro ha effetto solo sui motori di archiviazione che utilizzano solo il blocco a livello di tabella (MyISAM, MEMORY, MERGE). Questo parametro si applica ad Aurora MySQL versione 3.

Nome del parametro	Modificabili	Note
`lower_case_table_names`	Sì (Aurora MySQL versione 2) Solo al momento della creazione del cluster (Aurora MySQL versione 3)	In Aurora MySQL versione 2.10 e successive 2.x, assicurati di riavviare tutte le istanze di lettura dopo aver modificato questa impostazione e riavviato l'istanza di scrittura. Per informazioni dettagliate, consultar e Riavvio di un cluster Aurora con disponibilità di lettura. In Aurora MySQL versione 3, il valore di questo parametro viene impostato in modo permanente al momento della creazione del cluster. Se si utilizza un valore non predefinito per questa opzione, impostare il gruppo di parametri personalizzati Aurora MySQL versione 3 prima dell'aggiornamento e specificare il gruppo di parametri durante l'operazione di ripristino dello snapshot che crea il cluster versione 3. Con un database globale Aurora basato su Aurora MySQL, non puoi eseguire un aggiornamento locale da Aurora MySQL versione 2 alla versione 3 se il parametro `lower_case_table_names` è attivato. Per ulteriori informazioni sui metodi disponibili all'uso, consulta Aggiornamenti di una versione principale.
`master-info-repository`	Sì	Rimosso da Aurora MySQL versione 3.

Nome del parametro	Modificabili	Note
`master_verify_checksum`	Sì	Aurora MySQL versione 2. Utilizza `source_verify_checksum` in Aurora MySQL versione 3.
`max_delayed_threads`	Sì	Imposta il numero massimo di thread per gestire le istruzioni `INSERT DELAYED`. Questo parametro si applica ad Aurora MySQL versione 3.
`max_error_count`	Sì	Il numero massimo di messaggi di errore, avviso e note da archiviare per la visualizzazione. Questo parametro si applica ad Aurora MySQL versione 3.
`max_execution_time`	Sì	Il timeout per l'esecuzione delle istruzioni `SELECT`, in millisecondi. I valori possono essere compresi tra `0` e `18446744073709551615`. Se il parametro è impostato su `0`, non è previsto alcun timeout. Per ulteriori informazioni, consulta la sezione dedicata a max_execution_time nella documentazione di MySQL.
`min_examined_row_limit`	Sì	Utilizza questo parametro per impedire la registrazione delle query che esaminano un numero di righe inferiore a quello specificato.
`partial_revokes`	No	Questo parametro si applica ad Aurora MySQL versione 3.

Nome del parametro	Modificabili	Note
`preload_buffer_size`	Sì	La dimensione del buffer allocato durante il precaricamento degli indici. Questo parametro si applica ad Aurora MySQL versione 3.
`query_cache_type`	Sì	Rimosso da Aurora MySQL versione 3.

Nome del parametro	Modificabili	Note
`read_only`	Sì	Quando questo parametro è attivato, il server non consente aggiornamenti tranne quelli eseguiti dai thread di replica. I valori validi per Aurora MySQL versione 2 sono i seguenti: • `0` – `OFF` • `1` – `ON` • `{TrueIfReplica}` - `ON` per le repliche di lettura. Si tratta del valore di default. • `{TrueIfClusterReplica}` - `ON` per i cluster di replica come le repliche di lettura tra Regioni, i cluster secondari in un database globale Aurora e le implementazioni blu/verdi. I valori validi per Aurora MySQL versione 3 sono i seguenti: • `0` - `OFF`. Si tratta del valore di default. • `1` – `ON` • `{TrueIfClusterReplica}` - `ON` per i cluster di replica come le repliche di lettura tra Regioni, i cluster secondari in un database globale Aurora e le implementazioni blu/verdi. In Aurora MySQL versione 3, questo parametro non si applica agli utenti che dispongono del privilegio `CONNECTION_ADMIN` , incluso l'utente master

Nome del parametro	Modificabili	Note
		Aurora. Per ulteriori informazioni, consulta Privilegio basato sui ruoli.
`relay-log-space-limit`	Sì	Questo parametro si applica ad Aurora MySQL versione 3.
`replica_parallel_type`	Sì	Questo parametro consente l'esecuzione parallela sulla replica di tutti i thread di cui non è stato eseguito il commit già in fase di preparazione, senza violare la coerenza. Si applica ad Aurora MySQL versione 3. In Aurora MySQL versione 3.03.* e versioni precedenti, il valore predefinito è DATABASE. In Aurora MySQL versione 3.04 e versioni successive, il valore predefinito è LOGICAL_CLOCK.
`replica_preserve_commit_order`	Sì	Questo parametro si applica ad Aurora MySQL versione 3.
`replica_transaction_retries`	Sì	Questo parametro si applica ad Aurora MySQL versione 3.
`replica_type_conversions`	Sì	Questo parametro determina il tipo di conversione utilizzato nelle repliche. I valori consentiti sono `ALL_LOSSY` , `ALL_NON_LOSSY` , `ALL_SIGNED` e `ALL_UNSIGNED` . Per ulteriori informazioni, consulta l'argomento relativo alla replica con definizioni di tabelle diverse per origine e replica nella documentazione di MySQL. Questo parametro si applica ad Aurora MySQL versione 3.

Nome del parametro	Modificabili	Note
`replicate-do-db`	Sì	Questo parametro si applica ad Aurora MySQL versione 3.
`replicate-do-table`	Sì	Questo parametro si applica ad Aurora MySQL versione 3.
`replicate-ignore-db`	Sì	Questo parametro si applica ad Aurora MySQL versione 3.
`replicate-ignore-table`	Sì	Questo parametro si applica ad Aurora MySQL versione 3.
`replicate-wild-do-table`	Sì	Questo parametro si applica ad Aurora MySQL versione 3.
`replicate-wild-ignore-table`	Sì	Questo parametro si applica ad Aurora MySQL versione 3.
`require_secure_transport`	Sì	Questo parametro si applica ad Aurora MySQL versione 2 e 3. Per ulteriori informazioni, consulta Connessioni TLS a cluster di database Aurora MySQL.
`rpl_read_size`	Sì	Questo parametro si applica ad Aurora MySQL versione 3.
`server_audit_cw_upload`	No	Questo parametro è obsoleto in Aurora MySQL. Utilizza `server_audit_logs_upload` . Per ulteriori informazioni, consulta Pubblicazione di log Amazon Aurora MySQL in Amazon CloudWatch Logs.

Nome del parametro	Modificabili	Note
`server_audit_events`	Sì	Per ulteriori informazioni, consulta Utilizzo dell'audit avanzato con un cluster di database Amazon Aurora MySQL.
`server_audit_excl_users`	Sì	Per ulteriori informazioni, consulta Utilizzo dell'audit avanzato con un cluster di database Amazon Aurora MySQL.
`server_audit_incl_users`	Sì	Per ulteriori informazioni, consulta Utilizzo dell'audit avanzato con un cluster di database Amazon Aurora MySQL.
`server_audit_logging`	Sì	Per istruzioni sul caricamento dei log su Amazon CloudWatch Logs, consulta Pubblicazione di log Amazon Aurora MySQL in Amazon CloudWatch Logs.
`server_audit_logs_upload`	Sì	È possibile pubblicare i log di audit su CloudWatch Logs abilitando l'auditin g avanzato e impostando questo parametro su `1`. Il valore predefinito per il parametro `server_audit_logs_ upload` è `0`. Per ulteriori informazioni, consulta Pubblicazione di log Amazon Aurora MySQL in Amazon CloudWatch Logs.
`server_id`	No	Nessuno
`skip-character-set-client-h andshake`	Sì	Nessuno

Nome del parametro	Modificabili	Note
skip_name_resolve	No	Nessuno
slave-skip-errors	Sì	Si applica solo ai cluster Aurora MySQL versione 2, con compatibilità MySQL 5.7.
source_verify_checksum	Sì	Questo parametro si applica ad Aurora MySQL versione 3.
sync_frm	Sì	Rimosso da Aurora MySQL versione 3.
thread_cache_size	Sì	Il numero di thread da memorizza re nella cache. Questo parametro si applica ad Aurora MySQL versioni 2 e 3.
time_zone	Sì	Per impostazione predefinita, il fuso orario di un cluster di database Aurora è in formato UTC (Universal Time Coordinated). Tuttavia, puoi impostare il fuso orario delle istanze del cluster DB sul fuso orario locale dell'applicazione. Per ulteriori informazioni, consulta Fuso orario locale per i cluster DB Amazon Aurora.
tls_version	Sì	Per ulteriori informazioni, consulta Versioni TLS per Aurora MySQL.

Parametri a livello di istanza

La tabella seguente mostra tutti i parametri che si applicano a una specifica istanza database di un cluster di database Aurora MySQL.

Nome del parametro	Modificabili	Note
`activate_all_roles_on_login`	Sì	Questo parametro si applica ad Aurora MySQL versione 3.
`allow-suspicious-udfs`	No	Nessuno
`aurora_disable_hash_join`	Sì	Imposta questo parametro su `ON` per disabilitare l'ottimizzazione dell'hash join in Aurora MySQL versione 2.09 o successiva. Non è supportato per la versione 3. Per ulteriori informazioni, consulta Query parallela per Amazon Aurora MySQL.
`aurora_lab_mode`	Sì	Per ulteriori informazioni, consulta Modalità di laboratorio per Amazon Aurora MySQL. Rimosso da Aurora MySQL versione 3.
`aurora_oom_response`	Sì	Questo parametro è supportato per Aurora MySQL versione 2 e 3. Per ulteriori informazioni, consulta Risoluzione dei problemi di memoria insufficiente per i database Aurora MySQL.
`aurora_parallel_query`	Sì	Imposta su `ON` per abilitare la query parallela in Aurora MySQL versione 2.09 o successive. Il vecchio parametro `aurora_pq` non viene utilizzato in queste versioni. Per ulteriori informazioni, consulta Query parallela per Amazon Aurora MySQL.
`aurora_pq`	Sì	Imposta su `OFF` per disattivare la query parallela per istanze database specifiche e nelle versioni di Aurora MySQL precedenti alla 2.09. Nella versione 2.09

Nome del parametro	Modificabili	Note
		o successive, attiva e disattiva la query parallela con `aurora_parallel_query` . Per ulteriori informazioni, consulta Query parallela per Amazon Aurora MySQL.
`aurora_read_replica_read_committed`	Sì	Abilita il livello di isolamento `READ COMMITTED` per le repliche Aurora e modifica il comportamento di isolamento per ridurre il ritardo di rimozione durante le query a esecuzione prolungata. Abilita questa impostazione solo se sei informato sulle modifiche al comportamento e su come influiscono sui risultati della query. Ad esempio, questa impostazione utilizza un isolamento meno rigoroso rispetto all'impostazione predefinita di MySQL. Quando è abilitata, le query a esecuzione prolungata potrebbero visualizzare più di una copia della stessa riga perché Aurora riorganizza i dati della tabella mentre la query è in esecuzione. Per ulteriori informazioni, consulta Livelli di isolamento di Aurora MySQL.

Nome del parametro	Modificabili	Note
`aurora_tmptable_enable_per_table_limit`	Sì	Determina se il parametro `tmp_table_size` controlla le dimensioni massime delle tabelle temporane e in memoria create dal motore di storage `TempTable` in Aurora MySQL versione 3.04 e successive. Per ulteriori informazioni, consulta Limitazione delle dimensioni delle tabelle temporanee interne in memoria.
`aurora_use_vector_instructions`	Sì	Quando questo parametro è abilitato , Aurora MySQL utilizza istruzioni di elaborazione vettoriale ottimizzate fornite dalle moderne CPU per migliorare le prestazioni sui carichi di lavoro che fanno un uso intensivo di I/O. Questa impostazione è abilitata per impostazione predefinita in Aurora MySQL 3.05 e versioni successive.
`autocommit`	Sì	Nessuno
`automatic_sp_privileges`	Sì	Nessuno
`back_log`	Sì	Nessuno
`basedir`	No	Aurora MySQL utilizza istanze gestite in cui non si accede direttamente al filesystem.
`binlog_cache_size`	Sì	Nessuno
`binlog_max_flush_queue_time`	Sì	Nessuno

Nome del parametro	Modificabili	Note
binlog_order_commits	Sì	Nessuno
binlog_stmt_cache_size	Sì	Nessuno
binlog_transaction_compression	Sì	Questo parametro si applica ad Aurora MySQL versione 3.
binlog_transaction_compression_level_zstd	Sì	Questo parametro si applica ad Aurora MySQL versione 3.
bulk_insert_buffer_size	Sì	Nessuno
concurrent_insert	Sì	Nessuno
connect_timeout	Sì	Nessuno
core-file	No	Aurora MySQL utilizza istanze gestite in cui non si accede direttamente al filesystem.
datadir	No	Aurora MySQL utilizza istanze gestite in cui non si accede direttamente al filesystem.
default_authentication_plugin	No	Questo parametro si applica ad Aurora MySQL versione 3.
default_time_zone	No	Nessuno
default_tmp_storage_engine	Sì	Il motore di archiviazione predefinito per le tabelle temporanee.
default_week_format	Sì	Nessuno
delay_key_write	Sì	Nessuno
delayed_insert_limit	Sì	Nessuno
delayed_insert_timeout	Sì	Nessuno

Nome del parametro	Modificabili	Note
delayed_queue_size	Sì	Nessuno
div_precision_increment	Sì	Nessuno
end_markers_in_json	Sì	Nessuno
eq_range_index_dive_limit	Sì	Nessuno
event_scheduler	A volte	Indica lo stato dell'utilità di pianificazione eventi. Modificabile solo a livello di cluster in Aurora MySQL versione 3.
explicit_defaults_for_timestamp	Sì	Nessuno
flush	No	Nessuno
flush_time	Sì	Nessuno
ft_boolean_syntax	No	Nessuno
ft_max_word_len	Sì	Nessuno
ft_min_word_len	Sì	Nessuno
ft_query_expansion_limit	Sì	Nessuno
ft_stopword_file	Sì	Nessuno
general_log	Sì	Per istruzioni sul caricamento dei log su CloudWatch Logs, consulta Pubblicazione di log Amazon Aurora MySQL in Amazon CloudWatch Logs.
general_log_file	No	Aurora MySQL utilizza istanze gestite in cui non si accede direttamente al filesystem.

Nome del parametro	Modificabili	Note
`group_concat_max_len`	Sì	Nessuno
`host_cache_size`	Sì	Nessuno
`init_connect`	Sì	Il comando che deve essere eseguito dal server per ogni client che si connette. Utilizza le virgolette doppie (") per le impostazioni per evitare errori di connessione, ad esempio: `SET optimizer_switch="hash_join =off"` In Aurora MySQL versione 3, questo parametro non si applica agli utenti che dispongono del privilegio `CONNECTIO N_ADMIN` , incluso l'utente master di Aurora. Per ulteriori informazioni, consulta Privilegio basato sui ruoli.
`innodb_adaptive_hash_index`	Sì	È possibile modificare questo parametro a livello di istanza database in Aurora MySQL versione 2. È modificabile solo a livello di cluster database in Aurora MySQL versione 3. L'indice adattivo Hash non è supportato nelle istanze database di lettura.
`innodb_adaptive_max_sleep_d elay`	Sì	La modifica di questo parametro non ha alcun effetto, perché `innodb_th read_concurrency` è sempre 0 per Aurora.

Nome del parametro	Modificabili	Note
innodb_aurora_max_partitions_for_range	Sì	In alcuni casi in cui le statistiche persistenti non sono disponibili, è possibile utilizzare questo parametro per migliorare le prestazioni delle stime del numero di righe sulle tabelle partizionate. È possibile impostarlo su un valore compreso tra 0 e 8192, ovvero il valore determina il numero di partizioni da controllare durante la stima del numero di righe. Il valore predefinito è 0, che stima l'utilizzo di tutte le partizioni, coerente con il comportamento predefinito di MySQL. Questo parametro è disponibile per Aurora MySQL versione 3.03.1 e versioni successive.
innodb_autoextend_increment	Sì	Nessuno
innodb_buffer_pool_dump_at_shutdown	No	Nessuno
innodb_buffer_pool_dump_now	No	Nessuno
innodb_buffer_pool_filename	No	Nessuno
innodb_buffer_pool_load_abort	No	Nessuno
innodb_buffer_pool_load_at_startup	No	Nessuno

Nome del parametro	Modificabili	Note
innodb_buffer_pool_load_now	No	Nessuno
innodb_buffer_pool_size	Sì	Il valore predefinito è rappresentato da una formula. Per informazioni dettagliate su come viene calcolato il valore DBInstanceClassMemory nella formula, vedi Variabili di formula dei parametri database.
innodb_change_buffer_max_size	No	Aurora MySQL non utilizza affatto il buffer di modifica InnoDB.
innodb_compression_failure_threshold_pct	Sì	Nessuno
innodb_compression_level	Sì	Nessuno
innodb_compression_pad_pct_max	Sì	Nessuno
innodb_concurrency_tickets	Sì	La modifica di questo parametro non ha alcun effetto, perché innodb_thread_concurrency è sempre 0 per Aurora.

Nome del parametro	Modificabili	Note
innodb_deadlock_detect	Sì	Questa opzione viene utilizzata per disabilitare il rilevamento del deadlock in Aurora MySQL versione 2.11 e successive e versione 3. Sui sistemi ad alta simultaneità, il rilevamento del deadlock può causare un rallentamento quando numerosi thread attendono lo stesso blocco. Per ulteriori informazioni su questo parametro, consulta la documentazione di MySQL.
innodb_file_format	Sì	Rimosso da Aurora MySQL versione 3.
innodb_flushing_avg_loops	No	Nessuno
innodb_force_load_corrupted	No	Nessuno
innodb_ft_aux_table	Sì	Nessuno
innodb_ft_cache_size	Sì	Nessuno
innodb_ft_enable_stopword	Sì	Nessuno
innodb_ft_server_stopword_table	Sì	Nessuno
innodb_ft_user_stopword_table	Sì	Nessuno
innodb_large_prefix	Sì	Rimosso da Aurora MySQL versione 3.
innodb_lock_wait_timeout	Sì	Nessuno
innodb_log_compressed_pages	No	Nessuno

Nome del parametro	Modificabili	Note
innodb_lru_scan_depth	Sì	Nessuno
innodb_max_purge_lag	Sì	Nessuno
innodb_max_purge_lag_delay	Sì	Nessuno
innodb_monitor_disable	Sì	Nessuno
innodb_monitor_enable	Sì	Nessuno
innodb_monitor_reset	Sì	Nessuno
innodb_monitor_reset_all	Sì	Nessuno
innodb_old_blocks_pct	Sì	Nessuno
innodb_old_blocks_time	Sì	Nessuno
innodb_open_files	Sì	Nessuno
innodb_print_all_deadlocks	Sì	Quando è attivo, registra le informazioni su tutti i deadlock di InnoDB nel log degli errori di Aurora MySQL. Per ulteriori informazioni, consulta Contenimento e risoluzione dei problemi di deadlock di Aurora MySQL.
innodb_random_read_ahead	Sì	Nessuno
innodb_read_ahead_threshold	Sì	Nessuno
innodb_read_io_threads	No	Nessuno

Nome del parametro	Modificabili	Note
innodb_read_only	No	Aurora MySQL gestisce lo stato di sola lettura e lo stato di lettura/scrittura delle istanze database in base al tipo di cluster. Ad esempio, un cluster in provisioning ha un'istanza database di lettura/scrittura (l'istanza primaria) e tutte le altre istanze nel cluster sono di sola lettura (le repliche Aurora).
innodb_replication_delay	Sì	Nessuno
innodb_sort_buffer_size	Sì	Nessuno
innodb_stats_auto_recalc	Sì	Nessuno
innodb_stats_method	Sì	Nessuno
innodb_stats_on_metadata	Sì	Nessuno
innodb_stats_persistent	Sì	Nessuno
innodb_stats_persistent_sample_pages	Sì	Nessuno
innodb_stats_transient_sample_pages	Sì	Nessuno
innodb_thread_concurrency	No	Nessuno
innodb_thread_sleep_delay	Sì	La modifica di questo parametro non ha alcun effetto, perché innodb_thread_concurrency è sempre 0 per Aurora.

Nome del parametro	Modificabili	Note
interactive_timeout	Sì	Aurora valuta il valore minimo di interactive_timeout e wait_timeout . Quindi usa quel minimo come timeout per terminare tutte le sessioni inattive, sia interattive sia non interattive.
internal_tmp_disk_storage_engine	Sì	Controlla quale motore di archiviazione in memoria viene utilizzato per le tabelle temporanee interne. I valori consentiti sono INNODB e MYISAM. Questo parametro si applica ad Aurora MySQL versione 2.
internal_tmp_mem_storage_engine	A volte	Controlla quale motore di archiviazione in memoria viene utilizzato per le tabelle temporanee interne. I valori consentiti per le istanze database di scrittura sono MEMORY e TempTable . Per le istanze database di lettura questo parametro è impostato su TempTable e non può essere modificato. Questo parametro si applica ad Aurora MySQL versione 3.
join_buffer_size	Sì	Nessuno
keep_files_on_create	Sì	Nessuno
key_buffer_size	Sì	Cache per chiave per tabelle MyISAM. Per ulteriori informazioni, consultare keycache->cache_lock mutex.
key_cache_age_threshold	Sì	Nessuno

Nome del parametro	Modificabili	Note
key_cache_block_size	Sì	Nessuno
key_cache_division_limit	Sì	Nessuno
local_infile	Sì	Nessuno
lock_wait_timeout	Sì	Nessuno
log-bin	No	L'impostazione di binlog_format su STATEMENT , MIXED o ROW imposta automaticamente log-bin su ON. L'impostazione di binlog_format su OFF imposta automaticamente log-bin su OFF. Per ulteriori informazioni, consulta Replica tra Aurora e MySQL o tra Aurora e un altro cluster di database Aurora (replica dei log binari).
log_bin_trust_function_creators	Sì	Nessuno
log_bin_use_v1_row_events	Sì	Rimosso da Aurora MySQL versione 3.
log_error	No	Nessuno
log_error_suppression_list	Sì	Specifica un elenco di codici di errore che non sono registrati nel log degli errori di MySQL. In questo modo è possibile ignorare determinate condizioni di errore non critiche per mantenere puliti i log degli errori. Per ulteriori informazioni, consulta la sezione relativa a log_error_suppression_list nella documentazione di MySQL. Questo parametro si applica ad Aurora MySQL versione 3.03 e successive.

Nome del parametro	Modificabili	Note
log_output	Sì	Nessuno
log_queries_not_using_indexes	Sì	Nessuno
log_slave_updates	No	Aurora MySQL versione 2. Utilizza log_replica_updates in Aurora MySQL versione 3.
log_replica_updates	No	Aurora MySQL versione 3
log_throttle_queries_not_using_indexes	Sì	Nessuno
log_warnings	Sì	Rimosso da Aurora MySQL versione 3.
long_query_time	Sì	Nessuno
low_priority_updates	Sì	Le operazioni INSERT, UPDATE, DELETE e LOCK TABLE WRITE attendono fino a quando non ci sono più operazioni SELECT in sospeso. Questo parametro ha effetto solo sui motori di archiviazione che utilizzano solo il blocco a livello di tabella (MyISAM, MEMORY, MERGE). Questo parametro si applica ad Aurora MySQL versione 3.
max_allowed_packet	Sì	Nessuno
max_binlog_cache_size	Sì	Nessuno
max_binlog_size	No	Nessuno
max_binlog_stmt_cache_size	Sì	Nessuno

Nome del parametro	Modificabili	Note
`max_connect_errors`	Sì	Nessuno
`max_connections`	Sì	Il valore predefinito è rappresentato da una formula. Per informazioni dettagliate su come viene calcolato il valore `DBInstanceClassMemory` nella formula, vedi Variabili di formula dei parametri database. Per i valori predefiniti a seconda della classe di istanza, vedi Numero massimo di connessioni a un'istanza database Aurora MySQL.
`max_delayed_threads`	Sì	Imposta il numero massimo di thread per gestire le istruzioni `INSERT DELAYED`. Questo parametro si applica ad Aurora MySQL versione 3.
`max_error_count`	Sì	Il numero massimo di messaggi di errore, avviso e note da archiviare per la visualizzazione. Questo parametro si applica ad Aurora MySQL versione 3.

Nome del parametro	Modificabili	Note
`max_execution_time`	Sì	Il timeout per l'esecuzione delle istruzioni `SELECT`, in millisecondi. I valori possono essere compresi tra `0` e `18446744073709551615`. Se il parametro è impostato su `0`, non è previsto alcun timeout. Per ulteriori informazioni, consulta la sezione dedicata a max_execution_time nella documentazione di MySQL.
`max_heap_table_size`	Sì	Nessuno
`max_insert_delayed_threads`	Sì	Nessuno
`max_join_size`	Sì	Nessuno
`max_length_for_sort_data`	Sì	Rimosso da Aurora MySQL versione 3.
`max_prepared_stmt_count`	Sì	Nessuno
`max_seeks_for_key`	Sì	Nessuno
`max_sort_length`	Sì	Nessuno
`max_sp_recursion_depth`	Sì	Nessuno
`max_tmp_tables`	Sì	Rimosso da Aurora MySQL versione 3.
`max_user_connections`	Sì	Nessuno
`max_write_lock_count`	Sì	Nessuno
`metadata_locks_cache_size`	Sì	Rimosso da Aurora MySQL versione 3.

Nome del parametro	Modificabili	Note
min_examined_row_limit	Sì	Utilizza questo parametro per impedire la registrazione delle query che esaminano un numero di righe inferiore a quello specificato. Questo parametro si applica ad Aurora MySQL versione 3.
myisam_data_pointer_size	Sì	Nessuno
myisam_max_sort_file_size	Sì	Nessuno
myisam_mmap_size	Sì	Nessuno
myisam_sort_buffer_size	Sì	Nessuno
myisam_stats_method	Sì	Nessuno
myisam_use_mmap	Sì	Nessuno
net_buffer_length	Sì	Nessuno
net_read_timeout	Sì	Nessuno
net_retry_count	Sì	Nessuno
net_write_timeout	Sì	Nessuno
old-style-user-limits	Sì	Nessuno
old_passwords	Sì	Rimosso da Aurora MySQL versione 3.
optimizer_prune_level	Sì	Nessuno
optimizer_search_depth	Sì	Nessuno

Nome del parametro	Modificabili	Note
optimizer_switch	Sì	Per informazioni sulle funzionalità Aurora MySQL che utilizzano questa opzione, consulta Best practice con Amazon Aurora MySQL.
optimizer_trace	Sì	Nessuno
optimizer_trace_features	Sì	Nessuno
optimizer_trace_limit	Sì	Nessuno
optimizer_trace_max_mem_size	Sì	Nessuno
optimizer_trace_offset	Sì	Nessuno
performance-schema-consumer-events-waits-current	Sì	Se la colonna Origine per il parametro performance_schema è impostata su Modified, significa che lo schema delle prestazioni utilizza il parametro performance-schema-consumer-events-waits-current. Per ulteriori informazioni sull'abilitazione dello schema delle prestazioni, consulta Determinazione della gestione dello schema delle prestazioni da parte di Performance Insights.

Nome del parametro	Modificabili	Note
`performance-schema-instrument`	Sì	Se la colonna Origine per il parametro `performance_schema` è impostata su `Modified`, significa che lo schema delle prestazioni utilizza il parametro `performance-schema-instrument`. Per ulteriori informazioni sull'abilitazione dello schema delle prestazioni, consulta Determinazione della gestione dello schema delle prestazioni da parte di Performance Insights.
`performance_schema`	Sì	Se la colonna Origine è impostata su `Modified`, significa che il servizio Approfondimenti sulle prestazioni gestisce lo schema delle prestazioni. Per ulteriori informazioni sull'abilitazione dello schema delle prestazioni, consulta Determinazione della gestione dello schema delle prestazioni da parte di Performance Insights.
`performance_schema_accounts_size`	Sì	Se la colonna Origine per il parametro `performance_schema` è impostata su `Modified`, significa che lo schema delle prestazioni utilizza il parametro `performance_schema_accounts_size`. Per ulteriori informazioni sull'abilitazione dello schema delle prestazioni, consulta Determinazione della gestione dello schema delle prestazioni da parte di Performance Insights.

Nome del parametro	Modificabili	Note
performance_schema_consumer_global_instrumentation	Sì	Se la colonna Origine per il parametro performance_schema è impostata su Modified, significa che lo schema delle prestazioni utilizza il parametro performance_schema_consumer_global_instrumentation. Per ulteriori informazioni sull'abilitazione dello schema delle prestazioni, consulta Determinazione della gestione dello schema delle prestazioni da parte di Performance Insights.
performance_schema_consumer_thread_instrumentation	Sì	Se la colonna Origine per il parametro performance_schema è impostata su Modified, significa che lo schema delle prestazioni utilizza il parametro performance_schema_consumer_thread_instrumentation. Per ulteriori informazioni sull'abilitazione dello schema delle prestazioni, consulta Determinazione della gestione dello schema delle prestazioni da parte di Performance Insights.
performance_schema_consumer_events_stages_current	Sì	Se la colonna Origine per il parametro performance_schema è impostata su Modified, significa che lo schema delle prestazioni utilizza il parametro performance_schema_consumer_events_stages_current. Per ulteriori informazioni sull'abilitazione dello schema delle prestazioni, consulta Determinazione della gestione dello schema delle prestazioni da parte di Performance Insights.

Nome del parametro	Modificabili	Note
performance_schema_consumer_events_stages_history	Sì	Se la colonna Origine per il parametro performance_schema è impostata su Modified, significa che lo schema delle prestazioni utilizza il parametro performance_schema_consumer_events_stages_history. Per ulteriori informazioni sull'abilitazione dello schema delle prestazioni, consulta Determinazione della gestione dello schema delle prestazioni da parte di Performance Insights.
performance_schema_consumer_events_stages_history_long	Sì	Se la colonna Origine per il parametro performance_schema è impostata su Modified, significa che lo schema delle prestazioni utilizza il parametro performance_schema_consumer_events_stages_history_long. Per ulteriori informazioni sull'abilitazione dello schema delle prestazioni, consulta Determinazione della gestione dello schema delle prestazioni da parte di Performance Insights.

Nome del parametro	Modificabili	Note
`performance_schema_consumer_events_statements_current`	Sì	Se la colonna Origine per il parametro `performance_schema` è impostata su `Modified`, significa che lo schema delle prestazioni utilizza il parametro `performance_schema_consumer_events_statements_current`. Per ulteriori informazioni sull'abilitazione dello schema delle prestazioni, consulta Determinazione della gestione dello schema delle prestazioni da parte di Performance Insights.
`performance_schema_consumer_events_statements_history`	Sì	Se la colonna Origine per il parametro `performance_schema` è impostata su `Modified`, significa che lo schema delle prestazioni utilizza il parametro `performance_schema_consumer_events_statements_history`. Per ulteriori informazioni sull'abilitazione dello schema delle prestazioni, consulta Determinazione della gestione dello schema delle prestazioni da parte di Performance Insights.

Nome del parametro	Modificabili	Note
performance_schema_consumer_events_statements_history_long	Sì	Se la colonna Origine per il parametro performance_schema è impostata su Modified, significa che lo schema delle prestazioni utilizza il parametro performance_schema_consumer_events_statements_history_long. Per ulteriori informazioni sull'abilitazione dello schema delle prestazioni, consulta Determinazione della gestione dello schema delle prestazioni da parte di Performance Insights.
performance_schema_consumer_events_waits_history	Sì	Se la colonna Origine per il parametro performance_schema è impostata su Modified, significa che lo schema delle prestazioni utilizza il parametro performance_schema_consumer_events_waits_history. Per ulteriori informazioni sull'abilitazione dello schema delle prestazioni, consulta Determinazione della gestione dello schema delle prestazioni da parte di Performance Insights.

Nome del parametro	Modificabili	Note
performance_schema_consumer_events_waits_history_long	Sì	Se la colonna Origine per il parametro performance_schema è impostata su Modified, significa che lo schema delle prestazioni utilizza il parametro performance_schema_consumer_events_waits_history_long. Per ulteriori informazioni sull'abilitazione dello schema delle prestazioni, consulta Determinazione della gestione dello schema delle prestazioni da parte di Performance Insights.
performance_schema_consumer_statements_digest	Sì	Se la colonna Origine per il parametro performance_schema è impostata su Modified, significa che lo schema delle prestazioni utilizza il parametro performance_schema_consumer_statements_digest. Per ulteriori informazioni sull'abilitazione dello schema delle prestazioni, consulta Determinazione della gestione dello schema delle prestazioni da parte di Performance Insights.
performance_schema_digests_size	Sì	Se la colonna Origine per il parametro performance_schema è impostata su Modified, significa che lo schema delle prestazioni utilizza il parametro performance_schema_digests_size. Per ulteriori informazioni sull'abilitazione dello schema delle prestazioni, consulta Determinazione della gestione dello schema delle prestazioni da parte di Performance Insights.

Nome del parametro	Modificabili	Note
performance_schema_events_stages_history_long_size	Sì	Se la colonna Origine per il parametro performance_schema è impostata su Modified, significa che lo schema delle prestazioni utilizza il parametro performance_schema_events_stages_history_long_size. Per ulteriori informazioni sull'abilitazione dello schema delle prestazioni, consulta Determinazione della gestione dello schema delle prestazioni da parte di Performance Insights.
performance_schema_events_stages_history_size	Sì	Se la colonna Origine per il parametro performance_schema è impostata su Modified, significa che lo schema delle prestazioni utilizza il parametro performance_schema_events_stages_history_size. Per ulteriori informazioni sull'abilitazione dello schema delle prestazioni, consulta Determinazione della gestione dello schema delle prestazioni da parte di Performance Insights.

Nome del parametro	Modificabili	Note
`performance_schema_events_statements_history_long_size`	Sì	Se la colonna Origine per il parametro `performance_schema` è impostata su `Modified`, significa che lo schema delle prestazioni utilizza il parametro `performance_schema_events_statements_history_long_size`. Per ulteriori informazioni sull'abilitazione dello schema delle prestazioni, consulta Determinazione della gestione dello schema delle prestazioni da parte di Performance Insights.
`performance_schema_events_statements_history_size`	Sì	Se la colonna Origine per il parametro `performance_schema` è impostata su `Modified`, significa che lo schema delle prestazioni utilizza il parametro `performance_schema_events_statements_history_size`. Per ulteriori informazioni sull'abilitazione dello schema delle prestazioni, consulta Determinazione della gestione dello schema delle prestazioni da parte di Performance Insights.

Nome del parametro	Modificabili	Note
performance_schema_events_transactions_history_long_size	Sì	Se la colonna Origine per il parametro performance_schema è impostata su Modified, significa che lo schema delle prestazioni utilizza il parametro performance_schema_events_transactions_history_long_size. Per ulteriori informazioni sull'abilitazione dello schema delle prestazioni, consulta Determinazione della gestione dello schema delle prestazioni da parte di Performance Insights.
performance_schema_events_transactions_history_size	Sì	Se la colonna Origine per il parametro performance_schema è impostata su Modified, significa che lo schema delle prestazioni utilizza il parametro performance_schema_events_transactions_history_size. Per ulteriori informazioni sull'abilitazione dello schema delle prestazioni, consulta Determinazione della gestione dello schema delle prestazioni da parte di Performance Insights.

Nome del parametro	Modificabili	Note
performance_schema_events_waits_history_long_size	Sì	Se la colonna Origine per il parametro performance_schema è impostata su Modified, significa che lo schema delle prestazioni utilizza il parametro performance_schema_events_waits_history_long_size. Per ulteriori informazioni sull'abilitazione dello schema delle prestazioni, consulta Determinazione della gestione dello schema delle prestazioni da parte di Performance Insights.
performance_schema_events_waits_history_size	Sì	Se la colonna Origine per il parametro performance_schema è impostata su Modified, significa che lo schema delle prestazioni utilizza il parametro performance_schema_events_waits_history_size. Per ulteriori informazioni sull'abilitazione dello schema delle prestazioni, consulta Determinazione della gestione dello schema delle prestazioni da parte di Performance Insights.
performance_schema_hosts_size	Sì	Se la colonna Origine per il parametro performance_schema è impostata su Modified, significa che lo schema delle prestazioni utilizza il parametro performance_schema_hosts_size. Per ulteriori informazioni sull'abilitazione dello schema delle prestazioni, consulta Determinazione della gestione dello schema delle prestazioni da parte di Performance Insights.

Nome del parametro	Modificabili	Note
performance_schema_max_cond_classes	Sì	Se la colonna Origine per il parametro performance_schema è impostata su Modified, significa che lo schema delle prestazioni utilizza il parametro performance_schema_max_cond_classes . Per ulteriori informazioni sull'abilitazione dello schema delle prestazioni, consulta Determinazione della gestione dello schema delle prestazioni da parte di Performance Insights.
performance_schema_max_cond_instances	Sì	Se la colonna Origine per il parametro performance_schema è impostata su Modified, significa che lo schema delle prestazioni utilizza il parametro performance_schema_max_cond_instances . Per ulteriori informazioni sull'abilitazione dello schema delle prestazioni, consulta Determinazione della gestione dello schema delle prestazioni da parte di Performance Insights.
performance_schema_max_digest_length	Sì	Se la colonna Origine per il parametro performance_schema è impostata su Modified, significa che lo schema delle prestazioni utilizza il parametro performance_schema_max_digest_length . Per ulteriori informazioni sull'abilitazione dello schema delle prestazioni, consulta Determinazione della gestione dello schema delle prestazioni da parte di Performance Insights.

Nome del parametro	Modificabili	Note
performance_schema_max_file_classes	Sì	Se la colonna Origine per il parametro performance_schema è impostata su Modified, significa che lo schema delle prestazioni utilizza il parametro performance_schema_max_file_classes . Per ulteriori informazioni sull'abilitazione dello schema delle prestazioni, consulta Determinazione della gestione dello schema delle prestazioni da parte di Performance Insights.
performance_schema_max_file_handles	Sì	Se la colonna Origine per il parametro performance_schema è impostata su Modified, significa che lo schema delle prestazioni utilizza il parametro performance_schema_max_file_handles . Per ulteriori informazioni sull'abilitazione dello schema delle prestazioni, consulta Determinazione della gestione dello schema delle prestazioni da parte di Performance Insights.
performance_schema_max_file_instances	Sì	Se la colonna Origine per il parametro performance_schema è impostata su Modified, significa che lo schema delle prestazioni utilizza il parametro performance_schema_max_file_instances . Per ulteriori informazioni sull'abilitazione dello schema delle prestazioni, consulta Determinazione della gestione dello schema delle prestazioni da parte di Performance Insights.

Nome del parametro	Modificabili	Note
performance_schema_max_index_stat	Sì	Se la colonna Origine per il parametro performance_schema è impostata su Modified, significa che lo schema delle prestazioni utilizza il parametro performance_schema_max_index_stat . Per ulteriori informazioni sull'abilitazione dello schema delle prestazioni, consulta Determinazione della gestione dello schema delle prestazioni da parte di Performance Insights.
performance_schema_max_memory_classes	Sì	Se la colonna Origine per il parametro performance_schema è impostata su Modified, significa che lo schema delle prestazioni utilizza il parametro performance_schema_max_memory_classes . Per ulteriori informazioni sull'abilitazione dello schema delle prestazioni, consulta Determinazione della gestione dello schema delle prestazioni da parte di Performance Insights.
performance_schema_max_metadata_locks	Sì	Se la colonna Origine per il parametro performance_schema è impostata su Modified, significa che lo schema delle prestazioni utilizza il parametro performance_schema_max_metadata_locks . Per ulteriori informazioni sull'abilitazione dello schema delle prestazioni, consulta Determinazione della gestione dello schema delle prestazioni da parte di Performance Insights.

Nome del parametro	Modificabili	Note
performance_schema_max_mutex_classes	Sì	Se la colonna Origine per il parametro performance_schema è impostata su Modified, significa che lo schema delle prestazioni utilizza il parametro performance_schema_max_mutex_classes. Per ulteriori informazioni sull'abilitazione dello schema delle prestazioni, consulta Determinazione della gestione dello schema delle prestazioni da parte di Performance Insights.
performance_schema_max_mutex_instances	Sì	Se la colonna Origine per il parametro performance_schema è impostata su Modified, significa che lo schema delle prestazioni utilizza il parametro performance_schema_max_mutex_instances. Per ulteriori informazioni sull'abilitazione dello schema delle prestazioni, consulta Determinazione della gestione dello schema delle prestazioni da parte di Performance Insights.
performance_schema_max_prepared_statements_instances	Sì	Se la colonna Origine per il parametro performance_schema è impostata su Modified, significa che lo schema delle prestazioni utilizza il parametro performance_schema_max_prepared_statements_instances. Per ulteriori informazioni sull'abilitazione dello schema delle prestazioni, consulta Determinazione della gestione dello schema delle prestazioni da parte di Performance Insights.

Nome del parametro	Modificabili	Note
performance_schema_max_program_instances	Sì	Se la colonna Origine per il parametro performance_schema è impostata su Modified, significa che lo schema delle prestazioni utilizza il parametro performance_schema_max_program_instances. Per ulteriori informazioni sull'abilitazione dello schema delle prestazioni, consulta Determinazione della gestione dello schema delle prestazioni da parte di Performance Insights.
performance_schema_max_rwlock_classes	Sì	Se la colonna Origine per il parametro performance_schema è impostata su Modified, significa che lo schema delle prestazioni utilizza il parametro performance_schema_max_rwlock_classes. Per ulteriori informazioni sull'abilitazione dello schema delle prestazioni, consulta Determinazione della gestione dello schema delle prestazioni da parte di Performance Insights.
performance_schema_max_rwlock_instances	Sì	Se la colonna Origine per il parametro performance_schema è impostata su Modified, significa che lo schema delle prestazioni utilizza il parametro performance_schema_max_rwlock_instances. Per ulteriori informazioni sull'abilitazione dello schema delle prestazioni, consulta Determinazione della gestione dello schema delle prestazioni da parte di Performance Insights.

Nome del parametro	Modificabili	Note
performance_schema_max_socket_classes	Sì	Se la colonna Origine per il parametro performance_schema è impostata su Modified, significa che lo schema delle prestazioni utilizza il parametro performance_schema_max_socket_classes. Per ulteriori informazioni sull'abilitazione dello schema delle prestazioni, consulta Determinazione della gestione dello schema delle prestazioni da parte di Performance Insights.
performance_schema_max_socket_instances	Sì	Se la colonna Origine per il parametro performance_schema è impostata su Modified, significa che lo schema delle prestazioni utilizza il parametro performance_schema_max_socket_instances. Per ulteriori informazioni sull'abilitazione dello schema delle prestazioni, consulta Determinazione della gestione dello schema delle prestazioni da parte di Performance Insights.
performance_schema_max_sql_text_length	Sì	Se la colonna Origine per il parametro performance_schema è impostata su Modified, significa che lo schema delle prestazioni utilizza il parametro performance_schema_max_sql_text_length. Per ulteriori informazioni sull'abilitazione dello schema delle prestazioni, consulta Determinazione della gestione dello schema delle prestazioni da parte di Performance Insights.

Nome del parametro	Modificabili	Note
performance_schema_max_stage_classes	Sì	Se la colonna Origine per il parametro performance_schema è impostata su Modified, significa che lo schema delle prestazioni utilizza il parametro performance_schema_max_stage_classes. Per ulteriori informazioni sull'abilitazione dello schema delle prestazioni, consulta Determinazione della gestione dello schema delle prestazioni da parte di Performance Insights.
performance_schema_max_statement_classes	Sì	Se la colonna Origine per il parametro performance_schema è impostata su Modified, significa che lo schema delle prestazioni utilizza il parametro performance_schema_max_statement_classes. Per ulteriori informazioni sull'abilitazione dello schema delle prestazioni, consulta Determinazione della gestione dello schema delle prestazioni da parte di Performance Insights.
performance_schema_max_statement_stack	Sì	Se la colonna Origine per il parametro performance_schema è impostata su Modified, significa che lo schema delle prestazioni utilizza il parametro performance_schema_max_statement_stack. Per ulteriori informazioni sull'abilitazione dello schema delle prestazioni, consulta Determinazione della gestione dello schema delle prestazioni da parte di Performance Insights.

Nome del parametro	Modificabili	Note
performance_schema_max_table_handles	Sì	Se la colonna Origine per il parametro performance_schema è impostata su Modified, significa che lo schema delle prestazioni utilizza il parametro performance_schema_max_table_handles. Per ulteriori informazioni sull'abilitazione dello schema delle prestazioni, consulta Determinazione della gestione dello schema delle prestazioni da parte di Performance Insights.
performance_schema_max_table_instances	Sì	Se la colonna Origine per il parametro performance_schema è impostata su Modified, significa che lo schema delle prestazioni utilizza il parametro performance_schema_max_table_instances. Per ulteriori informazioni sull'abilitazione dello schema delle prestazioni, consulta Determinazione della gestione dello schema delle prestazioni da parte di Performance Insights.
performance_schema_max_table_lock_stat	Sì	Se la colonna Origine per il parametro performance_schema è impostata su Modified, significa che lo schema delle prestazioni utilizza il parametro performance_schema_max_table_lock_stat. Per ulteriori informazioni sull'abilitazione dello schema delle prestazioni, consulta Determinazione della gestione dello schema delle prestazioni da parte di Performance Insights.

Nome del parametro	Modificabili	Note
performance_schema_max_thread_classes	Sì	Se la colonna Origine per il parametro performance_schema è impostata su Modified, significa che lo schema delle prestazioni utilizza il parametro performance_schema_max_thread_classes. Per ulteriori informazioni sull'abilitazione dello schema delle prestazioni, consulta Determinazione della gestione dello schema delle prestazioni da parte di Performance Insights.
performance_schema_max_thread_instances	Sì	Se la colonna Origine per il parametro performance_schema è impostata su Modified, significa che lo schema delle prestazioni utilizza il parametro performance_schema_max_thread_instances. Per ulteriori informazioni sull'abilitazione dello schema delle prestazioni, consulta Determinazione della gestione dello schema delle prestazioni da parte di Performance Insights.
performance_schema_session_connect_attrs_size	Sì	Se la colonna Origine per il parametro performance_schema è impostata su Modified, significa che lo schema delle prestazioni utilizza il parametro performance_schema_session_connect_attrs_size. Per ulteriori informazioni sull'abilitazione dello schema delle prestazioni, consulta Determinazione della gestione dello schema delle prestazioni da parte di Performance Insights.

Nome del parametro	Modificabili	Note
`performance_schema_setup_actors_size`	Sì	Se la colonna Origine per il parametro `performance_schema` è impostata su `Modified`, significa che lo schema delle prestazioni utilizza il parametro `performance_schema_setup_actors_size`. Per ulteriori informazioni sull'abilitazione dello schema delle prestazioni, consulta Determinazione della gestione dello schema delle prestazioni da parte di Performance Insights.
`performance_schema_setup_objects_size`	Sì	Se la colonna Origine per il parametro `performance_schema` è impostata su `Modified`, significa che lo schema delle prestazioni utilizza il parametro `performance_schema_setup_objects_size`. Per ulteriori informazioni sull'abilitazione dello schema delle prestazioni, consulta Determinazione della gestione dello schema delle prestazioni da parte di Performance Insights.

Nome del parametro	Modificabili	Note
`performance_schema_show_processlist`	Sì	Questo parametro determina quale implementazione `SHOW PROCESSLIST` utilizzare: • L'implementazione predefinita esegue iterazioni tra i thread attivi dall'interno del gestore dei thread mantenendo un mutex globale. Ciò può causare un rallentamento delle prestazioni, in particolare su sistemi occupati. • L'implementazione `SHOW PROCESSLIST` alternativa è basata sulla tabella `processlist` dello schema di prestazioni. Questa implementazione esegue la query sui dati dei thread attivi dallo schema di prestazioni anziché dal gestore dei thread e non richiede un mutex. Questo parametro si applica ad Aurora MySQL versione 2.12 e successive e versione 3. Se la colonna Origine per il parametro `performance_schema` è impostata su `Modified`, significa che lo schema delle prestazioni utilizza il parametro `performance_schema_show_processlist`. Per ulteriori informazioni sull'abilitazione dello schema delle prestazioni, consulta Determinazione della gestione dello schema delle prestazioni da parte di Performance Insights.

Nome del parametro	Modificabili	Note
performance_schema_users_size	Sì	Se la colonna Origine per il parametro performance_schema è impostata su Modified, significa che lo schema delle prestazioni utilizza il parametro performance_schema_users_size. Per ulteriori informazioni sull'abilitazione dello schema delle prestazioni, consulta Determinazione della gestione dello schema delle prestazioni da parte di Performance Insights.
pid_file	No	Nessuno
plugin_dir	No	Aurora MySQL utilizza istanze gestite in cui non si accede direttamente al filesystem.
port	No	Aurora MySQL gestisce le proprietà della connessione e applica impostazioni coerenti per tutte le istanze database in un cluster.
preload_buffer_size	Sì	La dimensione del buffer allocato durante il precaricamento degli indici. Questo parametro si applica ad Aurora MySQL versione 3.
profiling_history_size	Sì	Nessuno
query_alloc_block_size	Sì	Nessuno
query_cache_limit	Sì	Rimosso da Aurora MySQL versione 3.
query_cache_min_res_unit	Sì	Rimosso da Aurora MySQL versione 3.

Nome del parametro	Modificabili	Note
query_cache_size	Sì	Il valore predefinito è rappresentato da una formula. Per informazioni dettagliate su come viene calcolato il valore DBInstanceClassMemory nella formula, vedi Variabili di formula dei parametri database. Rimosso da Aurora MySQL versione 3.
query_cache_type	Sì	Rimosso da Aurora MySQL versione 3.
query_cache_wlock_invalidate	Sì	Rimosso da Aurora MySQL versione 3.
query_prealloc_size	Sì	Nessuno
range_alloc_block_size	Sì	Nessuno
read_buffer_size	Sì	Nessuno

Nome del parametro	Modificabili	Note
`read_only`	Sì	Quando questo parametro è attivato, il server non consente aggiornamenti tranne quelli eseguiti dai thread di replica. I valori validi per Aurora MySQL versione 2 sono i seguenti: • `0` – `OFF` • `1` – `ON` • `{TrueIfReplica}` - `ON` per le repliche di lettura. Si tratta del valore di default. • `{TrueIfClusterReplica}` - `ON` per le istanze nei cluster di replica come le repliche di lettura tra Regioni, i cluster secondari in un Database globale Aurora e le implementazioni blu/verdi. Ti consigliamo di utilizzare il gruppo di parametri del cluster di database in Aurora MySQL versione 2 per garantire che il parametro `read_only` venga applicato alle nuove istanze di scrittura in caso di failover. **Note** Le istanze di lettura sono sempre di sola lettura, perché Aurora MySQL imposta `innodb_read_only` su `1` su tutte le letture. Pertanto,

Nome del parametro	Modificabili	Note
		`read_only` è ridondante sulle istanze di lettura. È stato rimosso a livello di istanza da Aurora MySQL versione 3.
`read_rnd_buffer_size`	Sì	Nessuno
`relay-log`	No	Nessuno
`relay_log_info_repository`	Sì	Rimosso da Aurora MySQL versione 3.
`relay_log_recovery`	No	Nessuno
`replica_checkpoint_group`	Sì	Aurora MySQL versione 3
`replica_checkpoint_period`	Sì	Aurora MySQL versione 3
`replica_parallel_workers`	Sì	Aurora MySQL versione 3
`replica_pending_jobs_size_max`	Sì	Aurora MySQL versione 3
`replica_skip_errors`	Sì	Aurora MySQL versione 3
`replica_sql_verify_checksum`	Sì	Aurora MySQL versione 3
`safe-user-create`	Sì	Nessuno
`secure_auth`	Sì	Questo parametro è sempre attivato in Aurora MySQL versione 2. Il tentativo di disattivarlo genera un errore. Rimosso da Aurora MySQL versione 3.

Nome del parametro	Modificabili	Note
secure_file_priv	No	Aurora MySQL utilizza istanze gestite in cui non si accede direttamente al filesystem.
show_create_table_verbosity	Sì	Abilitando questa variabile, SHOW_CREATE_TABLE visualizza ROW_FORMAT a prescindere che si tratti del formato predefinito. Questo parametro si applica ad Aurora MySQL versione 2.12 e successive e versione 3.
skip-slave-start	No	Nessuno
skip_external_locking	No	Nessuno
skip_show_database	Sì	Nessuno
slave_checkpoint_group	Sì	Aurora MySQL versione 2. Utilizza replica_checkpoint_group in Aurora MySQL versione 3.
slave_checkpoint_period	Sì	Aurora MySQL versione 2. Utilizza replica_checkpoint_period in Aurora MySQL versione 3.
slave_parallel_workers	Sì	Aurora MySQL versione 2. Utilizza replica_parallel_workers in Aurora MySQL versione 3.
slave_pending_jobs_size_max	Sì	Aurora MySQL versione 2. Utilizza replica_pending_jobs_size_max in Aurora MySQL versione 3.

Nome del parametro	Modificabili	Note
slave_sql_verify_checksum	Sì	Aurora MySQL versione 2. Utilizza replica_sql_verify_checksum in Aurora MySQL versione 3.
slow_launch_time	Sì	Nessuno
slow_query_log	Sì	Per istruzioni sul caricamento dei log su CloudWatch Logs, consulta Pubblicazione di log Amazon Aurora MySQL in Amazon CloudWatch Logs.
slow_query_log_file	No	Aurora MySQL utilizza istanze gestite in cui non si accede direttamente al filesystem.
socket	No	Nessuno
sort_buffer_size	Sì	Nessuno
sql_mode	Sì	Nessuno
sql_select_limit	Sì	Nessuno
stored_program_cache	Sì	Nessuno
sync_binlog	No	Nessuno
sync_master_info	Sì	Nessuno
sync_source_info	Sì	Questo parametro si applica ad Aurora MySQL versione 3.
sync_relay_log	Sì	Rimosso da Aurora MySQL versione 3.
sync_relay_log_info	Sì	Nessuno
sysdate-is-now	Sì	Nessuno

Nome del parametro	Modificabili	Note
`table_cache_element_entry_ttl`	No	Nessuno
`table_definition_cache`	Sì	Il valore predefinito è rappresentato da una formula. Per informazioni dettagliate su come viene calcolato il valore `DBInstanceClassMemory` nella formula, vedi Variabili di formula dei parametri database.
`table_open_cache`	Sì	Il valore predefinito è rappresentato da una formula. Per informazioni dettagliate su come viene calcolato il valore `DBInstanceClassMemory` nella formula, vedi Variabili di formula dei parametri database.
`table_open_cache_instances`	Sì	Nessuno
`temp-pool`	Sì	Rimosso da Aurora MySQL versione 3.
`temptable_max_mmap`	Sì	Questo parametro si applica ad Aurora MySQL versione 3. Per informazioni dettagliate, consultare Nuovo comportamento della tabella temporanea in Aurora MySQL versione 3.
`temptable_max_ram`	Sì	Questo parametro si applica ad Aurora MySQL versione 3. Per informazioni dettagliate, consultare Nuovo comportamento della tabella temporanea in Aurora MySQL versione 3.

Nome del parametro	Modificabili	Note
`temptable_use_mmap`	Sì	Questo parametro si applica ad Aurora MySQL versione 3. Per informazioni dettagliate, consultare Nuovo comportamento della tabella temporanea in Aurora MySQL versione 3.
`thread_cache_size`	Sì	Il numero di thread da memorizzare nella cache. Questo parametro si applica ad Aurora MySQL versioni 2 e 3.
`thread_handling`	No	Nessuno
`thread_stack`	Sì	Nessuno
`timed_mutexes`	Sì	Nessuno
`tmp_table_size`	Sì	Definisce le dimensioni massime delle tabelle temporanee in memoria create dal motore di storage `MEMORY` in Aurora MySQL versione 3. In Aurora MySQL versione 3.04 e successive, definisce le dimensioni massime delle tabelle temporanee in memoria create dal motore di storage `TempTable` quando `aurora_tmptable_enable_per_table_limit` è `ON`. Per ulteriori informazioni, consulta Limitazione delle dimensioni delle tabelle temporanee interne in memoria.
`tmpdir`	No	Aurora MySQL utilizza istanze gestite in cui non si accede direttamente al filesystem.

Nome del parametro	Modificabili	Note
transaction_alloc_block_size	Sì	Nessuno
transaction_isolation	Sì	Questo parametro si applica ad Aurora MySQL versione 3. Sostituisce tx_isolation .
transaction_prealloc_size	Sì	Nessuno
tx_isolation	Sì	Rimosso da Aurora MySQL versione 3. È sostituito da transaction_isolation .
updatable_views_with_limit	Sì	Nessuno
validate-password	No	Nessuno
validate_password_dictionary_file	No	Nessuno
validate_password_length	No	Nessuno
validate_password_mixed_case_count	No	Nessuno
validate_password_number_count	No	Nessuno
validate_password_policy	No	Nessuno
validate_password_special_char_count	No	Nessuno

Nome del parametro	Modificabili	Note
`wait_timeout`	Sì	Aurora valuta il valore minimo di `interactive_timeout` e `wait_timeout` . Quindi usa quel minimo come timeout per terminare tutte le sessioni inattive, sia interattive sia non interattive.

I parametri MySQL seguenti non si applicano ad Aurora MySQL

A causa delle differenze tra l'architettura di Aurora MySQL e quella di MySQL, alcuni parametri e variabili di stato MySQL non si applicano ad Aurora MySQL.

I parametri MySQL seguenti non si applicano ad Aurora MySQL. L'elenco non è completo.

- `activate_all_roles_on_login`: questo parametro non si applica ad Aurora MySQL versione 2. E' disponibile in Aurora MySQL versione 3.
- `big_tables`
- `bind_address`
- `character_sets_dir`
- `innodb_adaptive_flushing`
- `innodb_adaptive_flushing_lwm`
- `innodb_buffer_pool_chunk_size`
- `innodb_buffer_pool_instances`
- `innodb_change_buffering`
- `innodb_checksum_algorithm`
- `innodb_data_file_path`
- `innodb_dedicated_server`
- `innodb_doublewrite`
- `innodb_flush_log_at_timeout`: questo parametro non si applica ad Aurora MySQL. Per ulteriori informazioni, consulta Configurazione della frequenza di svuotamento del buffer dei registri.
- `innodb_flush_method`

- innodb_flush_neighbors
- innodb_io_capacity
- innodb_io_capacity_max
- innodb_log_buffer_size
- innodb_log_file_size
- innodb_log_files_in_group
- innodb_log_spin_cpu_abs_lwm
- innodb_log_spin_cpu_pct_hwm
- innodb_log_writer_threads
- innodb_max_dirty_pages_pct
- innodb_numa_interleave
- innodb_page_size
- innodb_redo_log_capacity
- innodb_redo_log_encrypt
- innodb_undo_log_encrypt
- innodb_undo_log_truncate
- innodb_undo_logs
- innodb_undo_tablespaces
- innodb_use_native_aio
- innodb_write_io_threads

Variabili di stato globali di Aurora MySQL

Aurora MySQL include variabili di stato fornite da Community MySQL e variabili che sono specifiche di Aurora. È possibile esaminare queste variabili per scoprire cosa succede all'interno del motore di database. Per ulteriori informazioni sulle variabili di stato in Community MySQL, consulta Server Status Variables nella documentazione di Community MySQL 8.0.

È possibile individuare i valori correnti delle variabili di stato globali di Aurora MySQL utilizzando un'istruzione come la seguente:

```
show global status like '%aurora%';
```

> **Note**
>
> Il valore delle variabili di stato globali viene cancellato al riavvio del motore di database.

Nella tabella seguente vengono descritte le variabili di stato globali utilizzate da Aurora MySQL.

Nome	Descrizione
`AuroraDb_commits`	Il numero totale di commit dall'ultimo riavvio.
`AuroraDb_commit_latency`	La latenza del commit aggregata dall'ultimo riavvio.
`AuroraDb_ddl_stmt_duration`	La latenza DDL aggregata dall'ultimo riavvio.
`AuroraDb_select_stmt_duration`	La latenza dell'istruzione `SELECT` aggregata dall'ultimo riavvio.
`AuroraDb_insert_stmt_duration`	La latenza dell'istruzione `INSERT` aggregata dall'ultimo riavvio.
`AuroraDb_update_stmt_duration`	La latenza dell'istruzione `UPDATE` aggregata dall'ultimo riavvio.
`AuroraDb_delete_stmt_duration`	La latenza dell'istruzione `DELETE` aggregata dall'ultimo riavvio.
`Aurora_binlog_io_cache_allocated`	Il numero di byte allocati alla cache I/O binlog.
`Aurora_binlog_io_cache_read_requests`	Il numero di richieste di lettura effettuate alla cache I/O binlog.
`Aurora_binlog_io_cache_reads`	Il numero di richieste di lettura gestite dalla cache I/O binlog.
`Aurora_enhanced_binlog`	Indica se il binlog avanzato è abilitato o disabilitato per questa istanza database. Per ulteriori

Nome	Descrizione
	informazioni, consulta Configurazione del file di log binario avanzato per Aurora MySQL.
`Aurora_external_connection_count`	Il numero di connessioni database all'istanza database, ad esclusione delle connessioni al servizio RDS utilizzate per i controlli dell'integrità del database.
`Aurora_fast_insert_cache_hits`	Un contatore che viene incrementato quando il cursore memorizzato nella cache viene recuperato e verificato. Per ulteriori informazioni sulla cache di inserimento rapido, consulta Miglioramenti alle prestazioni di Amazon Aurora MySQL.
`Aurora_fast_insert_cache_misses`	Un contatore che viene incrementato quando il cursore memorizzato nella cache non è più valido e Aurora esegue un attraversamento di indice normale. Per ulteriori informazioni sulla cache di inserimento rapido, consulta Miglioramenti alle prestazioni di Amazon Aurora MySQL.
`Aurora_fts_cache_memory_used`	La quantità di memoria in byte utilizzata dal sistema di ricerca full-text InnoDB. Questa variabile si applica ad Aurora MySQL versione 3.07 e successive.
`Aurora_fwd_master_dml_stmt_count`	Il numero totale di istruzioni DML inoltrate a questa istanza database di scrittura. Questa variabile si applica ad Aurora MySQL versione 2.
`Aurora_fwd_master_dml_stmt_duration`	La durata totale delle istruzioni DML inoltrate a questa istanza database di scrittura. Questa variabile si applica ad Aurora MySQL versione 2.

Nome	Descrizione
`Aurora_fwd_master_errors_rpc_timeout`	Il numero di volte in cui non è stato possibile stabilire una connessione inoltrata sull'istanza di scrittura.
`Aurora_fwd_master_errors_session_limit`	Il numero di query inoltrate che vengono rifiutate a causa di `session full` sull'istanza di scrittura.
`Aurora_fwd_master_errors_session_timeout`	Il numero di volte in cui una sessione di inoltro viene terminata a causa di un timeout sull'istanza di scrittura.
`Aurora_fwd_master_open_sessions`	Il numero di sessioni inoltrate sull'istanza database di scrittura. Questa variabile si applica ad Aurora MySQL versione 2.
`Aurora_fwd_master_select_stmt_count`	Il numero totale di istruzioni `SELECT` inoltrate a questa istanza database di scrittura. Questa variabile si applica ad Aurora MySQL versione 2.
`Aurora_fwd_master_select_stmt_duration`	La durata totale delle istruzioni `SELECT` inoltrate a questa istanza database di scrittura. Questa variabile si applica ad Aurora MySQL versione 2.
`Aurora_fwd_writer_dml_stmt_count`	Il numero totale di istruzioni DML inoltrate a questa istanza database di scrittura. Questa variabile si applica ad Aurora MySQL versione 3.
`Aurora_fwd_writer_dml_stmt_duration`	La durata totale delle istruzioni DML inoltrate a questa istanza database di scrittura. Questa variabile si applica ad Aurora MySQL versione 3.

Nome	Descrizione
`Aurora_fwd_writer_errors_rpc_timeout`	Il numero di volte in cui non è stato possibile stabilire una connessione inoltrata sull'istanza di scrittura.
`Aurora_fwd_writer_errors_session_limit`	Il numero di query inoltrate che vengono rifiutate a causa di `session full` sull'istanza di scrittura.
`Aurora_fwd_writer_errors_session_timeout`	Il numero di volte in cui una sessione di inoltro viene terminata a causa di un timeout sull'istanza di scrittura.
`Aurora_fwd_writer_open_sessions`	Il numero di sessioni inoltrate sull'istanza database di scrittura. Questa variabile si applica ad Aurora MySQL versione 3.
`Aurora_fwd_writer_select_stmt_count`	Il numero totale di istruzioni `SELECT` inoltrate a questa istanza database di scrittura. Questa variabile si applica ad Aurora MySQL versione 3.
`Aurora_fwd_writer_select_stmt_duration`	La durata totale delle istruzioni `SELECT` inoltrate a questa istanza database di scrittura. Questa variabile si applica ad Aurora MySQL versione 3.
`Aurora_lockmgr_buffer_pool_memory_used`	La quantità di memoria del pool di buffer in byte utilizzata dalla gestione dei blocchi Aurora MySQL.
`Aurora_lockmgr_memory_used`	La quantità di memoria in byte utilizzata dalla gestione dei blocchi Aurora MySQL.

Nome	Descrizione
`Aurora_ml_actual_request_cnt`	Il conteggio delle richieste di aggregazione effettuate da Aurora MySQL ai servizi di machine learning di Aurora per tutte le query eseguite dagli utenti dell'istanza database. Per ulteriori informazioni, consulta Utilizzo di machine learning di Amazon Aurora con Aurora MySQL.
`Aurora_ml_actual_response_cnt`	Il conteggio delle risposte aggregate che Aurora MySQL riceve dai servizi di machine learning di Aurora per tutte le query eseguite dagli utenti dell'istanza database. Per ulteriori informazioni, consulta Utilizzo di machine learning di Amazon Aurora con Aurora MySQL.
`Aurora_ml_cache_hit_cnt`	Il conteggio degli hit della cache interna aggregata che Aurora MySQL riceve dai servizi di machine learning di Aurora per tutte le query eseguite dagli utenti dell'istanza database. Per ulteriori informazioni, consulta Utilizzo di machine learning di Amazon Aurora con Aurora MySQL.
`Aurora_ml_logical_request_cnt`	Il numero di richieste logiche valutate dall'istanza database per essere inviate ai servizi di machine learning di Aurora dall'ultimo ripristino dello stato. A seconda del fatto che sia stato utilizzato il batching, questo valore può essere superiore a `Aurora_ml_actual_request_cnt`. Per ulteriori informazioni, consulta Utilizzo di machine learning di Amazon Aurora con Aurora MySQL.

Nome	Descrizione
`Aurora_ml_logical_response_cnt`	Il conteggio delle risposte aggregate che Aurora MySQL riceve dai servizi di machine learning di Aurora per tutte le query eseguite dagli utenti dell'istanza database. Per ulteriori informazioni, consulta Utilizzo di machine learning di Amazon Aurora con Aurora MySQL.
`Aurora_ml_retry_request_cnt`	Il numero di richieste ripetute inviate dall'istanza database ai servizi di machine learning di Aurora dall'ultimo ripristino dello stato. Per ulteriori informazioni, consulta Utilizzo di machine learning di Amazon Aurora con Aurora MySQL.
`Aurora_ml_single_request_cnt`	Il conteggio aggregato delle funzioni di machine learning di Aurora valutate in modalità non batch per tutte le query eseguite dagli utenti dell'istanza database. Per ulteriori informazioni, consulta Utilizzo di machine learning di Amazon Aurora con Aurora MySQL.
`aurora_oom_avoidance_recovery_state`	Indica se la funzione di recupero con riduzione dei problemi di esaurimento della memoria di Aurora, è nello stato `ACTIVE` o `INACTIVE` per l'istanza database in questione. Questa variabile si applica ad Aurora MySQL versione 3.06.0 e successive.

Nome	Descrizione
`aurora_oom_reserved_mem_enter_kb`	Rappresenta la soglia per il passaggio allo stato `RESERVED` nel meccanismo di gestione dei problemi di esaurimento della memoria di Aurora. Quando la memoria disponibile sul server scende al di sotto di questa soglia, il valore di `aurora_oom_status` passa a `RESERVED`, a indicare che il server si sta avvicinando a un livello critico di utilizzo della memoria. Questa variabile si applica ad Aurora MySQL versione 3.06.0 e successive.
`aurora_oom_reserved_mem_exit_kb`	Rappresenta la soglia per l'uscita dallo stato `RESERVED` nel meccanismo di gestione dei problemi di esaurimento della memoria di Aurora. Quando la memoria disponibile sul server supera questa soglia, `aurora_oom_status` ritorna a `NORMAL`, a indicare che il server è tornato a uno stato più stabile con risorse di memoria sufficienti. Questa variabile si applica ad Aurora MySQL versione 3.06.0 e successive.

Nome	Descrizione
aurora_oom_status	Rappresenta lo stato corrente dell'istanza database relativo all'eventuale esaurimento della memoria. Quando il valore è NORMAL, significa che le risorse di memoria sono sufficienti. Se il valore passa a RESERVED, significa che la memoria disponibile sul server è insuffici ente. Vengono intraprese delle azioni in base alla configurazione del parametro aurora_oo m_response . Per ulteriori informazioni, consulta Risoluzio ne dei problemi di memoria insufficiente per i database Aurora MySQL. Questa variabile si applica ad Aurora MySQL versione 3.06.0 e successive.
Aurora_pq_bytes_returned	Il numero di byte per le strutture di dati tuple trasmesse al nodo head nel corso delle query in parallelo. Dividi questo valore per 16.384 per compararlo a Aurora_pq_pages_pu shed_down .
Aurora_pq_max_concurrent_re quests	Il numero massimo di sessioni di query in parallelo eseguibili simultaneamente su questa istanza database Aurora. È un numero fisso che dipende dalla classe di istanza DB AWS.
Aurora_pq_pages_pushed_down	Il numero di pagine di dati (ognuna con una dimensione fissa di 16 KiB) per le quali una query in parallelo ha evitato una trasmissione di rete al nodo head.

Nome	Descrizione
`Aurora_pq_request_attempted`	Il numero di sessioni di query in parallelo necessarie. Questo valore può rappresentare più di una sessione per query, a seconda dei costrutti SQL come sottoquery e join.
`Aurora_pq_request_executed`	Il numero di sessioni di query in parallelo riuscite.
`Aurora_pq_request_failed`	Il numero di sessioni di query in parallelo che hanno restituito un errore al client. In alcuni casi, una richiesta di query in parallelo può non riuscire, ad esempio a causa di un problema nel livello di storage. In questi casi, viene eseguito un nuovo tentativo per la parte di query non riuscita utilizzando il meccanismo di query non in parallelo. Se anche il nuovo tentativo non riesce, viene restituito un errore al client e questo contatore viene incrementato.
`Aurora_pq_request_in_progress`	Il numero di sessioni di query in parallelo attualmente in corso. Questo numero si applica all'istanza database Aurora a cui sei connesso e non all'intero cluster di database Aurora. Per determinare se un'istanza database è prossima al relativo limite di simultaneità, compara questo valore a `Aurora_pq_max_concurrent_requests` .
`Aurora_pq_request_not_chosen`	Il numero di volte che una query in parallelo non è stata scelta per una query. Questo valore è la somma di vari altri contatori più granulari . Un'istruzione `EXPLAIN` può incrementare questo contatore anche se la query non viene effettivamente eseguita.

Nome	Descrizione
`Aurora_pq_request_not_chosen_below_min_rows`	Il numero di volte che una query in parallelo non è stata scelta a causa del numero di righe nella tabella. Un'istruzione `EXPLAIN` può incrementare questo contatore anche se la query non viene effettivamente eseguita.
`Aurora_pq_request_not_chosen_column_bit`	Numero di richieste di query parallele che utilizzano il percorso di elaborazione delle query non parallele a causa di un tipo di dati non supportato nell'elenco delle colonne proiettate.
`Aurora_pq_request_not_chosen_column_geometry`	Numero di richieste di query parallele che utilizzano il percorso di elaborazione delle query non parallele perché la tabella contiene colonne con il tipo di `GEOMETRY` dati. Per informazioni sulle versioni di Aurora MySQL che rimuovono questa limitazione, consulta Aggiornare cluster di query paralleli a Aurora MySQL versione 3.
`Aurora_pq_request_not_chosen_column_lob`	Numero di richieste di query parallele che utilizzano il percorso di elaborazione delle query non parallele perché la tabella contiene colonne con un tipo di dati `LOB` o colonne `VARCHAR` memorizzate esternamente a causa della lunghezza dichiarata. Per informazioni sulle versioni di Aurora MySQL che rimuovono questa limitazione, consulta Aggiornare cluster di query paralleli a Aurora MySQL versione 3.
`Aurora_pq_request_not_chosen_column_virtual`	Numero di richieste di query parallele che utilizzano il percorso di elaborazione delle query non parallele perché la tabella contiene una colonna virtuale.

Nome	Descrizione
`Aurora_pq_request_not_chosen_custom_charset`	Numero di richieste di query parallele che utilizzano il percorso di elaborazione delle query non parallele perché la tabella dispone di colonne con un set di caratteri personalizzato.
`Aurora_pq_request_not_chosen_fast_ddl`	Il numero di richieste di query parallele che utilizzano il percorso di elaborazione delle query non parallele perché la tabella è attualmente in fase di modifica da un'istruzione `ALTER` DDL veloce.
`Aurora_pq_request_not_chosen_few_pages_outside_buffer_pool`	Il numero di volte che una query in parallelo non è stata scelta anche se meno del 95% dei dati della tabella era già nel pool di buffer, in quanto non c'erano dati non memorizzati nel buffer sufficienti per giustificare l'utilizzo della funzione di query in parallelo.
`Aurora_pq_request_not_chosen_full_text_index`	Numero di richieste di query parallele che utilizzano il percorso di elaborazione delle query non parallele perché la tabella dispone di indici full-text.
`Aurora_pq_request_not_chosen_high_buffer_pool_pct`	Il numero di volte che una query in parallelo non è stata scelta in quanto un'elevata percentuale di dati di tabella (attualmente maggiore del 95%) era già nel pool di buffer. In questi casi, l'ottimizzatore determina che la lettura dei dati del pool di buffer è più efficace. Un'istruzione `EXPLAIN` può incrementare questo contatore anche se la query non viene effettivamente eseguita.

Nome	Descrizione
`Aurora_pq_request_not_chosen_index_hint`	Numero di richieste di query parallele che utilizzano il percorso di elaborazione delle query non parallele perché la query include un hint di indice.
`Aurora_pq_request_not_chosen_innodb_table_format`	Il numero di richieste di query parallele che utilizzano il percorso di elaborazione delle query non parallele perché la tabella utilizza un formato di riga InnoDB non supportato. La query parallela Aurora si applica solo ai formati di riga `COMPACT`, `REDUNDANT` e `DYNAMIC`.
`Aurora_pq_request_not_chosen_long_trx`	Il numero di richieste di query in parallelo che hanno utilizzato il percorso di elaborazione di query non in parallelo in seguito all'avvio della query in una transazione di lunga durata. Un'istruzione `EXPLAIN` può incrementare questo contatore anche se la query non viene effettivamente eseguita.
`Aurora_pq_request_not_chosen_no_where_clause`	Numero di richieste di query parallele che utilizzano il percorso di elaborazione delle query non parallele perché la query non include alcuna clausola `WHERE`.
`Aurora_pq_request_not_chosen_range_scan`	Numero di richieste di query parallele che utilizzano il percorso di elaborazione delle query non parallele perché la query utilizza una scansione di intervallo su un indice.
`Aurora_pq_request_not_chosen_row_length_too_long`	Numero di richieste di query parallele che utilizzano il percorso di elaborazione delle query non parallele perché la lunghezza totale combinata di tutte le colonne è troppo lunga.

Nome	Descrizione
`Aurora_pq_request_not_chosen_small_table`	Il numero di volte che una query in parallelo non è stata scelta a causa della dimensione globale della tabella, come determinata dal numero di righe e dalla lunghezza media delle stesse. Un'istruzione `EXPLAIN` può incrementare questo contatore anche se la query non viene effettivamente eseguita.
`Aurora_pq_request_not_chosen_temporary_table`	Numero di richieste di query parallele che utilizzano il percorso di elaborazione delle query non parallele perché la query fa riferimento a tabelle temporanee che utilizzano i tipi di tabella `MyISAM` o `memory` non supportati.
`Aurora_pq_request_not_chosen_tx_isolation`	Numero di richieste di query parallele che utilizzano il percorso di elaborazione delle query non parallele perché la query utilizza un livello di isolamento delle transazioni non supportato. Sulle istanze del database di lettura, la query parallela si applica solo ai livelli di isolamento `REPEATABLE READ` e `READ COMMITTED` .
`Aurora_pq_request_not_chosen_update_delete_stmts`	Numero di richieste di query parallele che utilizzano il percorso di elaborazione delle query non parallele perché la query fa parte di un'istruzione `UPDATE` o `DELETE`.
`Aurora_pq_request_not_chosen_unsupported_access`	Il numero di richieste di query in parallelo che utilizzano il percorso di elaborazione di query non in parallelo in quanto la clausola `WHERE` non soddisfa i criteri delle query in parallelo . Ciò può avvenire se la query non richiede l'analisi di un grande volume di dati oppure se la query è un'istruzione `DELETE` o `UPDATE`.

Nome	Descrizione
`Aurora_pq_request_not_chosen_unsupported_storage_type`	Il numero di richieste di query in parallelo che utilizzano il percorso di elaborazione delle query non in parallelo perché il cluster database Aurora MySQL non utilizza una configurazione di archiviazione del cluster Aurora supportata. Per ulteriori informazioni, consulta Limitazioni. Questo parametro si applica solo ad Aurora MySQL versione 3.04 e successive.
`Aurora_pq_request_throttled`	Il numero di volte che una query in parallelo non è stata scelta a causa del numero massimo di query in parallelo simultanee già in esecuzione su una particolare istanza database Aurora.
`Aurora_repl_bytes_received`	Numero di byte replicati in un'istanza database di lettura Aurora MySQL dall'ultimo riavvio. Per ulteriori informazioni, consulta Replica con Amazon Aurora MySQL.
`Aurora_reserved_mem_exceeded_incidents`	Il numero di volte dall'ultimo riavvio in cui il motore ha superato i limiti di memoria prenotata. Se `aurora_oom_response` è configurato, questa soglia definisce quando vengono attivate le attività di aggiramento dell'esaurimento della memoria (OOM). Per ulteriori informazioni sulla risposta OOM di Aurora MySQL, consulta Risoluzione dei problemi di memoria insufficiente per i database Aurora MySQL.

Nome	Descrizione
aurora_temptable_max_ram_allocation	La quantità massima di memoria (espressa in byte) utilizzata in un momento qualsiasi dalle tabelle temporanee interne dopo l'ultimo riavvio.
aurora_temptable_ram_allocation	La quantità corrente di memoria (espressa in byte) utilizzata dalle tabelle temporanee interne.
Aurora_in_memory_relaylog_status	Lo stato attuale della funzionalità relativa al log di inoltro in memoria; il valore può essere ENABLED o DISABLED.
Aurora_in_memory_relaylog_disabled_reason	Mostra il motivo dello stato corrente della funzionalità relativa al log di inoltro in memoria. Se la funzionalità è disabilitata, visualizza un messaggio in cui viene spiegato il motivo di tale circostanza.
Aurora_in_memory_relaylog_fallback_count	Mostra il numero totale di fallback della funzionalità relativa al log di inoltro in memoria alla modalità log di inoltro persistente (legacy). Il fallback può essere causato da un singolo evento di dimensioni superiori a quelle della cache (attualmente 128 MB) o dal superamento del limite di tentativi di transazione di replica replica_transaction_retries.
Aurora_in_memory_relaylog_recovery_count	Mostra il numero totale di operazioni di ripristino del log di inoltro in memoria eseguite automaticamente. Il conteggio include il numero totale di fallback e il numero di passaggi automatici alla modalità di log di inoltro in memoria dopo i fallback temporanei.

Nome	Descrizione
`Aurora_thread_pool_thread_count`	Il numero attuale di thread nel pool di thread di Aurora. Per ulteriori informazioni sulla risposta in Aurora MySQL, consulta Pool di thread.
`Aurora_tmz_version`	Indica la versione corrente delle informazioni sul fuso orario utilizzate dal cluster database. I valori seguono il formato IANA (Internet Assigned Numbers Authority):`YYYYsuffix` , ad esempio `2022a` e `2023c`. Questo parametro si applica ad Aurora MySQL versione 2.12 e successive e versione 3.04 e successive.
`Aurora_zdr_oom_threshold`	Rappresenta la soglia di memoria, in kilobyte (KB), per un'istanza database Aurora per lanciare un riavvio senza tempi di inattività finalizzato al ripristino da potenziali problemi correlati alla memoria.
`server_aurora_das_running`	Indica se i flussi di attività del database sono abilitati o disabilitati su questa istanza database. Per ulteriori informazioni, consulta Monitoraggio di Amazon Aurora tramite i flussi di attività del database.

Le variabili di stato MySQL seguenti non si applicano ad Aurora MySQL

A causa delle differenze tra l'architettura di Aurora MySQL e quella di MySQL, alcuni parametri e variabili di stato MySQL non si applicano ad Aurora MySQL.

Le variabili di stato MySQL seguenti non si applicano ad Aurora MySQL. L'elenco non è completo.

- `innodb_buffer_pool_bytes_dirty`
- `innodb_buffer_pool_pages_dirty`
- `innodb_buffer_pool_pages_flushed`

Aurora MySQL versione 3 rimuove le seguenti variabili di stato presenti in Aurora MySQL versione 2:

- AuroraDb_lockmgr_bitmaps0_in_use
- AuroraDb_lockmgr_bitmaps1_in_use
- AuroraDb_lockmgr_bitmaps_mem_used
- AuroraDb_thread_deadlocks
- available_alter_table_log_entries
- Aurora_lockmgr_memory_used
- Aurora_missing_history_on_replica_incidents
- Aurora_new_lock_manager_lock_release_cnt
- Aurora_new_lock_manager_lock_release_total_duration_micro
- Aurora_new_lock_manager_lock_timeout_cnt
- Aurora_total_op_memory
- Aurora_total_op_temp_space
- Aurora_used_alter_table_log_entries
- Aurora_using_new_lock_manager
- Aurora_volume_bytes_allocated
- Aurora_volume_bytes_left_extent
- Aurora_volume_bytes_left_total
- Com_alter_db_upgrade
- Compression
- External_threads_connected
- Innodb_available_undo_logs
- Last_query_cost
- Last_query_partial_plans
- Slave_heartbeat_period
- Slave_last_heartbeat
- Slave_received_heartbeats
- Slave_retried_transactions
- Slave_running

- `Time_since_zero_connections`

Queste variabili dello stato MySQL sono disponibili in Aurora MySQL versione 2, ma non sono disponibili in Aurora MySQL versione 3:

- `Innodb_redo_log_enabled`
- `Innodb_undo_tablespaces_total`
- `Innodb_undo_tablespaces_implicit`
- `Innodb_undo_tablespaces_explicit`
- `Innodb_undo_tablespaces_active`

Eventi di attesa Aurora MySQL

Di seguito sono riportati alcuni tra gli eventi di attesa più frequenti di Aurora MySQL.

> **Note**
>
> Per informazioni sull'ottimizzazione delle prestazioni di Aurora MySQL utilizzando gli eventi di attesa, consulta Regolazione di Aurora MySQL con eventi di attesa.
> Per informazioni sulle convenzioni di denominazione utilizzate per gli eventi di attesa MySQL, consulta la pagina relativa alle convenzioni di denominazione per la strumentazione dello schema di prestazioni nella documentazione di MySQL.

cpu

Il numero di connessioni attive pronte per l'esecuzione è costantemente superiore al numero di vCPUs. Per ulteriori informazioni, vederecpu.

io/aurora_redo_log_flush

Una sessione sta mantenendo i dati permanenti nell'archiviazione Aurora. In genere, questo evento di attesa è per un'operazione di I/O di scrittura in Aurora MySQL. Per ulteriori informazioni, consulta io/aurora_redo_log_flush.

io/aurora_respond_to_client

L'elaborazione delle query è stata completata e i risultati vengono restituiti al client dell'applicazione per le seguenti versioni Aurora MySQL: 2.10.2 e versioni 2.10 successive, 2.09.3

e versioni 2.09 successive, 2.07.7 e versioni 2.07 successive. Confrontare la larghezza di banda di rete della classe di istanza database con la dimensione del set di risultati restituito. Inoltre, controlla i tempi di risposta lato client. Se il client non risponde e non è in grado di elaborare i pacchetti TCP, possono verificarsi perdite di pacchetti e ritrasmissioni TCP. Questa situazione influisce negativamente sulla larghezza di banda della rete. Nelle versioni inferiori a 2.10.2, 2.09.3 e 2.07.7, l'evento di attesa include erroneamente il tempo di inattività. Per informazioni su come ottimizzare il database quando questa attesa è importante, vedere io/aurora_respond_to_client.

io/file/csv/data

Dei thread scrivono su tabelle in formato CSV (Comma-Separated Value, valori separati da virgola). Controlla l'utilizzo delle tabelle CSV. Una causa tipica di questo evento è l'impostazione `log_output` in una tabella.

io/file/sql/binlog

Un thread è in attesa di un file di log binario (binlog) scritto su disco.

io/redo_log_flush

Una sessione sta mantenendo i dati permanenti nell'archiviazione Aurora. In genere, questo evento di attesa riguarda un' I/O operazione di scrittura in Aurora MySQL. Per ulteriori informazioni, consulta io/redo_log_flush.

io/socket/sql/client_connessione

Il programma `mysqld` è impegnato a creare thread per gestire le nuove connessioni client in arrivo. Per ulteriori informazioni, consulta io/socket/sql/client_connection.

io/table/sql/handler

Il motore è in attesa di accedere a una tabella. Questo evento si verifica indipendentemente dal fatto che i dati siano memorizzati nella cache nel buffer pool o se si accede su disco. Per ulteriori informazioni, consulta io/table/sql/handler.

lock/table/sql/handler

Questo evento di attesa è un gestore eventi di attesa di blocco di tabella. Per ulteriori informazioni sugli eventi "atom" e "molecule" nello schema di prestazioni, consulta eventi Atom e Molecule dello schema di prestazioni nella documentazione di MySQL.

synch/cond/innodb/row_blocca_aspetta

Le istruzioni DML (Data Manipulation Language) accedono contemporaneamente alle stesse righe del database. Per ulteriori informazioni, consulta synch/cond/innodb/row_lock_wait.

synch/cond/innodb/row_lock_wait_cond

Più istruzioni DML (Data Manipulation Language) accedono contemporaneamente alle stesse righe del database. Per ulteriori informazioni, consulta synch/cond/innodb/row_lock_wait_cond.

synch/cond/sql/MDL_contesto: :COND_WAIT_STATUS

I thread sono in attesa di un blocco dei metadati di tabella. Il motore utilizza questo tipo di blocco per gestire l'accesso simultaneo a uno schema di database e garantire la coerenza dei dati. Per ulteriori informazioni, consulta la pagina relativa all'ottimizzazione delle operazioni di blocco nella documentazione di MySQL. Per informazioni su come ottimizzare il database quando questo evento è importante, vedere synch/cond/sql/MDL_context::COND_wait_status.

synch/cond/sql/MYSQL_BIN_LOG: :cond_done

Hai attivato la registrazione binaria. Potrebbe esserci un' elevata velocità effettiva di esecuzione del commit, un numero elevato di transazioni o repliche che leggono i binlog. Prendi in considerazione l'utilizzo di istruzioni multiriga o di crezione di bundle di istruzioni in un'unica transazione. In Aurora, usa i database globali anziché la replica del log binario oppure usa i parametri `aurora_binlog_*`.

synch/mutex/innodb/aurora_lock_thread_slot_futex

Più istruzioni DML (Data Manipulation Language) accedono contemporaneamente alle stesse righe del database. Per ulteriori informazioni, consulta synch/mutex/innodb/aurora_lock_thread_slot_futex.

synch/mutex/innodb/buf_pool-mutex

Il pool di buffer non è abbastanza grande da contenere il set di dati funzionante. Oppure il carico di lavoro accede alle pagine da una tabella specifica, che porta a contendenze nel buffer pool. Per ulteriori informazioni, consulta synch/mutex/innodb/buf_pool_mutex.

synch/mutex/innodb/fil_sistema_mutex

Il processo è in attesa di accedere alla cache di memoria tablespace. Per ulteriori informazioni, consulta synch/mutex/innodb/fil_system_mutex.

synch/mutex/innodb/trx_sys_mutex

Le operazioni consistono nel controllare, aggiornare, eliminare o aggiungere transazioni IDs in InnoDB in modo coerente o controllato. Queste operazioni richiedono una chiamata mutex `trx_sys`, che viene tracciata dalla strumentazione Performance Schema. Le operazioni

includono la gestione del sistema di transazioni all'avvio o all'arresto del database, i rollback, le operazioni di cancellazione degli annullamenti, l'accesso in lettura delle righe e i carichi del pool di buffer. L'elevato carico del database con un numero elevato di transazioni comporta la frequente comparsa di questo evento di attesa. Per ulteriori informazioni, consulta synch/mutex/innodb/trx_sys_mutex.

synch/mutex/mysys/KEY_CACHE: :cache_lock

La mutex `keycache->cache_lock` controlla l'accesso alla cache delle chiavi per le tabelle myISAM. Sebbene Aurora MySQL non consenta l'utilizzo di tabelle MyISAM per archiviare dati persistenti, vengono utilizzate per archiviare tabelle temporanee interne. Valuta se controllare i conteggi dello stato `created_tmp_disk_tables` o `created_tmp_tables`, perché in determinate situazioni, le tabelle temporanee vengono scritte su disco quando non possono più essere contenute nella memoria.

synch/mutex/sql/FILE_AS_TABLE: :LOCK_offsets

Il motore acquisisce questo mutex durante l'apertura o la creazione di un file di metadati di tabella. Quando questo evento di attesa si verifica con frequenza eccessiva, il numero di tabelle create o aperte è aumentato.

synch/mutex/sql/FILE_AS_TABLE: :LOCK_SHIM_LIST

Il motore acquisisce questo mutex durante l'esecuzione di operazioni come `reset_size`,`detach_contents`, oppure `add_contents` sulla struttura interna che tiene traccia delle tabelle aperte. Il mutex sincronizza l'accesso al contenuto dell'elenco. Quando questo evento di attesa si verifica con alta frequenza, indica un cambiamento improvviso nel set di tabelle a cui si accedeva in precedenza. Il motore deve accedere a nuove tabelle o lasciare andare il contesto relativo alle tabelle precedentemente accessibili.

synch/mutex/sql/LOCK_apri

Il numero di tabelle aperte dalle sessioni supera la dimensione della cache delle definizioni di tabella o della cache aperta della tabella. Aumenta le dimensioni di queste cache. Per ulteriori informazioni, consulta la pagina relativa ad apertura e chiusura delle tabelle in MySQL.

synch/mutex/sql/LOCK_tabella_cache

Il numero di tabelle aperte dalle sessioni supera la dimensione della cache delle definizioni di tabella o della cache aperta della tabella. Aumenta le dimensioni di queste cache. Per ulteriori informazioni, consulta la pagina relativa ad apertura e chiusura delle tabelle in MySQL.

synch/mutex/sql/LOG

In questo evento di attesa, ci sono thread in attesa di un blocco di log. Ad esempio, un thread potrebbe attendere un blocco per scrivere nel file di log delle query con tempi di risposta molto lunghi.

synch/mutex/sql/MYSQL_BIN_LOG: :LOCK_COMMIT

In questo evento di attesa, c'è un thread in attesa di acquisire un blocco per eseguire il commit nel log binario. Può verificarsi un conflitto della registrazione binaria nei database con frequenza di modifica molto alta. A seconda della versione di MySQL, vengono usati alcuni blocchi per proteggere la coerenza e la durabilità del log binario. In RDS for MySQL i log binari vengono usati per la replica e il processo di backup automatico. In Aurora MySQL i log binari non sono necessari per la replica nativa o i backup. Sono disabilitati per impostazione predefinita, ma possono essere abilitati e usati per la replica esterna o l'acquisizione dei dati delle modifiche. Per ulteriori informazioni, consulta la pagina relativa al log binario nella documentazione di MySQL.

sync/mutex/sql/MYSQL_BIN_LOG: :Collezione LOCK_DUMP_THREAD_METRICS

Se la registrazione binaria è attivata, il motore acquisisce questo mutex quando stampa le metriche dei thread di dump attivi nel registro degli errori del motore e sulla mappa delle operazioni interne.

sync/mutex/sql/MYSQL_BIN_LOG: :LOCK_INACTIVE_BINLOGS_MAP

Se la registrazione binaria è attivata, il motore acquisisce questo mutex quando aggiunge, elimina o cerca l'elenco dei file binlog dietro l'ultimo.

sync/mutex/sql/MYSQL_BIN_LOG: :LOCK_IO_CACHE

Se la registrazione binaria è attivata, il motore acquisisce questo mutex durante le operazioni della cache I/O di Aurora binlog: allocazione, ridimensionamento, liberazione, scrittura, lettura, rimozione e accesso alle informazioni della cache. Se questo evento si verifica frequentemente, il motore accede alla cache in cui sono memorizzati gli eventi binlog. Per ridurre i tempi di attesa, riduci i commit. Prova a raggruppare più istruzioni in una singola transazione.

synch/mutex/sql/MYSQL_BIN_LOG: :LOCK_LOG

Hai attivato la registrazione binaria. Potrebbe esserci un'elevata velocità effettiva di esecuzione del commit, molte transazioni nell'esecuzione del commit o repliche che leggono i binlog. Prendi in considerazione l'utilizzo di istruzioni multiriga o di crezione di bundle di istruzioni in un'unica transazione. In Aurora, usa i database globali anziché la replica dei log binari o usa i parametri `aurora_binlog_*`.

synch/mutex/sql/SERVER_DISCUSSIONE: :LOCK_sync

Il `SERVER_THREAD::LOCK_sync` mutex viene acquisito durante la pianificazione, l'elaborazione o l'avvio di thread per la scrittura di file. L'eccessiva occorrenza di questo evento di attesa indica una maggiore attività di scrittura nel database.

synch/mutex/sql/TABLESPACES_THREAD: :Lock_sync ----sep----:lock

Il motore acquisisce il mutex `TABLESPACES:lock` durante le seguenti operazioni di tablespace: creare, eliminare, troncare ed estendere. L'eccessiva occorrenza di questo evento di attesa indica un'alta frequenza di operazioni di tablespace. Un esempio è il caricamento di una grande quantità di dati nel database.

synch/rwlock/innodb/dict

In questo evento di attesa, ci sono thread in attesa di un oggetto rwlock nel dizionario dei dati InnoDB.

synch/rwlock/innodb/dict_operazione_lock

In questo evento di attesa, ci sono thread che mantengono blocchi sulle operazioni del dizionario dei dati InnoDB.

synch/rwlock/innodb/dictserratura SYS RW

Un numero elevato di istruzioni simultanee del linguaggio di controllo dei dati (DCLs) nel codice del linguaggio di definizione dei dati (DDLs) viene attivato contemporaneamente. Riduci la dipendenza dell'applicazione DDLs durante la normale attività dell'applicazione.

synch/rwlock/innodb/index_tree_rw_lock

Un gran numero di istruzioni DML (Data Manipolation Language) simili stanno accedendo allo stesso oggetto di database contemporaneamente. Prova a usare istruzioni multiriga. Inoltre, distribuisci il carico di lavoro su diversi oggetti di database. Ad esempio, implementa il partizionamento.

synch/sxlock/innodb/dict_operation_lock

Viene attivato contemporaneamente un numero elevato di istruzioni del linguaggio di controllo dei dati (DCLs) nel codice del linguaggio di definizione dei dati (DDLs). Riduci la dipendenza dell'applicazione DDLs durante la normale attività dell'applicazione.

synch/sxlock/innodb/dict_sys_lock

Viene attivato contemporaneamente un numero elevato di istruzioni del linguaggio di controllo dei dati (DCLs) nel codice del linguaggio di definizione dei dati (DDLs). Riduci la dipendenza dell'applicazione DDLs durante la normale attività dell'applicazione.

synch/sxlock/innodb/hash_table_locks

La sessione non è in grado di trovare pagine nel pool di buffer. Il motore deve leggere un file o modificare l'elenco LRU (LRU) utilizzato meno di recente per il buffer pool. Considera di aumentare le dimensioni della cache del buffer e migliorare i percorsi di accesso per le query pertinenti.

synch/sxlock/innodb/index_albero_rw_lock

Molte istruzioni DML (Data Manipulation Language) simili accedono allo stesso oggetto di database contemporaneamente. Prova a usare istruzioni multiriga. Inoltre, distribuisci il carico di lavoro su diversi oggetti di database. Ad esempio, implementa il partizionamento.

synch/mutex/innodb/temp_pool_manager_mutex

Questo evento di attesa si verifica quando una sessione è in attesa di acquisire un mutex per la gestione del pool di tablespace temporanee della sessione.

Per ulteriori informazioni sulla risoluzione dei problemi di sincronizzazione degli eventi di attesa, consulta la sezione relativa al motivo per cui un'istanza database MySQL mostra un numero elevato di sessioni attive in attesa di eventi di attesa SYNCH in Approfondimenti sulle prestazioni.

Stati del thread Aurora MySQL

Di seguito sono riportati alcuni tra gli stati di thread più frequenti di Aurora MySQL.

Verifica delle autorizzazioni

Il thread sta verificando se il server dispone dei privilegi necessari per eseguire l'istruzione.

controllo della cache delle query per la query

Il server sta verificando se la query corrente è presente nella cache delle query.

ripulito

Questo è lo stato finale di una connessione il cui lavoro è completo ma che non è stato chiuso dal client. La soluzione migliore è chiudere esplicitamente la connessione nel codice. Oppure puoi impostare un valore inferiore per `wait_timeout` nel gruppo di parametri.

chiusura tabelle

Il thread sta scaricando i dati della tabella modificati su disco e chiude le tabelle utilizzate. Se non si tratta di un'operazione rapida, verificare le metriche di consumo della larghezza di banda di rete rispetto alla larghezza di banda di rete della classe di istanza. Inoltre, verificare che i valori dei parametri per `table_open_cachee table_definition_cache` il parametro consentano di aprire contemporaneamente una quantità sufficiente di tabelle in modo che il motore non debba aprire e chiudere frequentemente le tabelle. Questi parametri influenzano il consumo di memoria sull'istanza.

conversione da HEAP a myISAM

La query sta convertendo una tabella temporanea da memoria a su disco. Questa conversione è necessaria perché le tabelle temporanee create da MySQL nelle fasi intermedie dell'elaborazione delle query sono diventate troppo grandi per la memoria. Verifica i valori di `tmp_table_size` e `max_heap_table_size`. Nelle versioni successive, il nome dello stato del thread è `converting HEAP to ondisk`.

conversione di HEAP in ondisk

Il thread sta convertendo una tabella temporanea interna da una tabella in memoria a una tabella su disco.

copia nella tabella tmp

Il thread sta elaborando un'istruzione `ALTER TABLE`. Questo stato si verifica dopo che la tabella con la nuova struttura è stata creata ma prima che le righe vengano copiate. Per un thread in questo stato, è possibile utilizzare lo schema delle prestazioni per ottenere informazioni sullo stato di avanzamento dell'operazione di copia.

creazione di indice di ordinamento

Aurora MySQL sta eseguendo un ordinamento perché non può utilizzare un indice esistente per soddisfare la clausola `ORDER BY` o `GROUP BY` di una query. Per ulteriori informazioni, consulta creazione di indice di ordinamento.

Creazione di tabelle

Il thread sta creando una tabella permanente o temporanea.

commit ritardato ok fatto

Un commit asincrono in Aurora MySQL ha ricevuto un riconoscimento ed è completo.

ritardato commit ok avviato

Il thread Aurora MySQL ha avviato il processo di commit asincrono ma è in attesa di un riconoscimento. Di solito questo è il tempo di commit autentico di una transazione.

invio ritardato ok fatto

Un thread di lavoro Aurora MySQL collegato a una connessione può essere liberato mentre viene inviata una risposta al client. Il thread può iniziare altri lavori. Lo stato `delayed send ok` significa che la conferma asincrona al client è stata completata.

invio ritardato ok avviato

Un thread di lavoro Aurora MySQL ha inviato una risposta asincrona a un client ed è ora libero di lavorare per altre connessioni. La transazione ha avviato un processo di commit asincrono che non è ancora stato riconosciuto.

executing

Il thread ha iniziato a eseguire un'istruzione.

liberazione elementi

Il thread ha eseguito un comando. Alcuni liberazioni degli elementi eseguiti durante questo stato coinvolgono la cache delle query. Questo stato è solitamente seguito dalla pulizia.

init

Questo stato si verifica prima dell'inizializzazione di `ALTER TABLE`,`DELETE`,`INSERT`,`SELECT`, oppure istruzioni `UPDATE`. Le azioni in questo stato includono lo svuotamento del log binario o del log InnoDB e alcune operazioni di pulizia della cache delle query.

Source has sent all binlog to replica; waiting for more updates

Il nodo primario ha terminato la sua parte della replica. Il thread è in attesa di eseguire altre query in modo che possa scrivere nel log binario (binlog).

apertura tabella

Il thread sta tentando di aprire una tabella. Questa operazione è veloce a meno che un'istruzione `ALTER TABLE` o `LOCK TABLE` debba finire o superare il valore di `table_open_cache`.

Ottimizzazione

Il server sta eseguendo le ottimizzazioni iniziali per una query.

preparazione

Questo stato si verifica durante l'ottimizzazione delle query.

fine query

Questo stato si verifica dopo l'elaborazione di una query ma prima dello stato di liberazione degli elementi.

rimozione di duplicati

Aurora MySQL non è riuscita a ottimizzare un'operazione `DISTINCT` nella fase iniziale di una query. Aurora MySQL deve rimuovere tutte le righe duplicate prima di inviare il risultato al client.

ricerca di righe per l'aggiornamento

Il thread sta trovando tutte le righe corrispondenti prima di aggiornarle. Questa fase è necessaria se il `UPDATE` sta cambiando l'indice utilizzato dal motore per trovare le righe.

invio dell'evento binlog a slave

Il thread ha letto un evento dal log binario e lo sta inviando alla replica.

invio di risultati memorizzati nella cache al client

Il server sta prendendo il risultato di una query dalla cache delle query e la invia al client.

Invio dei dati

Il thread sta leggendo ed elaborando le righe per un'istruzione `SELECT` ma non ha ancora iniziato a inviare dati al client. Il processo sta identificando quali pagine contengono i risultati necessari per soddisfare la query. Per ulteriori informazioni, consulta invio dei dati.

invio al client

Il server sta scrivendo un pacchetto sul client. Nelle versioni precedenti di MySQL, questo evento di attesa è stato etichettato `writing to net`.

avvio in corso

Questa è la prima fase all'inizio dell'esecuzione dell'istruzione.

statistiche

Il server sta calcolando le statistiche per sviluppare un piano di esecuzione delle query. Se un thread è in questo stato per un lungo periodo, il server è probabilmente associato a disco durante l'esecuzione di altri lavori.

archiviazione dei risultati nella cache delle query

Il server sta memorizzando il risultato di una query nella cache delle query.

blocco del sistema

Il thread ha chiamato `mysql_lock_tables`, ma lo stato del thread non è stato aggiornato dalla chiamata. Questo stato generale si verifica per molte ragioni.

aggiorna

Il thread si sta preparando per iniziare ad aggiornare la tabella.

aggiornamento

Il thread sta cercando righe e le sta aggiornando.

blocco utente

Il thread ha emesso una chiamata `GET_LOCK`. Il thread ha richiesto un blocco consultivo e lo sta aspettando o sta pianificando di richiederlo.

in attesa di ulteriori aggiornamenti

Il nodo primario ha terminato la sua parte della replica. Il thread è in attesa di eseguire altre query in modo che possa scrivere nel log binario (binlog).

in attesa del blocco dei metadati dello schema

Questa è un'attesa per un blocco dei metadati.

in attesa del blocco dei metadati della funzione memorizzata

Questa è un'attesa per un blocco dei metadati.

in attesa del blocco dei metadati della procedura archiviata

Questa è un'attesa per un blocco dei metadati.

in attesa dello scarico della tabella

Il thread sta eseguendo `FLUSH TABLES` e sta aspettando che tutti i thread chiudano le loro tabelle. Oppure il thread ha ricevuto una notifica che la struttura sottostante per una tabella è cambiata, quindi deve riaprire la tabella per ottenere la nuova struttura. Per riaprire la tabella, il

thread deve attendere che tutti gli altri thread abbiano chiuso la tabella. Questa notifica avviene se un altro thread ha utilizzato una delle seguenti istruzioni sulla tabella: `FLUSH TABLES`,`ALTER TABLE`,`RENAME TABLE`,`REPAIR TABLE`,`ANALYZE TABLE`, oppure `OPTIMIZE TABLE`.

in attesa di blocco a livello tabella

Una sessione tiene un blocco su un tavolo mentre un'altra sessione tenta di acquisire lo stesso blocco sulla stessa tabella.

in attesa del blocco dei metadati della tabella

Aurora MySQL utilizza il blocco dei metadati per gestire l'accesso simultaneo agli oggetti del database e garantire la coerenza dei dati. In questo evento di attesa, una sessione tiene un blocco di metadati su una tabella mentre un'altra sessione tenta di acquisire lo stesso blocco sulla stessa tabella. Quando lo schema delle prestazioni è abilitato, questo stato del thread viene segnalato come evento di attesa `synch/cond/sql/MDL_context::COND_wait_status`.

scrittura su rete

Il server sta scrivendo un pacchetto sulla rete. Nelle versioni successive di MySQL, questo evento di attesa è etichettato `Sending to client`.

Livelli di isolamento di Aurora MySQL

Scopri come le istanze database in un cluster Aurora MySQL implementano la proprietà di isolamento del database. In questo argomento viene spiegato come il comportamento predefinito di Aurora MySQL si bilancia tra coerenza rigorosa e prestazioni elevate. Puoi utilizzare queste informazioni per decidere quando modificare le impostazioni predefinite in base alle caratteristiche del tuo carico di lavoro.

Livelli di isolamento disponibili per le istanze di scrittura

È possibile utilizzare i livelli di isolamento `REPEATABLE READ`, `READ COMMITTED`, `READ UNCOMMITTED` e `SERIALIZABLE` sull'istanza primaria di un cluster di database Aurora MySQL. Questi livelli di isolamento funzionano allo stesso modo in Aurora MySQL e in RDS for MySQL.

Livello di isolamento REPEATABLE READ per le istanze di lettura

Per impostazione predefinita, le istanze database Aurora MySQL configurate come repliche di Aurora di sola lettura utilizzano sempre il livello di isolamento `REPEATABLE READ`. Queste istanze database ignorano qualsiasi istruzione `SET TRANSACTION ISOLATION LEVEL` e continuano a usare il livello di isolamento `REPEATABLE READ`.

Livello di isolamento READ COMMITTED per le istanze di lettura

Se l'applicazione include un carico di lavoro ad alta intensità di scrittura nell'istanza primaria e query a esecuzione prolungata nelle repliche di Aurora, è possibile che si verifichi un notevole ritardo di eliminazione. Un ritardo di eliminazione si verifica quando la garbage collection interna è bloccata da query a esecuzione prolungata. Il sintomo è un valore elevato per `history list length` nell'output del comando `SHOW ENGINE INNODB STATUS`. È possibile monitorare questo valore utilizzando il parametro `RollbackSegmentHistoryListLength` in CloudWatch. Un notevole ritardo dell'eliminazione può ridurre l'efficacia degli indici secondari, diminuire le prestazioni complessive delle query e usare inutilmente lo spazio di archiviazione.

Se si verificano questi problemi, puoi usare l'impostazione di configurazione a livello di sessione Aurora MySQL, `aurora_read_replica_read_committed`, per utilizzare il livello di isolamento `READ COMMITTED` nelle repliche di Aurora. L'applicazione di questa impostazione può aiutare a ridurre i rallentamenti e lo spazio usato inutilmente che possono derivare dall'esecuzione di query a esecuzione prolungata contemporaneamente alle transazioni che modificano le tabelle.

Prima di usare questa impostazione, ti consigliamo di comprendere il funzionamento di Aurora MySQL specifico dell'isolamento `READ COMMITTED`. Il comportamento `READ COMMITTED` della replica di Aurora è conforme allo standard ANSI SQL. Tuttavia, l'isolamento è meno restrittivo del comportamento `READ COMMITTED` tipico di MySQL che potresti già conoscere. Pertanto, potresti vedere risultati diversi per la query in `READ COMMITTED` su una replica di lettura di Aurora MySQL rispetto alla stessa query in `READ COMMITTED` nell'istanza primaria Aurora MySQL o in RDS per MySQL. Potresti utilizzare l'impostazione `aurora_read_replica_read_committed` per questi casi d'uso, come un report completo che esegue la scansione di un database molto grande. Potresti invece evitarla per brevi query con piccoli set di risultati, dove la precisione e la ripetibilità sono importanti.

Il livello di isolamento `READ COMMITTED` non è disponibile per le sessioni all'interno di un cluster secondario in un Aurora Global Database che utilizzano la funzionalità di inoltro di scrittura. Per informazioni sull'inoltro di scrittura, consulta Utilizzo dell'inoltro di scrittura in un database globale Amazon Aurora.

Utilizzo di READ COMMITTED per le letture

Per usare il livello di isolamento `READ COMMITTED` per le repliche di Aurora, configura l'impostazione `aurora_read_replica_read_committed` su `ON`. Usa questa impostazione a livello di sessione mentre è connessa a una replica di Aurora specifica. Per farlo, esegui i comandi SQL seguenti.

```
set session aurora_read_replica_read_committed = ON;
set session transaction isolation level read committed;
```

Puoi usare temporaneamente questa impostazione di configurazione per eseguire query interattive una tantum. Potresti anche voler eseguire un'applicazione per la creazione di report o l'analisi dei dati che beneficia del livello di isolamento `READ COMMITTED`, lasciando invariata l'impostazione predefinita per altre applicazioni.

Quando l'impostazione `aurora_read_replica_read_committed` è attivata, usa il comando `SET TRANSACTION ISOLATION LEVEL` per specificare il livello di isolamento per le transazioni appropriate.

```
set transaction isolation level read committed;
```

Differenze nel comportamento di READ COMMITTED nelle repliche di Aurora

L'impostazione `aurora_read_replica_read_committed` rende il livello di isolamento `READ COMMITTED` disponibile per una replica di Aurora con un comportamento coerente che viene ottimizzato per le transazioni a esecuzione prolungata. Il livello di isolamento `READ COMMITTED` nelle repliche di Aurora ha un isolamento meno restrittivo di quello nelle istanze primarie di Aurora. Per questo motivo, abilita l'impostazione solo in repliche di Aurora in cui sai che le query possono accettare la possibilità di determinati tipi di risultati incoerenti.

Le tue query possono riscontrare alcuni tipi di anomalie di lettura quando l'impostazione `aurora_read_replica_read_committed` è attivata. Due tipi di anomalie sono particolarmente importanti per comprendere e gestire il codice dell'applicazione. Una lettura non ripetibile si verifica quando viene eseguita un'altra transazione durante l'esecuzione della query. Una query a esecuzione prolungata può visualizzare dati diversi all'inizio della query rispetto a quelli visualizzati alla fine. Una lettura fantasma si verifica quando altre transazioni causano la riorganizzazione delle righe esistenti mentre la query è in esecuzione e una o più righe vengono lette due volte dalla query.

Le query potrebbero presentare conteggi di riga incoerenti a seguito delle letture fantasma. Le tue query potrebbero anche restituire risultati incompleti o incoerenti a causa delle letture non ripetibili. Ad esempio, supponiamo che un'operazione di join si riferisca alle tabelle che sono modificate contemporaneamente dalle istruzioni SQL come `INSERT` o `DELETE`. In questo caso, la query di join potrebbe leggere una riga da una tabella ma non la riga corrispondente da un'altra tabella.

Lo standard ANSI SQL consente entrambi questi comportamenti per il livello di isolamento `READ COMMITTED`. Tuttavia, questi comportamenti sono diversi dalla tipica

implementazione di MySQL di `READ COMMITTED`. Pertanto, prima di abilitare l'impostazione `aurora_read_replica_read_committed`, controlla qualsiasi codice SQL esistente per verificare se funziona come previsto nel modello di coerenza più flessibile.

Il conteggio delle righe e altri risultati potrebbero non essere propriamente coerenti nel livello di isolamento `READ COMMITTED` mentre questa impostazione è abilitata. Pertanto, in genere abiliti l'impostazione solo durante l'esecuzione di query analitiche che aggregano grandi quantità di dati e non richiedono la precisione assoluta. Se non hai questo tipo di query a esecuzione prolungata insieme a un carico di lavoro ad alta intensità di scrittura, probabilmente non è necessaria l'impostazione `aurora_read_replica_read_committed`. Senza la combinazione di query a esecuzione prolungata e carico di lavoro ad alta intensità di scrittura, è improbabile che si verifichino problemi con la lunghezza dell'elenco della cronologia.

Example Query che mostrano il comportamento di isolamento per READ COMMITTED nelle repliche di Aurora

L'esempio seguente mostra come le query `READ COMMITTED` in una replica di Aurora potrebbero restituire risultati non ripetibili se le transazioni modificano contemporaneamente le tabelle associate. La tabella `BIG_TABLE` contiene 1 milione di righe prima dell'inizio di qualsiasi query. Altre istruzioni DML (Data Manipulation Language) aggiungono, rimuovono o modificano le righe mentre sono in esecuzione.

Le query nell'istanza primaria Aurora nel livello di isolamento `READ COMMITTED` producono risultati prevedibili. Tuttavia, il sovraccarico di mantenere la visualizzazione di lettura coerente per la durata di ogni query a esecuzione prolungata può portare a costose operazioni di garbage collection in un secondo momento.

Le query nella replica di Aurora nel livello di isolamento `READ COMMITTED` sono ottimizzate per ridurre al minimo questo sovraccarico di garbage collection. Il compromesso sta nei risultati che possono variare a seconda che le query recuperino o meno le righe aggiunte, rimosse o riorganizzate dalle transazioni che eseguono il commit mentre la query è in esecuzione. Le query possono prendere in considerazione queste righe, ma non sono obbligate a farlo. A scopo dimostrativo, le query controllano solo il numero di righe nella tabella utilizzando la funzione `COUNT(*)`.

Orario	Istruzione DML nell'istanza primaria Aurora	Query nell'istanza primaria Aurora con READ COMMITTED	Query nella replica di Aurora con READ COMMITTED
T1	`INSERT INTO big_table SELECT * FROM other_table LIMIT 1000000; COMMIT;`		
T2		Q1: `SELECT COUNT(*) FROM big_table;`	Q2: `SELECT COUNT(*) FROM big_table;`
T3	`INSERT INTO big_table (c1, c2) VALUES (1, 'one more row'); COMMIT;`		
T4		Se Q1 finisce ora, il risultato è 1.000.000.	Se Q2 finisce ora, il risultato è 1.000.000 o 1.000.001.
T5	`DELETE FROM big_table LIMIT 2; COMMIT;`		
T6		Se Q1 finisce ora, il risultato è 1.000.000.	Se Q2 finisce ora, il risultato è 1.000.000 o 1.000.001 o 999.999 o 999.998.
T7	`UPDATE big_table SET c2 = CONCAT(c2`		

Orario	Istruzione DML nell'istanza primaria Aurora	Query nell'istanza primaria Aurora con READ COMMITTED	Query nella replica di Aurora con READ COMMITTED
	`,c2,c2); COMMIT;`		
T8		Se Q1 finisce ora, il risultato è 1.000.000.	Se Q2 finisce ora, il risultato è 1.000.000 o 1.000.001 o 999.999 oppure possibilmente un numero più alto.
T9		Q3: `SELECT COUNT(*) FROM big_table;`	Q4: `SELECT COUNT(*) FROM big_table;`
T10		Se Q3 finisce ora, il risultato è 999.999.	Se Q4 finisce ora, il risultato è 999.999.
T11		Q5: `SELECT COUNT(*) FROM parent_table p JOIN child_table c ON (p.id = c.id) WHERE p.id = 1000;`	Q6: `SELECT COUNT(*) FROM parent_table p JOIN child_table c ON (p.id = c.id) WHERE p.id = 1000;`
T12	`INSERT INTO parent_table (id, s) VALUES (1000, 'hello'); INSERT INTO child_table (id, s) VALUES (1000, 'world'); COMMIT;`		

Orario	Istruzione DML nell'istanza primaria Aurora	Query nell'istanza primaria Aurora con READ COMMITTED	Query nella replica di Aurora con READ COMMITTED
T13		Se Q5 finisce ora, il risultato è 0.	Se Q6 finisce ora, il risultato è 0 o 1.

Se le query finiscono rapidamente, prima che qualsiasi altra transazione esegua istruzioni e commit DML, i risultati sono prevedibili e uguali tra l'istanza primaria e la replica di Aurora. Esaminiamo in dettaglio le differenze di comportamento, a partire dalla prima query.

I risultati di Q1 sono prevedibili perché `READ COMMITTED` nell'istanza primaria usa un modello di consistenza elevata simile al livello di isolamento `REPEATABLE READ`.

I risultati per Q2 possono variare a seconda delle transazioni sottoposte al commit durante l'esecuzione della query. Ad esempio, supponiamo che altre transazioni eseguano istruzioni DML ed eseguano il commit mentre le query sono in esecuzione. In questo caso, la query nella replica di Aurora con il livello di isolamento `READ COMMITTED` potrebbe o meno prendere in considerazione le modifiche. I conteggi delle righe non sono prevedibili come nel livello di isolamento `REPEATABLE READ`. Inoltre, non sono prevedibili come le query in esecuzione sotto il livello di isolamento `READ COMMITTED` nell'istanza primaria o in un'istanza RDS for MySQL.

L'istruzione `UPDATE` a T7 in effetti non cambia il numero di righe della tabella. Tuttavia, modificando la lunghezza di una colonna di lunghezza variabile, questa istruzione può causare la riorganizzazione interna delle righe. Una transazione `READ COMMITTED` a esecuzione prolungata potrebbe visualizzare la versione precedente di una riga e, successivamente, all'interno della stessa query, visualizzare la nuova versione della stessa riga. La query può anche ignorare sia la versione precedente che quella nuova della riga, quindi il conteggio delle righe potrebbe essere diverso da quello previsto.

I risultati di Q5 e Q6 potrebbero essere identici o leggermente diversi. La query Q6 nella replica di Aurora in `READ COMMITTED` è in grado di vedere, ma non è obbligata a vedere, le nuove righe che vengono confermate mentre la query è in esecuzione. Potrebbe anche vedere la riga di una tabella, ma non dell'altra tabella. Se la query di join non trova una riga corrispondente in entrambe le tabelle, restituisce un conteggio pari a zero. Se la query trova entrambe le nuove righe in `PARENT_TABLE` e `CHILD_TABLE`, la query restituisce un conteggio pari a uno. In una query a esecuzione prolungata, le ricerche dalle tabelle unite potrebbero avvenire in momenti ampiamente separati.

> **Note**
>
> Queste differenze di comportamento dipendono dal momento in cui vengono eseguite le transazioni e vengono elaborate dalle query le righe della tabella sottostante. Pertanto, è molto probabile che si notino tali differenze nelle query di report che richiedono minuti o ore e che vengono eseguite in cluster Aurora che elaborano contemporaneamente transazioni OLTP. Questi sono i tipi di carichi di lavoro misti che beneficiano maggiormente del livello di isolamento `READ COMMITTED` nelle repliche di Aurora.

Suggerimenti di Aurora MySQL

È possibile utilizzare i suggerimenti SQL con le query Aurora MySQL per ottimizzare le prestazioni. È inoltre possibile utilizzare i suggerimenti per impedire che i piani di esecuzione per query importanti vengano modificati a causa di condizioni imprevedibili.

> **Tip**
>
> Per verificare l'effetto di un suggerimento su una query, esaminare il piano di query prodotto dall'istruzione `EXPLAIN`. Confrontare i piani di query con e senza il suggerimento.

In Aurora MySQL versione 3, puoi usare tutti i suggerimenti disponibili in MySQL Community Edition 8.0. Per ulteriori informazioni su questi suggerimenti, consulta Suggerimenti di ottimizzazione nel Manuale di riferimento MySQL.

Questi suggerimenti sono disponibili in Aurora MySQL versione 2. Questi suggerimenti si applicano alle query che utilizzano la funzionalità hash join in Aurora MySQL versione 2, in particolare alle query che utilizzano l'ottimizzazione delle query parallele.

PQ, NO_PQ

Specifica se forzare l'ottimizzatore a utilizzare una query parallela per tabella o per query.

`PQ` forza l'ottimizzazione a utilizzare la query parallela per le tabelle specificate o per l'intera query (blocco). `NO_PQ` impedisce all'ottimizzazione di utilizzare la query parallela per le tabelle specificate o per l'intera query (blocco).

Questo suggerimento è disponibile in Aurora MySQL versione 2.11 e successive. Gli esempi seguenti mostrano come utilizzare questo suggerimento.

> **Note**
>
> La specificazione del nome di una tabella impone all'ottimizzazione di applicare il suggerimento PQ/NO_PQ solo su quelle tabelle selezionate. La mancata specificazione del nome di una tabella impone il suggerimento PQ/NO_PQ su tutte le tabelle interessate dal blocco di query.

```
EXPLAIN SELECT /*+ PQ() */ f1, f2
    FROM num1 t1 WHERE f1 > 10 and f2 < 100;

EXPLAIN SELECT /*+ PQ(t1) */ f1, f2
    FROM num1 t1 WHERE f1 > 10 and f2 < 100;

EXPLAIN SELECT /*+ PQ(t1,t2) */ f1, f2
    FROM num1 t1, num1 t2 WHERE t1.f1 = t2.f21;

EXPLAIN SELECT /*+ NO_PQ() */ f1, f2
    FROM num1 t1 WHERE f1 > 10 and f2 < 100;

EXPLAIN SELECT /*+ NO_PQ(t1) */ f1, f2
    FROM num1 t1 WHERE f1 > 10 and f2 < 100;

EXPLAIN SELECT /*+ NO_PQ(t1,t2) */ f1, f2
    FROM num1 t1, num1 t2 WHERE t1.f1 = t2.f21;
```

HASH_JOIN, NO_HASH_JOIN

Attiva o disattiva la possibilità dell'ottimizzazione di query parallele di scegliere se utilizzare il metodo di ottimizzazione hash join per una query. HASH_JOIN consente all'ottimizzazione di utilizzare hash join se tale meccanismo è più efficiente. NO_HASH_JOIN impedisce all'ottimizzazione di utilizzare hash join per la query. Questo suggerimento è disponibile in Aurora MySQL versione 2.08 e successive. Non ha alcun effetto in Aurora MySQL versione 3.

Gli esempi seguenti mostrano come utilizzare questo suggerimento.

```
EXPLAIN SELECT/*+ HASH_JOIN(t2) */ f1, f2
  FROM t1, t2 WHERE t1.f1 = t2.f1;

EXPLAIN SELECT /*+ NO_HASH_JOIN(t2) */ f1, f2
```

```
  FROM t1, t2 WHERE t1.f1 = t2.f1;
```

HASH_JOIN_PROBING, NO_HASH_JOIN_PROBING

In una query hash join, indica se utilizzare la tabella specificata per il lato probe del join. La query verifica se i valori delle colonne della tabella di generazione esistono nella tabella probe, anziché leggerne l'intero contenuto. È possibile utilizzare `HASH_JOIN_PROBING` e `HASH_JOIN_BUILDING` per specificare la modalità di elaborazione delle query di join hash senza riordinare le tabelle all'interno del testo della query. Questo suggerimento è disponibile in Aurora MySQL versione 2.08 e successive. Non ha alcun effetto in Aurora MySQL versione 3.

Gli esempi seguenti mostrano come utilizzare questo suggerimento. Specificare il suggerimento `HASH_JOIN_PROBING` per la tabella `T2` ha lo stesso effetto di specificare `NO_HASH_JOIN_PROBING` per la tabella `T1`.

```
EXPLAIN SELECT /*+ HASH_JOIN(t2) HASH_JOIN_PROBING(t2) */ f1, f2
  FROM t1, t2 WHERE t1.f1 = t2.f1;

EXPLAIN SELECT /*+ HASH_JOIN(t2) NO_HASH_JOIN_PROBING(t1) */ f1, f2
  FROM t1, t2 WHERE t1.f1 = t2.f1;
```

HASH_JOIN_BUILDING, NO_HASH_JOIN_BUILDING

In una query hash join, indica se utilizzare la tabella specificata per il lato build del join. La query elabora tutte le righe di questa tabella per creare l'elenco dei valori di colonna per un riferimento incrociato con l'altra tabella. È possibile utilizzare `HASH_JOIN_PROBING` e `HASH_JOIN_BUILDING` per specificare la modalità di elaborazione delle query di join hash senza riordinare le tabelle all'interno del testo della query. Questo suggerimento è disponibile in Aurora MySQL versione 2.08 e successive. Non ha alcun effetto in Aurora MySQL versione 3.

L'esempio seguente mostra come utilizzare questo suggerimento. Specificare il suggerimento `HASH_JOIN_BUILDING` per la tabella `T2` ha lo stesso effetto di specificare `NO_HASH_JOIN_BUILDING` per la tabella `T1`.

```
EXPLAIN SELECT /*+ HASH_JOIN(t2) HASH_JOIN_BUILDING(t2) */ f1, f2
  FROM t1, t2 WHERE t1.f1 = t2.f1;

EXPLAIN SELECT /*+ HASH_JOIN(t2) NO_HASH_JOIN_BUILDING(t1) */ f1, f2
  FROM t1, t2 WHERE t1.f1 = t2.f1;
```

JOIN_FIXED_ORDER

Specifica che le tabelle della query vengono unite in base all'ordine in cui sono elencate nella query. È utile con le query che coinvolgono almeno tre tabelle. È inteso come un sostituto per il suggerimento MySQL `STRAIGHT_JOIN` ed è equivalente al suggerimento MySQL JOIN_FIXED_ORDER. Questo suggerimento è disponibile in Aurora MySQL versione 2.08 e successive.

L'esempio seguente mostra come utilizzare questo suggerimento.

```
EXPLAIN SELECT /*+ JOIN_FIXED_ORDER() */ f1, f2
  FROM t1 JOIN t2 USING (id) JOIN t3 USING (id) JOIN t4 USING (id);
```

JOIN_ORDER

Specifica l'ordine join per le tabelle nella query. È utile con le query che coinvolgono almeno tre tabelle. È equivalente al suggerimento MySQL JOIN_ORDER. Questo suggerimento è disponibile in Aurora MySQL versione 2.08 e successive.

L'esempio seguente mostra come utilizzare questo suggerimento.

```
EXPLAIN SELECT /*+ JOIN_ORDER (t4, t2, t1, t3) */ f1, f2
  FROM t1 JOIN t2 USING (id) JOIN t3 USING (id) JOIN t4 USING (id);
```

JOIN_PREFIX

Specifica le tabelle da inserire per prime nell'ordine join. È utile con le query che coinvolgono almeno tre tabelle. È equivalente al suggerimento MySQL JOIN_PREFIX. Questo suggerimento è disponibile in Aurora MySQL versione 2.08 e successive.

L'esempio seguente mostra come utilizzare questo suggerimento.

```
EXPLAIN SELECT /*+ JOIN_PREFIX (t4, t2) */ f1, f2
  FROM t1 JOIN t2 USING (id) JOIN t3 USING (id) JOIN t4 USING (id);
```

JOIN_SUFFIX

Specifica le tabelle da inserire per ultime nell'ordine join. È utile con le query che coinvolgono almeno tre tabelle. È equivalente al suggerimento MySQL JOIN_SUFFIX. Questo suggerimento è disponibile in Aurora MySQL versione 2.08 e successive.

L'esempio seguente mostra come utilizzare questo suggerimento.

```
EXPLAIN SELECT /*+ JOIN_SUFFIX (t1) */ f1, f2
  FROM t1 JOIN t2 USING (id) JOIN t3 USING (id) JOIN t4 USING (id);
```

Per informazioni sull'utilizzo delle query di hash join, vedere Ottimizzazione di grandi query di join Aurora MySQL con hash join.

Informazioni di riferimento sulle stored procedure Aurora MySQL

Puoi gestire il cluster di database Aurora MySQL tramite la chiamata delle stored procedure integrate.

Argomenti

- Raccolta e gestione della cronologia dello stato globale
- Configurazione, avvio e arresto della replica dei log binari (binlog)
- Terminare una sessione o una query
- Replica delle transazioni tramite GTID
- Rotazione dei log di query
- Impostazione e visualizzazione della configurazione dei log binari

Raccolta e gestione della cronologia dello stato globale

Amazon RDS fornisce una serie di procedure che acquisiscono uno snapshot dei valori di queste variabili di stato nel tempo e li scrivono in una tabella insieme alle modifiche eseguite dopo l'ultimo snapshot. Questa infrastruttura si chiama cronologia di stato globale. Per ulteriori informazioni, consulta Gestione della cronologia di stato globale.

Le seguenti stored procedure gestiscono il modo in cui la cronologia di stato globale viene raccolta e gestita.

Argomenti

- mysql.rds_collect_global_status_history
- mysql.rds_disable_gsh_collector
- mysql.rds_disable_gsh_rotation
- mysql.rds_enable_gsh_collector
- mysql.rds_enable_gsh_rotation
- mysql.rds_rotate_global_status_history
- mysql.rds_set_gsh_collector
- mysql.rds_set_gsh_rotation

mysql.rds_collect_global_status_history

Acquisisce uno snapshot on demand per la cronologia di stato globale.

Sintassi

```
CALL mysql.rds_collect_global_status_history;
```

mysql.rds_disable_gsh_collector

Disabilita gli snapshot creati mediante la cronologia di stato globale.

Sintassi

```
CALL mysql.rds_disable_gsh_collector;
```

mysql.rds_disable_gsh_rotation

Disattiva la rotazione della tabella `mysql.global_status_history`.

Sintassi

```
CALL mysql.rds_disable_gsh_rotation;
```

mysql.rds_enable_gsh_collector

Abilita la cronologia di stato globale per acquisire snapshot predefiniti agli intervalli specificati da `rds_set_gsh_collector`.

Sintassi

```
CALL mysql.rds_enable_gsh_collector;
```

mysql.rds_enable_gsh_rotation

Attiva la rotazione dei contenuti della tabella `mysql.global_status_history` su `mysql.global_status_history_old` agli intervalli specificati da `rds_set_gsh_rotation`.

Sintassi

```
CALL mysql.rds_enable_gsh_rotation;
```

mysql.rds_rotate_global_status_history

Ruota i contenuti della tabella `mysql.global_status_history` su `mysql.global_status_history_old` a richiesta.

Sintassi

```
CALL mysql.rds_rotate_global_status_history;
```

mysql.rds_set_gsh_collector

Specifica l'intervallo, espresso in minuti, tra gli snapshot acquisiti mediante la cronologia di stato globale.

Sintassi

```
CALL mysql.rds_set_gsh_collector(intervalPeriod);
```

Parametri

intervalPeriod

L'intervallo, in minuti, tra gli snapshot. Il valore predefinito è 5.

mysql.rds_set_gsh_rotation

Specifica l'intervallo, in giorni, tra le conversioni della tabella `mysql.global_status_history`.

Sintassi

```
CALL mysql.rds_set_gsh_rotation(intervalPeriod);
```

Parametri

intervalPeriod

L'intervallo, in giorni, tra le conversioni delle tabelle. Il valore predefinito è 7.

Configurazione, avvio e arresto della replica dei log binari (binlog)

Puoi chiamare le seguenti procedure archiviate mentre sei connesso all'istanza primaria in un cluster Aurora MySQL. Queste procedure controllano il modo in cui le transazioni vengono replicate da un database esterno a Aurora MySQL o da Aurora MySQL a un database esterno.

Argomenti

- mysql.rds_disable_session_binlog (Aurora MySQL versione 2)
- mysql.rds_enable_session_binlog (Aurora MySQL versione 2)
- mysql.rds_import_binlog_ssl_material
- mysql.rds_next_master_log (Aurora MySQL versione 2)
- mysql.rds_next_source_log (Aurora MySQL versione 3)
- mysql.rds_remove_binlog_ssl_material
- mysql.rds_reset_external_master (Aurora MySQL versione 2)
- mysql.rds_reset_external_source (Aurora MySQL versione 3)
- mysql.rds_set_binlog_source_ssl (Aurora MySQL versione 3)
- mysql.rds_set_external_master (Aurora MySQL versione 2)
- mysql.rds_set_external_source (Aurora MySQL versione 3)
- mysql.rds_set_external_master_with_auto_position (Aurora MySQL versione 2)
- mysql.rds_set_external_source_with_auto_position (Aurora MySQL versione 3)
- mysql.rds_set_master_auto_position (Aurora MySQL versione 2)
- mysql.rds_set_read_only (Aurora MySQL versione 3)
- mysql.rds_set_session_binlog_format (Aurora MySQL versione 2)
- mysql.rds_set_source_auto_position (Aurora MySQL versione 3)
- mysql.rds_skip_repl_error
- mysql.rds_start_replication
- mysql.rds_start_replication_until (Aurora MySQL versione 3)
- mysql.rds_stop_replication

mysql.rds_disable_session_binlog (Aurora MySQL versione 2)

Disattiva la registrazione binaria per la sessione corrente impostando la variabile `sql_log_bin` su `OFF`.

Sintassi

```
CALL mysql.rds_disable_session_binlog;
```

Parametri

Nessuno

Note per l'utilizzo

Per un cluster di database Aurora MySQL, puoi chiamare questa procedura archiviata mentre sei connesso all'istanza primaria.

Per Aurora, questa procedura è supportata per Aurora MySQL versione 2.12 e versioni successive compatibili con MySQL 5.7.

> **Note**
>
> In Aurora MySQL versione 3, è possibile utilizzare il seguente comando per disabilitare la registrazione di log binari per la sessione corrente se si dispone del privilegio `SESSION_VARIABLES_ADMIN`:
>
> ```
> SET SESSION sql_log_bin = OFF;
> ```

mysql.rds_enable_session_binlog (Aurora MySQL versione 2)

Attiva la registrazione binaria per la sessione corrente impostando la variabile `sql_log_bin` su `ON`.

Sintassi

```
CALL mysql.rds_enable_session_binlog;
```

Parametri

Nessuno

Note per l'utilizzo

Per un cluster di database Aurora MySQL, puoi chiamare questa procedura archiviata mentre sei connesso all'istanza primaria.

Per Aurora, questa procedura è supportata per Aurora MySQL versione 2.12 e versioni successive compatibili con MySQL 5.7.

> **Note**
>
> In Aurora MySQL versione 3, è possibile utilizzare il seguente comando per abilitare la registrazione di log binari per la sessione corrente se si dispone del privilegio `SESSION_VARIABLES_ADMIN`:
>
> ```
> SET SESSION sql_log_bin = ON;
> ```

mysql.rds_import_binlog_ssl_material

Importa il certificato dell'autorità di certificazione, il certificato client e la chiave client in un cluster di database Aurora MySQL. Le informazioni sono necessarie per la comunicazione SSL e la replica crittografata.

> **Note**
>
> Attualmente, questa procedura è supportata per Aurora MySQL versione 2 (2.09.2, 2.10.0, 2.10.1 e 2.11.0) e versione 3 (3.01.1 e successive).

Sintassi

```
CALL mysql.rds_import_binlog_ssl_material (
  ssl_material
);
```

Parametri

ssl_material

Il payload JSON che contiene il contenuto dei seguenti file in formato .pem per un client MySQL:

- «ssl_it»:»» *Certificate authority certificate*
- «certificato SSL»:»» *Client certificate*
- «chiave_ssl»:»» *Client key*

Note per l'utilizzo

Prepara la replica crittografata prima di eseguire questa procedura:

- Se SSL non è abilitato sull'istanza database di origine MySQL esterna e non disponi di una chiave client e di un certificato client preparato, abilita SSL sul server di database MySQL e genera la chiave client e il certificato client necessari.
- Se SSL è abilitato sull'istanza database di origine esterna, fornisci un certificato e una chiave client per il cluster database Aurora MySQL. Se non disponi di questi elementi, genera una nuova chiave e un nuovo certificato per il cluster di database Aurora MySQL. Per firmare il certificato client, devi avere la chiave autorità certificato che hai utilizzato per configurare SSL nell'istanza database di origine esterna MySQL.

Per ulteriori informazioni, consulta Creating SSL Certificates and Keys Using openssl nella documentazione MySQL.

> **Important**
>
> Dopo aver preparato la replica crittografata, utilizza una connessione SSL per eseguire questa procedura. La chiave client non deve essere trasferita mediante una connessione non sicura.

Questa procedura importa le informazioni SSL da un database MySQL esterno in un cluster di database Aurora MySQL. Le informazioni SSL sono in file in formato .pem che contengono le informazioni SSL per il cluster di database Aurora MySQL. Durante la replica crittografata, il cluster di database Aurora MySQL agisce come un client per il server di database MySQL. I certificati e le chiavi per il client Aurora MySQL sono in file in formato .pem.

Puoi copiare le informazioni da questi file nel parametro `ssl_material` nel payload JSON corretto. Per supportare la replica crittografata, importa queste informazioni SSL nel cluster di database Aurora MySQL.

Il payload JSON deve avere il formato seguente.

```
'{"ssl_ca":"-----BEGIN CERTIFICATE-----
ssl_ca_pem_body_code
-----END CERTIFICATE-----\n","ssl_cert":"-----BEGIN CERTIFICATE-----
```

```
ssl_cert_pem_body_code
-----END CERTIFICATE-----\n","ssl_key":"-----BEGIN RSA PRIVATE KEY-----
ssl_key_pem_body_code
-----END RSA PRIVATE KEY-----\n"}'
```

Esempi

L'esempio seguente importa le informazioni SSL in Aurora MySQL. Nei file in formato .pem, il codice del corpo è in genere più lungo del codice del corpo riportato nell'esempio.

```
call mysql.rds_import_binlog_ssl_material(
'{"ssl_ca":"-----BEGIN CERTIFICATE-----
AAAAB3NzaC1yc2EAAAADAQABAAABAQClKsfkNkuSevGj3eYhCe53pcjqP3maAhDFcvBS7O6V
hz2ItxCih+PnDSUaw+WNQn/mZphTk/a/gU8jEzoOWbkM4yxyb/wB96xbiFveSFJuOp/d6RJhJOI0iBXr
lsLnBItntckiJ7FbtxJMXLvvwJryDUilBMTjYtwB+QhYXUMOzce5Pjz5/i8SeJtjnV3iAoG/cQk+0FzZ
qaeJAAHco+CY/5WrUBkrHmFJr6HcXkvJdWPkYQS3xqC0+FmUZofz221CBt5IMucxXPkX4rWi+z7wB3Rb
BQoQzd8v7yeb7OzlPnWOyN0qFU0XA246RA8QFYiCNYwI3f05p6KLxEXAMPLE
-----END CERTIFICATE-----\n","ssl_cert":"-----BEGIN CERTIFICATE-----
AAAAB3NzaC1yc2EAAAADAQABAAABAQClKsfkNkuSevGj3eYhCe53pcjqP3maAhDFcvBS7O6V
hz2ItxCih+PnDSUaw+WNQn/mZphTk/a/gU8jEzoOWbkM4yxyb/wB96xbiFveSFJuOp/d6RJhJOI0iBXr
lsLnBItntckiJ7FbtxJMXLvvwJryDUilBMTjYtwB+QhYXUMOzce5Pjz5/i8SeJtjnV3iAoG/cQk+0FzZ
qaeJAAHco+CY/5WrUBkrHmFJr6HcXkvJdWPkYQS3xqC0+FmUZofz221CBt5IMucxXPkX4rWi+z7wB3Rb
BQoQzd8v7yeb7OzlPnWOyN0qFU0XA246RA8QFYiCNYwI3f05p6KLxEXAMPLE
-----END CERTIFICATE-----\n","ssl_key":"-----BEGIN RSA PRIVATE KEY-----
AAAAB3NzaC1yc2EAAAADAQABAAABAQClKsfkNkuSevGj3eYhCe53pcjqP3maAhDFcvBS7O6V
hz2ItxCih+PnDSUaw+WNQn/mZphTk/a/gU8jEzoOWbkM4yxyb/wB96xbiFveSFJuOp/d6RJhJOI0iBXr
lsLnBItntckiJ7FbtxJMXLvvwJryDUilBMTjYtwB+QhYXUMOzce5Pjz5/i8SeJtjnV3iAoG/cQk+0FzZ
qaeJAAHco+CY/5WrUBkrHmFJr6HcXkvJdWPkYQS3xqC0+FmUZofz221CBt5IMucxXPkX4rWi+z7wB3Rb
BQoQzd8v7yeb7OzlPnWOyN0qFU0XA246RA8QFYiCNYwI3f05p6KLxEXAMPLE
-----END RSA PRIVATE KEY-----\n"}');
```

mysql.rds_next_master_log (Aurora MySQL versione 2)

Cambia la posizione del log dell'istanza database di origine all'inizio del successivo log binario nell'istanza database di origine. Utilizzare questa procedura solo se si riceve l' I/O errore di replica 1236 su una replica di lettura.

Sintassi

```
CALL mysql.rds_next_master_log(
```

```
curr_master_log
);
```

Parametri

curr_master_log

L'indice del file di log master corrente. Ad esempio, se il file corrente è denominato `mysql-bin-changelog.012345`, l'indice è 12345. Per determinare il nome del file di log master corrente, esegui il comando `SHOW REPLICA STATUS` e visualizza il campo `Master_Log_File`.

Note per l'utilizzo

La procedura `mysql.rds_next_master_log` deve essere eseguita dall'utente master.

> **Warning**
>
> Effettua la chiamata `mysql.rds_next_master_log` solo se la replica fallisce dopo un failover di un'istanza DB Multi-AZ che è l'origine della replica e il campo riporta l'errore 1236. `Last_IO_Errno SHOW REPLICA STATUS I/O`
> La chiamata di `mysql.rds_next_master_log` può comportare una perdita di dati nella replica di lettura se le transazioni nell'istanza di origine non sono state scritte nel log binario sul disco prima dell'evento di failover.

Esempi

Supponi che una replica di lettura Aurora MySQL non riesca. L'esecuzione di `SHOW REPLICA STATUS\G` nella replica di lettura restituisce il risultato seguente:

```
*************************** 1. row ***************************
            Replica_IO_State:
                 Source_Host: myhost.XXXXXXXXXXXXXX.rr-rrrr-1.rds.amazonaws.com
                 Source_User: MasterUser
                 Source_Port: 3306
               Connect_Retry: 10
             Source_Log_File: mysql-bin-changelog.012345
          Read_Source_Log_Pos: 1219393
               Relay_Log_File: relaylog.012340
                Relay_Log_Pos: 30223388
```

```
        Relay_Source_Log_File: mysql-bin-changelog.012345
           Replica_IO_Running: No
          Replica_SQL_Running: Yes
              Replicate_Do_DB:
          Replicate_Ignore_DB:
           Replicate_Do_Table:
       Replicate_Ignore_Table:
      Replicate_Wild_Do_Table:
  Replicate_Wild_Ignore_Table:
                   Last_Errno: 0
                   Last_Error:
                 Skip_Counter: 0
          Exec_Source_Log_Pos: 30223232
              Relay_Log_Space: 5248928866
              Until_Condition: None
               Until_Log_File:
                Until_Log_Pos: 0
           Source_SSL_Allowed: No
           Source_SSL_CA_File:
           Source_SSL_CA_Path:
              Source_SSL_Cert:
            Source_SSL_Cipher:
               Source_SSL_Key:
        Seconds_Behind_Master: NULL
Source_SSL_Verify_Server_Cert: No
                Last_IO_Errno: 1236
                Last_IO_Error: Got fatal error 1236 from master when reading data from
 binary log: 'Client requested master to start replication from impossible position;
 the first event 'mysql-bin-changelog.013406' at 1219393, the last event read from
 '/rdsdbdata/log/binlog/mysql-bin-changelog.012345' at 4, the last byte read from '/
rdsdbdata/log/binlog/mysql-bin-changelog.012345' at 4.'
               Last_SQL_Errno: 0
               Last_SQL_Error:
  Replicate_Ignore_Server_Ids:
             Source_Server_Id: 67285976
```

Il campo `Last_IO_Errno` mostra che l'istanza riceve l'errore I/O 1236. Il campo `Master_Log_File` mostra che il nome di file è `mysql-bin-changelog.012345`, il che significa che l'indice del file di log è `12345`. Per risolvere il problema, puoi chiamare `mysql.rds_next_master_log` con il seguente parametro:

```
CALL mysql.rds_next_master_log(12345);
```

mysql.rds_next_source_log (Aurora MySQL versione 3)

Cambia la posizione del log dell'istanza database di origine all'inizio del successivo log binario nell'istanza database di origine. Utilizzare questa procedura solo se si riceve l' I/O errore di replica 1236 su una replica di lettura.

Sintassi

```
CALL mysql.rds_next_source_log(
curr_source_log
);
```

Parametri

curr_source_log

L'indice del file di log di origine corrente. Ad esempio, se il file corrente è denominato `mysql-bin-changelog.012345`, l'indice è 12345. Per determinare il nome del file di log di origine corrente, esegui il comando `SHOW REPLICA STATUS` e visualizza il campo `Source_Log_File`.

Note per l'utilizzo

L'utente amministrativo deve eseguire la procedura `mysql.rds_next_source_log`.

> **Warning**
>
> Effettua la chiamata `mysql.rds_next_source_log` solo se la replica fallisce dopo un failover di un'istanza DB Multi-AZ che è l'origine della replica e il campo riporta l'errore 1236. `Last_IO_Errno` `SHOW REPLICA STATUS` I/O
> La chiamata di `mysql.rds_next_source_log` può comportare una perdita di dati nella replica di lettura se le transazioni nell'istanza di origine non sono state scritte nel log binario sul disco prima dell'evento di failover. Si può diminuire la possibilità che si verifichi una situazione di questo tipo impostando i parametri dell'istanza di origine `sync_binlog` e `innodb_support_xa` su 1, anche se ciò può compromettere le prestazioni.

Esempi

Supponi che una replica di lettura Aurora MySQL non riesca. L'esecuzione di `SHOW REPLICA STATUS\G` nella replica di lettura restituisce il risultato seguente:

```
*************************** 1. row ***************************
             Replica_IO_State:
                  Source_Host: myhost.XXXXXXXXXXXXXX.rr-rrrr-1.rds.amazonaws.com
                  Source_User: MasterUser
                  Source_Port: 3306
                Connect_Retry: 10
              Source_Log_File: mysql-bin-changelog.012345
          Read_Source_Log_Pos: 1219393
               Relay_Log_File: relaylog.012340
                Relay_Log_Pos: 30223388
        Relay_Source_Log_File: mysql-bin-changelog.012345
           Replica_IO_Running: No
          Replica_SQL_Running: Yes
              Replicate_Do_DB:
          Replicate_Ignore_DB:
           Replicate_Do_Table:
       Replicate_Ignore_Table:
      Replicate_Wild_Do_Table:
  Replicate_Wild_Ignore_Table:
                   Last_Errno: 0
                   Last_Error:
                 Skip_Counter: 0
          Exec_Source_Log_Pos: 30223232
              Relay_Log_Space: 5248928866
              Until_Condition: None
               Until_Log_File:
                Until_Log_Pos: 0
           Source_SSL_Allowed: No
           Source_SSL_CA_File:
           Source_SSL_CA_Path:
              Source_SSL_Cert:
            Source_SSL_Cipher:
               Source_SSL_Key:
        Seconds_Behind_Source: NULL
Source_SSL_Verify_Server_Cert: No
                Last_IO_Errno: 1236
                Last_IO_Error: Got fatal error 1236 from source when reading data from
 binary log: 'Client requested source to start replication from impossible position;
 the first event 'mysql-bin-changelog.013406' at 1219393, the last event read from
```

```
 '/rdsdbdata/log/binlog/mysql-bin-changelog.012345' at 4, the last byte read from '/
rdsdbdata/log/binlog/mysql-bin-changelog.012345' at 4.'
               Last_SQL_Errno: 0
               Last_SQL_Error:
  Replicate_Ignore_Server_Ids:
             Source_Server_Id: 67285976
```

Il campo `Last_IO_Errno` mostra che l'istanza riceve l'errore I/O 1236. Il campo `Source_Log_File` mostra che il nome di file è `mysql-bin-changelog.012345`, il che significa che l'indice del file di log è `12345`. Per risolvere il problema, puoi chiamare `mysql.rds_next_source_log` con il seguente parametro:

```
CALL mysql.rds_next_source_log(12345);
```

mysql.rds_remove_binlog_ssl_material

Elimina il certificato dell'autorità di certificazione, il certificato client e la chiave client per la comunicazione SSL e la replica crittografata. Queste informazioni sono importate utilizzando mysql.rds_import_binlog_ssl_material.

Sintassi

```
CALL mysql.rds_remove_binlog_ssl_material;
```

mysql.rds_reset_external_master (Aurora MySQL versione 2)

Riconfigura un'istanza database Aurora MySQL affinché non sia più una replica di lettura di un'istanza di MySQL in esecuzione all'esterno di Amazon RDS.

> **Important**
>
> Per eseguire questa procedura, è necessario abilitare `autocommit`. Per abilitarlo, impostare il parametro `autocommit` su 1. Per ulteriori informazioni sulla modifica dei parametri, consulta Modifica dei parametri in un gruppo di parametri database in Amazon Aurora.

Sintassi

```
CALL mysql.rds_reset_external_master;
```

Note per l'utilizzo

La procedura `mysql.rds_reset_external_master` deve essere eseguita dall'utente master. Questa procedura deve essere eseguita sull'istanza database MySQL da rimuovere come replica di lettura di un'istanza MySQL eseguita esternamente a Amazon RDS.

> **Note**
>
> Queste stored procedure sono fornite principalmente per abilitare la replica con le istanze MySQL eseguite esternamente a Amazon RDS. Ti consigliamo di usare le repliche Aurora per gestire la replica in un cluster database Aurora MySQL. Per informazioni sulla gestione della replica nei cluster database Aurora MySQL, consulta Utilizzo delle repliche di Aurora.

Per ulteriori informazioni sull'uso della replica per importare dati da un'istanza di MySQL in esecuzione all'esterno di Aurora MySQL, consulta Replica tra Aurora e MySQL o tra Aurora e un altro cluster di database Aurora (replica dei log binari).

mysql.rds_reset_external_source (Aurora MySQL versione 3)

Riconfigura un'istanza database Aurora MySQL affinché non sia più una replica di lettura di un'istanza di MySQL in esecuzione all'esterno di Amazon RDS.

> **Important**
>
> Per eseguire questa procedura, è necessario abilitare `autocommit`. Per abilitarlo, impostare il parametro `autocommit` su `1`. Per ulteriori informazioni sulla modifica dei parametri, consulta Modifica dei parametri in un gruppo di parametri database in Amazon Aurora.

Sintassi

```
CALL mysql.rds_reset_external_source;
```

Note per l'utilizzo

L'utente amministrativo deve eseguire la procedura `mysql.rds_reset_external_source`. Questa procedura deve essere eseguita sull'istanza database MySQL da rimuovere come replica di lettura di un'istanza MySQL eseguita esternamente a Amazon RDS.

> **Note**
>
> Queste stored procedure sono fornite principalmente per abilitare la replica con le istanze MySQL eseguite esternamente a Amazon RDS. Ti consigliamo di usare le repliche Aurora per gestire la replica in un cluster database Aurora MySQL. Per informazioni sulla gestione della replica nei cluster database Aurora MySQL, consulta Utilizzo delle repliche di Aurora.

mysql.rds_set_binlog_source_ssl (Aurora MySQL versione 3)

Abilita la crittografia SOURCE_SSL per la replica binlog. Per ulteriori informazioni, consulta CHANGE REPLICATION SOURCE TO statement nella documentazione MySQL.

Sintassi

```
CALL mysql.rds_set_binlog_source_ssl(mode);
```

Parametri

mode

Valore che indica se la crittografia SOURCE_SSL è abilitata:

- 0: la crittografia SOURCE_SSL è disabilitata. Il valore predefinito è 0.
- 1: la crittografia SOURCE_SSL è abilitata. È possibile configurare la crittografia utilizzando SSL o TLS.

Note per l'utilizzo

Questa procedura è supportata per Aurora MySQL versione 3.06 e successive.

mysql.rds_set_external_master (Aurora MySQL versione 2)

Configura un'istanza database Aurora MySQL come replica di lettura di un'istanza di MySQL in esecuzione all'esterno di Amazon RDS.

La procedura mysql.rds_set_external_master è obsoleta e verrà rimossa in una versione futura. Usare invece mysql.rds_set_external_source.

> **Important**
>
> Per eseguire questa procedura, è necessario abilitare `autocommit`. Per abilitarlo, impostare il parametro `autocommit` su 1. Per ulteriori informazioni sulla modifica dei parametri, consulta Modifica dei parametri in un gruppo di parametri database in Amazon Aurora.

Sintassi

```
CALL mysql.rds_set_external_master (
  host_name
  , host_port
  , replication_user_name
  , replication_user_password
  , mysql_binary_log_file_name
  , mysql_binary_log_file_location
  , ssl_encryption
);
```

Parametri

host_name

Il nome host o l'indirizzo IP dell'istanza di MySQL eseguita esternamente a Amazon RDS per diventare l'istanza database di origine.

host_port

La porta utilizzata dall'istanza di MySQL eseguita esternamente a Amazon RDS e da configurare come istanza database di origine. Se la configurazione della rete include la replica della porta Secure Shell (SSH) che converte il numero di porta, specifica il numero di porta esposto da SSH.

replication_user_name

L'ID di un utente con autorizzazioni `REPLICATION CLIENT` e `REPLICATION SLAVE` nell'istanza di MySQL eseguita esternamente a Amazon RDS. Ti consigliamo di fornire un account utilizzato unicamente per la replica con l'istanza esterna.

replication_user_password

La password dell'ID utente specificata in `replication_user_name`.

mysql_binary_log_file_name

Il nome del log binario sull'istanza database di origine che contiene le informazioni relative alla replica.

mysql_binary_log_file_location

La posizione nel log binario `mysql_binary_log_file_name` a partire dalla quale la replica inizia a leggere le informazioni a essa relative.

È possibile determinare il nome e la posizione del file binlog in esecuzione `SHOW MASTER STATUS` sull'istanza del database di origine.

ssl_encryption

Un valore che specifica se la crittografia Secure Socket Layer (SSL) è utilizzata sulla connessione di replica. 1 indica che la crittografia SSL deve essere utilizzata; 0 specifica che la crittografia non deve essere utilizzata. Il valore predefinito è 0.

> **Note**
>
> L'opzione `MASTER_SSL_VERIFY_SERVER_CERT` non è supportata. Questa opzione è impostata su 0, il che significa che la connessione è crittografata, ma i certificati non sono verificati.

Note per l'utilizzo

La procedura `mysql.rds_set_external_master` deve essere eseguita dall'utente master. Questa procedura deve essere eseguita sull'istanza database MySQL da configurare come replica di lettura di un'istanza MySQL eseguita esternamente a Amazon RDS.

Prima di eseguire `mysql.rds_set_external_master`, devi configurare l'istanza di MySQL in esecuzione all'esterno di Amazon RDS come istanza database di origine. Per connetterti all'istanza MySQL in esecuzione all'esterno di Amazon RDS, devi specificare i valori di `replication_user_name` e `replication_user_password` che indicano un utente della replica dotato delle autorizzazioni `REPLICATION CLIENT` e `REPLICATION SLAVE` per l'istanza esterna di MySQL.

Per configurare un'istanza esterna di MySQL come istanza database di origine

1. Mediante il client MySQL scelto, eseguire la connessione all'istanza esterna di MySQL e creare un account utente da utilizzare per la replica. Di seguito è riportato un esempio di :

   MySQL 5.7

   ```
   CREATE USER 'repl_user'@'mydomain.com' IDENTIFIED BY 'password';
   ```

   MySQL 8.0

   ```
   CREATE USER 'repl_user'@'mydomain.com' IDENTIFIED WITH mysql_native_password BY
    'password';
   ```

   > **Note**
   >
   > Specifica una password diversa dal prompt mostrato qui come best practice per la sicurezza.

2. Nell'istanza esterna di MySQL, concedere i privilegi `REPLICATION CLIENT` e `REPLICATION SLAVE` all'utente della replica. L'esempio seguente concede i privilegi `REPLICATION CLIENT` e `REPLICATION SLAVE` su tutti i database per l'utente "repl_user" del dominio:

   MySQL 5.7

   ```
   GRANT REPLICATION CLIENT, REPLICATION SLAVE ON *.* TO 'repl_user'@'mydomain.com'
    IDENTIFIED BY 'password';
   ```

   MySQL 8.0

   ```
   GRANT REPLICATION CLIENT, REPLICATION SLAVE ON *.* TO 'repl_user'@'mydomain.com';
   ```

Per utilizzare la replica crittografata, configura l'istanza database di origine per utilizzare le connessioni SSL. Importa inoltre il certificato dell'autorità di certificazione, il certificato client e la chiave client nell'istanza database o nel cluster di database utilizzando la procedura mysql.rds_import_binlog_ssl_material.

> **Note**
>
> Queste stored procedure sono fornite principalmente per abilitare la replica con le istanze MySQL eseguite esternamente a Amazon RDS. Ti consigliamo di usare le repliche Aurora per gestire la replica in un cluster database Aurora MySQL. Per informazioni sulla gestione della replica nei cluster database Aurora MySQL, consulta Utilizzo delle repliche di Aurora.

Dopo aver chiamato `mysql.rds_set_external_master` per configurare un'istanza database di Amazon RDS come replica di lettura, puoi chiamare mysql.rds_start_replication nella replica di lettura per avviare il processo di replica. Puoi chiamare mysql.rds_reset_external_master (Aurora MySQL versione 2) per rimuovere la configurazione della replica di lettura.

Quando `mysql.rds_set_external_master` viene chiamato, Amazon RDS registra l'ora, l'utente e un'operazione di `set master` nelle tabelle `mysql.rds_history` e `mysql.rds_replication_status`.

Esempi

Nel caso di esecuzione su un'istanza database MySQL, l'esempio seguente configura l'istanza database come replica di lettura di un'istanza di MySQL eseguita esternamente a Amazon RDS.

```
call mysql.rds_set_external_master(
  'Externaldb.some.com',
  3306,
  'repl_user',
  'password',
  'mysql-bin-changelog.0777',
  120,
  1);
```

mysql.rds_set_external_source (Aurora MySQL versione 3)

Configura un'istanza database Aurora MySQL come replica di lettura di un'istanza di MySQL in esecuzione all'esterno di Amazon RDS.

> **Important**
>
> Per eseguire questa procedura, è necessario abilitare `autocommit`. Per abilitarlo, impostare il parametro `autocommit` su 1. Per ulteriori informazioni sulla modifica dei parametri, consulta Modifica dei parametri in un gruppo di parametri database in Amazon Aurora.

Sintassi

```
CALL mysql.rds_set_external_source (
  host_name
  , host_port
  , replication_user_name
  , replication_user_password
  , mysql_binary_log_file_name
  , mysql_binary_log_file_location
  , ssl_encryption
);
```

Parametri

host_name

Il nome host o l'indirizzo IP dell'istanza di MySQL eseguita esternamente a Amazon RDS per diventare l'istanza database di origine.

host_port

La porta utilizzata dall'istanza di MySQL eseguita esternamente a Amazon RDS e da configurare come istanza database di origine. Se la configurazione della rete include la replica della porta Secure Shell (SSH) che converte il numero di porta, specifica il numero di porta esposto da SSH.

replication_user_name

L'ID di un utente con autorizzazioni `REPLICATION CLIENT` e `REPLICATION SLAVE` nell'istanza di MySQL eseguita esternamente a Amazon RDS. Ti consigliamo di fornire un account utilizzato unicamente per la replica con l'istanza esterna.

replication_user_password

La password dell'ID utente specificata in `replication_user_name`.

mysql_binary_log_file_name

Il nome del log binario sull'istanza database di origine che contiene le informazioni relative alla replica.

mysql_binary_log_file_location

La posizione nel log binario `mysql_binary_log_file_name` a partire dalla quale la replica inizia a leggere le informazioni a essa relative.

È possibile determinare il nome e la posizione del file binlog in esecuzione `SHOW MASTER STATUS` sull'istanza del database di origine.

ssl_encryption

Un valore che specifica se la crittografia Secure Socket Layer (SSL) è utilizzata sulla connessione di replica. 1 indica che la crittografia SSL deve essere utilizzata; 0 specifica che la crittografia non deve essere utilizzata. Il valore predefinito è 0.

> **Note**
>
> Per abilitare questa opzione, è necessario aver importato un certificato SSL personalizzato tramite mysql.rds_import_binlog_ssl_material. Se non è stato importato un certificato SSL personalizzato, impostare questo parametro su 0 e utilizzare mysql.rds_set_binlog_source_ssl (Aurora MySQL versione 3) per abilitare SSL per la replica dei log binari.
> L'opzione `SOURCE_SSL_VERIFY_SERVER_CERT` non è supportata. Questa opzione è impostata su 0, il che significa che la connessione è crittografata, ma i certificati non sono verificati.

Note per l'utilizzo

L'utente amministrativo deve eseguire la procedura `mysql.rds_set_external_source`. Questa procedura deve essere eseguita nell'istanza database Aurora MySQL da configurare come replica di lettura di un'istanza MySQL eseguita esternamente ad Amazon RDS.

Prima di eseguire `mysql.rds_set_external_source`, devi configurare l'istanza di MySQL in esecuzione all'esterno di Amazon RDS come istanza database di origine. Per connetterti all'istanza MySQL in esecuzione all'esterno di Amazon RDS, devi specificare i valori di

`replication_user_name` e `replication_user_password` che indicano un utente della replica dotato delle autorizzazioni `REPLICATION CLIENT` e `REPLICATION SLAVE` per l'istanza esterna di MySQL.

Per configurare un'istanza esterna di MySQL come istanza database di origine

1. Mediante il client MySQL scelto, eseguire la connessione all'istanza esterna di MySQL e creare un account utente da utilizzare per la replica. Di seguito è riportato un esempio di :

   ```
   CREATE USER 'repl_user'@'mydomain.com' IDENTIFIED BY 'password';
   ```

   > **Note**
   >
   > Specifica una password diversa dal prompt mostrato qui come best practice per la sicurezza.

2. Nell'istanza esterna di MySQL, concedere i privilegi `REPLICATION CLIENT` e `REPLICATION SLAVE` all'utente della replica. L'esempio seguente concede i privilegi `REPLICATION CLIENT` e `REPLICATION SLAVE` su tutti i database per l'utente "repl_user" del dominio:

   ```
   GRANT REPLICATION CLIENT, REPLICATION SLAVE ON *.* TO 'repl_user'@'mydomain.com';
   ```

Per utilizzare la replica crittografata, configura l'istanza database di origine per utilizzare le connessioni SSL. Importa inoltre il certificato dell'autorità di certificazione, il certificato client e la chiave client nell'istanza database o nel cluster database utilizzando la procedura mysql.rds_import_binlog_ssl_material.

> **Note**
>
> Queste stored procedure sono fornite principalmente per abilitare la replica con le istanze MySQL eseguite esternamente a Amazon RDS. Ti consigliamo di usare le repliche Aurora per gestire la replica in un cluster database Aurora MySQL. Per informazioni sulla gestione della replica nei cluster database Aurora MySQL, consulta Utilizzo delle repliche di Aurora.

Dopo aver chiamato `mysql.rds_set_external_source` per configurare un'istanza database Aurora MySQL come replica di lettura, è possibile chiamare mysql.rds_start_replication sulla replica

di lettura per avviare il processo di replica. Puoi chiamare mysql.rds_reset_external_source (Aurora MySQL versione 3) per rimuovere la configurazione della replica di lettura.

Quando `mysql.rds_set_external_source` viene chiamato, Amazon RDS registra l'ora, l'utente e un'operazione di `set master` nelle tabelle `mysql.rds_history` e `mysql.rds_replication_status`.

Esempi

Se in esecuzione in un'istanza database Aurora MySQL, l'esempio seguente configura l'istanza database come replica di lettura di un'istanza di MySQL eseguita esternamente ad Amazon RDS.

```
call mysql.rds_set_external_source(
  'Externaldb.some.com',
  3306,
  'repl_user',
  'password',
  'mysql-bin-changelog.0777',
  120,
  1);
```

mysql.rds_set_external_master_with_auto_position (Aurora MySQL versione 2)

Configura un'istanza primaria Aurora MySQL affinché accetti la replica in entrata da un'istanza di MySQL esterna. Questa procedura configura anche la replica in base agli identificatori globali delle transazioni (). GTIDs

Questa procedura non configura la replica ritardata in quanto non è supportata da Aurora MySQL.

Sintassi

```
CALL mysql.rds_set_external_master_with_auto_position (
  host_name
  , host_port
  , replication_user_name
  , replication_user_password
  , ssl_encryption
);
```

Parametri

host_name

Il nome host o l'indirizzo IP dell'istanza di MySQL eseguita esternamente a Aurora per diventare il fonte di replica.

host_port

La porta utilizzata dall'istanza di MySQL eseguita esternamente a Aurora e da configurare come fonte di replica. Se la configurazione della rete include la replica della porta Secure Shell (SSH) che converte il numero di porta, specifica il numero di porta esposto da SSH.

replication_user_name

L'ID di un utente con autorizzazioni `REPLICATION CLIENT` e `REPLICATION SLAVE` nell'istanza di MySQL eseguita esternamente a Aurora. Ti consigliamo di fornire un account utilizzato unicamente per la replica con l'istanza esterna.

replication_user_password

La password dell'ID utente specificata in `replication_user_name`.

ssl_encryption

Questa opzione non è al momento implementata. Il valore predefinito è 0.

Note per l'utilizzo

Per un cluster di database Aurora MySQL, puoi chiamare questa procedura archiviata mentre sei connesso all'istanza primaria.

La procedura `mysql.rds_set_external_master_with_auto_position` deve essere eseguita dall'utente master. L'utente master esegue questa procedura sull'istanza primaria di un cluster di database Aurora MySQL che opera da destinazione di replica. Questa può essere la destinazione di replica di un'istanza database MySQL esterna o di un cluster di database Aurora MySQL.

Questa procedura è supportata per Aurora MySQL versione 2. Per Aurora MySQL versione 3, utilizzare invece la procedura mysql.rds_set_external_source_with_auto_position (Aurora MySQL versione 3).

Prima di eseguire `mysql.rds_set_external_master_with_auto_position`, configura l'istanza database MySQL esterna come fonte di replica. Per connetterti all'istanza di MySQL esterna,

specifica i valori per `replication_user_name` e `replication_user_password`. Questi valori devono indicare un utente di replica che dispone delle autorizzazioni `REPLICATION CLIENT` e `REPLICATION SLAVE` sull'istanza di MySQL esterna.

Per configurare un'istanza di MySQL esterna come fonte di replica

1. Mediante il client MySQL scelto, eseguire la connessione all'istanza di MySQL esterna e creare un account utente da utilizzare per la replica. Di seguito è riportato un esempio.

   ```
   CREATE USER 'repl_user'@'mydomain.com' IDENTIFIED BY 'SomePassW0rd'
   ```

2. Per un'istanza di MySQL esterna, concedere i privilegi `REPLICATION CLIENT` e `REPLICATION SLAVE` all'utente della replica. L'esempio seguente concede i privilegi `REPLICATION CLIENT` e `REPLICATION SLAVE` su tutti i database per l'utente `'repl_user'` per il dominio.

   ```
   GRANT REPLICATION CLIENT, REPLICATION SLAVE ON *.* TO 'repl_user'@'mydomain.com'
   IDENTIFIED BY 'SomePassW0rd'
   ```

Quando chiami `mysql.rds_set_external_master_with_auto_position`, Amazon RDS registra determinate informazioni. Queste informazioni sono l'orario, l'utente e l'operazione di `"set master"` nelle tabelle `mysql.rds_history` e `mysql.rds_replication_status`.

Per passare a una specifica transazione basata su GTID che notoriamente causa un problema, puoi usare la procedura archiviata mysql.rds_skip_transaction_with_gtid (Aurora MySQL versione 2 e 3). Per ulteriori informazioni sull'utilizzo della replica basata su GTID, consulta Utilizzo della replica basata su GTID.

Esempi

Nel caso di esecuzione su un'istanza primaria Aurora, l'esempio seguente configura il cluster Aurora come replica di lettura di un'istanza di MySQL eseguita esternamente a Aurora.

```
call mysql.rds_set_external_master_with_auto_position(
  'Externaldb.some.com',
  3306,
  'repl_user'@'mydomain.com',
  'SomePassW0rd');
```

mysql.rds_set_external_source_with_auto_position (Aurora MySQL versione 3)

Configura un'istanza primaria Aurora MySQL affinché accetti la replica in entrata da un'istanza di MySQL esterna. Questa procedura configura anche la replica in base agli identificatori globali delle transazioni (). GTIDs

Sintassi

```
CALL mysql.rds_set_external_source_with_auto_position (
  host_name
  , host_port
  , replication_user_name
  , replication_user_password
  , ssl_encryption
);
```

Parametri

host_name

Il nome host o l'indirizzo IP dell'istanza di MySQL eseguita esternamente a Aurora per diventare il fonte di replica.

host_port

La porta utilizzata dall'istanza di MySQL eseguita esternamente a Aurora e da configurare come fonte di replica. Se la configurazione della rete include la replica della porta Secure Shell (SSH) che converte il numero di porta, specifica il numero di porta esposto da SSH.

replication_user_name

L'ID di un utente con autorizzazioni `REPLICATION CLIENT` e `REPLICATION SLAVE` nell'istanza di MySQL eseguita esternamente a Aurora. Ti consigliamo di fornire un account utilizzato unicamente per la replica con l'istanza esterna.

replication_user_password

La password dell'ID utente specificata in `replication_user_name`.

ssl_encryption

Questa opzione non è al momento implementata. Il valore predefinito è 0.

> **Note**
>
> Utilizzare mysql.rds_set_binlog_source_ssl (Aurora MySQL versione 3) per abilitare SSL per la replica di log binari.

Note per l'utilizzo

Per un cluster di database Aurora MySQL, puoi chiamare questa procedura archiviata mentre sei connesso all'istanza primaria.

L'utente amministrativo deve eseguire la procedura `mysql.rds_set_external_source_with_auto_position`. L'utente master esegue questa procedura sull'istanza primaria di un cluster di database Aurora MySQL che opera da destinazione di replica. Questa può essere la destinazione di replica di un'istanza database MySQL esterna o di un cluster di database Aurora MySQL.

Questa procedura è supportata per Aurora MySQL versione 3. Questa procedura non configura la replica ritardata in quanto non è supportata da Aurora MySQL.

Prima di eseguire `mysql.rds_set_external_source_with_auto_position`, configura l'istanza database MySQL esterna come fonte di replica. Per connetterti all'istanza di MySQL esterna, specifica i valori per `replication_user_name` e `replication_user_password`. Questi valori devono indicare un utente di replica che dispone delle autorizzazioni `REPLICATION CLIENT` e `REPLICATION SLAVE` sull'istanza di MySQL esterna.

Per configurare un'istanza di MySQL esterna come fonte di replica

1. Mediante il client MySQL scelto, eseguire la connessione all'istanza di MySQL esterna e creare un account utente da utilizzare per la replica. Di seguito è riportato un esempio.

   ```
   CREATE USER 'repl_user'@'mydomain.com' IDENTIFIED BY 'SomePassW0rd'
   ```

2. Per un'istanza di MySQL esterna, concedere i privilegi `REPLICATION CLIENT` e `REPLICATION SLAVE` all'utente della replica. L'esempio seguente concede i privilegi `REPLICATION CLIENT` e `REPLICATION SLAVE` su tutti i database per l'utente `'repl_user'` per il dominio.

   ```
   GRANT REPLICATION CLIENT, REPLICATION SLAVE ON *.* TO 'repl_user'@'mydomain.com'
   IDENTIFIED BY 'SomePassW0rd'
   ```

Quando chiami `mysql.rds_set_external_source_with_auto_position`, Amazon RDS registra determinate informazioni. Queste informazioni sono l'orario, l'utente e l'operazione di `"set master"` nelle tabelle `mysql.rds_history` e `mysql.rds_replication_status`.

Per passare a una specifica transazione basata su GTID che notoriamente causa un problema, puoi usare la procedura archiviata mysql.rds_skip_transaction_with_gtid (Aurora MySQL versione 2 e 3). Per ulteriori informazioni sull'utilizzo della replica basata su GTID, consulta Utilizzo della replica basata su GTID.

Esempi

Nel caso di esecuzione su un'istanza primaria Aurora, l'esempio seguente configura il cluster Aurora come replica di lettura di un'istanza di MySQL eseguita esternamente a Aurora.

```
call mysql.rds_set_external_source_with_auto_position(
  'Externaldb.some.com',
  3306,
  'repl_user'@'mydomain.com',
  'SomePassW0rd');
```

mysql.rds_set_master_auto_position (Aurora MySQL versione 2)

Imposta la modalità di replica in modo che sia basata sulle posizioni dei file di registro binari o sugli identificatori globali delle transazioni (). GTIDs

Sintassi

```
CALL mysql.rds_set_master_auto_position (
auto_position_mode
);
```

Parametri

auto_position_mode

Valore che indica se usare la replica basata sulla posizione del file di log o la replica basata su GTID:

- `0` – Usa il metodo di replica basato sulla posizione del file di log binario. Il valore di default è `0`.
- 1 – Usa il metodo di replica basato su GTID.

Note per l'utilizzo

La procedura `mysql.rds_set_master_auto_position` deve essere eseguita dall'utente master.

Questa procedura è supportata per Aurora MySQL versione 2.

mysql.rds_set_read_only (Aurora MySQL versione 3)

Attiva o disattiva la modalità `read_only` a livello globale per l'istanza database.

Sintassi

```
CALL mysql.rds_set_read_only(mode);
```

Parametri

mode

Valore che indica se la modalità `read_only` è attiva o disattiva a livello globale per l'istanza database:

- `0` – OFF. Il valore predefinito è `0`.
- `1` – ON

Note per l'utilizzo

La stored procedure `mysql.rds_set_read_only` modifica solo il parametro `read_only`. Il parametro `innodb_read_only` non può essere modificato sulle istanze database di lettura.

La modifica del parametro `read_only` non viene mantenuta al riavvio. Per apportare modifiche permanenti a `read_only`, è necessario utilizzare il parametro `read_only` del cluster di database.

Questa procedura è supportata per Aurora MySQL versione 3.06 e successive.

mysql.rds_set_session_binlog_format (Aurora MySQL versione 2)

Imposta il formato di log binario per la sessione corrente.

Sintassi

```
CALL mysql.rds_set_session_binlog_format(format);
```

Parametri

format

Un valore che indica il formato di log binario per la sessione corrente:

- `STATEMENT`: l'origine della replica scrive eventi nel log binario in base alle istruzioni SQL.
- `ROW`: l'origine della replica scrive eventi nel log binario che indicano le modifiche alle singole righe della tabella.
- `MIXED`: la registrazione si basa in genere su istruzioni SQL, ma passa alle righe in determinate condizioni. Per ulteriori informazioni, consulta Mixed Binary Logging Format nella documentazione di MySQL.

Note per l'utilizzo

Per un cluster di database Aurora MySQL, puoi chiamare questa procedura archiviata mentre sei connesso all'istanza primaria.

Per utilizzare questa procedura archiviata, la registrazione binaria deve essere configurata per la sessione corrente.

Per Aurora, questa procedura è supportata per Aurora MySQL versione 2.12 e versioni successive compatibili con MySQL 5.7.

mysql.rds_set_source_auto_position (Aurora MySQL versione 3)

Imposta la modalità di replica in modo che sia basata sulle posizioni dei file di registro binari o sugli identificatori globali delle transazioni (). GTIDs

Sintassi

```
CALL mysql.rds_set_source_auto_position (auto_position_mode);
```

Parametri

auto_position_mode

Valore che indica se usare la replica basata sulla posizione del file di log o la replica basata su GTID:

- `0` – Usa il metodo di replica basato sulla posizione del file di log binario. Il valore di default è `0`.

- 1 – Usa il metodo di replica basato su GTID.

Note per l'utilizzo

Per un cluster di database Aurora MySQL, puoi chiamare questa procedura archiviata mentre sei connesso all'istanza primaria.

L'utente amministrativo deve eseguire la procedura `mysql.rds_set_source_auto_position`.

mysql.rds_skip_repl_error

Ignora ed elimina un errore di replica su una replica di lettura database MySQL.

Sintassi

```
CALL mysql.rds_skip_repl_error;
```

Note per l'utilizzo

La procedura `mysql.rds_skip_repl_error` deve essere eseguita dall'utente master su una replica di lettura. Per ulteriori informazioni su questa procedura, consulta Ignorare l'errore di replica corrente.

Per determinare se ci sono errori, esegui il comando MySQL `SHOW REPLICA STATUS\G`. Se un errore di replica non è critico, puoi eseguire `mysql.rds_skip_repl_error` per ignorare l'errore. Se vi sono più errori, `mysql.rds_skip_repl_error` elimina il primo, quindi informa della presenza di altri errori. Puoi quindi utilizzare `SHOW REPLICA STATUS\G` per determinare l'operazione corretta per l'errore successivo. Per informazioni sui valori restituiti, consulta Istruzione SHOW REPLICA STATUS nella documentazione di MySQL.

Per ulteriori informazioni sulla risoluzione degli errori di replica con Aurora MySQL, consulta Diagnosi e risoluzione di un errore relativo alla replica di lettura MySQL.

Errore di replica interrotta

Quando si chiama la procedura `mysql.rds_skip_repl_error`, è possibile che venga visualizzato un messaggio di errore che indica che la replica è inattiva o disattivata.

Questo messaggio di errore viene visualizzato se si esegue la procedura sull'istanza primaria anziché sulla replica di lettura. È necessario eseguire questa procedura sulla replica di lettura affinché funzioni.

Questo messaggio di errore può essere visualizzato anche quando si esegue la procedura sulla replica di lettura, ma la replica non viene riavviata correttamente.

Se devi ignorare un numero elevato di errori, il ritardo della replica potrebbe superare il periodo di retention predefinito per i file di log binari (binlog). In questo caso può verificarsi un errore irreversibile causato dall'eliminazione dei file binlog prima della loro riproduzione nella replica di lettura. Questa eliminazione causa l'arresto della replica e non è più possibile chiamare il comando `mysql.rds_skip_repl_error` per ignorare errori di replica.

Puoi limitare questo problema aumentando il numero di ore di retention dei file binlog nell'istanza database di origine. Una volta aumentato il tempo di retention dei file binlog, puoi riavviare la replica e chiamare il comando `mysql.rds_skip_repl_error` secondo necessità.

Per impostare il periodo di retention dei file binlog, usa la procedura mysql.rds_set_configuration e specifica un parametro di configurazione di `'binlog retention hours'` insieme al numero di ore di retention dei file binlog nel cluster di database. Nell'esempio seguente il periodo di retention dei file binlog è impostato su 48 ore.

```
CALL mysql.rds_set_configuration('binlog retention hours', 48);
```

mysql.rds_start_replication

Avvia la replica da un cluster di database Aurora MySQL.

> **Note**
>
> Puoi usare la stored procedure mysql.rds_start_replication_until (Aurora MySQL versione 3) o mysql.rds_start_replication_until_gtid (Aurora MySQL versione 3) per avviare la replica da un'istanza database Aurora MySQL e arrestare la replica in corrispondenza della posizione del file di log binario specificato.

Sintassi

```
CALL mysql.rds_start_replication;
```

Note per l'utilizzo

La procedura `mysql.rds_start_replication` deve essere eseguita dall'utente master.

Per importare dati da un'istanza MySQL in esecuzione all'esterno di Amazon RDS, chiama `mysql.rds_start_replication` nella replica di lettura per avviare il processo di replica dopo aver chiamato mysql.rds_set_external_master (Aurora MySQL versione 2) o mysql.rds_set_external_source (Aurora MySQL versione 3) per creare la configurazione della replica. Per ulteriori informazioni, consulta Replica tra Aurora e MySQL o tra Aurora e un altro cluster di database Aurora (replica dei log binari).

Per esportare dati in un'istanza di MySQL in esecuzione all'esterno di Amazon RDS, devi chiamare `mysql.rds_start_replication` e `mysql.rds_stop_replication` nella replica di lettura per controllare alcune operazioni di replica, come l'eliminazione di log binari. Per ulteriori informazioni, consulta Replica tra Aurora e MySQL o tra Aurora e un altro cluster di database Aurora (replica dei log binari).

Puoi anche chiamare `mysql.rds_start_replication` nella replica di lettura per riavviare un processo di replica arrestato in precedenza chiamando `mysql.rds_stop_replication`. Per ulteriori informazioni, consulta Errore di replica interrotta.

mysql.rds_start_replication_until (Aurora MySQL versione 3)

Avvia la replica da cluster di database Aurora MySQL e la arresta in corrispondenza della posizione del file di log binario specificato.

Sintassi

```
CALL mysql.rds_start_replication_until (
replication_log_file
  , replication_stop_point
);
```

Parametri

`replication_log_file`

Il nome del log binario sull'istanza database di origine che contiene le informazioni relative alla replica.

`replication_stop_point`

Posizione nel log binario `replication_log_file` in corrispondenza di cui la replica verrà arrestata.

Note per l'utilizzo

La procedura `mysql.rds_start_replication_until` deve essere eseguita dall'utente master.

Questa procedura è supportata per Aurora MySQL versione 3.04 e successive.

La stored procedure `mysql.rds_start_replication_until` non è supportata per la replica gestita, che include quanto segue:

- Repliche di cluster di database Amazon Aurora MySQL tra Regioni AWS
- Migrazione di dati da un'istanza database RDS per MySQL a un cluster database Amazon Aurora MySQL utilizzando una replica di lettura Aurora

Il nome file specificato per il parametro `replication_log_file` deve corrispondere al nome file binlog dell'istanza database di origine.

Quando il parametro `replication_stop_point` specifica una posizione di arresto nel passato, la replica viene arrestata immediatamente.

Esempi

L'esempio seguente avvia la replica e replica le modifiche fino a raggiungere la posizione `120` nel file di log binario `mysql-bin-changelog.000777`.

```
call mysql.rds_start_replication_until(
  'mysql-bin-changelog.000777',
  120);
```

mysql.rds_stop_replication

Arresta la replica da un'istanza database MySQL.

Sintassi

```
CALL mysql.rds_stop_replication;
```

Note per l'utilizzo

La procedura `mysql.rds_stop_replication` deve essere eseguita dall'utente master.

Se configuri la replica per importare dati da un'istanza di MySQL in esecuzione all'esterno di Amazon RDS, puoi chiamare `mysql.rds_stop_replication` nella replica di lettura per arrestare il processo di replica al termine dell'importazione. Per ulteriori informazioni, consulta Replica tra Aurora e MySQL o tra Aurora e un altro cluster di database Aurora (replica dei log binari).

Se configuri la replica per esportare dati in un'istanza di MySQL esterna ad Amazon RDS, devi chiamare `mysql.rds_start_replication` e `mysql.rds_stop_replication` nella replica di lettura per controllare alcune operazioni di replica, come l'eliminazione di log binari. Per ulteriori informazioni, consulta Replica tra Aurora e MySQL o tra Aurora e un altro cluster di database Aurora (replica dei log binari).

La stored procedure `mysql.rds_stop_replication` non è supportata per la replica gestita, che include quanto segue:

- Repliche di cluster di database Amazon Aurora MySQL tra Regioni AWS
- Migrazione di dati da un'istanza database RDS per MySQL a un cluster database Amazon Aurora MySQL utilizzando una replica di lettura Aurora

Terminare una sessione o una query

Le seguenti stored procedure terminano una sessione o una query.

Argomenti

- mysql.rds_kill
- mysql.rds_kill_query

mysql.rds_kill

Termina una connessione al server MySQL.

Sintassi

```
CALL mysql.rds_kill(processID);
```

Parametri

processID

L'identità del thread di connessione da terminare.

Note per l'utilizzo

Ogni connessione al server MySQL viene eseguita in un thread distinto. Per terminare una connessione, utilizza la procedura `mysql.rds_kill` e passa l'ID di thread di quella connessione. Per ottenere l'ID di thread, utilizza il comando MySQL SHOW PROCESSLIST.

Esempi

L'esempio seguente termina una connessione con l'ID di thread 4243:

```
CALL mysql.rds_kill(4243);
```

mysql.rds_kill_query

Termina una query in esecuzione sul server MySQL.

Sintassi

```
CALL mysql.rds_kill_query(processID);
```

Parametri

processID

L'identità del processo o del thread che esegue la query da terminare.

Note per l'utilizzo

Per arrestare una query in esecuzione nel server MySQL, utilizza la procedura `mysql_rds_kill_query` e invia l'ID di connessione del thread che sta eseguendo la query. La procedura termina quindi la connessione.

Per ottenere l'ID, esegui una query sulla tabella INFORMATION_SCHEMA PROCESSLIST MySQL o utilizza il comando MySQL SHOW PROCESSLIST. Il valore nella colonna ID da `SHOW PROCESSLIST` o `SELECT * FROM INFORMATION_SCHEMA.PROCESSLIST` è *processID*.

Esempi

L'esempio seguente arresta una query con l'ID di thread di query 230040:

```
CALL mysql.rds_kill_query(230040);
```

Replica delle transazioni tramite GTID

Le stored procedure seguenti controllano il modo in cui le transazioni vengono replicate utilizzando gli ID globali di transazione (GTID) con Aurora MySQL. Per informazioni su come utilizzare la replica basata su GTID con Aurora MySQL, consulta Utilizzo della replica basata su GTID.

Argomenti

- mysql.rds_assign_gtids_to_anonymous_transactions (Aurora MySQL versione 3)
- mysql.rds_gtid_purged (Aurora MySQL versione 3)
- mysql.rds_skip_transaction_with_gtid (Aurora MySQL versione 2 e 3)
- mysql.rds_start_replication_until_gtid (Aurora MySQL versione 3)

mysql.rds_assign_gtids_to_anonymous_transactions (Aurora MySQL versione 3)

Configura l'opzione `ASSIGN_GTIDS_TO_ANONYMOUS_TRANSACTIONS` dell'istruzione `CHANGE REPLICATION SOURCE TO`. Il canale di replica assegna così un GTID alle transazioni replicate che non ne hanno uno. In questo modo, è possibile eseguire la replica del log binario da un'origine che non utilizza la replica basata su GTID in una replica che lo usa. Per ulteriori informazioni, consulta MODIFICA ORIGINE REPLICA A istruzione e Replica da una fonte senza GTID a una replica con GTID nel Manuale di riferimento MySQL.

Sintassi

```
CALL mysql.rds_assign_gtids_to_anonymous_transactions(gtid_option);
```

Parametri

`gtid_option`

Valore di stringa. I valori consentiti sono `OFF`, `LOCAL` o un UUID specificato.

Note per l'utilizzo

Questa procedura ha lo stesso effetto del rilascio dell'istruzione `CHANGE REPLICATION SOURCE TO ASSIGN_GTIDS_TO_ANONYMOUS_TRANSACTIONS = gtid_option` nella community MySQL.

GTID deve essere rivolto a `ON` per *`gtid_option`* da impostare su `LOCAL` o un UUID specifico.

Il valore predefinito è `OFF`, il che significa che la funzione non viene utilizzata.

`LOCAL` assegna un GTID incluso l'UUID della replica (l'impostazione `server_uuid`).

Il passaggio di un parametro che è un UUID assegna un GTID che include l'UUID specificato, ad esempio l'impostazione `server_uuid` per il server di fonte di replica.

Esempi

Per disattivare questa funzione:

```
mysql> call mysql.rds_assign_gtids_to_anonymous_transactions('OFF');
+------------------------------------------------------------+
| Message  |
+------------------------------------------------------------+
| ASSIGN_GTIDS_TO_ANONYMOUS_TRANSACTIONS has been set to: OFF |
+------------------------------------------------------------+
1 row in set (0.07 sec)
```

Per utilizzare l'UUID della replica:

```
mysql> call mysql.rds_assign_gtids_to_anonymous_transactions('LOCAL');
+-------------------------------------------------------------+
| Message  |
+-------------------------------------------------------------+
| ASSIGN_GTIDS_TO_ANONYMOUS_TRANSACTIONS has been set to: LOCAL |
+-------------------------------------------------------------+
1 row in set (0.07 sec)
```

Per utilizzare un UUID specificato:

```
mysql> call mysql.rds_assign_gtids_to_anonymous_transactions('317a4760-
f3dd-3b74-8e45-0615ed29de0e');
+-----------------------------------------------------------------------------------------------
+
| Message |
+-----------------------------------------------------------------------------------------------
+
| ASSIGN_GTIDS_TO_ANONYMOUS_TRANSACTIONS has been set to: 317a4760-
f3dd-3b74-8e45-0615ed29de0e |
+-----------------------------------------------------------------------------------------------
+
1 row in set (0.07 sec)
```

mysql.rds_gtid_purged (Aurora MySQL versione 3)

Imposta il valore globale della variabile di sistema `gtid_purged` su un set di ID globali di transazione (GTID) specificato. La variabile di sistema `gtid_purged` è un set di GTID composto dai GTID di tutte le transazioni che sono state eseguite sul server, ma che non esistono in nessun file di log binario sul server.

Per consentire la compatibilità con MySQL 8.0, esistono due modi per impostare il valore di `gtid_purged`:

- Sostituire il valore di `gtid_purged` con il set di GTID specificato.
- Aggiungere il set di GTID specificato al set di GTID già contenuto in `gtid_purged`.

Sintassi

Per sostituire il valore di `gtid_purged` con il set di GTID specificato:

```
CALL mysql.rds_gtid_purged (gtid_set);
```

Per aggiungere il valore di `gtid_purged` con al set di GTID specificato:

```
CALL mysql.rds_gtid_purged (+gtid_set);
```

Parametri

`gtid_set`

Il valore di *`gtid_set`* deve essere un superset del valore corrente di `gtid_purged` e non può intersecarsi con `gtid_subtract(gtid_executed,gtid_purged)`. In altre parole, il nuovo set di GTID deve includere tutti i GTID già presenti in `gtid_purged` e non può includere in `gtid_executed` alcun GTID che non è ancora stato rimosso. Inoltre, il parametro *`gtid_set`* non può includere gli eventuali GTID presenti nel set `gtid_owned` globale, i GTID per le transazioni attualmente in fase di elaborazione sul server.

Note per l'utilizzo

La procedura `mysql.rds_gtid_purged` deve essere eseguita dall'utente master.

Questa procedura è supportata per Aurora MySQL versione 3.04 e successive.

Esempi

L'esempio seguente assegna il GTID `3E11FA47-71CA-11E1-9E33-C80AA9429562:23` alla variabile globale `gtid_purged`.

```
CALL mysql.rds_gtid_purged('3E11FA47-71CA-11E1-9E33-C80AA9429562:23');
```

mysql.rds_skip_transaction_with_gtid (Aurora MySQL versione 2 e 3)

Ignora la replica di una transazione con l'ID globale di transazione (GTID) specificato in un'istanza primaria Aurora.

Puoi usare questa procedura per il ripristino di emergenza quando è noto che una specifica transazione GTID causa un problema. Usa questa stored procedure per saltare la transazione problematica. Esempi di transazioni problematiche includono le transazioni che disabilitano la replica, eliminano dati importanti o con le quali l'istanza database diventa non disponibile.

Sintassi

```
CALL mysql.rds_skip_transaction_with_gtid (
gtid_to_skip
);
```

Parametri

`gtid_to_skip`

GTID della transazione di replica da ignorare.

Note per l'utilizzo

La procedura `mysql.rds_skip_transaction_with_gtid` deve essere eseguita dall'utente master.

Questa procedura è supportata per Aurora MySQL versione 2 e 3.

Esempi

Nell'esempio seguente viene ignorata la replica della transazione con il GTID `3E11FA47-71CA-11E1-9E33-C80AA9429562:23`.

```
CALL mysql.rds_skip_transaction_with_gtid('3E11FA47-71CA-11E1-9E33-C80AA9429562:23');
```

mysql.rds_start_replication_until_gtid (Aurora MySQL versione 3)

Avvia la replica da un cluster di database Aurora MySQL e la arresta immediatamente dopo l'ID globale di transazione (GTID) specificato.

Sintassi

```
CALL mysql.rds_start_replication_until_gtid(gtid);
```

Parametri

gtid

Il GTID dopo il quale deve essere arrestata la replica.

Note per l'utilizzo

La procedura `mysql.rds_start_replication_until_gtid` deve essere eseguita dall'utente master.

Questa procedura è supportata per Aurora MySQL versione 3.04 e successive.

La stored procedure `mysql.rds_start_replication_until_gtid` non è supportata per la replica gestita, che include quanto segue:

- Repliche di cluster di database Amazon Aurora MySQL tra Regioni AWS
- Migrazione di dati da un'istanza database RDS per MySQL a un cluster database Amazon Aurora MySQL utilizzando una replica di lettura Aurora

Quando il parametro `gtid` specifica una transazione che è già stata eseguita dalla replica, la procedura viene arrestata immediatamente.

Esempi

L'esempio seguente avvia la replica e replica le modifiche finché non raggiunge il GTID `3E11FA47-71CA-11E1-9E33-C80AA9429562:23`.

```
call mysql.rds_start_replication_until_gtid('3E11FA47-71CA-11E1-9E33-C80AA9429562:23');
```

Rotazione dei log di query

Le seguenti stored procedure ruotano i log MySQL nelle tabelle di backup. Per ulteriori informazioni, consulta File di log del database Aurora MySQL.

Argomenti

- mysql.rds_rotate_general_log
- mysql.rds_rotate_slow_log

mysql.rds_rotate_general_log

Converte la tabella `mysql.general_log` in una tabella di backup.

Sintassi

```
CALL mysql.rds_rotate_general_log;
```

Note per l'utilizzo

Puoi convertire la tabella `mysql.general_log` in una tabella di backup chiamando la procedura `mysql.rds_rotate_general_log`. Quando le tabelle di log sono convertite, la tabella di log corrente è copiata in una tabella di logo di backup e le voci nella tabella di log corrente sono eliminate. Se una tabella di log di backup esiste, viene eliminata prima che la tabella di log corrente sia copiata nel backup. Puoi eseguire una query sulla tabella di log di backup, se necessario. La tabella di log di backup per la tabella `mysql.general_log` è denominata `mysql.general_log_backup`.

È possibile eseguire questa procedura solo quando il parametro `log_output` è impostato su `TABLE`.

mysql.rds_rotate_slow_log

Converte la tabella `mysql.slow_log` in una tabella di backup.

Sintassi

```
CALL mysql.rds_rotate_slow_log;
```

Note per l’utilizzo

Puoi convertire la tabella `mysql.slow_log` in una tabella di backup chiamando la procedura `mysql.rds_rotate_slow_log`. Quando le tabelle di log sono convertite, la tabella di log corrente è copiata in una tabella di logo di backup e le voci nella tabella di log corrente sono eliminate. Se una tabella di log di backup esiste, viene eliminata prima che la tabella di log corrente sia copiata nel backup.

Puoi eseguire una query sulla tabella di log di backup, se necessario. La tabella di log di backup per la tabella `mysql.slow_log` è denominata `mysql.slow_log_backup`.

Impostazione e visualizzazione della configurazione dei log binari

Le seguenti stored procedure impostano e mostrano i parametri di configurazione, ad esempio per la conservazione dei file di log binari.

Argomenti

- mysql.rds_set_configuration
- mysql.rds_show_configuration

mysql.rds_set_configuration

Specifica il numero di ore di conservazione dei log binari o il numero di secondi di ritardo della replica.

Sintassi

```
CALL mysql.rds_set_configuration(name,value);
```

Parametri

name

Il nome del parametro di configurazione da impostare.

value

Il valore del parametro di configurazione.

Note per l'utilizzo

la procedura archiviata `mysql.rds_set_configuration` supporta i parametri di configurazione seguenti:

- binlog retention hours

I parametri di configurazione vengono archiviati in modo permanente e restano effettivi dopo qualsiasi riavvio o failover dell'istanza database.

binlog retention hours

Il parametro `binlog retention hours` viene utilizzato per specificare il numero di ore di conservazione dei file di log binari. Amazon Aurora elimina in genere un log binario non appena possibile, tuttavia il log potrebbe continuare a essere necessario per la replica con un database MySQL esterno ad Aurora.

Il valore predefinito di `binlog retention hours` è `NULL`. Per Aurora MySQL, `NULL` significa che i log binari vengono eliminati lentamente. I log binari Aurora MySQL potrebbero rimanere nel sistema per un certo periodo, di solito non più di un giorno.

Per specificare il numero di ore per mantenere i log binari in un cluster di database, usa la stored procedure `mysql.rds_set_configuration` e specifica un periodo con tempo sufficiente per l'esecuzione della replica, come mostrato nell'esempio seguente.

```
call mysql.rds_set_configuration('binlog retention hours', 24);
```

> **Note**
>
> Non puoi utilizzare il valore `0` per `binlog retention hours`.

Il valore `binlog retention hours` massimo per i cluster database Aurora MySQL versione 2.11.0 e successive e versione 3 è 2160 (90 giorni).

Dopo l'impostazione del periodo di retention, monitora l'utilizzo dello storage per l'istanza database per verificare che i log binari conservati non occupino troppo spazio di storage.

mysql.rds_show_configuration

Il numero di ore di retention dei log binari.

Sintassi

```
CALL mysql.rds_show_configuration;
```

Note per l'utilizzo

Per verificare il numero di ore per cui Amazon RDS deve conservare i log binari, usa la stored procedure `mysql.rds_show_configuration`.

Esempi

L'esempio seguente visualizza il periodo di retention:

```
call mysql.rds_show_configuration;
                name                          value     description
                binlog retention hours        24        binlog retention hours specifies
 the duration in hours before binary logs are automatically deleted.
```

Tabelle information_schema specifiche di Aurora MySQL

Aurora MySQL dispone di determinate tabelle `information_schema` che sono specifiche di Aurora.

information_schema.aurora_global_db_instance_status

La tabella `information_schema.aurora_global_db_instance_status` contiene informazioni sullo stato di tutte le istanze database nei cluster database primario e secondario di un database globale. Nella tabella seguente vengono visualizzate le colonne che è possibile utilizzare. Le colonne rimanenti sono solo per uso interno di Aurora.

> **Note**
>
> Questa tabella dello schema informativo è disponibile solo con Aurora MySQL 3.04.0 e versioni successive.

Colonna	Tipo di dati	Descrizione
SERVER_ID	varchar(100)	Identificatore istanza database.
SESSION_ID	varchar(100)	Un identificatore univoco per la sessione corrente. Il valore di `MASTER_SESSION_ID` identifica l'istanza database di lettura (primaria).
AWS_REGION	varchar(100)	La Regione AWS in cui viene eseguita questa istanza

Colonna	Tipo di dati	Descrizione
		database globale. Per l'elenco delle regioni, consulta Disponibilità nelle regioni.
DUREVOLE_LSN	bigint unsigned	Il numero di sequenza di log (LSN) reso durevole nello storage. Numero di sequenza di log (LSN) è un numero sequenziale univoco che identifica un record nel log delle transazioni del database. Gli LSN sono ordinati in modo tale che un LSN più grande rappresenti una transazione successiva.
HIGHEST_LSN_RCVD	bigint unsigned	L'LSN più alto ricevuto dall'istanza database dall'istanza database di scrittura.
OLDEST_READ_VIEW_TRX_ID	bigint unsigned	L'ID della transazione più vecchia che può essere rimossa dall'istanza database di scrittura.
OLDEST_READ_VIEW_LSN	bigint unsigned	L'LSN più vecchio utilizzato dall'istanza database per le operazioni di lettura dallo storage.

Colonna	Tipo di dati	Descrizione
VISIBILITY_LAG_IN_MSEC	float(10,0) unsigned	Per le istanze di lettura nel cluster database primario, il ritardo in millisecondi di questa istanza database rispetto all'istanza database di scrittura . Per istanze di lettura in un cluster database secondario, il ritardo in millisecondi di questa istanza database rispetto al volume secondario.

information_schema.aurora_global_db_status

La tabella `information_schema.aurora_global_db_status` contiene informazioni su vari aspetti del ritardo del database globale Aurora, in particolare il ritardo dello storage Aurora sottostante (chiamato ritardo di durabilità) e il ritardo tra l'obiettivo del punto di ripristino (RPO). Nella tabella seguente vengono visualizzate le colonne che è possibile utilizzare. Le colonne rimanenti sono solo per uso interno di Aurora.

> **Note**
>
> Questa tabella dello schema informativo è disponibile solo con Aurora MySQL 3.04.0 e versioni successive.

Colonna	Tipo di dati	Descrizione
AWS_REGION	varchar(100)	La Regione AWS in cui viene eseguita questa istanza database globale. Per l'elenco delle regioni, consulta Disponibilità nelle regioni.
HIGHEST_LSN_WRITTEN	bigint unsigned	Il numero di sequenza di log (LSN) più alto attualmen

Colonna	Tipo di dati	Descrizione
		te esistente in questo cluster database. Numero di sequenza di log (LSN) è un numero sequenziale univoco che identifica un record nel log delle transazioni del database. Gli LSN sono ordinati in modo tale che un LSN più grande rappresenti una transazione successiva.
DURABILITY_LAG_IN_MILLISECONDS	float(10,0) unsigned	La differenza nei valori di timestamp tra il parametro `HIGHEST_LSN_WRITTEN` su un cluster database secondario e il parametro `HIGHEST_LSN_WRITTEN` sul cluster database primario. Questo valore è sempre 0 sul cluster database primario del database globale Aurora.

Colonna	Tipo di dati	Descrizione
RPO_LAG_IN_MILLISECONDS	float(10,0) unsigned	Il ritardo dell'obiettivo del punto di ripristino (RPO). Il ritardo dell'obiettivo del punto di ripristino (RPO) è il tempo necessario per memorizzare il COMMIT delle transazioni utente più recenti dopo la sua memorizzazione nel cluster di database primario del database globale Aurora. Questo valore è sempre 0 sul cluster database primario del database globale Aurora. In sintesi, questo parametro calcola l'obiettivo del punto di ripristino per ciascun cluster database Aurora MySQL nel database globale Aurora, ovvero quanti dati potrebbero andare perduti in caso di interruzione. Come per il ritardo, l'obiettivo del punto di ripristino (RPO) viene misurato nel tempo.

Colonna	Tipo di dati	Descrizione
LAST_LAG_CALCULATION_TIMESTAMP	datetime	Il timestamp che specifica quando sono stati calcolati i valori per `DURABILITY_LAG_IN_MILLISECONDS` e `RPO_LAG_IN_MILLISECONDS`. Il valore temporale `1970-01-01 00:00:00+00` indica che questo è il cluster di database primario.
OLDEST_READ_VIEW_TRX_ID	bigint unsigned	L'ID della transazione più vecchia che può essere rimossa dall'istanza database di scrittura.

information_schema.replica_host_status

La tabella `information_schema.replica_host_status` contiene informazioni sulla replica. Le colonne che è possibile utilizzare sono mostrate nella tabella seguente. Le colonne rimanenti sono solo per uso interno di Aurora.

Colonna	Tipo di dati	Descrizione
CPU	double	La percentuale di utilizzo della CPU dell'host di replica.
IS_CURRENT	tinyint	Se la replica è aggiornata.
LAST_UPDATE_TIMESTAMP	datetime(6)	L'ora dell'ultimo aggiornamento. Utilizzata per determinare se un record è obsoleto.
REPLICA_LAG_IN_MILLISECONDS	double	Il ritardo di replica in millisecondi.

Colonna	Tipo di dati	Descrizione
SERVER_ID	varchar(100)	L'ID del server di database.
SESSION_ID	varchar(100)	L'ID della sessione di database. Utilizzato per determinare se un'istanz a database è un'istanza di scrittura o di lettura.

> **Note**
>
> Quando un'istanza di replica rimane indietro, le informazioni interrogate dalla relativa tabelle `information_schema.replica_host_status` potrebbero essere obsolete. In questo caso, ti consigliamo di eseguire una query dall'istanza si scrittura.
> Sebbene la tabella `mysql.ro_replica_status` contenga informazioni simili, ti consigliamo di non utilizzarla.

information_schema.aurora_forwarding_processlist

La tabella `information_schema.aurora_forwarding_processlist` contiene informazioni sui processi coinvolti nell'inoltro di scrittura.

I contenuti di questa tabella sono visibili solo sull'istanza database di scrittura per un cluster database con l'inoltro di scrittura globale o interno al cluster attivato. Viene restituito un set di risultati vuoto sulle istanze database di lettura.

Campo	Tipo di dati	Descrizione
ID	bigint	L'identificatore della connessione sull'istanza DB di scrittura . Questo identificatore è lo stesso valore visualizzato nella colonna `Id`dell'istruzione `SHOW PROCESSLIST` e restituita dalla funzione `CONNECTION_ID()` all'interno del thread.

Campo	Tipo di dati	Descrizione
UTENTE	varchar(32)	L'utente MySQL che ha eseguito l'istruzione.
HOST	varchar(255)	Il client MySQL che ha eseguito l'istruzione. Per le istruzioni inoltrate, questo campo mostra l'indirizzo host del client dell'applicazione che ha stabilito la connessione sull'istanza database di lettura di inoltro.
DB	varchar(64)	Il database predefinito per il thread.
COMMAND	varchar(16)	Il tipo di comando eseguito dal thread per conto del client o `Sleep` se lo stato della sessione è inattivo. Per le descrizioni dei comandi di thread, consulta la documentazione MySQL su Thread Command Values nella documentazione MySQL.
TIME	int	Il tempo in secondi in cui il thread è rimasto nello stato corrente.
STATE	varchar(64)	Un'azione, un evento o uno stato che indica cosa sta facendo il thread. Per le descrizioni dei valori di stato, consulta Stati generali del thread nella documentazione di MySQL.
INFO	longtext	L'istruzione eseguita dal thread o `NULL` se non sta eseguendo un'istruzione. L'istruzione può essere quella inviata al server o un'istruzione più interna se l' istruzione esegue altre istruzioni.
IS_FORWARDED	bigint	Indica se il thread viene inoltrato da un'istanza database di lettura.

Campo	Tipo di dati	Descrizione
REPLICA_SESSION_ID	bigint	L'identificatore di connessione sulla replica di Aurora. Questo identificatore è lo stesso valore visualizzato nella colonna `Id`dell'istruzione `SHOW PROCESSLIST` sull'istanza database di lettura Aurora di inoltro.
REPLICA_INSTANCE_IDENTIFIER	varchar(64)	L'identificatore dell'istanza database del thread di inoltro.
REPLICA_CLUSTER_NAME	varchar(64)	L'identificatore cluster database del thread di inoltro. Per l'inoltro di scrittura all'interno del cluster, questo identificatore è lo stesso cluster database dell'istanza database di scrittura.
REPLICA_REGION	varchar(64)	La Regione AWS di origine del thread di inoltro. Per l'inoltro di scrittura all'interno del cluster, questa regione è la stessa Regione AWS dell'istanza database di scrittura.

Aggiornamenti del motore del database per Amazon Aurora MySQL

Gli aggiornamenti di Amazon Aurora vengono resi disponibili periodicamente. Gli aggiornamenti vengono applicati ai cluster di database di Aurora durante le finestre di manutenzione del sistema. I tempi per l'applicazione degli aggiornamenti dipendono dalla regione e dall'impostazione della finestra di manutenzione del cluster del DB, ma anche dal tipo di aggiornamento.

Le versioni di Amazon Aurora sono rese disponibili per tutte le Regioni AWSnel corso di più giorni. Alcune Regioni possono mostrare temporaneamente una versione del motore che non è ancora disponibile in una Regione diversa.

Gli aggiornamenti vengono applicati a tutte le istanze in un cluster di database contemporaneamente. Un aggiornamento richiede un riavvio del database per tutte le istanze di un cluster di database, pertanto c'è un tempo di inattività di 20-30 secondi, al termine del quale si potranno ricominciare a usare i cluster di database. Puoi visualizzare o modificare le impostazioni della finestra di manutenzione dalla Console di gestione AWS.

Per informazioni dettagliate sulle versioni di Aurora MySQL supportate da Amazon Aurora, consultare le Note di rilascio di Aurora MySQL.

Di seguito viene illustrato come scegliere la versione di Aurora MySQL corretta per il tuo cluster, come specificare la versione quando crei o aggiorni un cluster e le procedure per aggiornare un cluster da una versione all'altra con un'interruzione minima.

Argomenti

- Verifica dei numeri di versione di Aurora MySQL
- Versioni beta e con supporto a lungo termine (LTS, Long-Term Support) di Amazon Aurora MySQL
- Preparazione per la fine del supporto standard per Amazon Aurora edizione compatibile con MySQL versione 2
- Preparazione per la fine del ciclo di vita di Amazon Aurora edizione compatibile con MySQL versione 1
- Aggiornamento dei cluster database Amazon Aurora MySQL
- Aggiornamenti e correzioni per il motore del database per Amazon Aurora MySQL

Verifica dei numeri di versione di Aurora MySQL

Anche se Aurora edizione compatibile con MySQL è compatibile con i motori di database MySQL, Aurora MySQL include funzionalità e correzioni di bug specifici per versioni particolari di Aurora MySQL. Gli sviluppatori di applicazioni possono controllare la versione di Aurora MySQL nelle loro applicazioni utilizzando SQL. Gli amministratori di database possono controllare e specificare le versioni di Aurora MySQL durante la creazione o l'aggiornamento di cluster di database e istanze database Aurora MySQL.

Argomenti

- Verifica o specifica delle versioni del motore Aurora MySQL tramite AWS
- Controllo delle versioni Aurora MySQL utilizzando SQL

Verifica o specifica delle versioni del motore Aurora MySQL tramite AWS

Quando esegui attività amministrative utilizzando la Console di gestione AWS, la AWS CLI o l'API RDS devi specificare la versione di Aurora MySQL in un formato alfanumerico descrittivo.

A partire da Aurora MySQL versione 2, le versioni del motore Aurora hanno la seguente sintassi.

```
mysql-major-version.mysql_aurora.aurora-mysql-version
```

La parte *mysql-major-version-* è `5.7` o `8.0`. Questo valore rappresenta la versione del protocollo client e il livello generale del supporto della funzionalità MySQL per la versione di Aurora MySQL corrispondente.

Il valore *aurora-mysql-version* è tratteggiato in tre parti: la versione principale di Aurora MySQL, la versione secondaria di Aurora MySQL e il livello di patch. La versione principale è `2` o `3`. Questi valori rappresentano Aurora MySQL compatibile con MySQL 5.7 o 8.0 rispettivamente. La versione secondaria rappresenta la versione di funzionalità della serie 2.x o 3.x. Il livello di patch inizia da `0` per ogni versione secondaria e rappresenta l'insieme successivo di correzioni di bug che si applicano alla versione secondaria. Occasionalmente, una nuova funzionalità viene incorporata in una versione secondaria ma non resa immediatamente visibile. In questi casi, la funzionalità è sottoposta a ottimizzazione e viene resa pubblica in un livello di patch successivo.

Tutte le versioni 2.x del motore Aurora MySQL sono compatibili con Community MySQL versione 5.7.12 o successive. Tutte le versioni 3.x del motore Aurora MySQL sono compatibili con MySQL

versione 8.0.23 o successive. È possibile fare riferimento alle note di rilascio della versione 3.x specifica per conoscere la versione compatibile con MySQL corrispondente.

Ad esempio, le versioni del motore per Aurora MySQL 3.04.0 e 2.11.2 sono le seguenti.

```
8.0.mysql_aurora.3.04.0
5.7.mysql_aurora.2.11.2
```

> **Note**
>
> Non esiste una corrispondenza uno-a-uno tra le versioni MySQL della community e le versioni Aurora MySQL 2.x. Per la versione 3 di Aurora MySQL esiste una mappatura più diretta. Per verificare quali sono le correzioni di bug e le nuove funzionalità di una particolare versione di Aurora MySQL, consulta Aggiornamenti del motore del database per Amazon Aurora MySQL versione 3 e Aggiornamenti del motore del database Amazon Aurora MySQL versione 2 nelle Note di rilascio di Aurora MySQL. Per l'elenco cronologico delle nuove funzionalità e versioni, consulta Cronologia dei documenti. Per verificare la versione minima richiesta per una correzione relativa alla sicurezza, consultare Vulnerabilità di sicurezza risolte in Aurora MySQLnelle Note di rilascio di Aurora MySQL.

La versione del motore Aurora MySQL si specifica in alcuni comandi della AWS CLI e operazioni API RDS. Ad esempio, specifichi l'opzione `--engine-version` quando esegui i comandi AWS CLI create-db-cluster e modify-db-cluster. Specifichi il parametro `EngineVersion` quando esegui le operazioni API RDS CreateDBCluster e ModifyDBCluster.

In Aurora MySQL versione 2, la versione del motore nella Console di gestione AWS include anche la versione di Aurora. L'aggiornamento del cluster modifica il valore visualizzato. Questa modifica ti consente di specificare e controllare le versioni Aurora MySQL precise, senza la necessità di connettersi al cluster o eseguire comandi SQL.

> **Tip**
>
> Per i cluster Aurora gestiti tramite CloudFormation, questa modifica nell'impostazione `EngineVersion` può attivare le operazioni di CloudFormation. Per informazioni su come CloudFormation gestisce le modifiche all'impostazione `EngineVersion`, vedi la documentazione di CloudFormation.

Controllo delle versioni Aurora MySQL utilizzando SQL

I numeri di versione Aurora che è possibile recuperare nell'applicazione utilizzando query SQL utilizzano il formato *<major version>*.*<minor version>*.*<patch version>*. Puoi ottenere questo numero di versione per qualsiasi istanza database nel cluster Aurora MySQL eseguendo una query sulla variabile di sistema `AURORA_VERSION`. Per ottenere questo numero di versione puoi utilizzare una delle seguenti query.

```
select aurora_version();
select @@aurora_version;
```

Le query producono un output simile al seguente.

```
mysql> select aurora_version(), @@aurora_version;
+------------------+------------------+
| aurora_version() | @@aurora_version |
+------------------+------------------+
| 3.05.2           | 3.05.2           |
+------------------+------------------+
```

I numeri di versione restituiti dalla console, dalla CLI e dall'API RDS utilizzando le tecniche descritte in Verifica o specifica delle versioni del motore Aurora MySQL tramite AWS sono in genere più descrittivi.

Versioni beta e con supporto a lungo termine (LTS, Long-Term Support) di Amazon Aurora MySQL

Aurora MySQL fornisce versioni beta e con supporto a lungo termine (LTS) per alcune versioni del motore Aurora MySQL.

Argomenti

- Versioni con supporto a lungo termine (Long-Term Support, LTS) di Aurora MySQL
- Note di rilascio della versione beta di Aurora MySQL

Versioni con supporto a lungo termine (Long-Term Support, LTS) di Aurora MySQL

Ogni nuova versione di Aurora MySQL resta disponibile per l'utilizzo per un certo periodo di tempo quando crei o aggiorni un cluster di database. Trascorso questo periodo, devi aggiornare i cluster che

utilizzano questa versione. Puoi aggiornare manualmente il cluster prima della scadenza del periodo di supporto oppure lasciare che Aurora lo faccia al posto tuo quando la versione di Aurora MySQL non è più supportata.

Aurora indica alcune versioni di Aurora MySQL come versioni con supporto a lungo termine (LTS). I cluster di database che utilizzano le versioni LTS possono restare più a lungo con la stessa versione ed essere sottoposti a un numero minore di cicli di aggiornamento rispetto ai cluster che utilizzano versioni non LTS. I cluster di database che utilizzano le versioni LTS possono restare con la stessa versione secondaria per almeno tre anni o fino alla fine del supporto standard per la versione principale, a seconda di quale evento si verifica per primo. Quando un cluster di database con una versione LTS deve essere aggiornato, Aurora applica la versione LTS successiva. In questo modo, l'intervallo di tempo prima del nuovo aggiornamento del cluster diventa più lungo.

Nel corso della durata di una versione LTS di Aurora MySQL, nuovi livelli di patch introducono correzioni a problemi importanti. Questi livelli di patch non comprendono nuove funzionalità. Puoi scegliere se applicare queste patch ai cluster database che eseguono la versione LTS. Ai clienti che eseguono versioni LTS si consiglia di eseguire l'aggiornamento alla versione patch più recente della versione secondaria LTS almeno una volta all'anno per sfruttare le correzioni operative e di sicurezza di elevata gravità. Per alcune correzioni critiche, Amazon potrebbe eseguire un aggiornamento gestito a un livello di patch all'interno della stessa versione LTS. Tali aggiornamenti gestiti vengono eseguiti automaticamente nella finestra di manutenzione del cluster. Tutte le versioni di Aurora MySQL (sia LTS che non LTS) sono sottoposte a numerosi test operativi e di stabilità. Alcune versioni secondarie sono designate come versioni LTS per consentire ai clienti di utilizzare tali versioni secondarie più a lungo senza passare a una versione secondaria più recente.

Per la maggior parte dei cluster Aurora MySQL, consigliamo di eseguire l'aggiornamento all'ultima versione invece di utilizzare la versione LTS. In questo modo, puoi sfruttare Aurora anche come servizio gestito e puoi accedere alle ultime funzionalità e correzioni di bug. Le versioni LTS sono destinate a cluster con le seguenti caratteristiche:

- L'applicazione Aurora MySQL non può avere tempi di inattività dovuti agli aggiornamenti, ad eccezione di rari casi per la distribuzione di patch critiche.
- Il ciclo di test per il cluster e le applicazioni associate richiede molto tempo per ogni aggiornamento al motore del database di Aurora MySQL.
- La versione del database per il cluster Aurora MySQL ha tutte le funzionalità del motore del database e le correzioni di bug necessarie per l'applicazione.

Le versioni LTS correnti per Aurora MySQL sono le seguenti:

- Aurora MySQL versione 3.10.*.
- Aurora MySQL versione 3.04.*.

Per ulteriori informazioni sulla versione LTS consulta Aggiornamenti del motore del database per Amazon Aurora MySQL versione 3 nelle Note di rilascio di Aurora MySQL.

> **Note**
>
> Si consiglia di disabilitare gli aggiornamenti automatici a versioni secondarie per le versioni LTS. Imposta il parametro `AutoMinorVersionUpgrade` su `false` o deseleziona la casella di controllo Abilita aggiornamento automatico versione secondaria nella Console di gestione AWS.
> Se non disabiliti l'opzione, il tuo cluster di database potrebbe essere aggiornato a una versione non LTS.

Note di rilascio della versione beta di Aurora MySQL

Una versione beta di Aurora MySQL è una correzione di sicurezza anticipata, disponibile solo in un numero limitato di. Regioni AWS Queste correzioni vengono in seguito implementate in modo più ampio in tutte le Regioni con la successiva versione patch.

La numerazione di una versione beta è simile a quella di una versione secondaria di Aurora MySQL, ma con l'aggiunta di una quarta cifra, ad esempio 2.12.0.1 o 3.05.0.1.

Per maggiori informazioni, consulta Aggiornamenti del motore del database per Amazon Aurora MySQL versione 2 e Aggiornamenti del motore del database per Amazon Aurora MySQL versione 3 nelle Note di rilascio per Aurora MySQL.

Preparazione per la fine del supporto standard per Amazon Aurora edizione compatibile con MySQL versione 2

La fine del supporto standard per Amazon Aurora edizione compatibile con MySQL versione 2 (compatibile con MySQL 5.7) è pianificata per il 31 ottobre 2024. Si consiglia di aggiornare tutti i cluster che eseguono Aurora MySQL versione 2 alla versione predefinita di Aurora MySQL versione

3 (compatibile con MySQL 8.0) o successive prima che la versione 2 di Aurora MySQL raggiunga la fine del periodo di supporto standard. Il 31 ottobre 2024, Amazon RDS registrerà automaticamente i database nel supporto esteso per Amazon RDS. Questa modalità di aggiornamento non si applica se si ha in esecuzione Amazon Aurora MySQL versione 2 (compatibile con MySQL 5.7) in un cluster Aurora Serverless versione 1. Per aggiornare i cluster Aurora Serverless versione 1 ad Aurora MySQL versione 3, consulta Percorso di aggiornamento per i cluster di database Aurora Serverless v1.

Le successive date di fine del supporto per le versioni principali di Aurora MySQL sono disponibili nel calendario dei rilasci per le versioni principali di Aurora MySQL.

Se i cluster eseguono Aurora MySQL versione 2, riceverai notifiche periodiche con le informazioni più recenti su come effettuare un aggiornamento nel momento in cui la data di fine del supporto standard si avvicina. Aggiorneremo periodicamente questa pagina con le informazioni più recenti.

Tempistiche della fine del supporto standard

1. Da ora fino al 31 ottobre 2024 potrai aggiornare i cluster Aurora MySQL versione 2 (con compatibilità con MySQL 5.7) ad Aurora MySQL versione 3 (con compatibilità con MySQL 8.0).
2. 31 ottobre 2024: in questa data termina il supporto standard per Aurora MySQL versione 2 e Amazon RDS registra automaticamente i cluster nel supporto esteso per Amazon RDS.

Verrà effettuata la registrazione automatica nel supporto esteso per RDS. Per ulteriori informazioni, consulta Supporto esteso di Amazon RDS con Amazon Aurora.

Individuazione di cluster interessati da questo processo di fine del ciclo di vita

Per individuare i cluster interessati da questo processo di fine del ciclo di vita, utilizza le seguenti procedure.

> **Important**
>
> Assicurati di eseguire queste istruzioni in ogni Regione AWS e per ciascun Account AWS in cui sono presenti delle risorse.

Console

Per trovare un cluster Aurora MySQL versione 2

1. Accedi alla Console di gestione AWS e apri la console Amazon RDS all'indirizzo https://console.aws.amazon.com/rds/.
2. Nel riquadro di navigazione, scegliere Databases (Database).
3. Nella casella Filtra per database, inserisci 5.7.
4. Controlla Aurora MySQL nella colonna del motore.

AWS CLI

Per trovare i cluster interessati da questo processo di fine del ciclo di vita utilizzando la AWS CLI, chiama il comando describe-db-clusters. È possibile utilizzare il seguente script di esempio.

Example

```
aws rds describe-db-clusters --include-share --query 'DBClusters[?(Engine==`aurora-
mysql` && contains(EngineVersion,`5.7.mysql_aurora`))].{EngineVersion:EngineVersion,
 DBClusterIdentifier:DBClusterIdentifier, EngineMode:EngineMode}' --output table
 --region us-east-1

        +-----------------------------------------------------------------+
        |                       DescribeDBClusters                        |
        +---------------------+---------------+-------------------------+
        |         DBCI        |      EM       |          EV             |
        +---------------------+---------------+-------------------------+
        |    aurora-mysql2    |  provisioned  | 5.7.mysql_aurora.2.11.3 |
        | aurora-serverlessv1 |   serverless  | 5.7.mysql_aurora.2.11.3 |
        +---------------------+---------------+-------------------------+
```

API RDS

Per trovare i cluster di database Aurora MySQL che eseguono Aurora MySQL versione 2, utilizza l'operazione API RDS DescribeDBClusters con i parametri obbligatori seguenti:

- `DescribeDBClusters`
  - Filters.Filter.N

- Nome
  - engine
- Values.Value.N
  - ['aurora']

Estensione del supporto per Amazon RDS

Puoi utilizzare il supporto esteso per Amazon RDS tramite Community MySQL 5.7 gratuitamente fino alla data di fine del supporto, ossia il 31 ottobre 2024. Il 31 ottobre 2024, Amazon RDS registra automaticamente i database nel supporto esteso per RDS per Aurora MySQL versione 2. Il supporto esteso per RDS per Aurora è un servizio a pagamento che offre fino a 28 mesi di supporto aggiuntivo per Aurora MySQL versione 2, fino al termine del supporto esteso per RDS, ossia febbraio 2027. L'estensione del supporto RDS sarà offerta solo per le versioni secondarie di Aurora MySQL 2.11 e 2.12. Per utilizzare Amazon Aurora MySQL versione 2 oltre la fine del supporto standard, pianifica l'esecuzione dei database su una di queste versioni secondarie prima del 31 ottobre 2024.

Per ulteriori informazioni sul supporto esteso per RDS, ad esempio costi e altre considerazioni, consulta Supporto esteso di Amazon RDS con Amazon Aurora.

Esecuzione di un aggiornamento

L'aggiornamento tra le versioni principali richiede una pianificazione e test più estesi rispetto all'aggiornamento a una versione secondaria. Il processo può richiedere un tempo considerevole. Consideriamo l'aggiornamento come un processo in tre fasi, con attività da svolgere prima, per e dopo l'aggiornamento.

Prima dell'aggiornamento:

Prima dell'aggiornamento, si consiglia di verificare la compatibilità, le prestazioni, le procedure di manutenzione delle applicazioni nel cluster aggiornato e altre considerazioni simili, al fine di confermare che, dopo l'aggiornamento, le applicazioni funzionino come previsto. I cinque consigli seguenti ti aiuteranno a ottenere un'esperienza di aggiornamento migliore.

- Innanzitutto, è fondamentale comprendere Come funziona l'aggiornamento della versione principale di Aurora MySQL in loco.
- Quindi, esplora le tecniche di aggiornamento disponibili quando Aggiornamento da Aurora MySQL versione 2 alla versione 3.

- Per aiutarti a decidere il momento e l'approccio giusti per l'aggiornamento, puoi apprendere le differenze tra Aurora MySQL versione 3 e il tuo attuale ambiente con Confronto tra Aurora MySQL versione 2 e Aurora MySQL versione 3.
- Dopo aver scelto l'opzione più comoda ed efficace, prova un aggiornamento simulato in loco su un cluster clonato, utilizzando Pianificazione di un aggiornamento della versione principale per un cluster Aurora MySQL.
- Rivedere le Controlli preliminari per gli aggiornamenti a versioni principali per Aurora MySQL. Esegui il controllo preliminare dell'aggiornamento per determinare se l'aggiornamento del database può essere effettuato correttamente e se sussistono problemi successivi all'aggiornamento in termini di compatibilità con le applicazioni, prestazioni, procedure di manutenzione e altre considerazioni simili.
- Non tutti i tipi o le versioni di cluster Aurora MySQL possono utilizzare il meccanismo di aggiornamento in loco. Per ulteriori informazioni, consulta Procedure di aggiornamento a una versione principale di Aurora MySQL.

In caso di domande o dubbi, il team di supporto di AWS è disponibile sui forum della community e tramite Premium Support.

Esecuzione dell'aggiornamento:

Puoi utilizzare una delle seguenti tecniche di aggiornamento: Il tempo di inattività del sistema dipende dalla tecnica scelta.

- Implementazioni blu/verdi: per le situazioni in cui la massima priorità è ridurre i tempi di inattività, puoi utilizzare le implementazioni blu/verdi Amazon RDS per eseguire l'aggiornamento della versione principale nei cluster di database Amazon Aurora con provisioning. Un'implementazione blu/verde crea un ambiente di gestione temporanea che copia l'ambiente di produzione. È possibile apportare alcune modifiche al cluster di database Aurora nell'ambiente verde (gestione temporanea) senza influire sui carichi di lavoro di produzione. Lo switchover richiede in genere meno di un minuto e non comporta perdita di dati. Per ulteriori informazioni, consulta Panoramica delle implementazioni di Amazon Aurora Blue/Green . Questa tecnica riduce al minimo i tempi di inattività, ma comporta l'utilizzo di risorse aggiuntive durante l'esecuzione dell'aggiornamento.
- Aggiornamenti in loco: utilizzando un aggiornamento in loco, Aurora esegue automaticamente un processo di controllo preliminare, disconnette il cluster, effettua il backup del cluster, completa l'aggiornamento e riporta il cluster online. L'esecuzione di un aggiornamento in loco di una versione principale può essere completata in pochi clic e non richiede altri interventi di coordinamento o

failover con altri cluster, ma comporta tempi di inattività. Per ulteriori informazioni, consulta Come eseguire un aggiornamento in loco

- Ripristino tramite snapshot: permette di aggiornare il cluster Aurora MySQL versione 2 tramite il ripristino a partire da uno snapshot di Aurora MySQL versione 2 in un cluster Aurora MySQL versione 3. A tale scopo, devi seguire la procedura di acquisizione di uno snapshot ed eseguire il ripristino a partire da esso. Poiché stai eseguendo il ripristino da uno snapshot, questo processo comporta un'interruzione del database.

Dopo l'aggiornamento:

Dopo l'aggiornamento dovrai monitorare attentamente il sistema (applicazioni e database) e, se necessario, apportare modifiche tramite fine-tuning. Seguendo attentamente le fasi di pre-aggiornamento, si ridurranno al minimo le modifiche necessarie. Per ulteriori informazioni, consulta Risoluzione dei problemi alle prestazioni del database Amazon Aurora MySQL.

Per ulteriori informazioni sui metodi, sulla pianificazione, sui test e sulla risoluzione dei problemi degli aggiornamenti delle versioni principali di Aurora MySQL, leggi attentamente Aggiornamento della versione principale di un cluster di database Amazon Aurora MySQL, inclusa la sezione su Risoluzione dei problemi relativi all'aggiornamento in loco di Aurora MySQL. È importante sottolineare che Aurora MySQL versione 3 non supporta alcuni tipi di istanza. Per ulteriori informazioni, consulta Classi di istanze database Amazon Aurora.

Percorso di aggiornamento per i cluster di database Aurora Serverless v1

L'aggiornamento tra le versioni principali richiede una pianificazione e test più estesi rispetto all'aggiornamento a una versione secondaria. Il processo può richiedere un tempo considerevole. Consideriamo l'aggiornamento come un processo in tre fasi, con attività da svolgere prima, per e dopo l'aggiornamento.

Aurora MySQL versione 2 (compatibile con MySQL 5.7) continuerà a ricevere il supporto standard per i cluster Aurora Serverless v1.

Se desideri effettuare l'aggiornamento ad Amazon Aurora MySQL 3 (con compatibilità con MySQL 8.0) e continuare a utilizzare Aurora Serverless, usa Amazon Aurora Serverless v2. Per comprendere le differenze tra Aurora Serverless v1 e Aurora Serverless v2, consulta Confronto tra Aurora Serverless v2 e Aurora Serverless v1.

Esegui l'aggiornamento a Aurora Serverless v2: puoi aggiornare un cluster Aurora Serverless v1 a Aurora Serverless v2. Per ulteriori informazioni, consulta Aggiornamento da un cluster Aurora Serverless v1 ad Aurora Serverless v2.

Preparazione per la fine del ciclo di vita di Amazon Aurora edizione compatibile con MySQL versione 1

Amazon Aurora edizione compatibile con MySQL versione 1 (con la compatibilità MySQL 5.6) dovrebbe raggiungere la fine del ciclo di vita il 28 febbraio 2023. Amazon consiglia di aggiornare tutti i cluster (provisionati e Aurora Serverless) che eseguono Aurora MySQL versione 1 ad Aurora MySQL versione 2 (con la compatibilità a MySQL 5.7) o ad Aurora MySQL versione 3 (con la compatibilità a MySQL 8.0). Effettua questa operazione prima che Aurora MySQL versione 1 raggiunga la fine del periodo di supporto.

Per i cluster di database Aurora con provisioning, è possibile completare gli aggiornamenti da Aurora MySQL versione 1 ad Aurora MySQL versione 2 in modi diversi. Le istruzioni per il meccanismo di aggiornamento in loco sono disponibili in Come eseguire un aggiornamento in loco. Un altro modo per completare l'aggiornamento è quello di acquisire uno snapshot di un cluster Aurora MySQL versione 1 e ripristinarlo in un cluster Aurora MySQL versione 2. Oppure è possibile seguire un processo in più fasi che esegue i cluster vecchi e nuovi affiancati. Per ulteriori dettagli su ciascun metodo, consulta Aggiornamento della versione principale di un cluster di database Amazon Aurora MySQL.

Per i cluster database Aurora Serverless v1, puoi eseguire un aggiornamento in loco da Aurora MySQL versione 1 ad Aurora MySQL versione 2. Per informazioni dettagliate su questo metodo, consulta Modifica di un cluster di database Aurora Serverless v1

Per i cluster database con provisioning Aurora, puoi completare gli aggiornamenti da Aurora MySQL versione 1 ad Aurora MySQL versione 3 utilizzando un processo di aggiornamento in due fasi:

1. Esegui l'aggiornamento da Aurora MySQL versione 1 ad Aurora MySQL versione 2 utilizzando i metodi descritti in precedenza.
2. Esegui l'aggiornamento da Aurora MySQL versione 2 ad Aurora MySQL versione 3 utilizzando gli stessi metodi utilizzati per l'aggiornamento da versione 1 a versione 2. Per ulteriori dettagli, consulta Aggiornamento da Aurora MySQL versione 2 alla versione 3. Prendi nota di Differenze di funzionalità tra Aurora MySQL versione 2 e 3.

Le prossime date di fine vita per le versioni principali di Aurora sono disponibili in Versioni di Amazon Aurora. Amazon aggiorna automaticamente i cluster non aggiornati prima della data di fine vita. Dopo la data di fine vita, questi aggiornamenti automatici alla versione principale successiva si verificano durante una finestra di manutenzione pianificata per i cluster.

Di seguito sono riportati i milestone aggiuntivi per l'aggiornamento dei cluster (provisionati e Aurora Serverless) di Aurora MySQL versione 1 che stanno raggiungendo la fine del ciclo di vita. Per ognuno di essi, l'ora di inizio è 00:00 Universal Coordinated Time (UTC).

1. Da ora fino al 28 febbraio 2023: è possibile avviare in qualsiasi momento gli aggiornamenti dei cluster Aurora MySQL versione 1 (con compatibilità MySQL 5.6) ad Aurora MySQL versione 2 (con compatibilità MySQL 5.7). Da Aurora MySQL versione 2, è possibile completare l'aggiornamento ad Aurora MySQL versione 3 (con compatibilità MySQL 8.0) per i cluster di database Aurora con provisioning.
2. 16 gennaio 2023: dopo questa data, non è possibile creare nuovi cluster o istanze Aurora MySQL versione 1 dalla Console di gestione AWS o da AWS Command Line Interface (AWS CLI). Inoltre, non è possibile aggiungere nuove regioni secondarie a un database globale Aurora. Ciò potrebbe influire sulla capacità di recupero da un'interruzione non pianificata, come descritto in Ripristino di un database globale Amazon Aurora da un'interruzione non pianificata, perché dopo questo periodo non è possibile completare i passaggi 5 e 6. Non sarà inoltre possibile creare una nuova replica di lettura tra Regioni con Aurora MySQL versione 1. È comunque possibile effettuare le seguenti operazioni per i cluster Aurora MySQL versione 1 esistenti fino al 28 febbraio 2023:
   - Ripristina lo snapshot acquisito di un cluster Aurora MySQL versione 1 nella stessa versione del cluster snapshot originale.
   - Aggiunta di repliche di lettura (non applicabile ai cluster di database Aurora Serverless).
   - Modificare la configurazione dell'istanza.
   - Effettuare il ripristino point-in-time.
   - Creare cloni di cluster della versione 1 esistenti.
   - Crea una nuova replica di lettura tra Regioni che esegue Aurora MySQL versione 2 o successiva.
3. 28 febbraio 2023: dopo questa data, prevediamo di aggiornare automaticamente i cluster Aurora MySQL versione 1 alla versione predefinita di Aurora MySQL versione 2 all'interno di una finestra di manutenzione pianificata riportata di seguito. Il ripristino degli snapshot DB di Aurora MySQL versione 1 comporta un aggiornamento automatico del cluster ripristinato alla versione di default di Aurora MySQL versione 2 in quel momento.

L'aggiornamento tra le versioni principali richiede una pianificazione e test più estesi rispetto all'aggiornamento a una versione secondaria. Il processo può richiedere un tempo considerevole.

Per le situazioni in cui la massima priorità è ridurre i tempi di inattività, puoi utilizzare le implementazioni blu/verdi per eseguire l'aggiornamento della versione principale nei cluster database Amazon Aurora allocati. Un'implementazione blu/verde crea un ambiente di gestione temporanea che copia l'ambiente di produzione. È possibile apportare modifiche al cluster database Aurora nell'ambiente verde (gestione temporanea) senza influire sui carichi di lavoro di produzione. Lo switchover richiede in genere meno di un minuto senza perdita di dati e senza la necessità di modificare l'applicazione. Per ulteriori informazioni, consulta Panoramica delle implementazioni di Amazon Aurora Blue/Green .

Al termine dell'aggiornamento, è possibile che sia necessario compiere ulteriori operazioni. Ad esempio, ciò potrebbe verificarsi a causa di differenze nella compatibilità SQL, del modo in cui funzionano alcune funzionalità relative a MySQL o alle impostazioni dei parametri tra la vecchia e la nuova versione.

Per ulteriori informazioni sui metodi, sulla pianificazione, sui test e sulla risoluzione dei problemi degli aggiornamenti delle versioni principali di Aurora MySQL, leggi attentamente Aggiornamento della versione principale di un cluster di database Amazon Aurora MySQL.

Individuazione di cluster interessati da questo processo di fine del ciclo di vita

Per individuare i cluster interessati da questo processo di fine del ciclo di vita, utilizza le seguenti procedure.

> **Important**
>
> Assicurati di eseguire queste istruzioni in ogni Regione AWS e per ciascun Account AWS in cui sono presenti delle risorse.

Console

Per trovare un cluster Aurora MySQL versione 1

1. Accedi alla Console di gestione AWS e apri la console Amazon RDS all'indirizzo https://console.aws.amazon.com/rds/.
2. Nel riquadro di navigazione, scegliere Databases (Database).
3. Nella casella Filter by databases (Filtra per database), inserisci 5.6.

4. Controlla Aurora MySQL nella colonna del motore.

AWS CLI

Per trovare i cluster interessati da questo processo di fine del ciclo di vita utilizzando la AWS CLI, chiama il comando describe-db-clusters. È possibile utilizzare il seguente script di esempio.

Example

```
aws rds describe-db-clusters --include-share --query 'DBClusters[?Engine==`aurora`].
{EV:EngineVersion, DBCI:DBClusterIdentifier, EM:EngineMode}' --output table --region
 us-east-1

        +-----------------------------------------+
        |            DescribeDBClusters           |
        +---------------+--------------+----------+
        |     DBCI      |     EM       |    EV    |
        +---------------+--------------+----------+
        |  my-database-1|  serverless  |  5.6.10a |
        +---------------+--------------+----------+
```

API RDS

Per trovare i cluster di database Aurora MySQL che eseguono Aurora MySQL versione 1, utilizza l'operazione API RDS DescribeDBClusters con i parametri obbligatori seguenti:

- `DescribeDBClusters`
  - Filters.Filter.N
    - Nome
      - engine
    - Values.Value.N
      - ['aurora']

Aggiornamento dei cluster database Amazon Aurora MySQL

È possibile aggiornare un cluster database Aurora MySQL per ottenere correzioni di bug e nuove funzionalità Aurora MySQL o per passare a una versione completamente nuova del motore del database sottostante. Nelle sezioni seguenti viene illustrato come.

> **Note**
>
> Il tipo di aggiornamento che esegui dipende dalla quantità di tempo di inattività che puoi permetterti per il tuo cluster, dalla quantità di test di verifica che intendi fare, dall'importanza delle correzioni di bug specifiche o delle nuove funzionalità per il tuo caso d'uso e dalla previsione di eseguire frequenti piccoli aggiornamenti o aggiornamenti occasionali che saltano diversi versioni intermedie. Per ogni aggiornamento, è possibile modificare la versione principale, la versione secondaria e il livello di patch per il cluster. Se non hai familiarità con la distinzione tra versioni principali, versioni secondarie e livelli di patch di Aurora MySQL, consulta le informazioni di base in Verifica dei numeri di versione di Aurora MySQL.

> **Tip**
>
> È possibile ridurre al minimo i tempi di inattività necessari per l'aggiornamento di un cluster di database utilizzando un'implementazione blu/verde. Per ulteriori informazioni, consulta Utilizzo di Amazon Blue/Green Aurora Deployments per gli aggiornamenti del database.

Argomenti

- Aggiornamento della versione secondaria o del livello di patch di un cluster di database Aurora MySQL
- Aggiornamento della versione principale di un cluster di database Amazon Aurora MySQL

Aggiornamento della versione secondaria o del livello di patch di un cluster di database Aurora MySQL

Puoi utilizzare i seguenti metodi per aggiornare la versione secondaria di un cluster di database o per applicare le patch a un cluster di database:

- Aggiornamento di Aurora MySQL modificando la versione del motore (per Aurora MySQL versione 2 e 3)
- Abilitazione degli aggiornamenti automatici tra versioni secondarie di Aurora MySQL

Per informazioni su come l'applicazione di patch senza tempi di inattività può ridurre le interruzioni durante il processo di aggiornamento, consulta Utilizzo dell'applicazione di patch senza tempi di inattività.

Per informazioni sull'esecuzione di un aggiornamento a una versione secondaria per il cluster di database Aurora MySQL, consulta i seguenti argomenti.

Argomenti

- Operazioni preliminari all'aggiornamento a una versione secondaria
- Controlli preliminari per gli aggiornamenti a versioni secondarie per Aurora MySQL
- Aggiornamento di Aurora MySQL modificando la versione del motore
- Abilitazione degli aggiornamenti automatici tra versioni secondarie di Aurora MySQL
- Utilizzo dell'applicazione di patch senza tempi di inattività
- Tecnica alternativa di aggiornamento blu/verde

Operazioni preliminari all'aggiornamento a una versione secondaria

Si consiglia di eseguire le seguenti azioni per ridurre i tempi di inattività durante un aggiornamento a una versione secondaria:

- La manutenzione del cluster di database Aurora deve essere eseguita durante un periodo di traffico ridotto. Usa Approfondimenti sulle prestazioni per identificare questi periodi di tempo al fine di configurare correttamente le finestre di manutenzione. Per ulteriori informazioni su Approfondimenti sulle prestazioni, consulta Monitoraggio del carico DB con Performance Insights su Amazon RDS. Per ulteriori informazioni sulla finestra di manutenzione del cluster di database, consulta Impostazione della finestra di manutenzione preferita del cluster database.
- Utilizza AWS SDK che supportano il jitter e il backoff esponenziali come best practice. Per ulteriori informazioni, consulta Backoff esponenziale e jitter.

Controlli preliminari per gli aggiornamenti a versioni secondarie per Aurora MySQL

Quando avvii un aggiornamento a una versione secondaria, Amazon Aurora esegue automaticamente dei controlli preliminari.

Questi controlli preliminari sono obbligatori. Non puoi scegliere di saltarli. I controlli preliminari offrono i seguenti vantaggi:

- Ti consentono di evitare tempi di inattività non pianificati durante l'aggiornamento.
- Se sono presenti incompatibilità, Amazon Aurora impedisce l'aggiornamento e fornisce un log per ottenere informazioni sulle stesse. Puoi quindi utilizzare il log per preparare il database per l'aggiornamento riducendo le incompatibilità. Per informazioni dettagliate sulla rimozione di eventuali incompatibilità, consulta Preparazione dell'installazione per l'aggiornamento nella documentazione di MySQL.

I controlli preliminari vengono eseguiti prima dell'arresto dell'istanza database per l'aggiornamento, il che significa che non generano alcun tempo di inattività durante l'esecuzione. Se i controlli preliminari rilevano un'incompatibilità, Aurora annulla automaticamente l'aggiornamento prima che l'istanza database venga arrestata. Aurora genera anche un evento per l'incompatibilità. Per ulteriori informazioni sugli eventi di Amazon Aurora, consulta Utilizzo della notifica degli eventi di Amazon RDS.

Aurora memorizza informazioni dettagliate su ciascuna incompatibilità nel file di log `PrePatchCompatibility.log`. Nella maggior parte dei casi, la voce di log include un collegamento alla documentazione MySQL utile per correggere l'incompatibilità. Per ulteriori informazioni sulla visualizzazione dei file di log, consultare Visualizzazione ed elenco dei file di log del database.

A causa della natura dei controlli preliminari, questi analizzano gli oggetti nel database. Questa analisi comporta il consumo di risorse e incrementa il tempo di completamento dell'aggiornamento.

Aggiornamento di Aurora MySQL modificando la versione del motore

L'aggiornamento della versione secondaria di un cluster di database Aurora MySQL applica correzioni aggiuntive e nuove funzionalità a un cluster esistente.

Questo tipo di aggiornamento si applica ai cluster Aurora MySQL in cui la versione originale e la versione aggiornata sono entrambi nella versione principale di Aurora MySQL, ossia 2 o 3. Il processo è semplice e veloce perché non comporta alcuna conversione dei metadati di Aurora MySQL o la riorganizzazione dei dati della tabella.

Puoi eseguire questo tipo di aggiornamento modificando la versione del motore del cluster di database utilizzando la Console di gestione AWS, l'AWS CLI, o l'API RDS. Ad esempio, se nel cluster è in esecuzione Aurora MySQL 3.x, scegli una versione 3.x superiore.

Se si sta eseguendo un aggiornamento secondario su un Database globale Aurora, è necessario aggiornare tutti i cluster secondari prima di aggiornare il cluster primario.

> **Note**
>
> Per eseguire un aggiornamento della versione secondaria ad Aurora MySQL versione 3.04.* o successive, o alla versione 2.12*, utilizza la seguente procedura:
>
> 1. Rimuovi tutte le regioni secondarie dal cluster globale. Seguire la procedura riportata in Rimozione di un cluster da un database globale Amazon Aurora.
> 2. Aggiorna la versione del motore della regione primaria alla versione 3.04.* o successive o alla versione 2.12.*, in base alle esigenze. Seguire la procedura riportata in To modify the engine version of a DB cluster.
> 3. Aggiungi le regioni secondarie al cluster globale. Seguire la procedura riportata in Aggiunta di una Regione AWS a un database globale Amazon Aurora.

Per modificare la versione del motore di un cluster di database

- Utilizzando la console – Modificare le proprietà del cluster. Nella finestra Modify DB cluster (Modifica il cluster di database) cambia la versione del motore Aurora MySQL nella casella DB engine version (Versione motore database). Se non hai familiarità con la procedura generale per la modifica di un cluster, segui le istruzioni riportate in Modifica del cluster di database tramite la console, la CLI e l'API.
- Utilizzando la AWS CLI: richiamare il comando modify-db-cluster della AWS CLI e specificare il nome del cluster di database per l'opzione `--db-cluster-identifier` e la versione del motore per l'opzione `--engine-version`.

  Ad esempio, per eseguire l'aggiornamento ad Aurora MySQL versione 3.04.1, imposta l'opzione `--engine-version` su `8.0.mysql_aurora.3.04.1`. Specifica l'opzione `--apply-immediately` per aggiornare immediatamente la versione del motore per il cluster di database.
- Con l'API RDS – Chiama l'operazione API ModifyDBCluster e specifica il nome del cluster di database per il parametro `DBClusterIdentifier` e la versione del motore per il parametro `EngineVersion`. Imposta il parametro `ApplyImmediately` su `true` per aggiornare immediatamente la versione del motore per il cluster di database.

Abilitazione degli aggiornamenti automatici tra versioni secondarie di Aurora MySQL

Per un cluster di database Amazon Aurora MySQL, puoi specificare che Aurora aggiorni automaticamente il cluster di database alle nuove versioni secondarie. Puoi farlo impostando la

proprietà `AutoMinorVersionUpgrade` (Aggiornamento automatico versione secondaria nella Console di gestione AWS) del cluster di database.

Gli aggiornamenti automatici vengono eseguiti durante le finestre di manutenzione. Se le singole istanze database nel cluster database hanno finestre di manutenzione diverse dalla finestra di manutenzione del cluster, la finestra di manutenzione del cluster ha la precedenza.

L'aggiornamento automatico della versione secondaria non si applica ai seguenti tipi di cluster Aurora MySQL:

- Cluster che fanno parte di un database globale Aurora
- Cluster con repliche tra regioni

La durata dell'interruzione varia a seconda del carico di lavoro, della dimensione del cluster, della quantità di dati di log binari e se Aurora può utilizzare la funzionalità ZDP (zero-downtime patching). Aurora riavvia il cluster di database, pertanto potrebbe verificarsi un breve periodo di indisponibilità prima di riprendere l'utilizzo del cluster. In particolare, la quantità di dati dei log binari influisce sui tempi di ripristino. L'istanza database elabora i dati del log binario durante il ripristino. Pertanto un volume elevato di dati dei log binari aumenta il tempo di recupero.

> **Note**
>
> Aurora esegue gli aggiornamenti automatici se tutte le istanze database nel tuo cluster di database hanno l'impostazione `AutoMinorVersionUpgrade` abilitata. Per informazioni su come effettuare tale impostazione, e sul relativo funzionamento quando applicata a livello di cluster e istanza, consulta Aggiornamenti automatici delle versioni secondarie per i cluster di database Aurora.
>
> Quindi, se esiste un percorso di aggiornamento per le istanze del cluster di database a una versione secondaria del motore di database con l'opzione `AutoUpgrade` impostata su true, l'aggiornamento verrà eseguito. L'impostazione `AutoUpgrade` è dinamica e viene configurata da RDS.
>
> Gli aggiornamenti automatici delle versioni secondarie vengono eseguiti sulla versione secondaria predefinita.

È possibile utilizzare un comando CLI come il seguente per verificare lo stato dell'impostazione `AutoMinorVersionUpgrade` per tutte le istanze database nei cluster Aurora MySQL.

```
aws rds describe-db-instances \
  --query '*[].
{DBClusterIdentifier:DBClusterIdentifier,DBInstanceIdentifier:DBInstanceIdentifier,AutoMinorVer
```

Questo comando genera un output simile al seguente:

```
[
  {
      "DBInstanceIdentifier": "db-t2-medium-instance",
      "DBClusterIdentifier": "cluster-57-2020-06-03-6411",
      "AutoMinorVersionUpgrade": true
  },
  {
      "DBInstanceIdentifier": "db-t2-small-original-size",
      "DBClusterIdentifier": "cluster-57-2020-06-03-6411",
      "AutoMinorVersionUpgrade": false
  },
  {
      "DBInstanceIdentifier": "instance-2020-05-01-2332",
      "DBClusterIdentifier": "cluster-57-2020-05-01-4615",
      "AutoMinorVersionUpgrade": true
  },
... output omitted ...
```

In questo esempio, l'impostazione Abilita aggiornamento automatico versione secondaria è disattivata per il cluster di database `cluster-57-2020-06-03-6411`, perché è disattivata per una delle istanze database del cluster.

Utilizzo dell'applicazione di patch senza tempi di inattività

L'esecuzione di aggiornamenti per i cluster Aurora MySQL DB comporta la possibilità di un'interruzione quando il database viene spento e durante l'aggiornamento. Per impostazione predefinita, se si avvia l'aggiornamento mentre il database è occupato, si perdono tutte le connessioni e le transazioni elaborate dal cluster di database. Se si aspetta che il database sia inattivo per eseguire l'aggiornamento, potrebbe essere necessario attendere a lungo.

L'applicazione di patch senza tempi di inattività si basa sul miglior tentativo per preservare le connessioni client attraverso un aggiornamento Aurora MySQL. Se l'applicazione di patch senza tempi di inattività viene completata correttamente, le sessioni delle applicazioni vengono conservate e il motore del database si riavvia mentre l'aggiornamento è in corso. Il riavvio del motore del database può causare un calo del throughput della durata di alcuni secondi fino a circa un minuto.

ZDP non si applica a quanto segue:

- Patch e aggiornamenti del sistema operativo
- Aggiornamenti di una versione principale

ZDP è disponibile per tutte le versioni di Aurora MySQL e le classi di istanze database supportate.

ZDP non è supportato per i database globali Aurora Serverless v1 o Aurora.

> **Note**
>
> Consigliamo di utilizzare le classi di istanza database T solo per i server di sviluppo e test o altri server non di produzione. Per ulteriori informazioni sulle classi di istanza T, consulta Utilizzo delle classi di istanza T per lo sviluppo e i test.

È possibile visualizzare i parametri degli attributi importanti durante ZDP nel log degli errori MySQL. È inoltre possibile visualizzare informazioni su quando Aurora MySQL utilizza ZDP o sceglie di non utilizzare ZDP nella pagina Eventi nella Console di gestione AWS.

In Aurora MySQL, Aurora può eseguire una patch senza tempi di inattività indipendentemente dal fatto che la replica dei log binari sia abilitata o meno. Se la replica dei log binari è abilitata, Aurora MySQL elimina automaticamente la connessione alla destinazione binlog durante un'operazione di applicazione di patch senza tempi di inattività. Aurora MySQL si riconnette automaticamente alla destinazione binlog e riprende la replica al termine del riavvio.

ZDP funziona anche in combinazione con i miglioramenti del riavvio in Aurora MySQL. L'applicazione di patch all'istanza database scrittore applica automaticamente le patch ai lettori contemporaneamente. Dopo aver applicato la patch, Aurora ripristina le connessioni sia sulle istanze DB dello scrittore che del lettore.

L'applicazione di patch senza tempi di inattività potrebbe non essere completata correttamente nelle condizioni seguenti:

- Durante l'esecuzione di query o transazioni di lunga durata. Se Aurora è in grado di eseguire l’applicazione di patch senza tempi di inattività, qualsiasi transazione aperta è annullata, ma le relative connessioni vengono mantenute.

- Sono in uso tabelle temporanee, blocchi utente temporanei e blocchi tabella temporanei, per esempio durante l'esecuzione di istruzioni DDL (Data Definition Language). Aurora interrompe queste connessioni.
- Esistono modifiche ai parametri in attesa.

Se non è disponibile alcuna finestra temporale appropriata per l'esecuzione dell'applicazione di patch senza tempi di inattività a causa di una o più di queste condizioni, viene ripristinato il comportamento standard dell'applicazione delle patch.

Sebbene le connessioni rimangano intatte a seguito di un'operazione ZDP riuscita, alcune variabili e funzionalità vengono reinizializzate. I seguenti tipi di informazioni non vengono conservati tramite un riavvio causato da zero-downtime patching:

- Variabili globali. Aurora ripristina le variabili di sessione, ma non ripristina le variabili globali dopo il riavvio.
- Variabili di stato. In particolare, il valore di attività riportato dallo stato del motore viene ripristinato dopo un riavvio che utilizza i meccanismi ZDR o ZDP.
- `LAST_INSERT_ID`.
- Stato `auto_increment` in memoria per le tabelle. Lo stato di incremento automatico in memoria viene reinizializzato. Per ulteriori informazioni sui valori di incremento automatico, consulta il Manuale di riferimento di MySQL.
- Informazioni diagnostiche dalle tabelle `INFORMATION_SCHEMA` e `PERFORMANCE_SCHEMA`. Queste informazioni diagnostiche vengono visualizzate anche nell'output di comandi come `SHOW PROFILE` e `SHOW PROFILES`.

Le seguenti attività relative a zero-downtime restart sono riportate nella pagina Eventi:

- Tentativo di aggiornare il database senza tempi di inattività.
- Tentativo di aggiornare il database senza tempi di inattività terminato. L'evento riporta quanto tempo è durato il processo. L'evento segnala inoltre quante connessioni sono state mantenute durante il riavvio e quante connessioni sono state interrotte. È possibile consultare il registro degli errori del database per visualizzare ulteriori dettagli su ciò che è successo durante il riavvio.

Tecnica alternativa di aggiornamento blu/verde

In alcune situazioni, la priorità principale è eseguire il passaggio immediato dal cluster precedente a quello aggiornato. In tali situazioni, è possibile utilizzare un processo in più fasi che esegue i cluster vecchi e nuovi affiancati. Qui, i dati vengono replicati dal cluster precedente a quello nuovo fino a quando il nuovo cluster non prende il controllo. Per informazioni dettagliate, consultare Utilizzo di Amazon Blue/Green Aurora Deployments per gli aggiornamenti del database.

Aggiornamento della versione principale di un cluster di database Amazon Aurora MySQL

In un numero di versione di Aurora MySQL come 3.04.1, il 3 rappresenta la versione principale. Aurora MySQL versione 2 è compatibile con MySQL 5.7. Aurora MySQL versione 3 è compatibile con MySQL 8.0.

L'aggiornamento tra le versioni principali richiede una pianificazione e test più estesi rispetto all'aggiornamento a una versione secondaria. Il processo può richiedere un tempo considerevole. Al termine dell'aggiornamento, è possibile che sia necessario compiere ulteriori operazioni. Ad esempio, ciò potrebbe verificarsi a causa di differenze nella compatibilità SQL o della modalità di funzionamento di alcune caratteristiche correlate a MySQL. Oppure potrebbe verificarsi a causa delle diverse impostazioni dei parametri tra la vecchia e la nuova versione.

Indice

- Aggiornamento da Aurora MySQL versione 2 alla versione 3
- Procedure di aggiornamento a una versione principale di Aurora MySQL
- Come funziona l'aggiornamento della versione principale di Aurora MySQL in loco
- Pianificazione di un aggiornamento della versione principale per un cluster Aurora MySQL
  - Simulazione dell’aggiornamento mediante clonazione del cluster di database
  - Distribuzioni blu/verdi
- Controlli preliminari per gli aggiornamenti a versioni principali per Aurora MySQL
  - Processo di controllo preliminare per Aurora MySQL
  - Formato dei log dei controlli preliminari per Aurora MySQL
  - Esempi di output di log dei controlli preliminari per Aurora MySQL
  - Verifica preliminare delle prestazioni per Aurora MySQL
  - Riepilogo dei controlli preliminari per l’aggiornamento di Community MySQL
  - Riepilogo dei controlli preliminari per l’aggiornamento di Aurora MySQL

- Informazioni di riferimento per le descrizioni dei controlli preliminari per Aurora MySQL
  - Errori
    - Controlli preliminari MySQL che restituiscono errori
    - Controlli preliminari Aurora MySQL che restituiscono errori
  - Avvisi
    - Controlli preliminari MySQL che restituiscono avvisi
    - Controlli preliminari Aurora MySQL che restituiscono avvisi
  - Note
  - Errori, avvisi o notifiche
- Come eseguire un aggiornamento in loco
  - Come gli aggiornamenti in loco influiscono sui gruppi di parametri per un cluster
  - Modifiche delle proprietà del cluster tra versioni di Aurora MySQL
  - Principali aggiornamenti in loco per database globali
  - Aggiornamenti in loco per cluster di database con repliche di lettura tra Regioni
- Esercitazione sull'aggiornamento in loco di Aurora MySQL
- Individuazione delle cause degli errori di aggiornamento a una versione principale di Aurora MySQL
- Risoluzione dei problemi relativi all'aggiornamento in loco di Aurora MySQL
- Pulizia post-aggiornamento per Aurora MySQL versione 3
  - Indici spaziali

Aggiornamento da Aurora MySQL versione 2 alla versione 3

Se si dispone di un cluster compatibile con MySQL 5.7 e si desidera aggiornarlo a un cluster compatibile con MySQL 8.0, è possibile eseguire un processo di aggiornamento sul cluster stesso. Questo tipo di aggiornamento è un aggiornamento in loco, a differenza degli aggiornamenti che si eseguono creando un nuovo cluster. Questa tecnica mantiene lo stesso endpoint e altre caratteristiche del cluster. L'aggiornamento è relativamente veloce perché non richiede la copia di tutti i dati in un nuovo volume cluster. Questa stabilità consente di ridurre al minimo eventuali modifiche di configurazione nelle applicazioni. Inoltre, consente di ridurre la quantità di test per il cluster aggiornato. Questo perché il numero di istanze database e le relative classi di istanza rimangono identiche.

Il meccanismo di aggiornamento utilizzato comporta l'arresto del cluster di database durante l'operazione. Aurora esegue un arresto pulito e completa le operazioni in sospeso, come il rollback delle transazioni e l'eliminazione degli annullamenti. Per ulteriori informazioni, consulta Come funziona l'aggiornamento della versione principale di Aurora MySQL in loco.

Il metodo di aggiornamento locale è conveniente, in quanto è semplice da eseguire e riduce al minimo le modifiche alla configurazione delle applicazioni associate. Ad esempio, un aggiornamento in loco conserva gli endpoint e il set di istanze database per il cluster. Tuttavia, il tempo necessario per un aggiornamento in loco può variare a seconda delle proprietà dello schema e dell'attività del cluster. Pertanto, a seconda delle esigenze del cluster, puoi scegliere tra più tecniche di aggiornamento:

- Aggiornamento locale
- Implementazione blu/verde
- Ripristino da snapshot

> **Note**
>
> Se usi la AWS CLI o l'API RDS per il metodo di aggiornamento del ripristino dello snapshot, è necessario eseguire un'operazione successiva per creare un'istanza database di scrittura nel cluster di database ripristinato.

Per informazioni generali su Aurora MySQL versione 3 e sulle nuove funzionalità, consultare Aurora MySQL versione 3 compatibile con MySQL 8.0.

Per informazioni dettagliate sulla pianificazione di un aggiornamento, consultare Pianificazione di un aggiornamento della versione principale per un cluster Aurora MySQL e Come eseguire un aggiornamento in loco.

Procedure di aggiornamento a una versione principale di Aurora MySQL

Non tutti i tipi o le versioni di cluster Aurora MySQL possono utilizzare il meccanismo di aggiornamento in loco. È possibile conoscere la procedura di aggiornamento appropriata per ogni cluster Aurora MySQL consultando la tabella seguente.

Tipo di cluster di database Aurora MySQL	Può utilizzare l'aggiornamento in loco?	Azione
Cluster con provisioning Aurora MySQL, versione 2	Sì	L'aggiornamento in loco è supportato per i cluster Aurora MySQL compatibili con MySQL 5.7. Per informazioni sull'aggiornamento ad Aurora MySQL versione 3, consulta Pianificazione di un aggiornamento della versione principale per un cluster Aurora MySQL e Come eseguire un aggiornamento in loco.
Cluster con provisioning Aurora MySQL, versione 3	Non applicabile	Utilizza una procedura di aggiornamento della versione secondaria per eseguire l'aggiornamento tra le versioni di Aurora MySQL versione 3.
Aurora Serverless v1 cluster	Non applicabile	Aurora Serverless v1 è supportato solo per Aurora MySQL versione 2.
Aurora Serverless v2 cluster	Non applicabile	Aurora Serverless v2 è supportato solo per Aurora MySQL versione 3.
Cluster in un database globale Aurora	Sì	Per eseguire l'aggiornamento di Aurora MySQL versione 2 alla versione 3, segui la procedura per eseguire un aggiornamento locale per i cluster in un database globale Aurora. Esegui l'aggiornamento sul cluster globale. Aurora esegue l'aggiornamento del cluster primario e di tutti i cluster secondari nel database globale nello stesso momento. Se si utilizza la AWS CLI o l'API RDS, chiama il comando `modify-global-cluster` o l'operazione

Tipo di cluster di database Aurora MySQL	Può utilizzare l'aggiornamento in loco?	Azione
		`ModifyGlobalCluster` anziché `modify-db-cluster` o `ModifyDBCluster` . È possibile eseguire un aggiornamento in loco da Aurora MySQL versione 2 alla versione 3 solo se il parametro `lower_case_table_names` è impostato sull'opzione predefinita e il database globale viene riavviato. Per ulteriori informazioni, consulta Aggiornamenti di una versione principale.
Cluster di query parallele	Sì	È possibile eseguire l'aggiornamento locale.
Cluster che costituisce la destinazione della replica del log binario	Probabile	Se la replica del log binario proviene da un cluster Aurora MySQL, è possibile eseguire un aggiornamento locale. Non è possibile eseguire l'aggiornamento se la replica del log binario proviene da un'istanza MySQL RDS o da un'istanza database MySQL on-premise. In tal caso, è possibile eseguire l'aggiornamento utilizzando il meccanismo di ripristino degli snapshot.

Tipo di cluster di database Aurora MySQL	Può utilizzare l'aggiornamento in loco?	Azione
Cluster con zero istanze database	No	Utilizzando la AWS CLI o l'API RDS, è possibile creare un cluster Aurora MySQL senza istanze database collegate. Allo stesso modo, è inoltre possibile rimuovere tutte le istanze database da un cluster Aurora MySQL lasciando intatti i dati nel volume del cluster. Sebbene il cluster non abbia istanze database collegate, non è possibile eseguire l'aggiornamento in loco. Il meccanismo di aggiornamento richiede un'istanza di scrittura nel cluster per eseguire conversioni sulle tabelle di sistema, sui file di dati e così via. In questo caso, è necessario utilizzare la AWS CLI o l'API RDS per creare un'istanza di scrittura per il cluster. Dopodiché è possibile eseguire l'aggiornamento in loco.
Cluster con backtrack abilitato	Sì	È possibile eseguire un aggiornamento in loco per un cluster Aurora MySQL che utilizza la funzionalità backtrack. Tuttavia, dopo l'aggiornamento, non è possibile eseguire il backtrack del cluster a un momento precedente l'aggiornamento.

Come funziona l'aggiornamento della versione principale di Aurora MySQL in loco

Aurora MySQL esegue l'aggiornamento della versione principale come processo in più fasi. È possibile controllare lo stato corrente di un aggiornamento. Alcuni passaggi dell'aggiornamento forniscono anche informazioni sullo stato di avanzamento. All'inizio di ciascuna fase, Aurora MySQL registra un evento. È possibile esaminare gli eventi man mano che si verificano nella pagina Events (Eventi) della console RDS. Per maggiori informazioni sull'utilizzo degli eventi, consulta Utilizzo della notifica degli eventi di Amazon RDS.

> **Important**
>
> Una volta avviato, il processo viene eseguito fino a quando l'aggiornamento va a buon fine o non riesce. Non è possibile annullare l'aggiornamento mentre è in corso. Se l'aggiornamento non riesce, Aurora esegue il rollback di tutte le modifiche e il cluster mantiene la versione del motore, i metadati e così via precedenti.

Il processo di aggiornamento è costituito da tre fasi:

1. Prima di iniziare il processo di aggiornamento, Aurora esegue una serie di controlli preliminari. Il cluster continua a funzionare mentre Aurora esegue questi controlli. Ad esempio, il cluster non può avere transazioni XA nello stato preparato né avere istruzioni DDL (Data Definition Language) in corso di elaborazione. Ad esempio, potrebbe essere necessario arrestare le applicazioni che inviano determinati tipi di istruzioni SQL. Oppure si può semplicemente aspettare fino a quando alcune istruzioni di lunga durata non siano state completate. Dopodiché, si può riprovare a eseguire l'aggiornamento. Alcuni controlli verificano le condizioni che, pur non impedendo l'aggiornamento, potrebbero far sì che richieda molto tempo.

   Se Aurora rileva che le condizioni richieste non sono soddisfatte, modifica le condizioni identificate nei dettagli dell'evento. Segui le indicazioni riportate in Risoluzione dei problemi relativi all'aggiornamento in loco di Aurora MySQL. Se Aurora rileva condizioni che potrebbero rallentare l'aggiornamento, pianifica il monitoraggio dell'aggiornamento per un periodo prolungato.
2. Aurora mette il cluster offline. Quindi Aurora esegue un insieme di test analoghi a quelli della fase precedente, per confermare che non si sono presentati nuovi problemi durante il processo di arresto. Se, a questo punto, Aurora rileva condizioni che impediscono l'aggiornamento, Aurora annulla l'aggiornamento e riporta il cluster in linea. In questo caso, verifica quando le condizioni non sono più applicabili e riavvia l'aggiornamento.
3. Aurora crea uno snapshot del volume del cluster. Si supponga di scoprire problemi di compatibilità o di altro tipo di al termine dell'aggiornamento. Oppure si supponga di volere eseguire test utilizzando sia i cluster originali sia quelli aggiornati. In questi casi, è possibile eseguire il ripristino da questa snapshot per creare un nuovo cluster con la versione originale del motore e i dati originali.

> **Tip**
>
> Questo snapshot è di tipo manuale. Tuttavia, Aurora può crearlo e continuare con il processo di aggiornamento anche se è stata raggiunta la quota per gli snapshot manuali. Questo snapshot rimane permanentemente (se necessario) fino a quando non viene eliminato. Al termine di tutti i test successivi all'aggiornamento, è possibile eliminare questo snapshot per ridurre al minimo i costi di storage.

4. Aurora clona il volume del cluster. La clonazione è un'operazione veloce che non comporta la copia dei dati effettivi della tabella. Se Aurora riscontra un problema durante l'aggiornamento, ripristina i dati originali dal volume del cluster clonato e riporta il cluster in linea. Il volume temporaneo clonato durante l'aggiornamento non è soggetto al normale limite del numero di cloni per un singolo volume cluster.
5. Aurora esegue un arresto pulito per l'istanza database di scrittura. Durante l'arresto pulito, gli eventi di avanzamento vengono registrati ogni 15 minuti per le operazioni seguenti. È possibile esaminare gli eventi man mano che si verificano nella pagina Events (Eventi) della console RDS.
   - Aurora elimina i record di annullamento per le versioni precedenti delle righe.
   - Aurora esegue il rollback di tutte le transazioni non salvate.
6. Aurora aggiorna la versione del motore sull'istanza database di scrittura:
   - Aurora installa il binario per la nuova versione del motore nell'istanza database di scrittura.
   - Aurora utilizza l'istanza database di scrittura per aggiornare i dati in formato compatibile con MySQL 5.7. Durante questa fase, Aurora modifica le tabelle di sistema ed esegue altre conversioni che influiscono sui dati nel volume del cluster. In particolare, Aurora aggiorna i metadati delle partizioni nelle tabelle di sistema affinché siano compatibili con il formato di partizione MySQL 5.7. Questa fase può richiedere molto tempo se le tabelle del cluster dispongono di un numero elevato di partizioni.

     Se durante questa fase si verificano degli errori, è possibile trovare i dettagli nei log degli errori di MySQL. Dopo l'avvio di questa fase, se il processo di aggiornamento non riesce per qualsiasi motivo, Aurora ripristina i dati originali dal volume cluster clonato.
7. Aurora aggiorna la versione del motore sulle istanze database di lettura.
8. Il processo di aggiornamento è completato. Aurora registra un evento finale per indicare che il processo di aggiornamento è stato completato correttamente. Ora il cluster di database sta eseguendo la nuova versione principale.

Pianificazione di un aggiornamento della versione principale per un cluster Aurora MySQL

Per aiutarti a decidere il momento e l'approccio giusti per l'aggiornamento, puoi apprendere le differenze tra Aurora MySQL versione 3 e il tuo attuale ambiente:

- Se si esegue la conversione da RDS per MySQL 8.0 o MySQL 8.0 Community Edition, consulta Confronto tra Aurora MySQL versione 3 e MySQL 8.0 Community Edition.
- Se si esegue l'aggiornamento da Aurora MySQL versione 2, RDS per MySQL 5.7 o dalla versione Community di MySQL 5.7, consulta Confronto tra Aurora MySQL versione 2 e Aurora MySQL versione 3.
- Creare nuove versioni compatibili con MySQL 8.0 di qualsiasi gruppo di parametri personalizzati. Applicare i valori dei parametri personalizzati necessari ai nuovi gruppi di parametri. Consultare Modifiche ai parametri per Aurora MySQL versione 3 per conoscere le modifiche ai parametri.
- Esaminare lo schema del database Aurora MySQL versione 2 e le definizioni di oggetto per l'utilizzo di nuove parole chiave riservate introdotte in MySQL 8.0 Community Edition. Eseguire questa operazione prima di effettuare l'aggiornamento. Per ulteriori informazioni, consulta MySQL 8.0 New Keywords and Reserved Words nella documentazione di MySQL.

Puoi inoltre trovare ulteriori suggerimenti e considerazioni sull'aggiornamento specifici di MySQL in Changes in MySQL 8.0 (Modifiche in MySQL 8.0)nel Manuale di riferimento MySQL. Per esempio, è possibile utilizzare il comando `mysqlcheck --check-upgrade` per analizzare i database Aurora MySQL esistenti e identificare potenziali problemi di aggiornamento.

> **Note**
>
> Ti consigliamo di utilizzare classi di istanza database più grandi durante l'aggiornamento ad Aurora MySQL versione 3 utilizzando la tecnica di aggiornamento locale o di ripristino da snapshot. Esempi sono db.r5.24xlarge e db.r6g.16xlarge. Ciò consente di completare il processo di aggiornamento più rapidamente utilizzando la maggior parte della capacità della CPU disponibile sull'istanza database. Al termine dell'aggiornamento della versione principale, puoi passare alla classe di istanza database desiderata.

Dopo aver terminato l'aggiornamento stesso, è possibile seguire le procedure di post-aggiornamento in Pulizia post-aggiornamento per Aurora MySQL versione 3. Infine, testare la funzionalità e le prestazioni dell'applicazione.

Se si sta convertendo da RDS per MySQL o dalla versione Community di MySQL, segui la procedura di migrazione spiegata in Migrazione di dati a un cluster di database Amazon Aurora MySQL. In alcuni casi, è possibile utilizzare la replica del log binario per sincronizzare i dati con un cluster Aurora MySQL versione 3 come parte della migrazione. In tal caso, il sistema di origine deve eseguire una versione compatibile con il cluster di database di destinazione.

Per assicurarsi che le applicazioni e le procedure di amministrazione funzionino correttamente dopo l'aggiornamento di un cluster tra versioni principali, eseguire parte della pianificazione e della preparazione in anticipo. Per vedere quali tipi di codice di gestione aggiornare per gli script della AWS CLI o le applicazioni basate su API RDS, consultare Come gli aggiornamenti in loco influiscono sui gruppi di parametri per un cluster. Consultare anche Modifiche delle proprietà del cluster tra versioni di Aurora MySQL.

Per informazioni sui problemi che si possono verificare durante l'aggiornamento, consulta Risoluzione dei problemi relativi all'aggiornamento in loco di Aurora MySQL. In caso di problemi che potrebbero richiedere un lungo periodo di tempo per completare l'aggiornamento, è possibile verificare tali condizioni in anticipo e correggerle.

> **Note**
>
> L'aggiornamento in loco comporta l'arresto del cluster di database durante l'operazione. Aurora MySQL esegue un arresto pulito e completa le operazioni in sospeso, come l'eliminazione delle operazioni di undo. Un aggiornamento potrebbe richiedere molto tempo se ci sono molti record di undo da eliminare. Si consiglia di eseguire l'aggiornamento solo quando la lunghezza dell'elenco della cronologia (HLL) si riduce. Un valore generalmente accettabile per HLL è pari o inferiore a 100.000. Per ulteriori informazioni, consulta questo post del blog.

Simulazione dell'aggiornamento mediante clonazione del cluster di database

È possibile verificare la compatibilità dell'applicazione, le prestazioni, le procedure di manutenzione e considerazioni simili per il cluster aggiornato. A questo scopo, è possibile eseguire una simulazione dell'aggiornamento prima di eseguire l'aggiornamento reale. Questa tecnica può essere particolarmente utile per i cluster di produzione. In questo caso, è importante ridurre al minimo i tempi di inattività e disporre del cluster aggiornato pronto all'uso non appena l'aggiornamento è terminato.

Utilizza le fasi seguenti:

1. Crea un clone del cluster originale. Segui la procedura riportata in Clonazione di un volume per un cluster di database Amazon Aurora.
2. Imposta un insieme di istanze database di scrittura e di lettura analogo a quello del cluster originale.
3. Esegui un aggiornamento in loco del cluster clonato. Segui la procedura riportata in Come eseguire un aggiornamento in loco.

   Avvia l'aggiornamento immediatamente dopo aver creato il clone. In questo modo, il volume del cluster è ancora identico allo stato del cluster originale. Se il clone rimane inattivo prima di eseguire l'aggiornamento, Aurora esegue i processi di pulizia del database in background. In tal caso, l'aggiornamento del clone non è una simulazione accurata dell'aggiornamento del cluster originale.
4. Verifica la compatibilità e le prestazioni delle applicazioni, le procedure di amministrazione e così via, utilizzando il cluster clonato.
5. Se riscontri problemi, modifica i tuoi piani di aggiornamento per tenerne conto. Ad esempio, adatta qualsiasi codice dell'applicazione affinché sia compatibile con il set di caratteristiche della versione superiore. Stima il tempo necessario per l'aggiornamento in base alla quantità di dati nel cluster. È inoltre possibile scegliere di pianificare l'aggiornamento per un momento in cui il cluster non è occupato.
6. Dopo avere verificato che le applicazioni e il carico di lavoro funzionano correttamente con il cluster di test, puoi eseguire l'aggiornamento locale per il cluster di produzione.
7. Agisci per ridurre al minimo il tempo di inattività totale del cluster durante un aggiornamento della versione principale. A questo scopo, assicurati che il carico di lavoro sul cluster sia basso o zero al momento dell'aggiornamento. In particolare, assicurati che non vi siano transazioni di lunga durata in corso all'avvio dell'aggiornamento.

Distribuzioni blu/verdi

In alcune situazioni, la priorità principale è eseguire il passaggio immediato dal cluster precedente a quello aggiornato. In tali situazioni, è possibile utilizzare un processo in più fasi che esegue i cluster vecchi e nuovi affiancati. Qui, i dati vengono replicati dal cluster precedente a quello nuovo fino a quando il nuovo cluster non prende il controllo. Per informazioni dettagliate, consultare Utilizzo di Amazon Blue/Green Aurora Deployments per gli aggiornamenti del database.

Controlli preliminari per gli aggiornamenti a versioni principali per Aurora MySQL

L'aggiornamento di MySQL da una versione principale a un'altra, ad esempio da MySQL 5.7 a MySQL 8.0, comporta alcune significative modifiche architetturali che richiedono un'attenta pianificazione e preparazione. A differenza degli aggiornamenti a versioni secondarie, in cui l'attenzione si concentra principalmente sull'aggiornamento del software del motore di database e in alcuni casi sulle tabelle di sistema, gli aggiornamenti principali di MySQL spesso introducono modifiche fondamentali al modo in cui il database archivia e gestisce i propri metadati.

Per aiutarti a identificare tali incompatibilità, durante l'aggiornamento da Aurora MySQL versione 2 alla versione 3, Aurora esegue automaticamente controlli di compatibilità degli aggiornamenti (controlli preliminari) per esaminare gli oggetti nel cluster di database e identificare le incompatibilità note che possono impedire il proseguimento dell'aggiornamento. Per informazioni dettagliate sui controlli preliminari di Aurora MySQL, consulta Informazioni di riferimento per le descrizioni dei controlli preliminari per Aurora MySQL. I controlli preliminari di Aurora vengono eseguiti in aggiunta a quelli eseguiti dall'utilità di controllo aggiornamenti di Community MySQL.

Questi controlli preliminari sono obbligatori. Non puoi scegliere di saltarli. I controlli preliminari offrono i seguenti vantaggi:

- Possono ridurre la possibilità di incorrere in errori di aggiornamento che possono portare a tempi di inattività prolungati.
- Se sono presenti incompatibilità, Amazon Aurora impedisce la prosecuzione dell'aggiornamento e fornisce un log per ottenere informazioni sulle stesse. Puoi quindi utilizzare il log per preparare il database per l'aggiornamento alla versione 3 risolvendo le incompatibilità. Per informazioni dettagliate sulla risoluzione delle incompatibilità, consulta l'argomento relativo alla preparazione dell'installazione per l'aggiornamento nella documentazione di MySQL e il post relativo alle informazioni sull'aggiornamento a MySQL 8.0 di MySQL 8.0 nel blog di MySQL Server.

  Per ulteriori informazioni sull'aggiornamento a MySQL 8.0, consulta Upgrading MySQL nella documentazione di MySQL.

I controlli preliminari vengono eseguiti prima che il cluster di database venga messo offline per l'aggiornamento alla versione principale. Se i controlli preliminari rilevano un'incompatibilità, Aurora annulla automaticamente l'aggiornamento prima che l'istanza database venga arrestata. Aurora genera anche un evento per l'incompatibilità. Per ulteriori informazioni sugli eventi di Amazon Aurora, consulta Utilizzo della notifica degli eventi di Amazon RDS.

Una volta completati i controlli preliminari, Aurora memorizza le informazioni dettagliate su ciascuna incompatibilità nel file `upgrade-prechecks.log`. Nella maggior parte dei casi, la voce di log include un collegamento alla documentazione MySQL utile per correggere l'incompatibilità. Per ulteriori informazioni sulla visualizzazione dei file di log, consultare Visualizzazione ed elenco dei file di log del database.

> **Note**
>
> A causa della natura dei controlli preliminari, questi analizzano gli oggetti nel database. Questa analisi comporta il consumo di risorse e incrementa il tempo di completamento dell'aggiornamento. Per ulteriori informazioni sulle considerazioni relative alle prestazioni del controllo preliminare, consulta Processo di controllo preliminare per Aurora MySQL.

Indice

- Processo di controllo preliminare per Aurora MySQL
- Formato dei log dei controlli preliminari per Aurora MySQL
- Esempi di output di log dei controlli preliminari per Aurora MySQL
- Verifica preliminare delle prestazioni per Aurora MySQL
- Riepilogo dei controlli preliminari per l'aggiornamento di Community MySQL
- Riepilogo dei controlli preliminari per l'aggiornamento di Aurora MySQL
- Informazioni di riferimento per le descrizioni dei controlli preliminari per Aurora MySQL
  - Errori
    - Controlli preliminari MySQL che restituiscono errori
    - Controlli preliminari Aurora MySQL che restituiscono errori
  - Avvisi
    - Controlli preliminari MySQL che restituiscono avvisi
    - Controlli preliminari Aurora MySQL che restituiscono avvisi
  - Note
  - Errori, avvisi o notifiche

Processo di controllo preliminare per Aurora MySQL

Come descritto in precedenza, il processo di aggiornamento di Aurora MySQL prevede l'esecuzione di controlli di compatibilità (controlli preliminari) sul database prima di procedere all'aggiornamento a una versione principale.

Per gli aggiornamenti in loco, i controlli preliminari vengono eseguiti sull'istanza database di scrittura mentre è online. Se il controllo preliminare ha esito positivo, l'aggiornamento procede. Se vengono rilevati errori, vengono registrati i relativi log nel file `upgrade-prechecks.log` e l'aggiornamento viene annullato. Prima di tentare nuovamente l'aggiornamento, risolvi gli eventuali errori restituiti nel file `upgrade-prechecks.log`.

Per gli aggiornamenti con ripristino tramite snapshot, il controllo preliminare viene eseguito durante il processo di ripristino. In caso di esito positivo, il database verrà aggiornato alla nuova versione di Aurora MySQL. Se vengono rilevati errori, vengono registrati i relativi log nel file `upgrade-prechecks.log` e l'aggiornamento viene annullato. Prima di tentare nuovamente l'aggiornamento, risolvi gli eventuali errori restituiti nel file `upgrade-prechecks.log`.

Per ulteriori informazioni, consultare Individuazione delle cause degli errori di aggiornamento a una versione principale di Aurora MySQL e Informazioni di riferimento per le descrizioni dei controlli preliminari per Aurora MySQL.

Per monitorare lo stato del controllo preliminare, puoi visualizzare i seguenti eventi nel cluster di database.

Controllo preliminare dello stato	Messaggio di evento	Azione
Avviato	Preparazione dell'aggiornamento in corso: avvio dei controlli preliminari di aggiornamento online.	Nessuno
Non riuscito	Il cluster di database si trova in uno stato che non consente l'aggiornamento: esito negativo dei controlli preliminari di aggiornamento.	Controlla `upgrade-prechecks.log` per verificare la presenza di eventuali errori. Risolvi gli errori.

Controllo preliminare dello stato	Messaggio di evento	Azione
	Per ulteriori dettagli, consulta il file upgrade-prechecks.log. Per ulteriori informazioni sulla risoluzione della causa dell'errore di aggiornamento, consulta https://docs.aws.amazon.com/AmazonRDS/latest/AuroraUserGuide/AuroraMySQL.Upgrading.Troubleshooting.html	Tenta di eseguire nuovamente l'aggiornamento.
Riuscito	Preparazione dell'aggiornamento in corso: completamento dei controlli preliminari di aggiornamento online.	Il controllo preliminare ha avuto esito positivo e non sono stati restituiti errori. Controlla `upgrade-prechecks.log` per verificare la presenza di eventuali avvisi e notifiche.

Per ulteriori informazioni sulla visualizzazione degli eventi, consulta Visualizzazione di eventi Amazon RDS.

Formato dei log dei controlli preliminari per Aurora MySQL

Una volta completati i controlli di compatibilità dell'aggiornamento (controlli preliminari), puoi esaminare il file `upgrade-prechecks.log`. Il file di log contiene i risultati, gli oggetti interessati e le informazioni di correzione per ogni controllo preliminare.

Gli errori bloccano l'aggiornamento. È necessario risolverli prima di tentare di eseguire nuovamente l'aggiornamento.

Gli avvisi e le notifiche hanno un livello meno critico, ma è comunque consigliabile esaminarli attentamente per assicurarsi che non vi siano problemi di compatibilità con il carico di lavoro dell'applicazione. Risolvi tempestivamente gli eventuali problemi identificati.

Il file di log presenta il formato seguente:

- `targetVersion`: versione compatibile con MySQL dell'aggiornamento di Aurora MySQL.
- `auroraServerVersion`: versione di Aurora MySQL su cui è stato eseguito il controllo preliminare.
- `auroraTargetVersion`: versione di Aurora MySQL a cui stai effettuando l'aggiornamento.
- `checksPerformed`: contiene l'elenco dei controlli preliminari eseguiti.
- `id`: nome del controllo preliminare in esecuzione.
- `title`: descrizione del controllo preliminare in esecuzione.
- `status`: non indica se il controllo preliminare è riuscito o meno, ma mostra lo stato della query di controllo preliminare:
  - `OK`: query di controllo preliminare eseguita e completata correttamente.
  - `ERROR`: query di controllo preliminare non eseguita. Ciò può verificarsi a causa di problemi quali vincoli di risorse, riavvii imprevisti dell'istanza o interruzione della query per il controllo preliminare della compatibilità.

    Per ulteriori informazioni, consulta questo esempio.
- `description`: descrizione generale dell'incompatibilità e delle modalità per correggere l'errore.
- `documentationLink`: link alla documentazione pertinente di Aurora MySQL o MySQL (ove applicabile). Per ulteriori informazioni, consulta Informazioni di riferimento per le descrizioni dei controlli preliminari per Aurora MySQL.
- `detectedProblems`: se il controllo preliminare restituisce un errore, un avviso o una notifica, mostra i dettagli dell'incompatibilità e degli oggetti incompatibili (ove applicabile):
  - `level`: livello dell'incompatibilità rilevata dal controllo preliminare. I livelli validi sono i seguenti:
    - `Error`: l'aggiornamento non può proseguire finché non viene risolta l'incompatibilità.
    - `Warning`: l'aggiornamento può proseguire, ma sono stati rilevati un oggetto, una sintassi o una configurazione obsoleti. Esamina attentamente gli avvisi e risolvili al più presto per evitare problemi nelle versioni future.

    - `Notice`: l'aggiornamento può proseguire, ma sono stati rilevati un oggetto, una sintassi o una configurazione obsoleti. Esamina attentamente le notifiche e risolvile al più presto per evitare problemi nelle versioni future.
  - `dbObject`: nome dell'oggetto del database in cui è stata rilevata l'incompatibilità.
  - `description`: descrizione dettagliata dell'incompatibilità e delle modalità per correggere l'errore.
- `errorCount`: numero di errori di incompatibilità rilevati. Questi impediscono l'aggiornamento.
- `warningCount`: numero di avvisi di incompatibilità rilevati. Questi non impediscono l'aggiornamento, ma occorre risolverli tempestivamente per evitare problemi nelle versioni future.
- `noticeCount`: numero di notifiche di incompatibilità rilevate. Queste non impediscono l'aggiornamento, ma occorre risolverle tempestivamente per evitare problemi nelle versioni future.
- `Summary`: riepilogo del numero di errori, avvisi e notifiche relativi alla compatibilità del controllo preliminare.

Esempi di output di log dei controlli preliminari per Aurora MySQL

Gli esempi seguenti mostrano l'output del log dei controlli preliminari che potrebbe essere visualizzato. Per informazioni dettagliate sui controlli preliminari eseguiti, consulta Informazioni di riferimento per le descrizioni dei controlli preliminari per Aurora MySQL.

Stato di controllo preliminare OK, nessuna incompatibilità rilevata

La query di controllo preliminare è stata completata correttamente. Non sono state rilevate incompatibilità.

```
{
  "id": "auroraUpgradeCheckIndexLengthLimitOnTinytext",
  "title": "Check for the tables with indexes defined with prefix length greater
 than 255 bytes on tiny text columns",
  "status": "OK",
  "detectedProblems": []
},
```

Stato di controllo preliminare OK, errore rilevato

La query di controllo preliminare è stata completata correttamente. È stato rilevato un errore.

```
{
```

```
 "id": "auroraUpgradeCheckForPrefixIndexOnGeometryColumns",
 "title": "Check for geometry columns on prefix indexes",
 "status": "OK",
 "description": "Consider dropping the prefix indexes of geometry columns and
restart the upgrade.",
 "detectedProblems": [
     {
       "level": "Error",
       "dbObject": "test25.sbtest1",
       "description": "Table `test25`.`sbtest1` has an index `idx_t1` on geometry
column/s. Mysql 8.0 does not support this type of index on a geometry column
https://dev.mysql.com/worklog/task/?id=11808. To upgrade to MySQL 8.0, Run 'DROP
INDEX `idx_t1` ON `test25`.`sbtest1`;"
     },
}
```

Stato di controllo preliminare OK, avviso rilevato

Gli avvisi possono essere restituiti quando un controllo preliminare ha esito positivo o negativo.

Qui la query di controllo preliminare è stata completata correttamente. Sono stati rilevati due avvisi.

```
{
  "id": "zeroDatesCheck",
  "title": "Zero Date, Datetime, and Timestamp values",
  "status": "OK",
  "description": "Warning: By default zero date/datetime/timestamp values are no
longer allowed in MySQL, as of 5.7.8 NO_ZERO_IN_DATE and NO_ZERO_DATE are included
in SQL_MODE by default. These modes should be used with strict mode as they will be
merged with strict mode in a future release. If you do not include these modes in
your SQL_MODE setting, you are able to insert date/datetime/timestamp values that
contain zeros. It is strongly advised to replace zero values with valid ones, as
they may not work correctly in the future.",
  "documentationLink": "https://lefred.be/content/mysql-8-0-and-wrong-dates/",
  "detectedProblems": [
      {
        "level": "Warning",
        "dbObject": "global.sql_mode",
        "description": "does not contain either NO_ZERO_DATE or NO_ZERO_IN_DATE
which allows insertion of zero dates"
      },
      {
```

```
        "level": "Warning",
        "dbObject": "session.sql_mode",
        "description": " of 10 session(s) does not contain either NO_ZERO_DATE or
 NO_ZERO_IN_DATE which allows insertion of zero dates"
      }
    ]
}
```

Stato di controllo preliminare ERRORE, nessuna incompatibilità segnalata

La query di controllo preliminare ha avuto esito negativo con un errore, quindi non è stato possibile verificare le incompatibilità.

```
{
  "id": "auroraUpgradeCheckForDatafilePathInconsistency",
  "title": "Check for inconsistency related to ibd file path.",
  "status": "ERROR",
  "description": "Can't connect to MySQL server on 'localhost:3306' (111) at
 13/08/2024 12:22:20 UTC. This failure can occur due to low memory available on the
 instance for executing upgrade prechecks. Please check 'FreeableMemory' Cloudwatch
 metric to verify the available memory on the instance while executing prechecks. If
 instance ran out of memory, we recommend to retry the upgrade on a higher instance
 class."
}
```

Questo errore può verificarsi a causa di un riavvio imprevisto dell'istanza o dell'interruzione di una query di controllo preliminare della compatibilità sul database durante l'esecuzione. Ad esempio, su classi di istanza database di dimensioni inferiori, questo problema potrebbe verificarsi quando la memoria disponibile nell'istanza si esaurisce.

Puoi utilizzare la metrica `FreeableMemory` di Amazon CloudWatch per verificare la memoria disponibile nell'istanza durante l'esecuzione dei controlli preliminari. Se l'istanza ha esaurito la memoria, si consiglia di tentare di eseguire nuovamente l'aggiornamento su una classe di istanza database di dimensioni maggiori. In alcuni casi, è possibile utilizzare un'implementazione blu/verde. Ciò consente l'esecuzione di controlli preliminari e aggiornamenti sul cluster di database "verde" indipendentemente dal carico di lavoro di produzione, che consuma anche risorse di sistema.

Per ulteriori informazioni, consulta Risoluzione dei problemi di utilizzo della memoria per i database Aurora MySQL.

Riepilogo del controllo preliminare, sono stati rilevati un errore e tre avvisi

I controlli preliminari di compatibilità contengono anche informazioni sulle versioni di origine e di destinazione di Aurora MySQL, nonché un riepilogo del numero di errori, avvisi e notifiche alla fine dell'output dei controlli preliminari.

Ad esempio, l'output seguente mostra che è stato effettuato un tentativo di aggiornamento da Aurora MySQL 2.11.6 ad Aurora MySQL 3.07.1. L'aggiornamento ha restituito un errore, tre avvisi e nessuna notifica. Poiché gli aggiornamenti non possono proseguire quando viene restituito un errore, è necessario risolvere il problema di compatibilità routineSyntaxCheck e tentare di eseguire nuovamente l'aggiornamento.

```
{
  "serverAddress": "/tmp%2Fmysql.sock",
  "serverVersion": "5.7.12 - MySQL Community Server (GPL)",
  "targetVersion": "8.0.36",
  "auroraServerVersion": "2.11.6",
  "auroraTargetVersion": "3.07.1",
  "outfilePath": "/rdsdbdata/tmp/PreChecker.log",
  "checksPerformed": [{
      ... output for each individual precheck ...
      .
      .
      {
        "id": "oldTemporalCheck",
        "title": "Usage of old temporal type",
        "status": "OK",
          "detectedProblems": []
      },
      {
        "id": "routinesSyntaxCheck",
        "title": "MySQL 8.0 syntax check for routine-like objects",
        "status": "OK",
        "description": "The following objects did not pass a syntax check with
 the latest MySQL 8.0 grammar. A common reason is that they reference names that
 conflict with new reserved keywords. You must update these routine definitions and
 `quote` any such references before upgrading.",
        "documentationLink": "https://dev.mysql.com/doc/refman/en/keywords.html",
        "detectedProblems": [{
            "level": "Error",
            "dbObject": "test.select_res_word",
            "description": "at line 2,18: unexpected token 'except'"
        }]
```

```
      },
      .
      .
      .
      {
        "id": "zeroDatesCheck",
        "title": "Zero Date, Datetime, and Timestamp values",
        "status": "OK",
        "description": "Warning: By default zero date/datetime/timestamp values
 are no longer allowed in MySQL, as of 5.7.8 NO_ZERO_IN_DATE and NO_ZERO_DATE are
 included in SQL_MODE by default. These modes should be used with strict mode as
 they will be merged with strict mode in a future release. If you do not include
 these modes in your SQL_MODE setting, you are able to insert date/datetime/
timestamp values that contain zeros. It is strongly advised to replace zero values
 with valid ones, as they may not work correctly in the future.",
        "documentationLink": "https://lefred.be/content/mysql-8-0-and-wrong-dates/",
        "detectedProblems": [{
            "level": "Warning",
            "dbObject": "global.sql_mode",
            "description": "does not contain either NO_ZERO_DATE or NO_ZERO_IN_DATE
 which allows insertion of zero dates"
            },
            {
            "level": "Warning",
            "dbObject": "session.sql_mode",
            "description": " of 8 session(s) does not contain either NO_ZERO_DATE or
 NO_ZERO_IN_DATE which allows insertion of zero dates"
            }
          ]
       },
       .
       .
       .
  }],
  "errorCount": 1,
  "warningCount": 3,
  "noticeCount": 0,
  "Summary": "1 errors were found. Please correct these issues before upgrading to
 avoid compatibility issues."
}
```

Verifica preliminare delle prestazioni per Aurora MySQL

I controlli preliminari di compatibilità vengono eseguiti prima della disconnessione dell'istanza database per l'aggiornamento, quindi in circostanze normali non determinano alcun tempo di inattività dell'istanza database durante l'esecuzione. Tuttavia, possono influire sul carico di lavoro dell'applicazione in esecuzione sull'istanza database di scrittura. I controlli preliminari accedono al dizionario dei dati tramite le tabelle information_schema, che possono subire rallentamenti se vi sono molti oggetti del database. Considera i fattori seguenti:

- La durata del controllo preliminare varia in base al numero di oggetti del database come tabelle, colonne, routine e vincoli. L'esecuzione dei cluster di database con un numero elevato di oggetti può richiedere più tempo.

  Ad esempio, removedFunctionsCheck può richiedere più tempo e utilizzare più risorse in base al numero di oggetti memorizzati.
- Per gli aggiornamenti in loco, l'utilizzo di una classe di istanza database di dimensioni maggiori (ad esempio, db.r5.24xlarge o db.r6g.16xlarge) può aiutare a completare l'aggiornamento più rapidamente utilizzando più CPU. È possibile ridurre le dimensioni dopo l'aggiornamento.
- Le query su `information_schema` eseguite su più database possono subire rallentamenti, specialmente con molti oggetti e su istanze database di dimensioni inferiori. In questi casi, valuta la possibilità di eseguire la clonazione, il ripristino tramite snapshot o un'implementazione blu/verde per gli aggiornamenti.
- L'utilizzo delle risorse (CPU, memoria) per il controllo preliminare può aumentare se il numero di oggetti aumenta, determinando tempi di esecuzione più lunghi su istanze database di dimensioni inferiori. In questi casi, valuta la possibilità di effettuare dei test eseguendo la clonazione, il ripristino tramite snapshot o un'implementazione blu/verde per gli aggiornamenti.

  Se i controlli preliminari hanno esito negativo a causa della carenza di risorse, è possibile rilevare tale condizione nel log del controllo preliminare utilizzando l'output di stato:

```
"status": "ERROR",
```

Per ulteriori informazioni, consultare Come funziona l'aggiornamento della versione principale di Aurora MySQL in loco e Pianificazione di un aggiornamento della versione principale per un cluster Aurora MySQL.

Riepilogo dei controlli preliminari per l’aggiornamento di Community MySQL

Di seguito è riportato un elenco generale delle incompatibilità tra MySQL 5.7 e 8.0:

- Il cluster di database compatibile con MySQL 5.7 non deve utilizzare funzionalità che non sono supportate in MySQL 8.0.

  Per ulteriori informazioni, consulta Features Removed in MySQL 8.0 nella documentazione MySQL.
- Non devono essere presenti violazioni di parole chiave o parole riservate. Alcune parole chiave, che non erano riservate in precedenza, possono essere riservate in MySQL 8.0.

  Per ulteriori informazioni, consulta Keywords and Reserved Words nella documentazione MySQL.
- Per supporto Unicode migliorato, valuta la conversione di oggetti che utilizzano il charset `utf8mb3` per utilizzare il charset `utf8mb4`. Il set di caratteri `utf8mb3` è obsoleto. Inoltre, valuta l'utilizzo di `utf8mb4` per i riferimenti al set di caratteri anziché `utf8`, perché attualmente `utf8` è un'alias per il charset `utf8mb3`.

  Per ulteriori informazioni, consulta The utf8mb3 Character Set (3-Byte UTF-8 Unicode Encoding) nella documentazione MySQL.
- Non devono essere presenti tabelle InnoDB con un formato di riga non predefinito.
- Non devono essere presenti attributi di tipo lunghezza `ZEROFILL` o `display`.
- Non devono essere presenti tabelle partizionate che utilizzano un motore di storage che non dispone di supporto di partizionamento nativo.
- Non devono essere presenti tabelle nel database di sistema MySQL 5.7 `mysql` che hanno lo stesso nome di una tabella utilizzata dal dizionario dati MySQL 8.0.
- Non devono essere presenti tabelle che utilizzano tipi di dati o funzioni obsolete.
- Non devono essere presenti nomi di vincoli della chiave più lunghi di 64 caratteri.
- Non devono esistere modalità SQL obsolete definite nell'impostazione della variabile di sistema `sql_mode`.
- Non devono essere presenti tabelle o stored procedure con singoli elementi di colonna `ENUM` o `SET` la cui lunghezza è superiore a 255 caratteri.
- Non devono essere presenti partizioni di tabella che risiedono in tablespace InnoDB condivisi.
- Non devono essere presenti riferimenti circolari nei percorsi dei file di dati del tablespace.
- Non devono essere presenti definizioni di query e di programmi memorizzati che utilizzano qualificatori `ASC` o `DESC` per clausole `GROUP BY`.

- Non devono essere presenti variabili di sistema rimosse e le variabili di sistema devono utilizzare i nuovi valori predefiniti per MySQL 8.0.
- Non devono essere presenti valori di date, datetime o timestamp pari a zero (`0`).
- Non devono essere presenti incoerenze di schema derivanti dalla rimozione o dal danneggiamento dei file.
- Non devono essere presenti nomi di tabella contenenti la stringa di caratteri `FTS`.
- Non devono essere presenti tabelle InnoDB appartenenti a un motore diverso.
- Non devono essere presenti nomi di tabelle o schemi non validi per MySQL 5.7.

Per informazioni dettagliate sui controlli preliminari eseguiti, consulta Informazioni di riferimento per le descrizioni dei controlli preliminari per Aurora MySQL.

Per ulteriori informazioni sull'aggiornamento a MySQL 8.0, consulta Upgrading MySQL nella documentazione di MySQL. Per una descrizione generale delle modifiche apportate in MySQL 8.0, consulta What is new in MySQL 8.0 nella documentazione di MySQL.

Riepilogo dei controlli preliminari per l'aggiornamento di Aurora MySQL

Aurora MySQL ha i propri requisiti specifici per l'aggiornamento dalla versione 2 alla versione 3, tra cui:

- Non deve essere presente alcuna sintassi SQL obsoleta, ad esempio `SQL_CACHE`, `SQL_NO_CACHE` e `QUERY_CACHE` nelle viste, nelle routine, nei trigger e negli eventi.
- Non deve essere presente alcuna colonna `FTS_DOC_ID` in nessuna tabella senza l'indice `FTS`.
- Non deve essere presente alcuna mancata corrispondenza delle definizioni delle colonne tra il dizionario dei dati InnoDB e la definizione effettiva della tabella.
- Tutti i nomi di database e tabelle devono avere lettere minuscole quando il parametro `lower_case_table_names` è impostato su `1`.
- Trigger ed eventi non devono avere un definer mancante o vuoto oppure un contesto di creazione non valido.
- Tutti i nomi dei trigger in un database devono essere univoci.
- Il ripristino DDL e Fast DDL non sono supportati in Aurora MySQL versione 3. Non devono essere presenti artefatti nei database correlati a queste funzionalità.
- Le tabelle con il formato di riga `REDUNDANT` o `COMPACT` non possono avere indici di dimensioni superiori a 767 byte.

- La lunghezza dei prefissi degli indici definiti nelle colonne di testo `tiny` non può superare i 255 byte. Con il set di caratteri `utf8mb4`, ciò limita la lunghezza dei prefissi supportata a 63 caratteri.

  Una lunghezza maggiore era consentita in MySQL 5.7 utilizzando il parametro `innodb_large_prefix`. Questo parametro è obsoleto in MySQL 8.0.
- Non deve essere presente alcuna incoerenza di metadati InnoDB nella tabella `mysql.host`.
- Non deve essere presente alcuna mancata corrispondenza tra i tipi di dati delle colonne nelle tabelle di sistema.
- Non devono essere presenti transazioni XA nello stato `prepared`.
- I nomi delle colonne nelle viste non possono contenere più di 64 caratteri.
- I caratteri speciali nelle stored procedure non possono essere incoerenti.
- Le tabelle non possono presentare incoerenze nei percorsi dei file di dati.

Per informazioni dettagliate sui controlli preliminari eseguiti, consulta Informazioni di riferimento per le descrizioni dei controlli preliminari per Aurora MySQL.

Informazioni di riferimento per le descrizioni dei controlli preliminari per Aurora MySQL

I controlli preliminari per l'aggiornamento per Aurora MySQL sono descritti qui in dettaglio.

Indice

- Errori
  - Controlli preliminari MySQL che restituiscono errori
  - Controlli preliminari Aurora MySQL che restituiscono errori
- Avvisi
  - Controlli preliminari MySQL che restituiscono avvisi
  - Controlli preliminari Aurora MySQL che restituiscono avvisi
- Note
- Errori, avvisi o notifiche

Errori

In caso di esito negativo, i seguenti controlli preliminari generano errori e l'aggiornamento non può essere eseguito.

Argomenti

- Controlli preliminari MySQL che restituiscono errori
- Controlli preliminari Aurora MySQL che restituiscono errori

Controlli preliminari MySQL che restituiscono errori

I seguenti controlli preliminari sono forniti da Community MySQL:

- checkTableMysqlSchema
- circularDirectoryCheck
- columnsWhichCannotHaveDefaultsCheck
- depreciatedSyntaxCheck
- engineMixupCheck
- enumSetElementLengthCheck
- foreignKeyLengthCheck
- getDuplicateTriggers
- getEventsWithNullDefiner
- getMismatchedMetadata
- getTriggersWithNullDefiner
- getValueOfVariablelower_case_table_names
- groupByAscSyntaxCheck
- mysqlEmptyDotTableSyntaxCheck
- mysqlIndexTooLargeCheck
- mysqlInvalid57NamesCheck
- mysqlOrphanedRoutinesCheck
- mysqlSchemaCheck
- nonNativePartitioningCheck
- oldTemporalCheck
- partitionedTablesInSharedTablespaceCheck
- removedFunctionsCheck
- routineSyntaxCheck
- schemaInconsistencyCheck

checkTableMysqlSchema

Livello di controllo preliminare: errore

Problemi segnalati dal comando **`check table x for upgrade`** per lo schema **`mysql`**

Prima di iniziare l'aggiornamento ad Aurora MySQL versione 3, `check table for upgrade` viene eseguito su ogni tabella dello schema `mysql` dell'istanza database. Il comando `check table for upgrade` esamina le tabelle per individuare eventuali problemi che potrebbero verificarsi durante l'aggiornamento a una versione più recente di MySQL. L'esecuzione di questo comando prima di tentare un aggiornamento può aiutare a identificare e risolvere eventuali incompatibilità in anticipo, rendendo più agevole l'effettivo processo di aggiornamento.

Questo comando esegue diversi controlli su ogni tabella, come i seguenti:

- Verifica della compatibilità della struttura della tabella e dei metadati con la versione MySQL di destinazione
- Verifica della presenza di eventuali funzionalità obsolete o rimosse utilizzate dalla tabella
- Garanzia che la tabella possa essere aggiornata correttamente senza perdita di dati

Per ulteriori informazioni, consulta CHECK TABLE statement nella documentazione di MySQL.

Output di esempio:

```
{
  "id": "checkTableMysqlSchema",
  "title": "Issues reported by 'check table x for upgrade' command for mysql
 schema.",
  "status": "OK",
  "detectedProblems": []
}
```

L'output di questo controllo preliminare dipende dall'errore riscontrato e dal momento in cui si verifica l'errore, poiché `check table for upgrade` esegue più controlli.

Se riscontri errori con questo controllo preliminare, apri un caso con il Supporto AWS per richiedere che l'incoerenza dei metadati venga risolta. In alternativa, puoi tentare nuovamente l'aggiornamento eseguendo un dump logico, quindi effettuando il ripristino su un nuovo cluster di database Aurora MySQL versione 3.

circularDirectoryCheck

Livello di controllo preliminare: errore

Riferimenti circolari alle directory nei percorsi dei file di dati del tablespace

A partire da MySQL 8.0.17, la clausola `CREATE TABLESPACE ... ADD DATAFILE` non consente più i riferimenti circolari alle directory. Per evitare problemi di aggiornamento, rimuovi tutti i riferimenti circolari alle directory dai percorsi dei file di dati del tablespace prima di eseguire l'aggiornamento ad Aurora MySQL versione 3.

Output di esempio:

```
{
  "id": "circularDirectory",
  "title": "Circular directory references in tablespace data file paths",
  "status": "OK",
  "description": "Error: Following tablespaces contain circular directory references
 (e.g. the reference '/../') in data file paths which as of MySQL 8.0.17 are not
 permitted by the CREATE TABLESPACE ... ADD DATAFILE clause. An exception to the
 restriction exists on Linux, where a circular directory reference is permitted if
 the preceding directory is a symbolic link. To avoid upgrade issues, remove any
 circular directory references from tablespace data file paths before upgrading.",
  "documentationLink": "https://dev.mysql.com/doc/refman/8.0/en/upgrading-from-
previous-series.html#upgrade-innodb-changes",
  "detectedProblems": [
      {
        "level": "Error",
        "dbObject": "ts2",
        "description": "circular reference in datafile path: '/home/ec2-user/dbdata/
mysql_5_7_44/../ts2.ibd'",
        "dbObjectType": "Tablespace"
      }
  ]
}
```

Se ricevi questo errore, crea nuovamente le tabelle utilizzando un tablespace file-per-table. Utilizza i percorsi di file predefiniti per tutti i tablespace e le definizioni delle tabelle.

> **Note**
>
> Aurora MySQL non supporta i tablespace o i comandi `CREATE TABLESPACE` generici. Prima di creare nuovamente i tablespace, consulta le operazioni Online DDL nella documentazione di MySQL per comprendere gli effetti del blocco e dello spostamento dei dati sulle transazioni in primo piano.

Una volta creati nuovamente i tablespace, il controllo preliminare viene superato e l'aggiornamento può procedere.

```
{
  "id": "circularDirectoryCheck",
  "title": "Circular directory references in tablespace data file paths",
  "status": "OK",
  "detectedProblems": []
},
```

columnsWhichCannotHaveDefaultsCheck

Livello di controllo preliminare: errore

Colonne che non possono avere valori predefiniti

Prima di MySQL 8.0.13, le colonne `BLOB`, `TEXT`, `GEOMETRY` e `JSON` non possono avere valori predefiniti. Rimuovi tutte le clausole predefinite in queste colonne prima dell'aggiornamento ad Aurora MySQL versione 3. Per ulteriori informazioni sulle modifiche alla gestione predefinita per questi tipi di dati, consulta Data type default values nella documentazione di MySQL.

Output di esempio:

```
{
  "id": "columnsWhichCannotHaveDefaultsCheck",
  "title": "Columns which cannot have default values",
  "status": "OK",
  "description": "Error: The following columns are defined as either BLOB, TEXT,
 GEOMETRY or JSON and have a default value set. These data types cannot have default
 values in MySQL versions prior to 8.0.13, while starting with 8.0.13, the default
 value must be specified as an expression. In order to fix this issue, please use
 the ALTER TABLE ... ALTER COLUMN ... DROP DEFAULT statement.",
  "documentationLink": "https://dev.mysql.com/doc/refman/8.0/en/data-type-
defaults.html#data-type-defaults-explicit",
  "detectedProblems": [
      {
        "level": "Error",
        "dbObject": "test.test_blob_default.geo_col",
        "description": "geometry"
      }
  ]
},
```

Il controllo preliminare restituisce un errore perché la colonna `geo_col` della tabella `test.test_blob_default` utilizza un tipo di dati `BLOB`, `TEXT`, `GEOMETRY`, o `JSON` con un valore predefinito specificato.

Guardando la definizione della tabella, possiamo vedere che la colonna `geo_col` è definita come `geo_col geometry NOT NULL default ''`.

```
mysql> show create table test_blob_default\G
*************************** 1. row ***************************
       Table: test_blob_default
Create Table: CREATE TABLE `test_blob_default` (
  `geo_col` geometry NOT NULL DEFAULT ''
) ENGINE=InnoDB DEFAULT CHARSET=latin1
```

Rimuovi questa clausola predefinita per consentire il superamento del controllo preliminare.

> **Note**
>
> Prima di eseguire le istruzioni `ALTER TABLE` o creare nuovamente i tablespace, consulta le operazioni Online DDL nella documentazione di MySQL per comprendere gli effetti del blocco e dello spostamento dei dati sulle transazioni in primo piano.

```
mysql> ALTER TABLE test_blob_default modify COLUMN geo_col geometry NOT NULL;
Query OK, 0 rows affected (0.02 sec)
Records: 0  Duplicates: 0  Warnings: 0

mysql> show create table test_blob_default\G
*************************** 1. row ***************************
       Table: test_blob_default
Create Table: CREATE TABLE `test_blob_default` (
  `geo_col` geometry NOT NULL
) ENGINE=InnoDB DEFAULT CHARSET=latin1
1 row in set (0.00 sec)
```

Il controllo preliminare viene superato e puoi tentare nuovamente di eseguire l'aggiornamento.

```
{
  "id": "columnsWhichCannotHaveDefaultsCheck",
  "title": "Columns which cannot have default values",
  "status": "OK",
```

```
  "detectedProblems": []
},
```

depreciatedSyntaxCheck

Livello di controllo preliminare: errore

Utilizzo di parole chiave obsolete nella definizione

MySQL 8.0 ha rimosso la cache delle query. Di conseguenza, è stata rimossa una parte della sintassi SQL specifica della cache delle query. Se uno qualsiasi degli oggetti del database contiene le parole chiave `QUERY CACHE`, `SQL_CACHE` o `SQL_NO_CACHE`, viene restituito un errore di controllo preliminare. Per risolvere il problema, crea nuovamente gli oggetti rimuovendo le parole chiave menzionate.

Output di esempio:

```
{
  "id": "depreciatedSyntaxCheck",
  "title": "Usage of depreciated keywords in definition",
  "status": "OK",
  "description": "Error: The following DB objects contain keywords like 'QUERY
 CACHE', 'SQL_CACHE', 'SQL_NO_CACHE' which are not supported in major version 8.0.
 It is recommended to drop these DB objects or rebuild without any of the above
 keywords before upgrade.",
  "detectedProblems": [
      {
"level": "Error",
"dbObject": "test.no_query_cache_check",
"description": "PROCEDURE uses depreciated words in definition"
      }
  ]
}
```

Il controllo preliminare segnala che la stored procedure `test.no_query_cache_check` utilizza una delle parole chiave rimosse. Guardando la definizione della procedura, possiamo vedere che utilizza `SQL_NO_CACHE`.

```
mysql> show create procedure test.no_query_cache_check\G
*************************** 1. row ***************************
           Procedure: no_query_cache_check
            sql_mode:
```

```
    Create Procedure: CREATE DEFINER=`reinvent`@`%` PROCEDURE
 `no_query_cache_check`()
BEGIN
    SELECT SQL_NO_CACHE k from sbtest1 where id > 10 and id < 20 group by k asc;
END
character_set_client: utf8mb4
collation_connection: utf8mb4_0900_ai_ci
  Database Collation: latin1_swedish_ci
1 row in set (0.00 sec)
```

Rimuovi la parola chiave.

```
mysql> drop procedure test.no_query_cache_check;
Query OK, 0 rows affected (0.01 sec)

mysql> delimiter //

mysql> CREATE DEFINER=`reinvent`@`%` PROCEDURE `no_query_cache_check`() BEGIN
 SELECT k from sbtest1 where id > 10 and id < 20 group by k asc; END//
Query OK, 0 rows affected (0.00 sec)

mysql> delimiter ;
```

Dopo aver rimosso la parola chiave, il controllo preliminare viene completato correttamente.

```
{
  "id": "depreciatedSyntaxCheck",
  "title": "Usage of depreciated keywords in definition",
  "status": "OK",
  "detectedProblems": []
}
```

engineMixupCheck

Livello di controllo preliminare: errore

Tabelle riconosciute da InnoDB che appartengono a un motore diverso

In modo analogo a schemaInconsistencyCheck, questo controllo preliminare verifica che i metadati delle tabelle in MySQL siano coerenti prima di procedere con l'aggiornamento.

Se riscontri errori con questo controllo preliminare, apri un caso con il Supporto AWS per richiedere che l'incoerenza dei metadati venga risolta. In alternativa, puoi tentare nuovamente

l’aggiornamento eseguendo un dump logico, quindi effettuando il ripristino su un nuovo cluster di database Aurora MySQL versione 3.

Output di esempio:

```
{
  "id": "engineMixupCheck",
  "title": "Tables recognized by InnoDB that belong to a different engine",
  "status": "OK",
  "description": "Error: Following tables are recognized by InnoDB engine while the
 SQL layer believes they belong to a different engine. Such situation may happen
 when one removes InnoDB table files manually from the disk and creates e.g. a
 MyISAM table with the same name.\n\nA possible way to solve this situation is to
 e.g. in case of MyISAM table:\n\n1. Rename the MyISAM table to a temporary name
 (RENAME TABLE).\n2. Create some dummy InnoDB table (its definition does not need to
 match), then copy (copy, not move) and rename the dummy .frm and .ibd files to the
 orphan name using OS file commands.\n3. The orphan table can be then dropped (DROP
 TABLE), as well as the dummy table.\n4. Finally the MyISAM table can be renamed
 back to its original name.",
  "detectedProblems": [
      {
        "level": "Error",
        "dbObject": "mysql.general_log_backup",
        "description": "recognized by the InnoDB engine but belongs to CSV"
      }
  ]
}
```

enumSetElementLengthCheck

Livello di controllo preliminare: errore

Definizioni delle colonne **`ENUM`** e **`SET`** che contengono elementi con più di 255 caratteri

Le tabelle e le stored procedure non devono contenere elementi colonna `ENUM` o `SET` con più di 255 caratteri o 1020 byte. Prima di MySQL 8.0, la lunghezza massima combinata era di 64K, ma la versione 8.0 limita i singoli elementi a 255 caratteri o 1020 byte (con supporto per i multibyte). Se si verifica un errore di controllo preliminare per `enumSetElementLengthCheck`, modifica tutti gli elementi che superano i nuovi limiti prima di tentare di eseguire nuovamente l’aggiornamento.

Output di esempio:

```
{
```

```
  "id": "enumSetElementLengthCheck",
  "title": "ENUM/SET column definitions containing elements longer than 255
 characters",
  "status": "OK",
  "description": "Error: The following columns are defined as either ENUM or SET and
 contain at least one element longer that 255 characters. They need to be altered so
 that all elements fit into the 255 characters limit.",
  "documentationLink": "https://dev.mysql.com/doc/refman/8.0/en/string-type-
overview.html",
  "detectedProblems": [
      {
        "level": "Error",
        "dbObject": "test.large_set.s",
        "description": "SET contains element longer than 255 characters"
      }
  ]
},
```

Il controllo preliminare segnala un errore perché la colonna `s` della tabella `test.large_set` contiene un elemento `SET` con più di 255 caratteri.

Dopo aver ridotto le dimensioni `SET` di questa colonna, il controllo preliminare viene eseguito e l'aggiornamento può continuare.

```
{
  "id": "enumSetElementLenghtCheck",
  "title": "ENUM/SET column definitions containing elements longer than 255
 characters",
  "status": "OK",
  "detectedProblems": []
},
```

foreignKeyLengthCheck

Livello di controllo preliminare: errore

Nomi di vincoli di chiave esterna con più di 64 caratteri

In MySQL, la lunghezza degli identificatori è limitata a 64 caratteri, come indicato nella documentazione di MySQL. A causa dei problemi identificati per cui la lunghezza delle chiavi esterne poteva essere uguale o superiore a questo valore, con conseguenti errori di aggiornamento, è stato implementato questo controllo preliminare. Se si verificano errori con

questo controllo preliminare, è necessario modificare o rinominare il vincolo interessato in modo che sia inferiore a 64 caratteri prima di tentare di eseguire nuovamente l'aggiornamento.

Output di esempio:

```
{
  "id": "foreignKeyLength",
  "title": "Foreign key constraint names longer than 64 characters",
  "status": "OK",
  "detectedProblems": []
}
```

getDuplicateTriggers

Livello di controllo preliminare: errore

Tutti i nomi dei trigger in un database devono essere univoci.

A causa delle modifiche nell'implementazione del dizionario dei dati, MySQL 8.0 non supporta i trigger con distinzione tra maiuscole e minuscole all'interno di un database. Questo controllo preliminare verifica che il cluster di database non disponga di uno o più database contenenti trigger duplicati. Per ulteriori informazioni, consulta Identifier case sensitivity nella documentazione di MySQL.

Output di esempio:

```
{
  "id": "getDuplicateTriggers",
  "title": "MySQL pre-checks that all trigger names in a database are unique or
 not.",
  "status": "OK",
  "description": "Error: You have one or more database containing duplicate
 triggers. Mysql 8.0 does not support case sensitive triggers within a database
 https://dev.mysql.com/doc/refman/8.0/en/identifier-case-sensitivity.html. To
 upgrade to MySQL 8.0, drop the triggers with case-insensitive duplicate names and
 recreate with distinct names.",
  "detectedProblems": [
      {
        "level": "Error",
        "dbObject": "test",
        "description": "before_insert_product"
      },
```

```
      {
        "level": "Error",
        "dbObject": "test",
        "description": "before_insert_PRODUCT"
      }
  ]
}
```

Il controllo preliminare segnala un errore per cui il cluster di database ha due trigger con lo stesso nome, ma che utilizzano lettere maiuscole e minuscole diverse: `test.before_insert_product` e `test.before_insert_PRODUCT`.

Prima dell'aggiornamento, rinomina i trigger o eliminali e creali di nuovo con un nuovo nome.

Dopo la ridenominazione di `test.before_insert_PRODUCT` in `test.before_insert_product_2`, il controllo preliminare ha esito positivo.

```
{
  "id": "getDuplicateTriggers",
  "title": "MySQL pre-checks that all trigger names in a database are unique or
 not.",
  "status": "OK",
  "detectedProblems": []
}
```

getEventsWithNullDefiner

Livello di controllo preliminare: errore

La colonna definer per **`mysql.event`** non può essere nulla o vuota.

L'attributo `DEFINER` specifica l'account MySQL che possiede una definizione di oggetto memorizzato, come un trigger, una stored procedure o un evento. Questo attributo è particolarmente utile in situazioni in cui si desidera controllare il contesto di sicurezza in cui viene eseguito l'oggetto memorizzato. Quando si crea un oggetto memorizzato, se non è specificato un `DEFINER`, l'impostazione predefinita è l'utente che ha creato l'oggetto.

Quando si esegue l'aggiornamento a MySQL 8.0, non è possibile avere oggetti memorizzati che abbiano un definer `null` o vuoto nel dizionario dei dati di MySQL. Se si dispone di tali oggetti memorizzati, viene generato un errore di controllo preliminare. È necessario correggerlo prima di procedere con l'aggiornamento.

Esempio di errore:

```
{
  "id": "getEventsWithNullDefiner",
  "title": "The definer column for mysql.event cannot be null or blank.",
  "status": "OK",
  "description": "Error: Set definer column in mysql.event to a valid non-null
 definer.",
  "detectedProblems": [
      {
        "level": "Error",
        "dbObject": "test.get_version",
        "description": "Set definer for event get_version in Schema test"
      }
  ]
}
```

Il controllo preliminare restituisce un errore per l'evento `test.get_version` perché ha un definer `null`.

Per risolvere questo problema puoi controllare la definizione dell'evento. Come puoi vedere, il definer è `null` o vuoto.

```
mysql> select db,name,definer from mysql.event where name='get_version';
+------+-------------+---------+
| db   | name        | definer |
+------+-------------+---------+
| test | get_version |         |
+------+-------------+---------+
1 row in set (0.00 sec)
```

Elimina o crea nuovamente l'evento con un definer valido.

> **Note**
>
> Prima di eliminare o ridefinire un `DEFINER`, esamina e controlla attentamente i requisiti dell'applicazione e dei privilegi. Per ulteriori informazioni, consulta Stored object access control nella documentazione di MySQL.

```
mysql> drop event test.get_version;
```

```
Query OK, 0 rows affected (0.00 sec)

mysql> DELIMITER ;
mysql> delimiter $$
mysql> CREATE EVENT get_version
    ->     ON SCHEDULE
    ->       EVERY 1 DAY
    ->     DO
    ->      ///DO SOMETHING //
    -> $$
Query OK, 0 rows affected (0.01 sec)

mysql> DELIMITER ;

mysql> select db,name,definer from mysql.event where name='get_version';
+------+-------------+------------+
| db   | name        | definer    |
+------+-------------+------------+
| test | get_version | reinvent@% |
+------+-------------+------------+
1 row in set (0.00 sec)
```

Ora il controllo preliminare viene superato.

```
{
  "id": "getEventsWithNullDefiner",
  "title": "The definer column for mysql.event cannot be null or blank.",
  "status": "OK",
  "detectedProblems": []},
```

getMismatchedMetadata

Livello di controllo preliminare: errore

Mancata corrispondenza delle definizioni delle colonne tra il dizionario dei dati InnoDB e la definizione effettiva della tabella

In modo analogo a schemaInconsistencyCheck, questo controllo preliminare verifica che i metadati delle tabelle in MySQL siano coerenti prima di procedere con l'aggiornamento. In questo caso, il controllo preliminare verifica che le definizioni delle colonne corrispondano tra il dizionario dei dati InnoDB e la definizione della tabella MySQL. Se viene rilevata una mancata corrispondenza, l'aggiornamento non procede.

Se riscontri errori con questo controllo preliminare, apri un caso con il Supporto AWS per richiedere che l'incoerenza dei metadati venga risolta. In alternativa, puoi tentare nuovamente l'aggiornamento eseguendo un dump logico, quindi effettuando il ripristino su un nuovo cluster di database Aurora MySQL versione 3.

Output di esempio:

```
{
  "id": "getMismatchedMetadata",
  "title": "Column definition mismatch between InnoDB Data Dictionary and actual
 table definition.",
  "status": "OK",
  "description": "Error: Your database has mismatched metadata. The upgrade to mysql
 8.0 will not succeed until this is fixed.",
  "detectedProblems": [
      {
        "level": "Error",
        "dbObject": "test.mismatchTable",
        "description": "Table `test/mismatchTable` column names mismatch with InnoDb
 dictionary column names: iD <> id"
      }
  ]
}
```

Il controllo preliminare segnala una mancata corrispondenza nei metadati per la colonna `id` della tabella `test.mismatchTable`. In particolare, i metadati MySQL hanno il nome della colonna `iD`, mentre InnoDB ha il nome `id`.

getTriggersWithNullDefiner

Livello di controllo preliminare: errore

La colonna definer per **`information_schema.triggers`** non può essere **`null`** o vuota.

Il controllo preliminare verifica che il database non abbia trigger definiti con definer `null` o vuoti. Per ulteriori informazioni sui requisiti del definer per gli oggetti memorizzati, consulta getEventsWithNullDefiner.

Output di esempio:

```
{
  "id": "getTriggersWithNullDefiner",
```

```
  "title": "The definer column for information_schema.triggers cannot be null or
 blank.",
  "status": "OK",
  "detectedProblems": [
      {
        "level": "Error",
        "dbObject": "test.example_trigger",
        "description": "Set definer for trigger example_trigger in Schema test"
      }
  ]
}
```

Il controllo preliminare restituisce un errore perché il trigger `example_trigger` nello schema `test` ha un definer `null`. Per risolvere questo problema, correggi il definer creando nuovamente il trigger con un utente valido oppure elimina il trigger. Per ulteriori informazioni, consulta l'esempio in getEventsWithNullDefiner.

> **Note**
>
> Prima di eliminare o ridefinire un `DEFINER`, esamina e controlla attentamente i requisiti dell'applicazione e dei privilegi. Per ulteriori informazioni, consulta Stored object access control nella documentazione di MySQL.

getValueOfVariablelower_case_table_names

Livello di controllo preliminare: errore

Tutti i nomi di database o tabelle devono avere lettere minuscole quando il parametro **`lower_case_table_names`** è impostato su **`1`**.

Prima di MySQL 8.0, i nomi dei database, i nomi delle tabelle e altri oggetti corrispondevano ai file nella directory dei dati, come i metadati basati su file (.frm). La variabile di sistema lower_case_table_names consente di controllare il modo in cui il server gestisce la distinzione tra maiuscole e minuscole degli identificatori per gli oggetti del database e l'archiviazione di tali oggetti di metadati. Questo parametro può essere modificato su un server già inizializzato dopo un riavvio.

Tuttavia, in MySQL 8.0, sebbene questo parametro controlli ancora il modo in cui il server gestisce la distinzione tra maiuscole e minuscole degli identificatori, non può essere modificato dopo l'inizializzazione del dizionario dei dati. Quando si aggiorna o si crea un database MySQL

8.0, il valore impostato per `lower_case_table_names` al primo avvio del dizionario dei dati su MySQL viene utilizzato per tutta la durata del database. Questa restrizione è stata messa in atto nell'ambito dell'implementazione di Atomic Data Dictionary, in cui viene eseguita la migrazione degli oggetti del database dai metadati basati su file alle tabelle InnoDB interne dello schema `mysql`.

Per ulteriori informazioni, consulta Data dictionary changes nella documentazione di MySQL.

Per evitare problemi durante l'aggiornamento mentre si aggiornano i metadati basati su file al nuovo Atomic Data Dictionary, questo controllo preliminare verifica che quando `lower_case_table_names = 1` tutte le tabelle vengano memorizzate su disco in lettere minuscole. In caso contrario, viene restituito un errore di controllo preliminare ed è necessario correggere i metadati prima di procedere con l'aggiornamento.

Output di esempio:

```
{
  "id": "getValueOfVariablelower_case_table_names",
  "title": "MySQL pre-checks that all database or table names are lowercase when the
 lower_case_table_names parameter is set to 1.",
  "status": "OK",
  "description": "Error: You have one or more databases or tables with uppercase
 letters in the names, but the lower_case_table_names parameter is set to 1. To
 upgrade to MySQL 8.0, either change all database or table names to lowercase, or
 set the parameter to 0.",
  "detectedProblems": [
      {
        "level": "Error",
        "dbObject": "test.TEST",
        "description": "Table test.TEST contains one or more capital letters in name
 while lower_case_table_names = 1"
      }
  ]
}
```

Viene restituito un errore perché la tabella `test.TEST` contiene lettere maiuscole, ma `lower_case_table_names` è impostato su 1.

Per risolvere questo problema, è possibile rinominare la tabella utilizzando lettere minuscole o modificare il parametro `lower_case_table_names` nel cluster di database prima di iniziare l'aggiornamento.

> **Note**
>
> Verifica ed esamina attentamente la documentazione sulla distinzione tra maiuscole e minuscole in MySQL e scopri come tali modifiche potrebbero influire sulla tua applicazione.
> Consulta anche la documentazione di MySQL 8.0 che spiega come i lower_case_table_names vengono gestiti in modo diverso in MySQL 8.0.

groupByAscSyntaxCheck

Livello di controllo preliminare: errore

Utilizzo della sintassi **`GROUP BY ASC/DESC`** rimossa

A partire da MySQL 8.0.13, la sintassi `ASC` o `DESC` obsoleta per le clausole `GROUP BY` è stata rimossa. Le query che si basano sull'ordinamento `GROUP BY` potrebbero ora produrre risultati diversi. Per ottenere un ordinamento specifico, utilizza invece una clausola `ORDER BY`. Se nel database sono presenti oggetti che utilizzano questa sintassi, è necessario crearli nuovamente utilizzando una clausola `ORDER BY` prima di tentare di eseguire di nuovo l'aggiornamento. Per ulteriori informazioni, consulta SQL changes nella documentazione di MySQL.

Output di esempio:

```
{
  "id": "groupbyAscSyntaxCheck",
  "title": "Usage of removed GROUP BY ASC/DESC syntax",
  "status": "OK",
  "description": "Error: The following DB objects use removed GROUP BY ASC/DESC
 syntax. They need to be altered so that ASC/DESC keyword is removed from GROUP BY
 clause and placed in appropriate ORDER BY clause.",
  "documentationLink": "https://dev.mysql.com/doc/relnotes/mysql/8.0/en/
news-8-0-13.html#mysqld-8-0-13-sql-syntax",
  "detectedProblems": [
      {
        "level": "Error",
        "dbObject": "test.groupbyasc",
        "description": "PROCEDURE uses removed GROUP BY ASC syntax",
        "dbObjectType": "Routine"
      }
  ]
```

```
}
```

mysqlEmptyDotTableSyntaxCheck

Livello di controllo preliminare: errore

Verifica la presenza di una sintassi **`.<table>`** obsoleta utilizzata nelle routine.

In MySQL 8.0, le routine non possono più contenere la sintassi dell'identificatore obsoleta (`\".<table>\"`). Se alcune routine o trigger memorizzati contengono tali identificatori, l'aggiornamento non riesce. Ad esempio, il seguente riferimento `.dot_table` non è più consentito:

```
mysql> show create procedure incorrect_procedure\G
*************************** 1. row ***************************
           Procedure: incorrect_procedure
            sql_mode:
    Create Procedure: CREATE DEFINER=`reinvent`@`%` PROCEDURE
 `incorrect_procedure`()
BEGIN delete FROM .dot_table; select * from .dot_table where 1=1; END
character_set_client: utf8mb4
collation_connection: utf8mb4_0900_ai_ci
  Database Collation: latin1_swedish_ci
1 row in set (0.00 sec)
```

Dopo aver creato nuovamente le routine e i trigger per utilizzare la sintassi dell'identificatore e l'escape corretti, il controllo preliminare viene superato e l'aggiornamento può procedere. Per ulteriori informazioni, consulta Schema object names nella documentazione di MySQL.

Output di esempio:

```
{
  "id": "mysqlEmptyDotTableSyntaxCheck",
  "title": "Check for deprecated '.<table>' syntax used in routines.",
  "status": "OK",
  "description": "Error: The following routines contain identifiers in deprecated
 identifier syntax (\".<table>\"), and should be corrected before upgrade:\n",
  "detectedProblems": [
      {
        "level": "Error",
        "dbObject": "test.incorrect_procedure",
```

```
        "description": " routine body contains deprecated identifiers."
      }
  ]
}
```

Il controllo preliminare restituisce un errore per la routine `incorrect_procedure` nel database `test` perché contiene una sintassi obsoleta.

Dopo aver corretto la routine, il controllo preliminare ha esito positivo ed è possibile tentare di eseguire nuovamente l'aggiornamento.

mysqlIndexTooLargeCheck

Livello di controllo preliminare: errore

Verifica la presenza di indici troppo grandi per funzionare su versioni di MySQL successive alla 5.7

Per i formati di riga compatti o ridondanti, non dovrebbe essere possibile creare un indice con un prefisso con più di 767 byte. Tuttavia, prima della versione 5.7.35 di MySQL questo era possibile. Per ulteriori informazioni, consulta le note di rilascio di MySQL 5.7.35.

Tutti gli indici interessati da questo bug diventeranno inaccessibili dopo l'aggiornamento a MySQL 8.0. Questo controllo preliminare identifica gli indici che causano l'errore, che devono essere creati nuovamente prima che l'aggiornamento possa procedere.

```
{
 "id": "mysqlIndexTooLargeCheck",
 "title": "Check for indexes that are too large to work on higher versions of MySQL
Server than 5.7",
 "status": "OK",
 "description": "Error: The following indexes ware made too large for their format
in an older version of MySQL (older than 5.7.34). Normally those indexes within
tables with compact or redundant row formats shouldn't be larger than 767 bytes. To
fix this problem those indexes should be dropped before upgrading or those tables
will be inaccessible.",
 "detectedProblems": [
     {
       "level": "Error",
       "dbObject": "test.table_with_large_idx",
       "description": "IDX_2"
     }
```

```
  ]
}
```

Il controllo preliminare restituisce un errore perché la tabella `test.table_with_large_idx` contiene un indice su una tabella che utilizza un formato di riga compatto o ridondante con più di 767 byte. Queste tabelle diventerebbero inaccessibili dopo l'aggiornamento a MySQL 8.0. Prima di procedere con l'aggiornamento, esegui una delle seguenti operazioni:

- Elimina l'indice indicato nel controllo preliminare.
- Aggiungi un indice menzionato nel controllo preliminare.
- Cambia il formato di riga utilizzato dalla tabella.

In questo esempio, creiamo nuovamente la tabella per risolvere l'errore di controllo preliminare. Prima di ricostruire la tabella, assicurati che innodb_file_format sia impostato su `Barracuda` e che innodb_default_row_format sia impostato su `dynamic`. Queste sono le impostazioni predefinite in MySQL 5.7. Per ulteriori informazioni, consulta InnoDB row formats e InnoDB file-format management nella documentazione di MySQL.

> **Note**
>
> Prima di creare nuovamente i tablespace, consulta le operazioni Online DDL nella documentazione di MySQL per comprendere gli effetti del blocco e dello spostamento dei dati sulle transazioni in primo piano.

```
mysql > select @@innodb_file_format,@@innodb_default_row_format;
+----------------------+-----------------------------+
| @@innodb_file_format | @@innodb_default_row_format |
+----------------------+-----------------------------+
| Barracuda            | dynamic                     |
+----------------------+-----------------------------+
1 row in set (0.00 sec)

mysql> optimize table table_with_large_idx;
+---------------------------+----------+----------
+-------------------------------------------------------------------------+
| Table                     | Op       | Msg_type | Msg_text
                            |
+---------------------------+----------+----------
+-------------------------------------------------------------------------+
```

```
| test.table_with_large_idx | optimize | note     | Table does not support optimize,
 doing recreate + analyze instead |
| test.table_with_large_idx | optimize | status   | OK
                                  |
+---------------------------+----------+----------
+-------------------------------------------------------------------+
2 rows in set (0.02 sec)

# Verify FILE_FORMAT and ROW_FORMAT
mysql>  select * from information_schema.innodb_sys_tables where name like 'test/
table_with_large_idx';
+----------+---------------------------+------+--------+-------+-------------
+------------+---------------+------------+
| TABLE_ID | NAME                      | FLAG | N_COLS | SPACE | FILE_FORMAT |
 ROW_FORMAT | ZIP_PAGE_SIZE | SPACE_TYPE |
+----------+---------------------------+------+--------+-------+-------------
+------------+---------------+------------+
|       43 | test/table_with_large_idx |   33 |      4 |    26 | Barracuda   |
 Dynamic    |             0 | Single     |
+----------+---------------------------+------+--------+-------+-------------
+------------+---------------+------------+
1 row in set (0.00 sec)
```

Una volta creata nuovamente la tabella, il controllo preliminare viene superato e l'aggiornamento può procedere.

```
{
  "id": "mysqlIndexTooLargeCheck",
  "title": "Check for indexes that are too large to work on higher versions of MySQL
 Server than 5.7",
  "status": "OK",
  "detectedProblems": []
},
```

mysqlInvalid57NamesCheck

Livello di controllo preliminare: errore

Verifica la presenza di nomi di tabelle e schemi non validi utilizzati in MySQL 5.7

Durante la migrazione al nuovo dizionario dei dati in MySQL 8.0, l'istanza database in uso non può contenere schemi o tabelle con il prefisso `#mysql50#`. Se esistono oggetti di questo tipo,

l'aggiornamento non riesce. Per risolvere questo problema, esegui mysqlcheck sugli schemi e sulle tabelle restituiti.

> **Note**
>
> Assicurati di utilizzare una versione MySQL 5.7 di `mysqlcheck`, poiché --fix-db-names e --fix-table-names sono stati rimossi da MySQL 8.0.

Output di esempio:

```
{
  "id": "mysqlInvalid57NamesCheck",
  "title": "Check for invalid table names and schema names used in 5.7",
  "status": "OK",
  "description": "The following tables and/or schemas have invalid names. In order
 to fix them use the mysqlcheck utility as follows:\n\n  $ mysqlcheck --check-
upgrade --all-databases\n  $ mysqlcheck --fix-db-names --fix-table-names --all-
databases\n\nOR via mysql client, for eg:\n\n  ALTER DATABASE `#mysql50#lost+found`
 UPGRADE DATA DIRECTORY NAME;",
  "documentationLink": "https://dev.mysql.com/doc/refman/5.7/en/identifier-
mapping.html https://dev.mysql.com/doc/refman/5.7/en/alter-database.html https://
dev.mysql.com/doc/refman/8.0/en/mysql-nutshell.html#mysql-nutshell-removals",
  "detectedProblems": [
      {
        "level": "Error",
        "dbObject": "#mysql50#fix_db_names",
        "description": "Schema name"
      }
  ]
}
```

Il controllo preliminare segnala che lo schema `#mysql50#fix_db_names` ha un nome non valido.

Una volta corretto il nome dello schema, il controllo preliminare viene superato e l'aggiornamento può procedere.

```
{
  "id": "mysqlInvalid57NamesCheck",
  "title": "Check for invalid table names and schema names used in 5.7",
```

```
  "status": "OK",
  "detectedProblems": []
},
```

mysqlOrphanedRoutinesCheck

Livello di controllo preliminare: errore

Verifica la presenza di routine orfane in 5.7

Durante la migrazione al nuovo dizionario dei dati in MySQL 8.0, se nel database sono presenti stored procedure in cui lo schema non esiste più, l'aggiornamento non riesce. Questo controllo preliminare verifica che tutti gli schemi a cui si fa riferimento nelle stored procedure sull'istanza database esistano ancora. Per consentire il proseguimento dell'aggiornamento, elimina queste stored procedure.

Output di esempio:

```
{
  "id": "mysqlOrphanedRoutinesCheck",
  "title": "Check for orphaned routines in 5.7",
  "status": "OK",
  "description": "Error: The following routines have been orphaned. Schemas that
 they are referencing no longer exists.\nThey have to be cleaned up or the upgrade
 will fail.",
  "detectedProblems": [
      {
        "level": "Error",
        "dbObject": "dropped_db.get_version",
        "description": "is orphaned"
      }
  ]
},
```

Il controllo preliminare segnala che la stored procedure `get_version` nel database `dropped_db` è orfana.

Per eseguire la pulizia di questa procedura, è possibile creare nuovamente lo schema mancante.

```
mysql> create database dropped_db;
Query OK, 1 row affected (0.01 sec)
```

Una volta creato nuovamente lo schema, è possibile interrompere la procedura per consentire il proseguimento dell'aggiornamento.

```
{
  "id": "mysqlOrphanedRoutinesCheck",
  "title": "Check for orphaned routines in 5.7",
  "status": "OK",
  "detectedProblems": []
},
```

mysqlSchemaCheck

Livello di controllo preliminare: errore

I nomi delle tabelle nello schema **`mysql`** sono in conflitto con le nuove tabelle in MySQL 8.0

Il nuovo Atomic Data Dictionary introdotto in MySQL 8.0 memorizza tutti i metadati in un set di tabelle InnoDB interne nello schema `mysql`. Durante l'aggiornamento, le nuove tabelle interne del dizionario dei dati vengono create nello schema `mysql`. Per evitare collisioni di denominazione, che comporterebbero errori di aggiornamento, il controllo preliminare esamina tutti i nomi delle tabelle dello schema `mysql` per assicurarsi che nessuno dei nuovi nomi di tabella sia già in uso. In caso affermativo, viene restituito un errore e l'aggiornamento non può continuare.

Output di esempio:

```
{
  "id": "mysqlSchema",
  "title": "Table names in the mysql schema conflicting with new tables in the
 latest MySQL.",
  "status": "OK",
  "description": "Error: The following tables in mysql schema have names that will
 conflict with the ones introduced in the latest version. They must be renamed or
 removed before upgrading (use RENAME TABLE command). This may also entail changes
 to applications that use the affected tables.",
  "documentationLink": "https://dev.mysql.com/doc/refman/8.0/en/upgrade-before-you-
begin.html",
  "detectedProblems": [
      {
        "level": "Error",
        "dbObject": "mysql.tablespaces",
        "description": "Table name used in mysql schema.",
        "dbObjectType": "Table"
```

```
      }
  ]
}
```

Viene restituito un errore perché nello schema `mysql` è presente una tabella denominata `tablespaces`. Questo è uno dei nuovi nomi di tabella del dizionario dei dati interno di MySQL 8.0. È necessario rinominare o rimuovere tali tabelle prima dell'aggiornamento, utilizzando il comando `RENAME TABLE`.

nonNativePartitioningCheck

Livello di controllo preliminare: errore

Tabelle partizionate utilizzando motori con partizionamento non nativo

Secondo la documentazione di MySQL 8.0, due motori di archiviazione attualmente forniscono supporto nativo per il partizionamento: InnoDB e NDB. Di questi, solo InnoDB è supportato in Aurora MySQL versione 3, che è compatibile con MySQL 8.0. Ogni tentativo di creare tabelle partizionate in MySQL 8.0 utilizzando qualsiasi altro motore di archiviazione fallisce. Questo controllo preliminare cerca le tabelle nel cluster di database che utilizzano il partizionamento non nativo. Se ne vengono restituite alcune, è necessario rimuovere il partizionamento o convertire il motore di archiviazione in InnoDB.

Output di esempio:

```
{
  "id": "nonNativePartitioning",
  "title": "Partitioned tables using engines with non native partitioning",
  "status": "OK",
  "description": "Error: In the latest MySQL storage engine is responsible for
 providing its own partitioning handler, and the MySQL server no longer provides
 generic partitioning support. InnoDB and NDB are the only storage engines that
 provide a native partitioning handler that is supported in the latest MySQL. A
 partitioned table using any other storage engine must be altered—either to convert
 it to InnoDB or NDB, or to remove its partitioning—before upgrading the server,
 else it cannot be used afterwards.",
  "documentationLink": "https://dev.mysql.com/doc/refman/8.0/en/upgrading-from-
previous-series.html#upgrade-configuration-changes",
  "detectedProblems": [
      {
        "level": "Error",
        "dbObject": "test.partMyisamTable",
         "description": "MyISAM engine does not support native partitioning",
```

```
            "dbObjectType": "Table"
        }
    ]
}
```

In questo esempio, una tabella MyISAM utilizza il partizionamento, che richiede un'azione prima che l'aggiornamento possa procedere.

oldTemporalCheck

Livello di controllo preliminare: errore

Utilizzo di un tipo di dato temporale obsoleto

I dati temporali obsoleti sono le colonne di tipo temporale (come `TIMESTAMP` e `DATETIME`) create nelle versioni di MySQL 5.5 e precedenti. In MySQL 8.0, il supporto per questi tipi di dati temporali obsoleti è stato rimosso, il che significa che gli aggiornamenti in loco da MySQL 5.7 a 8.0 non sono possibili se il database contiene questi tipi di dati temporali obsoleti. Per risolvere questo problema, è necessario creare nuovamente tutte le tabelle contenenti questi tipi di dati temporali obsoleti prima di procedere con l'aggiornamento.

Per ulteriori informazioni sulla definizione come obsoleti dei suddetti tipi di dati temporali in MySQL 5.7, consulta questo blog. Per ulteriori informazioni sulla rimozione dei tipi di dati temporali obsoleti in MySQL 8.0, consulta questo blog.

> **Note**
>
> Prima di creare nuovamente i tablespace, consulta le operazioni Online DDL nella documentazione di MySQL per comprendere gli effetti del blocco e dello spostamento dei dati sulle transazioni in primo piano.

Output di esempio:

```
{
  "id": "oldTemporalCheck",
  "title": "Usage of old temporal type",
  "status": "OK",
  "description": "Error: Following table columns use a deprecated and no longer
 supported temporal disk storage format. They must be converted to the new format
 before upgrading. It can by done by rebuilding the table using 'ALTER TABLE
 <table_name> FORCE' command",
```

```
  "documentationLink": "https://dev.mysql.com/blog-archive/mysql-8-0-removing-
support-for-old-temporal-datatypes/",
  "detectedProblems": [
      {
        "level": "Error",
        "dbObject": "test.55_temporal_table.timestamp_column",
        "description": "timestamp /* 5.5 binary format */",
        "dbObjectType": "Column"
      }
  ]
},
```

Viene segnalato un errore per la colonna `timestamp_column` della tabella `test.55_temporal_table`, poiché utilizza un formato di archiviazione su disco con dati temporali obsoleti che non è più supportato.

Per risolvere questo problema e consentire il proseguimento dell'aggiornamento, crea nuovamente la tabella per convertire il formato di archiviazione su disco temporale obsoleto in quello nuovo introdotto in MySQL 5.6. Per ulteriori informazioni e prerequisiti prima di procedere, consulta Converting between 3-byte and 4-byte Unicode character sets nella documentazione di MySQL.

L'esecuzione del comando seguente per ricostruire le tabelle menzionate in questo controllo preliminare converte i tipi di dati temporali obsoleti nel formato più recente con una precisione di frazioni di secondo.

```
ALTER TABLE ... ENGINE=InnoDB;
```

Per ulteriori informazioni su come creare nuovamente le tabelle, consulta ALTER TABLE statement nella documentazione di MySQL.

Dopo aver creato nuovamente la tabella in questione e riavviato l'aggiornamento, il controllo di compatibilità viene superato e l'aggiornamento può procedere.

```
{
  "id": "oldTemporalCheck",
  "title": "Usage of old temporal type",
  "status": "OK",
  "detectedProblems": []
}
```

partitionedTablesInSharedTablespaceCheck

Livello di controllo preliminare: errore

Utilizzo di tabelle partizionate in tablespace condivisi

A partire da MySQL 8.0.13, il supporto per l'inserimento di partizioni di tabelle in tablespace condivisi è stato rimosso. Prima dell'aggiornamento, sposta tali tabelle dai tablespace condivisi ai tablespace file-per-table.

> **Note**
>
> Prima di creare nuovamente i tablespace, consulta le operazioni di partizionamento nella documentazione di MySQL per comprendere gli effetti del blocco e dello spostamento dei dati sulle transazioni in primo piano.

Output di esempio:

```
{
  "id": "partitionedTablesInSharedTablespaceCheck",
  "title": "Usage of partitioned tables in shared tablespaces",
  "status": "OK",
  "description": "Error: The following tables have partitions in shared tablespaces.
 They need to be moved to file-per-table tablespace before upgrading. You can do
 this by running query like 'ALTER TABLE table_name REORGANIZE PARTITION X INTO
 (PARTITION X VALUES LESS THAN (30) TABLESPACE=innodb_file_per_table);'",
  "documentationLink": "https://dev.mysql.com/doc/refman/8.0/en/mysql-
nutshell.html#mysql-nutshell-removals",
  "detectedProblems": [
      {
        "level": "Error",
        "dbObject": "test.partInnoDBTable",
        "description": "Partition p1 is in shared tablespace innodb",
        "dbObjectType": "Table"
      }
  ]
}
```

Il controllo preliminare non riesce perché la partizione `p1` dalla tabella `test.partInnoDBTable` si trova nel tablespace di sistema.

Per risolvere questo problema, crea nuovamente la tabella `test.partInnodbTable` inserendo la partizione `p1` che causa il problema in un tablespace file-per-table.

```
mysql > ALTER TABLE partInnodbTable REORGANIZE PARTITION p1
    ->   INTO (PARTITION p1 VALUES LESS THAN ('2014-01-01')
 TABLESPACE=innodb_file_per_table);
Query OK, 0 rows affected, 1 warning (0.02 sec)
Records: 0  Duplicates: 0  Warnings: 0
```

A questo punto, il controllo preliminare viene superato.

```
{
  "id": "partitionedTablesInSharedTablespaceCheck",
  "title": "Usage of partitioned tables in shared tablespaces",
  "status": "OK",
  "detectedProblems": []
}
```

removedFunctionsCheck

Livello di controllo preliminare: errore

Utilizzo di funzioni rimosse dalla versione più recente di MySQL

In MySQL 8.0, sono state rimosse alcune funzioni integrate. Questo controllo preliminare esamina il database alla ricerca di oggetti che potrebbero utilizzare queste funzioni. Se vengono trovati tali oggetti, viene restituito un errore. È necessario risolvere i problemi prima di tentare di eseguire nuovamente l'aggiornamento.

La maggior parte delle funzioni rimosse sono funzioni spaziali, che sono state sostituite con funzioni `ST_*` equivalenti. In questi casi, occorre modificare gli oggetti del database in modo da utilizzare la nuova denominazione delle procedure. Per ulteriori informazioni, consulta Features Removed in MySQL 8.0 nella documentazione MySQL.

Output di esempio:

```
{
  "id": "removedFunctionsCheck",
  "title": "Usage of removed functions",
  "status": "OK",
```

```
  "description": "Error: The following DB objects use functions that were removed
 in the latest MySQL version. Please make sure to update them to use supported
 alternatives before upgrade.",
  "documentationLink": "https://dev.mysql.com/doc/refman/8.0/en/mysql-
nutshell.html#mysql-nutshell-removals",
  "detectedProblems": [
      {
        "level": "Error",
        "dbObject": "test.GetLocationsInPolygon",
        "description": "PROCEDURE uses removed function POLYGONFROMTEXT (consider
 using ST_POLYGONFROMTEXT instead)",
        "dbObjectType": "Routine"
      },
      {
        "level": "Error",
        "dbObject": "test.InsertLocation",
        "description": "PROCEDURE uses removed function POINTFROMTEXT (consider
 using ST_POINTFROMTEXT instead)",
        "dbObjectType": "Routine"
      }
  ]
},
```

Il controllo preliminare segnala che la stored procedure `test.GetLocationsInPolygon` utilizza due funzioni rimosse: POLYGONFROMTEXT e POINTFROMTEXT. Suggerisce inoltre di utilizzare le nuove funzioni ST_POLYGONFROMTEXT e ST_POINTFROMTEXT come sostitute. Dopo aver creato nuovamente la procedura utilizzando i suggerimenti, il controllo preliminare viene completato correttamente.

```
{
  "id": "removedFunctionsCheck",
  "title": "Usage of removed functions",
  "status": "OK",
  "detectedProblems": []
},
```

> **Note**
>
> Sebbene nella maggior parte dei casi le funzioni obsolete siano sostituite direttamente, assicurati di testare l'applicazione e di consultare la documentazione per eventuali cambiamenti nel comportamento dovuti alla modifica.

routineSyntaxCheck

Livello di controllo preliminare: errore

Controllo della sintassi MySQL per oggetti simili a routine

MySQL 8.0 ha introdotto delle parole chiave riservate che non erano riservate in precedenza. Il controllo preliminare dell'aggiornamento valuta l'uso di parole chiave riservate nei nomi degli oggetti del database, nonché nelle relative definizioni e nel relativo corpo. Se il controllo rileva l'uso di parole chiave riservate negli oggetti del database, ad esempio stored procedure, funzioni, eventi e trigger, l'aggiornamento non riesce e viene pubblicato un errore nel file `upgrade-prechecks.log`. Per risolvere il problema, è necessario aggiornare le definizioni degli oggetti e racchiudere tali riferimenti tra virgolette singole (') prima dell'aggiornamento. Per ulteriori informazioni sull'escape delle parole riservate in MySQL, consulta String literals nella documentazione di MySQL.

In alternativa, è possibile modificare il nome specificandone uno diverso, il che potrebbe richiedere modifiche all'applicazione.

Output di esempio:

```
{
  "id": "routineSyntaxCheck",
  "title": "MySQL syntax check for routine-like objects",
  "status": "OK",
  "description": "The following objects did not pass a syntax check with the latest
 MySQL grammar. A common reason is that they reference names that conflict with new
 reserved keywords. You must update these routine definitions and `quote` any such
 references before upgrading.",
  "documentationLink": "https://dev.mysql.com/doc/refman/en/keywords.html",
  "detectedProblems": [
      {
         "level": "Error",
         "dbObject": "test.select_res_word",
         "description": "at line 2,18: unexpected token 'except'",
         "dbObjectType": "Routine"
      }
  ]
}
```

Per risolvere il problema, consulta la definizione della routine.

```
SHOW CREATE PROCEDURE test.select_res_word\G

*************************** 1. row ***************************
           Procedure: select_res_word
            sql_mode:
 ONLY_FULL_GROUP_BY,STRICT_TRANS_TABLES,NO_ZERO_IN_DATE,NO_ZERO_DATE,ERROR_FOR_DIVISION_BY_Z
    Create Procedure: CREATE PROCEDURE 'select_res_word'()
BEGIN
    SELECT * FROM except;
END
character_set_client: utf8
collation_connection: utf8_general_ci
  Database Collation: latin1_swedish_ci
1 row in set (0.00 sec)
```

La procedura utilizza una tabella denominata `except`, che è una nuova parola chiave in MySQL 8.0. Crea nuovamente la procedura evitando il valore letterale della stringa.

```
> drop procedure if exists select_res_word;
Query OK, 0 rows affected (0.00 sec)

> DELIMITER $$
 > CREATE PROCEDURE select_res_word()
    -> BEGIN
    ->     SELECT * FROM 'except';
    -> END$$
Query OK, 0 rows affected (0.00 sec)

 > DELIMITER ;
```

A questo punto, il controllo preliminare viene superato.

```
{
  "id": "routineSyntaxCheck",
  "title": "MySQL syntax check for routine-like objects",
  "status": "OK",
  "detectedProblems": []
}
```

schemaInconsistencyCheck

Livello di controllo preliminare: errore

Incongruenze dello schema derivanti dalla rimozione o dal danneggiamento dei file

Come descritto in precedenza, MySQL 8.0 ha introdotto l'Atomic Data Dictionary, che archivia tutti i metadati in un set di tabelle InnoDB interne nello schema `mysql`. Questa nuova architettura offre un modo transazionale e conforme agli standard ACID per gestire i metadati del database, risolvendo il problema di "Atomic DDL" derivante dal precedente approccio basato su file. Prima di MySQL 8.0, era possibile che gli oggetti dello schema diventassero orfani se un'operazione DDL veniva interrotta inaspettatamente. La migrazione dei metadati basati su file nelle nuove tabelle Atomic Data Dictionary durante l'aggiornamento garantisce che tali oggetti dello schema orfani non siano presenti nell'istanza database. Se vengono rilevati oggetti orfani, l'aggiornamento non riesce.

Per contribuire al rilevamento di questi oggetti orfani prima di iniziare l'aggiornamento, viene eseguito il controllo preliminare `schemaInconsistencyCheck` per garantire che tutti gli oggetti di metadati del dizionario di dati siano sincronizzati. Se vengono rilevati oggetti di metadati orfani, l'aggiornamento non procede. Per procedere con l'aggiornamento, esegui la pulizia degli oggetti di metadati orfani.

Se riscontri errori con questo controllo preliminare, apri un caso con il Supporto AWS per richiedere che l'incoerenza dei metadati venga risolta. In alternativa, puoi tentare nuovamente l'aggiornamento eseguendo un dump logico, quindi effettuando il ripristino su un nuovo cluster di database Aurora MySQL versione 3.

Output di esempio:

```
{
  "id": "schemaInconsistencyCheck",
  "title": "Schema inconsistencies resulting from file removal or corruption",
  "status": "OK",
  "description": "Error: Following tables show signs that either table datadir
 directory or frm file was removed/corrupted. Please check server logs, examine
 datadir to detect the issue and fix it before upgrade",
  "detectedProblems": [
      {
        "level": "Error",
        "dbObject": "test.schemaInconsistencyCheck_failure",
        "description": "present in INFORMATION_SCHEMA's INNODB_SYS_TABLES table but
 missing from TABLES table"
      }
  ]
}
```

Il controllo preliminare segnala che la tabella `test.schemaInconsistencyCheck_failure` contiene metadati incoerenti. In questo caso, la tabella esiste nei metadati del motore di archiviazione InnoDB (`information_schema.INNODB_SYS_TABLES`), ma non nei metadati di MySQL (`information_schema.TABLES`).

Controlli preliminari Aurora MySQL che restituiscono errori

I seguenti controlli preliminari sono specifici di Aurora MySQL:

- auroraCheckDDLRecovery
- auroraCheckRdsUpgradePrechecksTable
- auroraFODUpgradeCheck
- auroraGetDanglingFulltextIndex
- auroraUpgradeCheckForDatafilePathInconsistency
- auroraUpgradeCheckForFtsSpaceIdZero
- auroraUpgradeCheckForIncompleteXATransactions
- auroraUpgradeCheckForInstanceLimit
- auroraUpgradeCheckForInternalUsers
- auroraUpgradeCheckForInvalidUtf8mb3CharacterStringInViews
- auroraUpgradeCheckForInvalidUtf8mb3ColumnComments
- auroraUpgradeCheckForInvalidUtf8mb3IndexComments
- auroraUpgradeCheckForInvalidUtf8mb3TableComments
- auroraUpgradeCheckForMasterUser
- auroraUpgradeCheckForPrefixIndexOnGeometryColumns
- auroraUpgradeCheckForSpecialCharactersInProcedures
- auroraUpgradeCheckForSysSchemaObjectTypeMismatch
- auroraUpgradeCheckForViewColumnNameLength
- auroraUpgradeCheckIndexLengthLimitOnTinyColumns
- auroraUpgradeCheckMissingInnodbMetadataForMysqlHostTable

auroraCheckDDLRecovery

Livello di controllo preliminare: errore

Verifica la presenza di artefatti relativi alla funzionalità di ripristino Aurora DDL

Nell'ambito dell'implementazione della funzionalità di ripristino DDL (Data Definition Language) in Aurora MySQL, i metadati delle istruzioni DDL inflight vengono mantenuti nelle tabelle `ddl_log_md_table` e `ddl_log_table` nello schema `mysql`. L'implementazione di Aurora della funzionalità di ripristino DDL non è supportata dalla versione 3 in poi, poiché la funzionalità fa parte della nuova implementazione Atomic Data Dictionary in MySQL 8.0. Se durante i controlli di compatibilità sono in esecuzione delle istruzioni DDL, questo controllo preliminare potrebbe avere esito negativo. Si consiglia di eseguire l'aggiornamento mentre non è in esecuzione alcuna istruzione DDL.

Se questo controllo preliminare ha esito negativo senza che siano in esecuzione una o più istruzioni DDL, apri un caso con il Supporto AWS per richiedere che l'incoerenza dei metadati venga risolta. In alternativa, puoi tentare nuovamente l'aggiornamento eseguendo un dump logico, quindi effettuando il ripristino su un nuovo cluster di database Aurora MySQL versione 3.

Se sono in esecuzione delle istruzioni DDL, l'output del controllo preliminare stampa il seguente messaggio:

```
“There are DDL statements in process. Please allow DDL statements to finish before
 upgrading.”
```

Output di esempio:

```
{
  "id": "auroraCheckDDLRecovery",
  "title": "Check for artifacts related to Aurora DDL recovery feature",
  "status": "OK",
  "description": "Aurora implementation of DDL recovery is not supported from 3.x
 onwards. This check verifies that the database do not have artifacts realted to the
 feature",
  "detectedProblems": [
      {
        "level": "Error",
        "dbObject": "mysql.ddl_log_md_table",
        "description": "Table mysql.ddl_log_md_table is not empty. Your database has
 pending DDL recovery operations. Reachout to AWS support for assistance."
      },
      {
        "level": "Error",
        "dbObject": "mysql.ddl_log_table",
```

```
        "description": "Table mysql.ddl_log_table is not empty. Your database has
 pending DDL recovery operations. Reachout to AWS support for assistance."
      },
      {
        "level": "Error",
        "dbObject": "information_schema.processlist",
        "description": "There are DDL statements in process. Please allow DDL
 statements to finish before upgrading."
      }
  ]
}
```

Il controllo preliminare ha restituito un errore dovuto a un DDL inflight eseguito contemporaneamente ai controlli di compatibilità. Si consiglia di tentare di eseguire nuovamente l'aggiornamento senza che siano in esecuzione le istruzioni DDL.

auroraCheckRdsUpgradePrechecksTable

Livello di controllo preliminare: errore

Verifica l'esistenza della tabella **`mysql.rds_upgrade_prechecks`**

Si tratta di un controllo preliminare solo interno eseguito dal servizio RDS. Eventuali errori vengono gestiti automaticamente durante l'aggiornamento e possono essere ignorati senza problemi.

Se riscontri errori con questo controllo preliminare, apri un caso con il Supporto AWS per richiedere che l'incoerenza dei metadati venga risolta. In alternativa, puoi tentare nuovamente l'aggiornamento eseguendo un dump logico, quindi effettuando il ripristino su un nuovo cluster di database Aurora MySQL versione 3.

```
{
  "id": "auroraCheckRdsUpgradePrechecksTable",
  "title": "Check existence of mysql.rds_upgrade_prechecks table",
  "status": "OK",
  "detectedProblems": []
}
```

auroraFODUpgradeCheck

Livello di controllo preliminare: errore

Verifica la presenza di artefatti relativi alla funzionalità Fast DDL di Aurora

L'ottimizzazione Fast DDL è stata introdotta in modalità Lab in Aurora MySQL versione 2 per migliorare l'efficienza di alcune operazioni DDL. In Aurora MySQL versione 3, la modalità Lab è stata rimossa e l'implementazione Fast DDL è stata sostituita dalla funzionalità MySQL 8.0 denominata Instant DDL.

Prima dell'aggiornamento ad Aurora MySQL versione 3, tutte le tabelle che utilizzano Fast DDL in modalità Lab dovranno essere create nuovamente eseguendo il comando `OPTIMIZE TABLE` o `ALTER TABLE ... ENGINE=InnoDB` per garantire la compatibilità con Aurora MySQL versione 3.

Questo controllo preliminare restituisce un elenco di tutte le tabelle di questo tipo. Dopo che le tabelle restituite sono state create nuovamente, è possibile tentare di eseguire nuovamente l'aggiornamento.

Output di esempio:

```
{
  "id": "auroraFODUpgradeCheck",
  "title": "Check for artifacts related to Aurora fast DDL feature",
  "status": "OK",
  "description": "Aurora fast DDL is not supported from 3.x onwards. This check
 verifies that the database does not have artifacts related to the feature",
  "documentationLink": "https://docs.aws.amazon.com/AmazonRDS/latest/
AuroraUserGuide/AuroraMySQL.Managing.FastDDL.html#AuroraMySQL.Managing.FastDDL-v2",
  "detectedProblems": [
      {
        "level": "Error",
        "dbObject": "test.test",
        "description": "Your table has pending Aurora fast DDL operations. Run
 'OPTIMIZE TABLE <table name>' for the table to apply all the pending DDL updates.
 Then try the upgrade again."
      }
  ]
}
```

Il controllo preliminare segnala che nella tabella `test.test` sono presenti operazioni Fast DDL in sospeso.

Per consentire il proseguimento dell'aggiornamento, è possibile creare nuovamente la tabella, quindi tentare di eseguire di nuovo l'aggiornamento.

> **Note**
>
> Prima di creare nuovamente i tablespace, consulta le operazioni Online DDL nella documentazione di MySQL per comprendere gli effetti del blocco e dello spostamento dei dati sulle transazioni in primo piano.

```
mysql> optimize table test.test;
+-----------+----------+----------
+-------------------------------------------------------------------+
| Table     | Op       | Msg_type | Msg_text
                  |
+-----------+----------+----------
+-------------------------------------------------------------------+
| test.test | optimize | note     | Table does not support optimize, doing recreate
 + analyze instead |
| test.test | optimize | status   | OK
                  |
+-----------+----------+----------
+-------------------------------------------------------------------+
2 rows in set (0.04 sec)
```

Dopo aver creato nuovamente la tabella, il controllo preliminare ha esito positivo.

```
{
  "id": "auroraFODUpgradeCheck",
  "title": "Check for artifacts related to Aurora fast DDL feature",
  "status": "OK",
  "detectedProblems": []
}
```

auroraGetDanglingFulltextIndex

Livello di controllo preliminare: errore

Tabelle con riferimento all'indice **FULLTEXT** dangling

Prima di MySQL 5.6.26, era possibile che, dopo aver eliminato un indice di ricerca full-text, le colonne nascoste `FTS_DOC_ID` e `FTS_DOC_ID_INDEX` diventassero orfane. Per ulteriori informazioni, consulta Bug #76012.

Se sono presenti tabelle create in versioni precedenti di MySQL per cui è stato riscontrato quanto descritto sopra, l'aggiornamento ad Aurora MySQL versione 3 può non riuscire. Questo controllo preliminare verifica che nel cluster di database non esistano tali indici full-text orfani o dangling prima dell'aggiornamento a MySQL 8.0. Se questo controllo preliminare ha esito negativo, crea nuovamente tutte le tabelle che contengono tali indici full-text dangling.

Output di esempio:

```
{
  "id": "auroraGetDanglingFulltextIndex",
  "title": "Tables with dangling FULLTEXT index reference",
  "status": "OK",
  "description": "Error: The following tables contain dangling FULLTEXT index which
 is not supported. It is recommended to rebuild the table before upgrade.",
  "detectedProblems": [
      {
        "level": "Error",
        "dbObject": "test.table_with_fts_index",
        "description": "Table `test.table_with_fts_index` contains dangling FULLTEXT
 index. Kindly recreate the table before upgrade."
      }
  ]
},
```

Il controllo preliminare segnala un errore per la tabella `test.table_with_fts_index` perché contiene un indice full-text dangling. Per consentire il proseguimento dell'aggiornamento, crea nuovamente la tabella per eseguire la pulizia delle tabelle ausiliarie con indice full-text. Utilizza `OPTIMIZE TABLE test.table_with_fts_index` o `ALTER TABLE test.table_with_fts_index, ENGINE=INNODB`.

Dopo aver creato nuovamente la tabella, il controllo preliminare ha esito positivo.

```
{
  "id": "auroraGetDanglingFulltextIndex",
  "title": "Tables with dangling FULLTEXT index reference",
  "status": "OK",
  "detectedProblems": []
},
```

auroraUpgradeCheckForDatafilePathInconsistency

Livello di controllo preliminare: errore

Verifica l'eventuale presenza di incoerenze relative al percorso di file **`ibd`**

Questo controllo preliminare si applica solo ad Aurora MySQL versione 3.03.0 e precedenti. Se si verifica un errore con questo controllo preliminare, esegui l'aggiornamento ad Aurora MySQL versione 3.04 o successive.

Output di esempio:

```
{
  "id": "auroraUpgradeCheckForDatafilePathInconsistency",
  "title": "Check for inconsistency related to ibd file path.",
  "status": "OK",
  "detectedProblems": []
}
```

auroraUpgradeCheckForFtsSpaceIdZero

Livello di controllo preliminare: errore

Verifica la presenza di un indice full-text con ID di spazio pari a zero

In MySQL, quando si aggiunge un indice full-text a una tabella InnoDB, vengono creati alcuni tablespace di indici ausiliari. A causa di un bug nelle versioni precedenti di MySQL, poi corretto nella versione 5.6.20, era possibile che queste tabelle di indici ausiliari fossero state create nel tablespace di sistema, anziché nel tablespace InnoDB.

Se esistono tablespace ausiliari di questo tipo, l'aggiornamento non riesce. Crea nuovamente gli indici full-text indicati nell'errore del controllo preliminare, quindi tenta di eseguire di nuovo l'aggiornamento.

Output di esempio:

```
{
  "id": "auroraUpgradeCheckForFtsSpaceIdZero",
  "title": "Check for fulltext index with space id as zero",
  "status": "OK",
  "description": "The auxiliary tables of FTS indexes on the table are created
 in system table-space. Due to this DDL queries executed on MySQL8.0 shall cause
 database unavailability. To avoid that, drop and recreate all the FTS indexes on
 the table or rebuild the table using ALTER TABLE query before the upgrade.",
  "detectedProblems": [
      {
        "level": "Error",
```

```
        "dbObject": "test.fts_space_zero_check",
        "description": " The auxiliary tables of FTS indexes on the table
 'test.fts_space_zero_check' are created in system table-space due to https://
bugs.mysql.com/bug.php?id=72132. In MySQL8.0, DDL queries executed on this table
 shall cause database unavailability. To avoid that, drop and recreate all the
 FTS indexes on the table or rebuild the table using ALTER TABLE query before the
 upgrade."
      }
  ]
},
```

Il controllo preliminare segnala un errore per la tabella `test.fts_space_zero_check`, poiché include tabelle ausiliarie di ricerca full-text (FTS) nel tablespace di sistema.

Dopo aver eliminato e creato nuovamente gli indici FTS associati a questa tabella, il controllo preliminare ha esito positivo.

> **Note**
>
> Prima di creare nuovamente i tablespace, consulta le operazioni di partizionamento nella documentazione di MySQL per comprendere gli effetti del blocco e dello spostamento dei dati sulle transazioni in primo piano.

```
{
 "id": "auroraUpgradeCheckForFtsSpaceIdZero",
 "title": "Check for fulltext index with space id as zero",
 "status": "OK",
 "detectedProblems": []
}
```

auroraUpgradeCheckForIncompleteXATransactions

Livello di controllo preliminare: errore

Verifica la presenza di transazioni XA in stato preparato

Durante l'esecuzione del processo di aggiornamento a una versione principale, è essenziale che l'istanza database Aurora MySQL versione 2 sia sottoposta a un arresto pulito. Ciò garantisce che tutte le transazioni vengano eseguite o ripristinate e che InnoDB abbia eliminato tutti i record del log di undo. Poiché il rollback delle transazioni è necessario, se il database contiene transazioni

XA in stato preparato, può impedire il proseguimento dell'arresto pulito. Per questo motivo, se vengono rilevate transazioni XA preparate, l'aggiornamento non potrà procedere finché non si interverrà per eseguirne il commit o il rollback.

Per ulteriori informazioni sulla ricerca di transazioni XA in stato preparato utilizzando `XA RECOVER`, consulta XA transaction SQL statements nella documentazione di MySQL. Per ulteriori informazioni sugli stati delle transazioni XA, consulta XA transaction states nella documentazione di MySQL.

Output di esempio:

```
{
  "id": "auroraUpgradeCheckForIncompleteXATransactions",
  "title": "Pre-checks for XA Transactions in prepared state.",
  "status": "OK",
  "description": "Your cluster currently has XA transactions in the prepared state.
 To proceed with the upgrade, commit or rollback these transactions.",
  "detectedProblems": [
      {
        "level": "Error",
        "dbObject": "all",
        "description": "Your cluster currently has XA transactions in the prepared
 state. To proceed with the upgrade, commit or rollback these transactions."
      }
  ]
}
```

Questo controllo preliminare segnala un errore perché vi sono transazioni in stato preparato di cui deve essere eseguito il commit o il rollback.

Dopo aver effettuato l'accesso al database, puoi controllare la tabella information_schema.innodb_trx e l'output `XA RECOVER` per ulteriori informazioni.

> **Important**
>
> Prima di eseguire il commit o il rollback di una transazione, è consigliabile consultare la documentazione di MySQL e i requisiti dell'applicazione.

```
mysql> select trx_started,
    trx_mysql_thread_id,
```

```
    trx_id,trx_state,
    trx_operation_state,
    trx_rows_modified,
    trx_rows_locked
from
    information_schema.innodb_trx;
+---------------------+--------------------+---------+-----------
+---------------------+-------------------+-----------------+
| trx_started         | trx_mysql_thread_id | trx_id  | trx_state |
 trx_operation_state | trx_rows_modified | trx_rows_locked |
+---------------------+--------------------+---------+-----------
+---------------------+-------------------+-----------------+
| 2024-08-12 01:09:39 |                   0 | 2849470 | RUNNING   | NULL
    |                 1 |               0 |
+---------------------+--------------------+---------+-----------
+---------------------+-------------------+-----------------+
1 row in set (0.00 sec)

mysql> xa recover;
+----------+--------------+--------------+--------+
| formatID | gtrid_length | bqual_length | data   |
+----------+--------------+--------------+--------+
|        1 |            6 |            0 | xatest |
+----------+--------------+--------------+--------+
1 row in set (0.00 sec)
```

In questo esempio, viene eseguito il rollback della transazione preparata.

```
mysql> XA ROLLBACK 'xatest';
Query OK, 0 rows affected (0.00 sec)
v
mysql> xa recover;
Empty set (0.00 sec)
```

Dopo il rollback della transazione XA, il controllo preliminare ha esito positivo.

```
{
  "id": "auroraUpgradeCheckForIncompleteXATransactions",
  "title": "Pre-checks for XA Transactions in prepared state.",
  "status": "OK",
  "detectedProblems": []
}
```

auroraUpgradeCheckForInstanceLimit

Livello di controllo preliminare: errore

Verifica se l'aggiornamento è supportato nella classe di istanza corrente

L'esecuzione di un aggiornamento in loco da Aurora MySQL versione 2.12.0 o 2.12.1, in cui la classe di istanza database di scrittura è db.r6i.32xlarge, al momento non è supportata. In questo caso, il controllo preliminare restituisce un errore. Per consentire il proseguimento dell'aggiornamento, puoi modificare la classe dell'istanza database impostandola su db.r6i.24xlarge o una versione precedente. Altrimenti puoi eseguire l'aggiornamento ad Aurora MySQL versione 2.12.2 o successiva, in cui l'aggiornamento in loco ad Aurora MySQL versione 3 è supportato su db.r6i.32xlarge.

Output di esempio:

```
{
  "id": "auroraUpgradeCheckForInstanceLimit",
  "title": "Checks if upgrade is supported on the current instance class",
  "status": "OK",
  "description": "Upgrade from Aurora Version 2.12.0 and 2.12.1 may fail for 32.xl
 and above instance class.",
  "detectedProblems": [
      {
        "level": "Error",
        "dbObject": "all",
        "description": "Upgrade is not supported on this instance size for Aurora
 MySql Version 2.12.1. Before upgrading to Aurora MySql 3, please consider either:
 1. Changing the instance class to 24.xl or lower. -or- 2. Upgrading to patch
 version 2.12.2 or higher."
      }
  ]
},
```

Il controllo preliminare restituisce un errore perché l'istanza database di scrittura utilizza la classe di istanza db.r6i.32xlarge ed è in esecuzione su Aurora MySQL versione 2.12.1.

auroraUpgradeCheckForInternalUsers

Livello di controllo preliminare: errore

Verifica la presenza di utenti interni 8.0

Questo controllo preliminare si applica solo ad Aurora MySQL versione 3.03.0 e precedenti. Se si verifica un errore con questo controllo preliminare, esegui l'aggiornamento ad Aurora MySQL versione 3.04 o successive.

Output di esempio:

```
{
  "id": "auroraUpgradeCheckForInternalUsers",
  "title": "Check for 8.0 internal users.",
  "status": "OK",
  "detectedProblems": []
}
```

auroraUpgradeCheckForInvalidUtf8mb3CharacterStringInViews

Livello di controllo preliminare: errore

Verifica la presenza di caratteri utf8mb3 non validi nella definizione della vista

Questo controllo preliminare identifica le viste che contengono commenti con codifica dei caratteri non validi `utf8mb3`. In MySQL 8.0, viene applicata una convalida più rigorosa alla codifica dei caratteri nei metadati, inclusi i commenti delle viste. Se la definizione di una vista contiene caratteri che non sono validi nel set di caratteri `utf8mb3`, l'aggiornamento non riesce.

Per risolvere questo problema, modifica la definizione della vista per rimuovere o sostituire eventuali caratteri non BMP prima di tentare l'aggiornamento.

Output di esempio:

```
{
"id": "auroraUpgradeCheckForInvalidUtf8mb3CharacterStringInViews",
"title": "Check for invalid utf8mb3 character string.",
"status": "OK",
"description": "Definition of following view(s) has/have invalid utf8mb3 character
 string.",
"detectedProblems": [
        {
            "level": "Error",
            "dbObject": "precheck.utf8mb3_invalid_char_view",
            "description": "Definition of view precheck.utf8mb3_invalid_char_view
 contains an invalid utf8mb3 character string. This is due to https://
bugs.mysql.com/bug.php?id=110177. To fix the inconsistency, we recommend you to
```

```
 modify the view definition to not use non-BMP characters and try the upgrade
 again."
        }
    ]
},
```

Il controllo preliminare segnala che la definizione della vista `utf8mb3_invalid_char_view` contiene caratteri `utf8mb3` non validi nella relativa definizione.

Per risolvere questo problema, identifica la vista che contiene i caratteri non supportati e aggiorna i commenti. Innanzitutto, esamina la struttura della vista e identifica i commenti.

```
MySQL> SHOW CREATE VIEW precheck.utf8mb3_invalid_char_view\G
*************************** 1. row ***************************
                View: utf8mb3_invalid_char_view
         Create View: CREATE ALGORITHM=UNDEFINED DEFINER=`admin`@`%` SQL SECURITY
 DEFINER VIEW `utf8mb3_invalid_char_view` AS select 'This row contains a dolphin #'
 AS `message`
character_set_client: utf8
collation_connection: utf8_general_ci
1 row in set, 1 warning (0.00 sec)
```

Dopo aver identificato la vista che contiene l'errore, sostituisci la vista con l'istruzione `CREATE OR REPLACE VIEW`.

```
MySQL> CREATE OR REPLACE VIEW precheck.utf8mb3_invalid_char_view AS select 'This
 view definition to not use non-BMP characters' AS message;
```

Dopo aver aggiornato tutte le definizioni di viste che contengono caratteri non supportati, il controllo preliminare viene superato e l'aggiornamento può procedere.

```
{
"id": "auroraUpgradeCheckForInvalidUtf8mb3ColumnComments",
"title": "Check for invalid utf8mb3 column comments.",
"status": "OK",
"detectedProblems": []
}
```

auroraUpgradeCheckForInvalidUtf8mb3ColumnComments

Livello di controllo preliminare: errore

Verifica la presenza di commenti non validi nella colonna utf8mb3

Questo controllo preliminare identifica le tabelle che contengono commenti delle colonne con codifica dei caratteri non validi `utf8mb3`. In MySQL 8.0, viene applicata una convalida più rigorosa alla codifica dei caratteri nei metadati, inclusi i commenti delle colonne. Se un commento di colonna contiene caratteri che non sono validi nel set di caratteri utf8mb3, l'aggiornamento non riuscirà.

Per risolvere questo problema, è necessario modificare i commenti delle colonne per rimuovere o sostituire eventuali caratteri non BMP prima di tentare l'aggiornamento. È possibile utilizzare l'istruzione `ALTER TABLE` per aggiornare i commenti della colonna.

Output di esempio:

```
{
  "id": "auroraUpgradeCheckForInvalidUtf8mb3ColumnComments",
  "title": "Check for invalid utf8mb3 column comments.",
  "status": "OK",
  "description": "Following table(s) has/have invalid utf8mb3 comments on the
 column/columns.",
  "detectedProblems": [
      {
        "level": "Error",
        "dbObject": "test.t2",
        "description": "Table test.t2 has invalid utf8mb3 comments in it's column/
columns. This is due to non-BMP characters in the comment field. To fix the
 inconsistency, we recommend you to modify comment fields to not use non-BMP
 characters and try the upgrade again."
      }
  ]
}
```

Il controllo preliminare segnala che la tabella `test.t2` contiene caratteri `utf8mb3` non validi in uno o più commenti della colonna, in particolare a causa della presenza di caratteri non BMP.

Per risolvere il problema, è possibile identificare le colonne problematiche e aggiornare i relativi commenti. Innanzitutto, esamina la struttura della tabella per identificare le colonne con commenti:

```
mysql> SHOW CREATE TABLE test.t2\G
```

Dopo aver identificato le colonne con commenti problematici, aggiornale utilizzando l'istruzione `ALTER TABLE`. Per esempio:

```
mysql> ALTER TABLE test.t2 MODIFY COLUMN column_name data_type COMMENT 'Updated
 comment without non-BMP characters';
```

In alternativa, è possibile rimuovere completamente il commento:

```
mysql> ALTER TABLE test.t2 MODIFY COLUMN column_name data_type COMMENT '';
```

Dopo aver aggiornato tutti i commenti delle colonne problematiche, il controllo preliminare verrà superato e l'aggiornamento potrà procedere:

```
{
  "id": "auroraUpgradeCheckForInvalidUtf8mb3ColumnComments",
  "title": "Check for invalid utf8mb3 column comments.",
  "status": "OK",
  "detectedProblems": []
}
```

> **Note**
>
> Prima di modificare i commenti delle colonne, assicurati che eventuale documentazione o codice dell'applicazione che si basa su questi commenti siano aggiornati di conseguenza. Valuta la possibilità di eseguire la migrazione al set di caratteri utf8mb4 per un migliore supporto Unicode se l'applicazione richiede caratteri non BMP.

auroraUpgradeCheckForInvalidUtf8mb3IndexComments

Livello di controllo preliminare: errore

Verifica la presenza di commenti non validi nell'indice utf8mb3

Questo controllo preliminare identifica le tabelle che contengono commenti degli indici con codifica dei caratteri non validi `utf8mb3`. In MySQL 8.0, viene applicata una convalida più rigorosa alla codifica dei caratteri nei metadati, inclusi i commenti degli indici. Se i commenti degli indici contengono caratteri che non sono validi nel set di caratteri `utf8mb3`, l'aggiornamento non riesce.

Per risolvere questo problema, è necessario modificare i commenti degli indici per rimuovere o sostituire eventuali caratteri non BMP prima di tentare l'aggiornamento.

Output di esempio:

```
{
    "id": "auroraUpgradeCheckForInvalidUtf8mb3IndexComments",
    "title": "Check for invalid utf8mb3 index comments.",
    "status": "OK",
    "description": "Following table(s) has/have invalid utf8mb3 comments on the
 index.",
    "detectedProblems": [
        {
            "level": "Error",
            "dbObject": "precheck.utf8mb3_tab_index_comment",
            "description": "Table precheck.utf8mb3_tab_index_comment has invalid
 utf8mb3 comments in it's index. This is due to https://bugs.mysql.com/bug.php?
id=110177. To fix the inconsistency, we recommend you to modify comment fields to
 not use non-BMP characters and try the upgrade again."
        }
    ]
},
```

Il controllo preliminare segnala che la tabella `utf8mb3_tab_index_comment` contiene caratteri `utf8mb3` non validi in uno o più commenti della colonna, in particolare a causa della presenza di caratteri non BMP.

Per risolvere questo problema, esamina innanzitutto la struttura della tabella per identificare l'indice con commenti problematici.

```
MySQL> SHOW CREATE TABLE precheck.utf8mb3_tab_index_comment\G
*************************** 1. row ***************************
    Table: utf8mb3_tab_index_comment
Create Table: CREATE TABLE `utf8mb3_tab_index_comment` (
`id` int(11) DEFAULT NULL,
`name` varchar(100) DEFAULT NULL,
KEY `idx_name` (`name`) COMMENT 'Name index #'
) ENGINE=InnoDB DEFAULT CHARSET=utf8
1 row in set (0.01 sec)
```

Una volta identificato l'indice che contiene commenti che utilizzano caratteri non supportati, eliminalo e crealo di nuovo.

> **Note**
>
> L'eliminazione e la nuova creazione di un indice della tabella possono causare tempi di inattività. Si consiglia di pianificare e programmare l'operazione durante la manutenzione.

```
MySQL> ALTER TABLE precheck.utf8mb3_tab_index_comment DROP INDEX idx_name;
MySQL> ALTER TABLE precheck.utf8mb3_tab_index_comment ADD INDEX idx_name(name);
```

L'esempio seguente mostra un altro modo per creare nuovamente l'indice.

```
MySQL> ALTER TABLE utf8mb3_tab_index_comment DROP INDEX idx_name, ADD INDEX idx_name
 (name) COMMENT 'Updated comment without non-BMP characters';
```

Dopo aver rimosso o aggiornato tutti i commenti degli indici non supportati, il controllo preliminare viene superato e l'aggiornamento può procedere.

```
{
"id": "auroraUpgradeCheckForInvalidUtf8mb3IndexComments",
"title": "Check for invalid utf8mb3 index comments.",
"status": "OK",
"detectedProblems": []
},
```

auroraUpgradeCheckForInvalidUtf8mb3TableComments

Livello di controllo preliminare: errore

Verifica la presenza di caratteri utf8mb3 non validi nella definizione della tabella

Questo controllo preliminare identifica le tabelle che contengono commenti con codifica dei caratteri non validi `utf8mb3`. In MySQL 8.0, viene applicata una convalida più rigorosa alla codifica dei caratteri nei metadati, inclusi i commenti delle tabelle. Se i commenti delle tabelle contengono caratteri che non sono validi nel set di caratteri `utf8mb3`, l'aggiornamento non riesce.

Per risolvere questo problema, è necessario modificare i commenti delle tabelle per rimuovere o sostituire eventuali caratteri non BMP prima di tentare l'aggiornamento.

Output di esempio:

```
{
    "id": "auroraUpgradeCheckForInvalidUtf8mb3TableComments",
    "title": "Check for invalid utf8mb3 table comments.",
    "status": "OK",
    "description": "Following table(s) has/have invalid utf8mb3 comments.",
    "detectedProblems": [
        {
            "level": "Error",
            "dbObject": "precheck.utf8mb3_table_with_comment",
            "description": "Table precheck.utf8mb3_table_with_comment has invalid
 utf8mb3 comments. This is due to https://bugs.mysql.com/bug.php?id=110177. To fix
 the inconsistency, we recommend you to modify comment fields to not use non-BMP
 characters and try the upgrade again."
        }

    ]
},
```

Il controllo preliminare segnala i commenti `utf8mb3` non validi definiti per le tabelle `utf8mb3_table_with_comment` nel database di test.

Per risolvere questo problema, identifica la tabella che contiene i caratteri non supportati e aggiorna i commenti. Innanzitutto, esamina la struttura della tabella vista e identifica i commenti.

```
MySQL> SHOW CREATE TABLE precheck.utf8mb3_table_with_comment\G
*************************** 1. row ***************************
    Table: utf8mb3_table_with_comment
Create Table: CREATE TABLE `utf8mb3_table_with_comment` (
`id` int(11) NOT NULL,
`name` varchar(100) DEFAULT NULL,
PRIMARY KEY (`id`)
) ENGINE=InnoDB DEFAULT CHARSET=latin1 COMMENT='This table comment contains flag ##'
1 row in set (0.00 sec)
```

Dopo aver identificato i commenti delle tabelle che contengono caratteri non supportati, aggiorna i commenti con l'istruzione `ALTER TABLE`.

```
MySQL> ALTER TABLE precheck.utf8mb3_table_with_comment COMMENT='Updated comment
 without non-BMP characters';
```

In alternativa, è possibile rimuovere il commento.

```
MySQL> ALTER TABLE precheck.utf8mb3_table_with_comment COMMENT='';
```

Dopo aver rimosso tutti i caratteri non supportati da tutti i commenti delle tabelle, il controllo preliminare ha esito positivo.

```
{
"id": "auroraUpgradeCheckForInvalidUtf8mb3TableComments",
"title": "Check for invalid utf8mb3 table comments.",
"status": "OK",
"detectedProblems": []
},
```

auroraUpgradeCheckForMasterUser

Livello di controllo preliminare: errore

Verifica se esiste un utente master RDS

MySQL 8 ha aggiunto un nuovo modello di privilegi con supporto per il ruolo e i privilegi dinamici per rendere la gestione dei privilegi più semplice e dettagliata. Come parte di questa modifica, Aurora MySQL ha introdotto il nuovo `rds_superuser_role`, che viene automaticamente concesso all'utente master del database al momento dell'aggiornamento da Aurora MySQL versione 2 alla versione 3.

Per ulteriori informazioni sui ruoli e i privilegi assegnati all'utente master in Aurora MySQL, consulta Privilegi dell'account utente master. Per ulteriori informazioni sul modello di privilegi basato su ruolo in Aurora MySQL versione 3, consulta Privilegio basato sui ruoli.

Questo controllo preliminare verifica che l'utente master esista nel database. Se l'utente master non esiste, il controllo preliminare avrà esito negativo. Per consentire il proseguimento dell'aggiornamento, crea nuovamente l'utente master reimpostando la password dell'utente principale o creando manualmente l'utente. Prova quindi a eseguire nuovamente l'aggiornamento. Per ulteriori informazioni sulla reimpostazione della password dell'utente master, consulta Modifica della password per l'utente master del database.

Output di esempio:

```
{
  "id": "auroraUpgradeCheckForMasterUser",
  "title": "Check if master user exists",
  "status": "OK",
```

```
  "description": "Throws error if master user has been dropped!",
  "documentationLink": "https://docs.aws.amazon.com/AmazonRDS/latest/
AuroraUserGuide/UsingWithRDS.MasterAccounts.html",
  "detectedProblems": [
      {
        "level": "Error",
        "dbObject": "all",
        "description": "Your Master User on host '%' has been dropped. To proceed
 with the upgrade, recreate the master user `reinvent` on default host '%'"
      }
  ]
}
```

Dopo aver reimpostato la password dell'utente master, il controllo preliminare verrà superato e potrai tentare di eseguire nuovamente l'aggiornamento.

L'esempio seguente utilizza l'interfaccia AWS CLI per reimpostare la password. Le modifiche alla password vengono applicate immediatamente.

```
aws rds modify-db-cluster \
    --db-cluster-identifier my-db-cluster \
    --master-user-password my-new-password
```

Quindi il controllo preliminare ha esito positivo.

```
{
  "id": "auroraUpgradeCheckForMasterUser",
  "title": "Check if master user exists",
  "status": "OK",
  "detectedProblems": []
}
```

auroraUpgradeCheckForPrefixIndexOnGeometryColumns

Livello di controllo preliminare: errore

Verifica la presenza di colonne di geometria negli indici dei prefissi

A partire da MySQL 8.0.12, non è più possibile creare un indice con prefisso in una colonna utilizzando il tipo di dati GEOMETRY. Per ulteriori informazioni, consulta WL#11808.

Se esistono indici di questo tipo, l'aggiornamento non riesce. Per risolvere il problema, elimina gli indici `GEOMETRY` con prefisso nelle tabelle menzionate nell'errore del controllo preliminare.

Output di esempio:

```
{
  "id": "auroraUpgradeCheckForPrefixIndexOnGeometryColumns",
  "title": "Check for geometry columns on prefix indexs",
  "status": "OK",
  "description": "Consider dropping the prefix Indexes of geometry columns and
 restart the upgrade.",
  "detectedProblems": [
      {
        "level": "Error",
        "dbObject": "test.geom_index_prefix",
        "description": "Table `test`.`geom_index_prefix` has an index `LatLon` on
 geometry column/s. Mysql 8.0 does not support this type of index on a geometry
 column https://dev.mysql.com/worklog/task/?id=11808. To upgrade to MySQL 8.0, Run
 'DROP INDEX `LatLon` ON `test`.`geom_index_prefix`;"
      }
  ]
}
```

Il controllo preliminare segnala un errore perché la tabella `test.geom_index_prefix` contiene un indice con un prefisso in una colonna `GEOMETRY`.

Dopo aver eliminato questo indice, il controllo preliminare ha esito positivo.

```
{
  "id": "auroraUpgradeCheckForPrefixIndexOnGeometryColumns",
  "title": "Check for geometry columns on prefix indexs",
  "status": "OK",
  "detectedProblems": []
}
```

auroraUpgradeCheckForSpecialCharactersInProcedures

Livello di controllo preliminare: errore

Verifica l’eventuale presenza di incoerenze relative ai caratteri speciali nelle stored procedure

Prima di MySQL 8.0, i nomi dei database, i nomi delle tabelle e altri oggetti corrispondevano ai file nella directory dei dati, ossia a metadati basati su file. Nell’ambito dell’aggiornamento a MySQL 8.0, viene eseguita la migrazione di tutti gli oggetti del database nelle nuove tabelle

interne del dizionario dei dati, che sono archiviate nello schema `mysql` per supportare l'Atomic Data Dictionary appena implementato. Nell'ambito della migrazione delle stored procedure, la definizione e il corpo di ciascuna procedura vengono convalidati man mano che vengono acquisiti nel nuovo dizionario di dati.

Prima di MySQL 8, in alcuni casi era possibile creare routine memorizzate o inserire direttamente nella tabella `mysql.proc` procedure che contenevano caratteri speciali. Ad esempio, era possibile creare una stored procedure contenente un commento con il carattere spazio non divisibile non conforme `\xa0`. Se viene rilevata una di queste procedure, l'aggiornamento non riesce.

Questo controllo preliminare verifica che i corpi e le definizioni delle stored procedure non contengano tali caratteri. Per consentire il proseguimento dell'aggiornamento, crea nuovamente queste stored procedure senza caratteri nascosti o speciali.

Output di esempio:

```
{
  "id": "auroraUpgradeCheckForSpecialCharactersInProcedures",
  "title": "Check for inconsistency related to special characters in stored
 procedures.",
  "status": "OK",
  "description": "Following procedure(s) has special characters inconsistency.",
  "detectedProblems": [
      {
        "level": "Error",
        "dbObject": "information_schema.routines",
        "description": "Data Dictionary Metadata is inconsistent for the procedure
 `get_version_proc` in the database `test` due to usage of special characters in
 procedure body. To avoid that, drop and recreate the procedure without any special
 characters before proceeding with the Upgrade."
      }
  ]
}
```

Il controllo preliminare segnala che il cluster di database contiene una procedura denominata `get_version_proc` nel database `test` che contiene caratteri speciali nel corpo della procedura.

Dopo aver eliminato e creato nuovamente la stored procedure, il controllo preliminare ha esito positivo e consente di procedere con l'aggiornamento.

```
{
  "id": "auroraUpgradeCheckForSpecialCharactersInProcedures",
  "title": "Check for inconsistency related to special characters in stored
 procedures.",
  "status": "OK",
  "detectedProblems": []
},
```

auroraUpgradeCheckForSysSchemaObjectTypeMismatch

Livello di controllo preliminare: errore

Verifica l'eventuale mancata corrispondenza del tipo di oggetto per lo schema **`sys`**

Lo schema sys è un set di oggetti e viste in un database MySQL che aiuta gli utenti a eseguire la risoluzione dei problemi, l'ottimizzazione e il monitoraggio per le relative istanze database. Quando si esegue un aggiornamento di versione principale da Aurora MySQL versione 2 alla versione 3, le viste dello schema `sys` vengono create nuovamente e aggiornate alle nuove definizioni di Aurora MySQL versione 3.

Nell'ambito dell'aggiornamento, se alcuni oggetti dello schema `sys` vengono definiti utilizzando i motori di archiviazione (`sys_config/BASE TABLE` in INFORMATION_SCHEMA.TABLES) anziché le viste, l'aggiornamento non riesce. Tali tabelle si trovano nella tabella `information_schema.tables`. Questo non è un comportamento previsto, ma in alcuni casi può verificarsi a causa di un errore dell'utente.

Questo controllo preliminare convalida tutti gli oggetti dello schema `sys` per garantire che utilizzino le definizioni di tabella corrette e che le viste non vengano erroneamente definite come tabelle InnoDB o MyISAM. Per risolvere il problema, correggi manualmente gli oggetti restituiti rinominandoli o eliminandoli. Prova quindi a eseguire nuovamente l'aggiornamento.

Output di esempio:

```
{
  "id": "auroraUpgradeCheckForSysSchemaObjectTypeMismatch",
  "title": "Check object type mismatch for sys schema.",
  "status": "OK",
  "description": "Database contains objects with type mismatch for sys schema.",
  "detectedProblems": [
      {
        "level": "Error",
```

```
        "dbObject": "sys.waits_global_by_latency",
        "description": "Your object sys.waits_global_by_latency has a type mismatch.
 To fix the inconsistency we recommend to rename or remove the object before
 upgrading (use RENAME TABLE command). "
      }
  ]
}
```

Il controllo preliminare segnala che la vista sys.waits_global_by_latency nello schema `sys` presenta una mancata corrispondenza del tipo che impedisce il proseguimento dell'aggiornamento.

Dopo aver effettuato l'accesso all'istanza database, puoi vedere che questo oggetto è definito come una tabella InnoDB, mentre invece dovrebbe essere una vista.

```
mysql> show create table sys.waits_global_by_latency\G
*************************** 1. row ***************************
       Table: waits_global_by_latency
Create Table: CREATE TABLE `waits_global_by_latency` (
  `events` varchar(128) DEFAULT NULL,
  `total` bigint(20) unsigned DEFAULT NULL,
  `total_latency` text,
  `avg_latency` text,
  `max_latency` text
) ENGINE=InnoDB DEFAULT CHARSET=utf8
1 row in set (0.00 sec)
```

Per risolvere questo problema, è possibile eliminare e creare nuovamente la vista con la definizione corretta o rinominarla. Durante il processo di aggiornamento, la vista viene creata automaticamente con la definizione di tabella corretta.

```
mysql> RENAME TABLE sys.waits_global_by_latency to sys.waits_global_by_latency_old;
Query OK, 0 rows affected (0.01 sec)
```

A questo punto, il controllo preliminare viene superato.

```
{
  "id": "auroraUpgradeCheckForSysSchemaObjectTypeMismatch",
  "title": "Check object type mismatch for sys schema.",
  "status": "OK",
  "detectedProblems": []
```

```
}
```

auroraUpgradeCheckForViewColumnNameLength

Livello di controllo preliminare: errore

Controlla il limite massimo per il nome della colonna nella vista

La lunghezza massima consentita per un nome di colonna in MySQL è di 64 caratteri. Prima di MySQL 8.0, in alcuni casi era possibile creare una vista con un nome di colonna più lungo di 64 caratteri. Se nell'istanza database sono presenti viste di questo tipo, viene restituito un errore di controllo preliminare e l'aggiornamento non riesce. Per consentire il proseguimento dell'aggiornamento, è necessario creare nuovamente le viste in questione, assicurandosi che la lunghezza delle colonne sia inferiore a 64 caratteri. Prova quindi a eseguire nuovamente l'aggiornamento.

Output di esempio:

```
{
  "id": "auroraUpgradeCheckForViewColumnNameLength",
  "title": "Check for upperbound limit related to column name in view.",
  "status": "OK",
  "description": "Following view(s) has column(s) with length greater than 64.",
  "detectedProblems": [
      {
        "level": "Error",
        "dbObject":
 "test.colname_view_test.col2_pad_pad_pad_pad_pad_pad_pad_pad_pad_pad_pad_pad_pad_pad_pad_pad_pa
        "description": "View `test`.`colname_view_test`has column
 `col2_pad_pad_pad_pad_pad_pad_pad_pad_pad_pad_pad_pad_pad_pad_pad_pad` with
 invalid column name length. To avoid Upgrade errors, view should be altered by
 renaming the column name so that its length is not 0 and doesn't exceed 64."
      }
  ]
}
```

Il controllo preliminare segnala che la vista `test.colname_view_test` contiene una colonna `col2_pad_pad_pad_pad_pad_pad_pad_pad_pad_pad_pad_pad_pad_pad_pad_pad` che supera la lunghezza massima consentita di 64 caratteri.

Osservando la definizione della vista, possiamo individuare la colonna che causa l'errore.

```
mysql> desc `test`.`colname_view_test`;
+-------------------------------------------------------------------------+-------------
+------+-----+---------+-------+
| Field                                                                   | Type        |
 Null | Key | Default | Extra |
+-------------------------------------------------------------------------+-------------
+------+-----+---------+-------+
| col1                                                                    | varchar(20) |
 YES  |     | NULL    |       |
| col2_pad_pad_pad_pad_pad_pad_pad_pad_pad_pad_pad_pad_pad_pad_pad_pad_pad | int(11)     |
 YES  |     | NULL    |       |
+-------------------------------------------------------------------------+-------------
+------+-----+---------+-------+
2 rows in set (0.00 sec)
```

Per consentire il proseguimento dell'aggiornamento, crea nuovamente la vista, assicurandoti che la lunghezza delle colonne non superi i 64 caratteri.

```
mysql> drop view `test`.`colname_view_test`;
Query OK, 0 rows affected (0.01 sec)

mysql> create view `test`.`colname_view_test`(col1, col2_nopad) as select inf,
 fodcol from test;
Query OK, 0 rows affected (0.01 sec)

mysql> desc `test`.`colname_view_test`;
+------------+-------------+------+-----+---------+-------+
| Field      | Type        | Null | Key | Default | Extra |
+------------+-------------+------+-----+---------+-------+
| col1       | varchar(20) | YES  |     | NULL    |       |
| col2_nopad | int(11)     | YES  |     | NULL    |       |
+------------+-------------+------+-----+---------+-------+
2 rows in set (0.00 sec)
```

A questo punto, il controllo preliminare viene superato.

```
{
  "id": "auroraUpgradeCheckForViewColumnNameLength",
  "title": "Check for upperbound limit related to column name in view.",
  "status": "OK",
  "detectedProblems": []
}
```

auroraUpgradeCheckIndexLengthLimitOnTinyColumns

Livello di controllo preliminare: errore

Verifica la presenza di tabelle con indici definiti con una lunghezza del prefisso superiore a 255 byte in colonne di dimensioni piccolissime

Quando si crea un indice su una colonna utilizzando un tipo di dati binario in MySQL, è necessario aggiungere una lunghezza del prefisso nella definizione dell'indice. Prima di MySQL 8.0, in alcuni casi era possibile specificare una lunghezza del prefisso superiore alle dimensioni massime consentite per tali tipi di dati. Un esempio è rappresentato dalle colonne `TINYTEXT` e `TINYBLOB`, in cui le dimensioni massime consentite dei dati sono di 255 byte, ma sono consentiti prefissi di indice di dimensioni superiori. Per ulteriori informazioni, consulta InnoDB limits nella documentazione di MySQL.

Se questo controllo preliminare ha esito negativo, elimina l'indice che causa l'errore o riduci la lunghezza del prefisso delle colonne `TINYTEXT` e `TINYBLOB` dell'indice a meno di 255 byte. Prova quindi a eseguire nuovamente l'aggiornamento.

Output di esempio:

```
{
  "id": "auroraUpgradeCheckIndexLengthLimitOnTinyColumns",
  "title": "Check for the tables with indexes defined with prefix length greater
 than 255 bytes on tiny columns",
  "status": "OK",
  "description": "Prefix length of the indexes defined on tiny columns cannot
 exceed 255 bytes. With utf8mb4 char set, this limits the prefix length supported
 upto 63 characters only. A larger prefix length was allowed in MySQL5.7 using
 innodb_large_prefix parameter. This parameter is deprecated in MySQL 8.0.",
  "documentationLink": "https://dev.mysql.com/doc/refman/8.0/en/innodb-limits.html,
 https://dev.mysql.com/doc/refman/8.0/en/storage-requirements.html",
  "detectedProblems": [
      {
        "level": "Error",
        "dbObject": "test.tintxt_prefixed_index.col1",
        "description": "Index 'PRIMARY' on tinytext/tinyblob column `col1` of table
 `test.tintxt_prefixed_index` is defined with prefix length exceeding 255 bytes.
 Reduce the prefix length to <=255 bytes depending on character set used. For
 utf8mb4, it should be <=63."
      }
  ]
```

```
}
```

Il controllo preliminare segnala un errore per la tabella `test.tintxt_prefixed_index`, poiché ha un indice PRIMARY con un prefisso di dimensioni superiori a 255 byte in una colonna TINYTEXT o TINYBLOB.

Osservando la definizione di questa tabella, possiamo vedere che la chiave primaria ha un prefisso 65 sulla colonna TINYTEXT `col1`. Poiché la tabella è definita utilizzando il set di caratteri `utf8mb4`, che memorizza 4 byte per carattere, il prefisso supera il limite di 255 byte.

```
mysql> show create table `test`.`tintxt_prefixed_index`\G
*************************** 1. row ***************************
       Table: tintxt_prefixed_index
Create Table: CREATE TABLE `tintxt_prefixed_index` (
  `col1` tinytext NOT NULL,
  `col2` tinytext,
  `col_id` tinytext,
  PRIMARY KEY (`col1`(65))
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 ROW_FORMAT=DYNAMIC
1 row in set (0.00 sec)
```

Modificando il prefisso dell'indice su 63 durante l'utilizzo del set di caratteri `utf8mb4`, l'aggiornamento potrà proseguire.

```
mysql> alter table `test`.`tintxt_prefixed_index` drop PRIMARY KEY, ADD  PRIMARY KEY
 (`col1`(63));
Query OK, 0 rows affected (0.04 sec)
Records: 0  Duplicates: 0  Warnings: 0
```

A questo punto, il controllo preliminare viene superato.

```
{
  "id": "auroraUpgradeCheckIndexLengthLimitOnTinyColumns",
  "title": "Check for the tables with indexes defined with prefix length greater
 than 255 bytes on tiny columns",
  "status": "OK",
  "detectedProblems": []
}
```

auroraUpgradeCheckMissingInnodbMetadataForMysqlHostTable

Livello di controllo preliminare: errore

Verifica eventuali incoerenze nei metadati InnoDB mancanti per la tabella **`mysql.host`**

Si tratta di un controllo preliminare solo interno eseguito dal servizio RDS. Eventuali errori vengono gestiti automaticamente durante l'aggiornamento e possono essere ignorati senza problemi.

Se riscontri errori con questo controllo preliminare, apri un caso con il Supporto AWS per richiedere che l'incoerenza dei metadati venga risolta. In alternativa, puoi tentare nuovamente l'aggiornamento eseguendo un dump logico, quindi effettuando il ripristino su un nuovo cluster di database Aurora MySQL versione 3.

Avvisi

In caso di esito negativo, i seguenti controlli preliminari generano avvisi ma l'aggiornamento può comunque essere eseguito.

Argomenti

- Controlli preliminari MySQL che restituiscono avvisi
- Controlli preliminari Aurora MySQL che restituiscono avvisi

Controlli preliminari MySQL che restituiscono avvisi

I seguenti controlli preliminari sono forniti da Community MySQL:

- defaultAuthenticationPlugin
- maxdbFlagCheck
- mysqlDollarSignNameCheck
- reservedKeywordsCheck
- utf8mb3Check
- zeroDatesCheck

defaultAuthenticationPlugin

Livello di controllo preliminare: avviso

Nuove considerazioni sul plugin di autenticazione predefinito

In MySQL 8.0, è stato introdotto il plugin di autenticazione `caching_sha2_password`, che offre una crittografia delle password più sicura e prestazioni migliori rispetto al plugin `mysql_native_password` obsoleto. Per Aurora MySQL versione 3, il plugin di autenticazione predefinito utilizzato per gli utenti del database è il plugin `mysql_native_password`.

Questo controllo preliminare avverte che tale plugin verrà rimosso e l'impostazione predefinita verrà modificata in una futura versione principale. Prendi in considerazione la possibilità di valutare la compatibilità dei client e degli utenti delle applicazioni prima di questa modifica.

Per ulteriori informazioni, consulta caching_sha2_password compatibility issues and solutions nella documentazione di MySQL.

Output di esempio:


```
{
  "id": "defaultAuthenticationPlugin",
  "title": "New default authentication plugin considerations",
  "description": "Warning: The new default authentication plugin
 'caching_sha2_password' offers more secure password hashing than previously
 used 'mysql_native_password' (and consequent improved client connection
 authentication). However, it also has compatibility implications that
 may affect existing MySQL installations. If your MySQL installation
 must serve pre-8.0 clients and you encounter compatibility issues after
 upgrading, the simplest way to address those issues is to reconfigure
 the server to revert to the previous default authentication plugin
 (mysql_native_password). For example, use these lines in the server option file:
\n\n[mysqld]\ndefault_authentication_plugin=mysql_native_password\n\nHowever, the
 setting should be viewed as temporary, not as a long term or permanent solution,
 because it causes new accounts created with the setting in effect to forego the
 improved authentication security.\nIf you are using replication please take time to
 understand how the authentication plugin changes may impact you.",
  "documentationLink": "https://dev.mysql.com/doc/refman/8.0/en/upgrading-from-
previous-series.html#upgrade-caching-sha2-password-compatibility-issues\nhttps://
dev.mysql.com/doc/refman/8.0/en/upgrading-from-previous-series.html#upgrade-caching-
sha2-password-replication"
},
```


maxdbFlagCheck

Livello di controllo preliminare: avviso

Utilizzo di un flag **`sql_mode MAXDB`** obsoleto

In MySQL 8.0, sono state rimosse alcune opzioni di variabili di sistema sql_mode obsolete, una delle quali era MAXDB. Questo controllo preliminare esamina tutte le sessioni attualmente connesse, insieme alle routine e ai trigger, per garantire che sql_mode non sia impostato su una combinazione contenente MAXDB in alcun caso.

Output di esempio:

```
{
  "id": "maxdbFlagCheck",
  "title": "Usage of obsolete MAXDB sql_mode flag",
  "status": "OK",
  "description": "Warning: The following DB objects have the obsolete MAXDB
 option persisted for sql_mode, which will be cleared during the upgrade. It can
 potentially change the datatype DATETIME into TIMESTAMP if it is used inside
 object's definition, and this in turn can change the behavior in case of dates
 earlier than 1970 or later than 2037. If this is a concern, please redefine these
 objects so that they do not rely on the MAXDB flag before running the upgrade.",
  "documentationLink": "https://dev.mysql.com/doc/refman/8.0/en/mysql-
nutshell.html#mysql-nutshell-removals",
  "detectedProblems": [
      {
        "level": "Warning",
        "dbObject": "test.maxdb_stored_routine",
        "description": "PROCEDURE uses obsolete MAXDB sql_mode",
        "dbObjectType": "Routine"
      }
  ]
}
```

Il controllo preliminare segnala che la routine test.maxdb_stored_routine contiene un'opzione sql_mode non supportata.

Dopo aver effettuato l'accesso al database, è possibile visualizzare la definizione di routine in cui sql_mode contiene MAXDB.

```
 > SHOW CREATE PROCEDURE test.maxdb_stored_routine\G

*************************** 1. row ***************************
           Procedure: maxdb_stored_routine
            sql_mode:
 PIPES_AS_CONCAT,ANSI_QUOTES,IGNORE_SPACE,MAXDB,NO_KEY_OPTIONS,NO_TABLE_OPTIONS,NO_FIELD_OPT
```

```
    Create Procedure: CREATE DEFINER="msandbox"@"localhost" PROCEDURE
 "maxdb_stored_routine"()
BEGIN
    SELECT * FROM test;
END
character_set_client: utf8
collation_connection: utf8_general_ci
  Database Collation: latin1_swedish_ci
1 row in set (0.00 sec)
```

Per risolvere il problema, crea nuovamente la procedura dopo aver impostato l'opzione `sql_mode` corretta sul client.

> **Note**
>
> Secondo la documentazione di MySQL, MySQL memorizza l'impostazione `sql_mode` attiva quando una routine viene creata o modificata. Esegue sempre la routine con questa impostazione, indipendentemente dall'impostazione `sql_mode` al momento dell'avvio della routine.
> Prima di modificare `sql_mode`, consulta Server SQL modes nella documentazione di MySQL. Valuta attentamente qualsiasi potenziale impatto di questa operazione sulla tua applicazione.

Crea nuovamente la procedura senza l'opzione `sql_mode` non supportata.

```
mysql > set sql_mode='PIPES_AS_CONCAT,ANSI_QUOTES,IGNORE_SPACE';
Query OK, 0 rows affected, 1 warning (0.00 sec)

mysql > DROP PROCEDURE test.maxdb_stored_routine\G
Query OK, 0 rows affected (0.00 sec)

mysql >
mysql > DELIMITER $$
mysql >
mysql > CREATE PROCEDURE test.maxdb_stored_routine()
    -> SQL SECURITY DEFINER
    -> BEGIN
    ->     SELECT * FROM test;
    -> END$$
Query OK, 0 rows affected (0.00 sec)
```

```
mysql >
mysql > DELIMITER ;
mysql > show create procedure test.maxdb_stored_routine\G
*************************** 1. row ***************************
           Procedure: maxdb_stored_routine
            sql_mode: PIPES_AS_CONCAT,ANSI_QUOTES,IGNORE_SPACE
    Create Procedure: CREATE DEFINER="msandbox"@"localhost" PROCEDURE
 "maxdb_stored_routine"()
BEGIN
    SELECT * FROM test;
END
character_set_client: utf8
collation_connection: utf8_general_ci
  Database Collation: latin1_swedish_ci
1 row in set (0.00 sec)
```

Il controllo preliminare ha esito positivo.

```
{
  "id": "maxdbFlagCheck",
  "title": "Usage of obsolete MAXDB sql_mode flag",
  "status": "OK",
  "detectedProblems": []
}
```

mysqlDollarSignNameCheck

Livello di controllo preliminare: avviso

Verifica l'utilizzo obsoleto dei simboli del dollaro singoli nei nomi degli oggetti

A partire da MySQL 8.0.32, l'utilizzo del simbolo del dollaro ($) come primo carattere di un identificatore senza virgolette è obsoleto. Se sono presenti schemi, tabelle, viste, colonne o routine contenenti un $ come primo carattere, il controllo preliminare restituisce un avviso. Sebbene questo avviso non impedisca il proseguimento dell'aggiornamento, è consigliabile agire subito per risolvere il problema. A partire da MySQL 8.4 qualsiasi identificatore di questo tipo restituisce un errore di sintassi anziché un avviso.

Output di esempio:

```
{
```

```
  "id": "mysqlDollarSignNameCheck",
  "title": "Check for deprecated usage of single dollar signs in object names",
  "status": "OK",
  "description": "Warning: The following objects have names with deprecated usage
 of dollar sign ($) at the begining of the identifier. To correct this warning,
 ensure, that names starting with dollar sign, also end with it, similary to quotes
 ($example$). ",
  "detectedProblems": [
      {
        "level": "Warning",
        "dbObject": "test.$deprecated_syntx",
        "description": " name starts with $ sign."
      }
  ]
},
```

Il controllo preliminare segnala un avviso perché la tabella `$deprecated_syntx` nello schema `test` contiene un $ come primo carattere.

reservedKeywordsCheck

Livello di controllo preliminare: avviso

Utilizzo di oggetti di database con nomi in conflitto con nuove parole chiave riservate

Questo controllo è simile a routineSyntaxCheck, in quanto verifica l'utilizzo di oggetti di database con nomi in conflitto con nuove parole chiave riservate. Sebbene i due controlli non impediscano gli aggiornamenti, è necessario valutare attentamente gli avvisi.

Output di esempio:

Utilizzando l'esempio precedente con la tabella denominata `except`, il controllo preliminare restituisce un avviso:

```
{
  "id": "reservedKeywordsCheck",
  "title": "Usage of db objects with names conflicting with new reserved keywords",
  "status": "OK",
  "description": "Warning: The following objects have names that conflict with
 new reserved keywords. Ensure queries sent by your applications use `quotes` when
 referring to them or they will result in errors.",
  "documentationLink": "https://dev.mysql.com/doc/refman/en/keywords.html",
  "detectedProblems": [
      {
```

```
        "level": "Warning",
        "dbObject": "test.except",
        "description": "Table name",
        "dbObjectType": "Table"
      }
  ]
}
```

Questo avviso segnala che potrebbero esserci alcune query dell'applicazione da esaminare. Se le query dell'applicazione non eseguono correttamente l'escape dei valori letterali delle stringhe, potrebbero verificarsi degli errori dopo l'aggiornamento a MySQL 8.0. Esamina le tue applicazioni per verificare, eseguendo test rispetto a un clone o uno snapshot del cluster di database Aurora MySQL in esecuzione sulla versione 3.

Esempio di una query dell'applicazione senza escape che avrà esito negativo dopo l'aggiornamento:

```
SELECT * FROM escape;
```

Esempio di una query dell'applicazione con escape corretto che avrà esito positivo dopo l'aggiornamento:

```
SELECT * FROM 'escape';
```

utf8mb3Check

Livello di controllo preliminare: avviso

Utilizzo del set di caratteri **`utf8mb3`**

In MySQL 8.0, il set di caratteri `utf8mb3` è obsoleto e verrà rimosso in una futura versione principale di MySQL. Questo controllo preliminare è implementato per generare un avviso se vengono rilevati oggetti di database che utilizzano tale set di caratteri. Anche se ciò non impedisce il proseguimento dell'aggiornamento, si consiglia vivamente di valutare la possibilità di eseguire la migrazione delle tabelle al set di caratteri `utf8mb4`, che è l'impostazione predefinita a partire da MySQL 8.0. Per ulteriori informazioni su utf8mb3 e utf8mb4, consulta Converting between 3-byte and 4-byte Unicode character sets nella documentazione di MySQL.

Output di esempio:

```
{
```

```
  "id": "utf8mb3",
  "title": "Usage of utf8mb3 charset",
  "status": "OK",
  "description": "Warning: The following objects use the deprecated utf8mb3
 character set. It is recommended to convert them to use utf8mb4 instead, for
 improved Unicode support. The utf8mb3 character is subject to removal in the
 future.",
  "documentationLink": "https://dev.mysql.com/doc/refman/8.0/en/charset-unicode-
utf8mb3.html",
  "detectedProblems": [
      {
        "level": "Warning",
        "dbObject": "test.t1.col1",
        "description": "column's default character set: utf8",
        "dbObjectType": "Column"
      },
      {
        "level": "Warning",
        "dbObject": "test.t1.col2",
        "description": "column's default character set: utf8",
        "dbObjectType": "Column"
      }
  ]
}
```


Per risolvere il problema, è necessario creare nuovamente gli oggetti e le tabelle a cui si fa riferimento. Per ulteriori informazioni e prerequisiti prima di procedere, consulta Converting between 3-byte and 4-byte Unicode character sets nella documentazione di MySQL.

zeroDatesCheck

Livello di controllo preliminare: avviso

Valori zero in date, datetime e timestamp

MySQL ora applica regole più rigide per quanto riguardo l'uso di valori zero nelle colonne di date, datetime e timestamp. Si consiglia di utilizzare le modalità `NO_ZERO_IN_DATE` e `NO_ZERO_DATE` SQL insieme alla modalità `strict`, poiché verranno unite alla modalità `strict` in una futura versione di MySQL.

Se, al momento dell'esecuzione del controllo preliminare, l'impostazione `sql_mode` per una qualsiasi delle connessioni al database non include queste modalità, viene generato un avviso nel controllo preliminare. Gli utenti potrebbero comunque essere in grado di inserire valori di

date, datetime e timestamp contenenti valori zero. Tuttavia, si consiglia vivamente di sostituire i valori zero con valori validi, poiché potrebbero avere un comportamento diverso in futuro e non funzionare correttamente. Poiché si tratta di un avviso, non impedisce l'esecuzione degli aggiornamenti, ma si consiglia di iniziare a pianificare le azioni necessarie.

Output di esempio:


```
{
  "id": "zeroDatesCheck",
  "title": "Zero Date, Datetime, and Timestamp values",
  "status": "OK",
  "description": "Warning: By default zero date/datetime/timestamp values are no
 longer allowed in MySQL, as of 5.7.8 NO_ZERO_IN_DATE and NO_ZERO_DATE are included
 in SQL_MODE by default. These modes should be used with strict mode as they will be
 merged with strict mode in a future release. If you do not include these modes in
 your SQL_MODE setting, you are able to insert date/datetime/timestamp values that
 contain zeros. It is strongly advised to replace zero values with valid ones, as
 they may not work correctly in the future.",
  "documentationLink": "https://lefred.be/content/mysql-8-0-and-wrong-dates/",
  "detectedProblems": [
      {
        "level": "Warning",
        "dbObject": "global.sql_mode",
        "description": "does not contain either NO_ZERO_DATE or NO_ZERO_IN_DATE
 which allows insertion of zero dates"
      },
      {
        "level": "Warning",
        "dbObject": "session.sql_mode",
        "description": " of 10 session(s) does not contain either NO_ZERO_DATE or
 NO_ZERO_IN_DATE which allows insertion of zero dates"
      }
  ]
}
```


Controlli preliminari Aurora MySQL che restituiscono avvisi

I seguenti controlli preliminari sono specifici di Aurora MySQL:

- auroraUpgradeCheckForRollbackSegmentHistoryLength
- auroraUpgradeCheckForUncommittedRowModifications

auroraUpgradeCheckForRollbackSegmentHistoryLength

Livello di controllo preliminare: avviso

Verifica se la lunghezza dell'elenco della cronologia dei segmenti di rollback per il cluster è elevata

Come indicato in auroraUpgradeCheckForIncompleteXATransactions, durante l'esecuzione del processo di aggiornamento a una versione principale, è essenziale che venga eseguito un arresto pulito dell'istanza database Aurora MySQL versione 2. Ciò garantisce che tutte le transazioni vengano eseguite o ripristinate e che InnoDB abbia eliminato tutti i record del log di undo.

Se l'elenco della cronologia dei segmenti di rollback (HLL) del cluster di database ha una lunghezza elevata, il cluster può prolungare il tempo impiegato da InnoDB per completare l'eliminazione dei record del log di undo, con conseguenti tempi di inattività prolungati durante il processo di aggiornamento alla versione principale. Se il controllo preliminare rileva che l'HLL del cluster di database è elevata, genera un avviso. Sebbene ciò non impedisca il proseguimento dell'aggiornamento, ti consigliamo di monitorare attentamente l'HLL del tuo cluster di database. Mantenendola bassa, è possibile ridurre i tempi di inattività necessari per un aggiornamento a una versione principale. Per ulteriori informazioni sul monitoraggio dell'HLL, consulta La lunghezza dell'elenco della cronologia di InnoDB è aumentata in modo significativo.

Output di esempio:

```
{
  "id": "auroraUpgradeCheckForRollbackSegmentHistoryLength",
  "title": "Checks if the rollback segment history length for the cluster is high",
  "status": "OK",
  "description": "Rollback Segment History length is greater than 1M. Upgrade may
 take longer time.",
  "detectedProblems": [
      {
        "level": "Warning",
        "dbObject": "information_schema.innodb_metrics",
        "description": "The InnoDB undo history list length('trx_rseg_history_len')
 is 82989114. Upgrade may take longer due to purging of undo information for old row
 versions."
      }
  ]
}
```

Il controllo preliminare restituisce un avviso perché ha rilevato che l'HLL di undo di InnoDB era elevata nel cluster di database (82989114). Sebbene l'aggiornamento proceda, a seconda della quantità di operazioni di undo da eliminare, i tempi di inattività necessari durante il processo di aggiornamento vengono prolungati.

Ti consigliamo di esaminare le transazioni aperte nel tuo cluster di database prima di eseguire l'aggiornamento nell'ambiente di produzione, per assicurarti che l'HLL rimanga di dimensioni gestibili.

auroraUpgradeCheckForUncommittedRowModifications

Livello di controllo preliminare: avviso

Verifica se ci sono numerose modifiche alle righe di cui non è stato eseguito il commit

Come indicato in auroraUpgradeCheckForIncompleteXATransactions, durante l'esecuzione del processo di aggiornamento a una versione principale, è essenziale che venga eseguito un arresto pulito dell'istanza database Aurora MySQL versione 2. Ciò garantisce che tutte le transazioni vengano eseguite o ripristinate e che InnoDB abbia eliminato tutti i record del log di undo.

Se il cluster di database ha transazioni che hanno modificato un elevato numero di righe, può prolungare il tempo impiegato da InnoDB per completare un rollback della transazione nell'ambito del processo di arresto pulito. Se il controllo preliminare rileva transazioni di lunga durata, con un elevato numero di righe modificate nel cluster di database, genera un avviso. Sebbene ciò non impedisca il proseguimento dell'aggiornamento, ti consigliamo di monitorare attentamente le dimensioni delle transazioni attive del tuo cluster di database. Mantenendola bassa, è possibile ridurre i tempi di inattività necessari per un aggiornamento a una versione principale.

Output di esempio:

```
{
  "id": "auroraUpgradeCheckForUncommittedRowModifications",
  "title": "Checks if there are many uncommitted modifications to rows",
  "status": "OK",
  "description": "Database contains uncommitted row changes greater than 10M.
 Upgrade may take longer time.",
  "detectedProblems": [
      {
        "level": "Warning",
        "dbObject": "information_schema.innodb_trx",
        "description": "The database contains 11000000 uncommitted row change(s) in
 1 transaction(s). Upgrade may take longer due to transaction rollback."
```

```
      }
  ]
},
```

Il controllo preliminare segnala che il cluster di database contiene una transazione con 11.000.000 di modifiche alle righe senza commit, di cui dovrà essere eseguito il rollback durante il processo di arresto pulito. L'aggiornamento proseguirà, ma per ridurre i tempi di inattività durante il processo di aggiornamento, si consiglia di monitorare ed esaminare la situazione prima di eseguire l'aggiornamento sui cluster di produzione.

Per visualizzare le transazioni attive nell'istanza database di scrittura, puoi utilizzare la tabella information_schema.innodb_trx. La seguente query dell'istanza database di scrittura mostra le transazioni correnti, il tempo di esecuzione, lo stato e le righe modificate per il cluster di database.

```
# Example of uncommitted transaction
mysql> SELECT trx_started,
       TIME_TO_SEC(TIMEDIFF(now(), trx_started)) AS seconds_trx_has_been_running,
       trx_mysql_thread_id AS show_processlist_connection_id,
       trx_id,
       trx_state,
       trx_rows_modified AS rows_modified
FROM information_schema.innodb_trx;
+---------------------+-----------------------------
+---------------------------------+----------+-----------+---------------+
| trx_started         | seconds_trx_has_been_running |
 show_processlist_connection_id | trx_id   | trx_state | rows_modified |
+---------------------+-----------------------------
+---------------------------------+----------+-----------+---------------+
| 2024-08-12 18:32:52 |                          1592 |
 20041 | 52866130 | RUNNING   |      11000000 |
+---------------------+-----------------------------
+---------------------------------+----------+-----------+---------------+
1 row in set (0.01 sec)

# Example of transaction rolling back
mysql> SELECT trx_started,
       TIME_TO_SEC(TIMEDIFF(now(), trx_started)) AS seconds_trx_has_been_running,
       trx_mysql_thread_id AS show_processlist_connection_id,
       trx_id,
       trx_state,
       trx_rows_modified AS rows_modified
FROM information_schema.innodb_trx;
```

```
+---------------------+------------------------------
+--------------------------------+----------+-------------+--------------+
| trx_started         | seconds_trx_has_been_running |
 show_processlist_connection_id | trx_id   | trx_state    | rows_modified |
+---------------------+------------------------------
+--------------------------------+----------+-------------+--------------+
| 2024-08-12 18:32:52 |                          1719 |
 20041 | 52866130 | ROLLING BACK |      10680479 |
+---------------------+------------------------------
+--------------------------------+----------+-------------+--------------+
1 row in set (0.01 sec)
```

Dopo il commit o il rollback della transazione, il controllo preliminare non restituisce più un avviso. Consulta la documentazione di MySQL e contatta il tuo team applicativo prima di eseguire il rollback di transazioni di grandi dimensioni, poiché il completamento del rollback può richiedere tempo, a seconda delle dimensioni delle transazioni.

```
{
  "id": "auroraUpgradeCheckForUncommittedRowModifications",
  "title": "Checks if there are many uncommitted modifications to rows",
  "status": "OK",
  "detectedProblems": []
},
```

Per ulteriori informazioni sull'ottimizzazione della gestione delle transazioni InnoDB e sul potenziale impatto dell'esecuzione e del rollback di transazioni di grandi dimensioni sulle istanze database MySQL, consulta Optimizing InnoDB transaction management nella documentazione di MySQL.

Note

In caso di esito negativo, i seguenti controlli preliminari generano notifiche ma l'aggiornamento può comunque procedere.

sqlModeFlagCheck

Livello di controllo preliminare: notifica

Utilizzo di flag **`sql_mode`** obsoleti

Oltre a `MAXDB`, altre opzioni `sql_mode` sono state rimosse: `DB2`, `MSSQL`, `MYSQL323`, `MYSQL40`, `ORACLE`, `POSTGRESQL`, `NO_FIELD_OPTIONS`, `NO_KEY_OPTIONS` e `NO_TABLE_OPTIONS`. A partire da MySQL 8.0, nessuno di questi valori può essere assegnato alla variabile di sistema `sql_mode`. Se questo controllo preliminare rileva sessioni aperte che utilizzano queste impostazioni `sql_mode`, assicurati che l'istanza database e i gruppi di parametri del cluster di database, nonché le applicazioni e le configurazioni client, siano aggiornati per disabilitarle. Per ulteriori informazioni, consulta la documentazione di MySQL.

Output di esempio:

```
{
  "id": "sqlModeFlagCheck",
  "title": "Usage of obsolete sql_mode flags",
  "status": "OK",
  "detectedProblems": []
}
```

Per risolvere questi errori di controllo preliminare, consulta maxdbFlagCheck.

Errori, avvisi o notifiche

Il seguente controllo preliminare può restituire un errore, un avviso o una notifica a seconda dell'output del controllo preliminare.

checkTableOutput

Livello di controllo preliminare: errore, avviso o notifica

Problemi segnalati dal comando **`check table x for upgrade`**

Prima di avviare l'aggiornamento ad Aurora MySQL versione 3, `check table for upgrade` viene eseguito su ogni tabella degli schemi utente del cluster di database. Questo controllo preliminare non equivale a checkTableMysqlSchema.

Il comando `check table for upgrade` esamina le tabelle per individuare eventuali problemi che potrebbero verificarsi durante l'aggiornamento a una versione più recente di MySQL. L'esecuzione di questo comando prima di tentare un aggiornamento può aiutare a identificare e risolvere eventuali incompatibilità in anticipo, rendendo più agevole l'effettivo processo di aggiornamento.

Questo comando esegue diversi controlli su ogni tabella, come i seguenti:

- Verifica della compatibilità della struttura della tabella e dei metadati con la versione MySQL di destinazione
- Verifica della presenza di eventuali funzionalità obsolete o rimosse utilizzate dalla tabella
- Garanzia che la tabella possa essere aggiornata correttamente senza perdita di dati

A differenza di altri controlli preliminari, questo può restituire un errore, un avviso o una notifica a seconda dell'output di `check table`. Se questo controllo preliminare restituisce delle tabelle, esaminale attentamente insieme al messaggio e al codice restituito prima di iniziare l'aggiornamento. Per ulteriori informazioni, consulta CHECK TABLE statement nella documentazione di MySQL.

Vengono riportati un esempio di errore e un esempio di avviso.

Esempio di errore:

```
{
  "id": "checkTableOutput",
  "title": "Issues reported by 'check table x for upgrade' command",
  "status": "OK",
  "detectedProblems": [
      {
        "level": "Error",
        "dbObject": "test.parent",
        "description": "Table 'test.parent' doesn't exist"
      }
  ]
},
```

Il controllo preliminare segnala un errore che indica che la tabella `test.parent` non esiste.

Il file `mysql-error.log` per l'istanza database di scrittura mostra che c'è un errore di chiave esterna.

```
2024-08-13T15:32:10.676893Z 62 [Warning] InnoDB: Load table `test`.`parent` failed,
 the table has missing foreign key indexes. Turn off 'foreign_key_checks' and try
 again.
2024-08-13T15:32:10.676905Z 62 [Warning] InnoDB: Cannot open table test/parent from
 the internal data dictionary of InnoDB though the .frm file for the table exists.
Please refer to http://dev.mysql.com/doc/refman/5.7/en/innodb-troubleshooting.html
 for how to resolve the issue.
```

Accedi all'istanza database di scrittura ed esegui `show engine innodb status\G` per ottenere maggiori informazioni sull'errore della chiave esterna.

```
mysql> show engine innodb status\G
*************************** 1. row ***************************
  Type: InnoDB
  Name:
Status:
=====================================
2024-08-13 15:33:33 0x14ef7b8a1700 INNODB MONITOR OUTPUT
=====================================
.
.
.
------------------------
LATEST FOREIGN KEY ERROR
------------------------
2024-08-13 15:32:10 0x14ef6dbbb700 Error in foreign key constraint of table test/
child:
there is no index in referenced table which would contain
the columns as the first columns, or the data types in the
referenced table do not match the ones in table. Constraint:
,
  CONSTRAINT `fk_pname` FOREIGN KEY (`p_name`) REFERENCES `parent` (`name`)
The index in the foreign key in table is p_name_idx
Please refer to http://dev.mysql.com/doc/refman/5.7/en/innodb-foreign-key-
constraints.html for correct foreign key definition.
.
.
```

Il messaggio `LATEST FOREIGN KEY ERROR` segnala che il vincolo di chiave esterna `fk_pname` nella tabella `test.child`, che fa riferimento alla tabella `test.parent`, presenta un indice mancante o una mancata corrispondenza del tipo di dati. La documentazione di MySQL sui vincoli di chiave esterna spiega che le colonne a cui si fa riferimento in una chiave esterna devono avere un indice associato e che le colonne principale/secondaria devono utilizzare lo stesso tipo di dati.

Per verificare se il problema è correlato a un indice mancante o a una mancata corrispondenza del tipo di dati, accedi al database e controlla le definizioni della tabella disabilitando temporaneamente la variabile di sessione foreign_key_checks. Una volta fatto ciò, possiamo vedere che il vincolo secondario in questione (`fk_pname`) utilizza `p_name varchar(20) CHARACTER SET latin1 DEFAULT NULL` per fare riferimento alla tabella principale `name`

`varchar(20) NOT NULL`. La tabella principale utilizza `DEFAULT CHARSET=utf8`, ma la colonna `p_name` della tabella secondaria utilizza `latin1`, quindi viene generato l'errore di mancata corrispondenza del tipo di dati.

```
mysql> show create table parent\G
ERROR 1146 (42S02): Table 'test.parent' doesn't exist

mysql> show create table child\G
*************************** 1. row ***************************
       Table: child
Create Table: CREATE TABLE `child` (
  `id` int(11) NOT NULL,
  `p_name` varchar(20) CHARACTER SET latin1 DEFAULT NULL,
  PRIMARY KEY (`id`),
  KEY `p_name_idx` (`p_name`),
  CONSTRAINT `fk_pname` FOREIGN KEY (`p_name`) REFERENCES `parent` (`name`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8
1 row in set (0.00 sec)

mysql> set foreign_key_checks=0;
Query OK, 0 rows affected (0.00 sec)

mysql> show create table parent\G
*************************** 1. row ***************************
       Table: parent
Create Table: CREATE TABLE `parent` (
  `name` varchar(20) NOT NULL,
  PRIMARY KEY (`name`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8
1 row in set (0.00 sec)

mysql> show create table child\G
*************************** 1. row ***************************
       Table: child
Create Table: CREATE TABLE `child` (
  `id` int(11) NOT NULL,
  `p_name` varchar(20) CHARACTER SET latin1 DEFAULT NULL,
  PRIMARY KEY (`id`),
  KEY `p_name_idx` (`p_name`),
  CONSTRAINT `fk_pname` FOREIGN KEY (`p_name`) REFERENCES `parent` (`name`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8
1 row in set (0.00 sec)
```

Per risolvere questo problema, possiamo modificare la tabella secondaria in modo che utilizzi lo stesso set di caratteri della tabella principale oppure modificare la tabella principale in modo che utilizzi lo stesso set di caratteri della tabella secondaria. In questo esempio, poiché la tabella secondaria utilizza esplicitamente `latin1` nella definizione della colonna `p_name`, viene eseguito `ALTER TABLE` per modificare il set di caratteri in `utf8`.

```
mysql> alter table child modify p_name varchar(20) character set utf8 DEFAULT NULL;
Query OK, 0 rows affected (0.06 sec)
Records: 0  Duplicates: 0  Warnings: 0

mysql> flush tables;
Query OK, 0 rows affected (0.01 sec)
```

Una volta fatto ciò, il controllo preliminare viene superato e l'aggiornamento può procedere.

Esempio di avviso:

```
{
  "id": "checkTableOutput",
  "title": "Issues reported by 'check table x for upgrade' command",
  "status": "OK",
  "detectedProblems": [
      {
        "level": "Warning",
        "dbObject": "test.orders",
        "description": "Trigger test.orders.delete_audit_trigg does not have CREATED
 attribute."
      }
  ]
}
```

Il controllo preliminare segnala un avviso per il trigger `delete_audit_trigg` sulla tabella `test.orders` perché non ha un attributo `CREATED`. Secondo quanto riportato in Checking version compatibility nella documentazione di MySQL, questo messaggio è informativo e viene stampato per i trigger creati prima di MySQL 5.7.2.

Poiché si tratta di un avviso, non impedisce il proseguimento dell'aggiornamento. Tuttavia, se desideri risolvere il problema, puoi creare nuovamente il trigger in questione; una volta fatto ciò, il controllo preliminare ha esito positivo senza avvisi.

```
{
```

```
  "id": "checkTableOutput",
  "title": "Issues reported by 'check table x for upgrade' command",
  "status": "OK",
  "detectedProblems": []
},
```

Come eseguire un aggiornamento in loco

Ti consigliamo di rivedere il materiale di background in Come funziona l'aggiornamento della versione principale di Aurora MySQL in loco.

Esegui eventuali operazioni di pianificazione e test pre-aggiornamento, come descritto in Pianificazione di un aggiornamento della versione principale per un cluster Aurora MySQL.

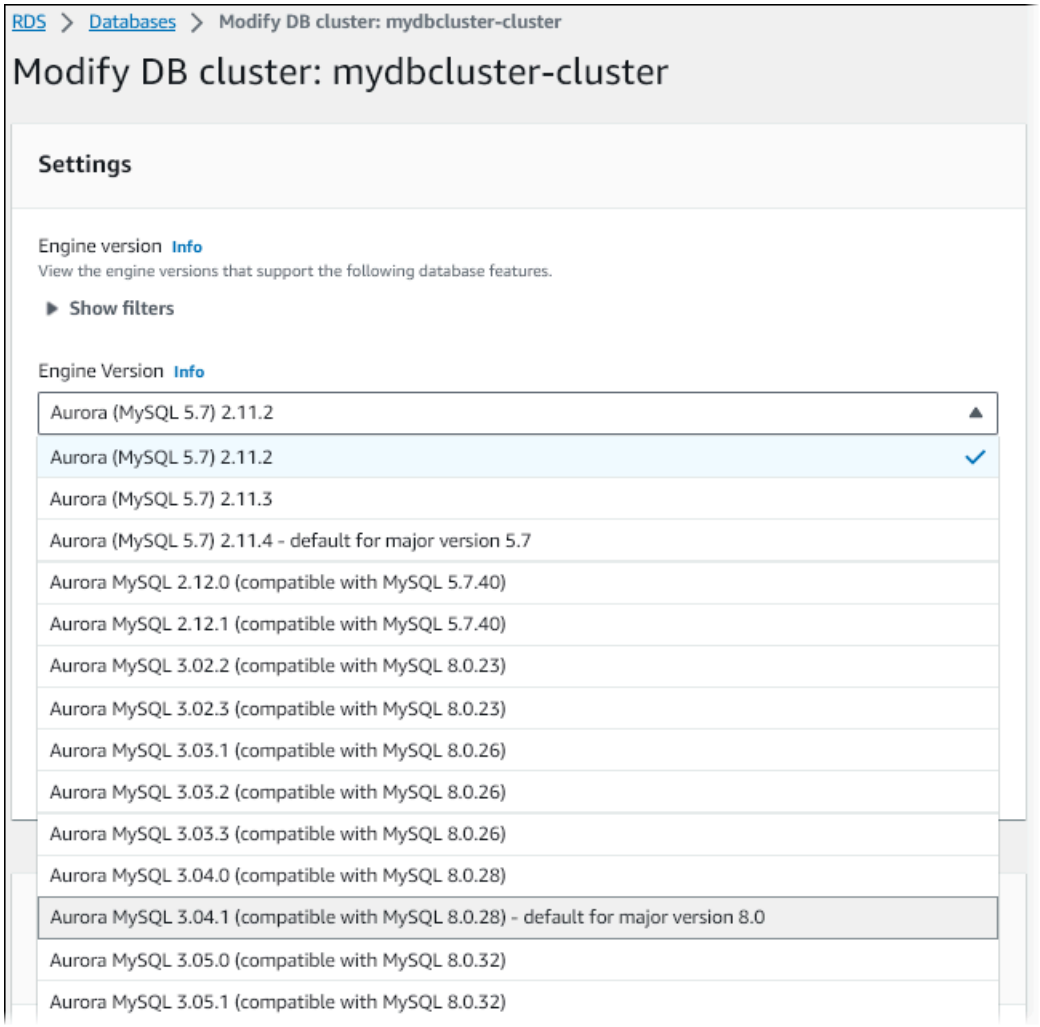
Console

Nell'esempio seguente viene aggiornato il cluster database `mydbcluster-cluster` ad Aurora MySQL versione 3.04.1.

Per aggiornare la versione principale di un cluster di database Aurora MySQL

1. Accedi alla Console di gestione AWS e apri la console Amazon RDS all'indirizzo https://console.aws.amazon.com/rds/.
2. Se è stato utilizzato un gruppo di parametri personalizzato con il cluster di database originale, crea un gruppo di parametri corrispondente compatibile con la nuova versione principale. Apportare le modifiche necessarie ai parametri di configurazione nel nuovo gruppo di parametri. Per ulteriori informazioni, consulta Come gli aggiornamenti in loco influiscono sui gruppi di parametri per un cluster.
3. Nel riquadro di navigazione, scegliere Databases (Database).
4. Scegliere il cluster di database che si desidera modificare.
5. Scegliere Modify (Modifica).
6. Per Versione, scegli una versione principale di Aurora MySQL.

   In genere consigliamo di utilizzare la versione secondaria più recente della versione principale. In questo caso viene scelta la versione predefinita corrente.



7. Scegli Continua.
8. Nella pagina successiva specificare quando eseguire l'aggiornamento. Scegliere During the next scheduled maintenance window (Durante la successiva finestra di manutenzione programmata) o Immediately (Immediatamente).
9. (Facoltativo) Esaminare periodicamente la pagina Events (Eventi) nella console RDS durante l'aggiornamento. In questo modo è possibile monitorare lo stato di avanzamento dell'aggiornamento e identificare eventuali problemi. Se l'aggiornamento incontra dei problemi, consulta Risoluzione dei problemi relativi all'aggiornamento in loco di Aurora MySQL per la procedura da intraprendere.
10. Se all'inizio di questa procedura è stato creato un nuovo gruppo di parametri, associa il gruppo di parametri personalizzati al cluster aggiornato. Per ulteriori informazioni, consulta Come gli aggiornamenti in loco influiscono sui gruppi di parametri per un cluster.

> **Note**
>
> Per eseguire questo passaggio, è necessario riavviare nuovamente il cluster per applicare il nuovo gruppo di parametri.

11. (Facoltativo) Dopo avere completato i test successivi all'aggiornamento, eliminare lo snapshot manuale che Aurora ha creato all'inizio dell'aggiornamento.

AWS CLI

Per aggiornare la versione principale di un cluster di database Aurora MySQL, utilizzare il comando AWS CLI CLI modify-db-cluster con i seguenti parametri obbligatori:

- `--db-cluster-identifier`
- `--engine-version`
- `--allow-major-version-upgrade`
- `--apply-immediately` o `--no-apply-immediately`

Se il cluster utilizza gruppi di parametri personalizzati, includere anche una o entrambe le opzioni seguenti:

- `--db-cluster-parameter-group-name`, se il cluster utilizza un gruppo di parametri cluster personalizzato
- `--db-instance-parameter-group-name`, se alcune istanze del cluster utilizzano un gruppo di parametri database personalizzato

Nell'esempio seguente viene aggiornato il cluster database `sample-cluster` ad Aurora MySQL versione 3.04.1. L'aggiornamento avviene immediatamente, invece di attendere la successiva finestra di manutenzione.

Example

Per Linux, macOS o Unix:

```
aws rds modify-db-cluster \
          --db-cluster-identifier sample-cluster \
          --engine-version 8.0.mysql_aurora.3.04.1 \
```

```
          --allow-major-version-upgrade \
          --apply-immediately
```

Per Windows:

```
aws rds modify-db-cluster ^
          --db-cluster-identifier sample-cluster ^
          --engine-version 8.0.mysql_aurora.3.04.1 ^
          --allow-major-version-upgrade ^
          --apply-immediately
```

È possibile combinare altri comandi della CLI con `modify-db-cluster` per creare un processo end-to-end automatizzato per l'esecuzione e la verifica degli aggiornamenti. Per maggiori informazioni ed esempi, vedi Esercitazione sull'aggiornamento in loco di Aurora MySQL.

> **Note**
>
> Se il cluster fa parte di un database globale Aurora, la procedura di aggiornamento in loco è leggermente diversa. Si chiama l'operazione del comando modify-global-cluster invece di `modify-db-cluster`. Per ulteriori informazioni, consulta Principali aggiornamenti in loco per database globali.

API RDS

Per aggiornare la versione principale di un cluster di database Aurora MySQL, utilizza l'operazione ModifyDBCluster dell'API RDS con i seguenti parametri obbligatori:

- `DBClusterIdentifier`
- `Engine`
- `EngineVersion`
- `AllowMajorVersionUpgrade`
- `ApplyImmediately` (impostato su `true` o `false`)

> **Note**
>
> Se il cluster fa parte di un database globale Aurora, la procedura di aggiornamento in loco è leggermente diversa. Si chiama l'operazione ModifyGlobalCluster invece di

`ModifyDBCluster`. Per ulteriori informazioni, consulta Principali aggiornamenti in loco per database globali.

Come gli aggiornamenti in loco influiscono sui gruppi di parametri per un cluster

I gruppi di parametri di Aurora hanno diversi insiemi di impostazioni di configurazione per i cluster compatibili con MySQL 5.7 o 8.0. Quando si esegue un aggiornamento locale, il cluster aggiornato e tutte le relative istanze devono utilizzare i corrispondenti gruppi di parametri cluster e istanza corrispondenti.

Il cluster e le istanze potrebbero utilizzare i gruppi di parametri compatibili con 5.7 predefiniti. In tal caso, il cluster e l'istanza aggiornati iniziano con i gruppi di parametri compatibili con 8.0 predefiniti. Se il cluster e le istanze utilizzano gruppi di parametri personalizzati, accertarsi di creare i gruppi di parametri compatibili con 8.0 corrispondenti. Accertarsi inoltre di specificarli durante il processo di aggiornamento.

> **Note**
>
> Per la maggior parte delle impostazioni dei parametri, è possibile scegliere il gruppo di parametri personalizzati in due punti. Quando si crea il cluster o quando si associa il gruppo di parametri al cluster in un secondo momento.
> Tuttavia, se si utilizza un'impostazione non predefinita per il parametro `lower_case_table_names`, è necessario impostare il gruppo di parametri personalizzati con questa impostazione in anticipo. Quindi specificare il gruppo di parametri quando si esegue il ripristino dello snapshot per creare il cluster. Qualsiasi modifica al parametro `lower_case_table_names` non ha effetto dopo la creazione del cluster.
> È opportuno utilizzare la stessa impostazione per `lower_case_table_names` durante l'aggiornamento da Aurora MySQL versione 2 alla versione 3.
> Con un Database globale Aurora basato su Aurora MySQL, è possibile eseguire un aggiornamento in loco da Aurora MySQL versione 2 alla versione 3 solo se il parametro `lower_case_table_names` è impostato sull'opzione predefinita e il database globale viene riavviato. Per ulteriori informazioni sui metodi disponibili all'uso, consulta Aggiornamenti di una versione principale.

Modifiche delle proprietà del cluster tra versioni di Aurora MySQL

Quando esegui l'aggiornamento da Aurora MySQL versione 2 alla versione 3, assicurati di modificare le applicazioni o gli script utilizzati per configurare o gestire i cluster e le istanze database Aurora MySQL.

Inoltre, modifica il codice che manipola i gruppi di parametri per tenere conto del fatto che i nomi dei gruppi di parametri predefiniti sono diversi per i cluster compatibili con 5.7 e 8.0. I nomi dei gruppi di parametri predefiniti per i cluster di Aurora MySQL versione 2 e 3 sono `default.aurora-mysql5.7` e `default.aurora-mysql8.0`, rispettivamente.

Ad esempio, è possibile che si disponga di codice simile al seguente applicabile al cluster prima di un aggiornamento.

```
# Check the default parameter values for MySQL 5.7-compatible clusters.
aws rds describe-db-parameters --db-parameter-group-name default.aurora-mysql5.7 --
region us-east-1
```

Dopo avere aggiornato la versione principale del cluster, è necessario modificare tale codice nel modo seguente.

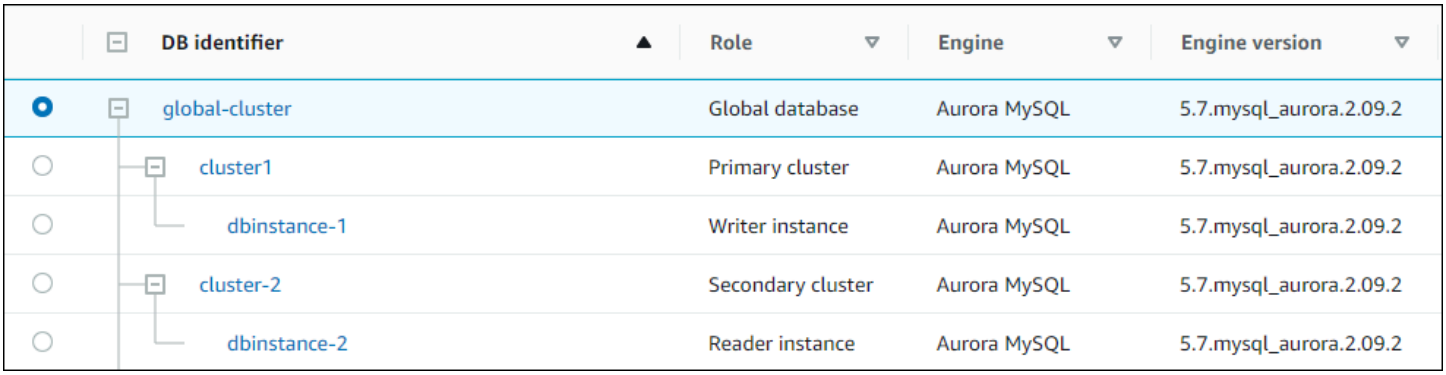
```
# Check the default parameter values for MySQL 8.0-compatible clusters.
aws rds describe-db-parameters --db-parameter-group-name default.aurora-mysql8.0 --
region us-east-1
```

Principali aggiornamenti in loco per database globali

Per un database globale Aurora, è possibile aggiornare il cluster di database globale. Aurora aggiorna automaticamente tutti i cluster nello stesso momento e assicura che tutti eseguano la stessa versione del motore. Questo requisito è dovuto al fatto che tutte le modifiche apportate alle tabelle di sistema, ai formati di file di dati e così via vengono replicate automaticamente in tutti i cluster secondari.

Segui le istruzioni in Come funziona l'aggiornamento della versione principale di Aurora MySQL in loco. Quando specifichi cosa aggiornare, assicurati di scegliere il cluster di database globale anziché uno dei cluster in esso contenuti.

Se utilizzi il plugin Console di gestione AWS, scegli l'articolo con il ruolo Global database (Database globale).



DB identifier	Role	Engine	Engine version
global-cluster	Global database	Aurora MySQL	5.7.mysql_aurora.2.09.2
cluster1	Primary cluster	Aurora MySQL	5.7.mysql_aurora.2.09.2
dbinstance-1	Writer instance	Aurora MySQL	5.7.mysql_aurora.2.09.2
cluster-2	Secondary cluster	Aurora MySQL	5.7.mysql_aurora.2.09.2
dbinstance-2	Reader instance	Aurora MySQL	5.7.mysql_aurora.2.09.2

Se si utilizza la AWS CLI o l'API RDS, avviare il processo di aggiornamento chiamando il comando modify-global-cluster o l'operazione ModifyGlobalCluster. Utilizzare uno di questi anziché `modify-db-cluster` o `ModifyDBCluster`.

> **Note**
>
> Non è possibile specificare un gruppo di parametri personalizzato per il cluster di database globale mentre si esegue un aggiornamento della versione principale del database globale Aurora. Creare i gruppi di parametri personalizzati in ciascuna regione del cluster globale. Applicarli quindi manualmente ai cluster regionali dopo l'aggiornamento.

Per aggiornare la versione principale di un cluster database globale Aurora MySQL mediante la AWS CLI, utilizza il comando modify-global-cluster con i seguenti parametri obbligatori:

- `--global-cluster-identifier`
- `--engine aurora-mysql`
- `--engine-version`
- `--allow-major-version-upgrade`

Nell'esempio seguente il cluster database globale viene aggiornato ad Aurora MySQL versione 3.04.2.

Example

Per Linux, macOS o Unix:

```
aws rds modify-global-cluster \
```

```
          --global-cluster-identifier global_cluster_identifier \
          --engine aurora-mysql \
          --engine-version 8.0.mysql_aurora.3.04.2 \
          --allow-major-version-upgrade
```

Per Windows:

```
aws rds modify-global-cluster ^
          --global-cluster-identifier global_cluster_identifier ^
          --engine aurora-mysql ^
          --engine-version 8.0.mysql_aurora.3.04.2 ^
          --allow-major-version-upgrade
```

Aggiornamenti in loco per cluster di database con repliche di lettura tra Regioni

È possibile aggiornare un cluster di database Aurora con una replica di lettura tra Regioni utilizzando la procedura di aggiornamento in loco, ma occorre tenere a mente alcune considerazioni:

- È necessario prima aggiornare il cluster di database della replica di lettura. Se si tenta di eseguire prima l'aggiornamento del cluster primario, si riceverà un messaggio di errore del tipo seguente:

  Unable to upgrade DB cluster test-xr-primary-cluster because the associated Aurora cross-Region replica test-xr-replica-cluster isn't patched yet. Upgrade the Aurora cross-Region replica and try again.

  Ciò significa che il cluster di database primario non può avere una versione del motore di database superiore a quella del cluster di replica.
- Prima di aggiornare il cluster di database primario, interrompi il carico di lavoro di scrittura e disabilita tutte le nuove richieste di connessione all'istanza database di scrittura del cluster primario.
- Quando aggiorni il cluster primario, scegli un gruppo di parametri del cluster di database personalizzato con il parametro `binlog_format` impostato su un valore che supporti la replica con registrazione di log binari, ad esempio `MIXED`.

  Per ulteriori informazioni sull'utilizzo di registrazione binaria con Aurora, consulta Replica tra Aurora e MySQL o tra Aurora e un altro cluster di database Aurora (replica dei log binari). Per ulteriori informazioni sulla modifica dei parametri di configurazione di Aurora MySQL, consulta Parametri di configurazione Aurora MySQL e Gruppi di parametri per Amazon Aurora.

- Non lasciar passare troppo tempo tra l'aggiornamento del cluster di replica e l'aggiornamento del cluster di database primario. Si consiglia di non attendere oltre la finestra di manutenzione successiva.
- Dopo aver aggiornato il cluster di database primario, riavvia la relativa istanza database di scrittura. Il gruppo di parametri del cluster di database personalizzato che abilita la replica binlog non ha effetto finché l'istanza database di scrittura non viene riavviata.
- Non riavviare il carico di lavoro di scrittura e non abilitare le connessioni all'istanza database di scrittura prima di aver verificato che la replica tra Regioni sia stata riavviata e che il ritardo di replica nella Regione AWS sia pari a 0.

Esercitazione sull'aggiornamento in loco di Aurora MySQL

I seguenti esempi di Linux mostrano come è possibile eseguire i passaggi generali della procedura di aggiornamento in loco utilizzando la AWS CLI.

Questo primo esempio crea un cluster di database Aurora che esegue una versione 2.x di Aurora MySQL. Il cluster include un'istanza database di scrittura e un'istanza database di lettura. Il comando `wait db-instance-available` si interrompe fino a quando l'istanza database di scrittura non è disponibile. Questo è il momento in cui il cluster è pronto per essere aggiornato.

```
aws rds create-db-cluster --db-cluster-identifier mynewdbcluster --engine aurora-mysql
 \
  --db-cluster-version 5.7.mysql_aurora.2.11.2
...
aws rds create-db-instance --db-instance-identifier mynewdbcluster-instance1 \
  --db-cluster-identifier mynewdbcluster --db-instance-class db.t4g.medium --engine
 aurora-mysql
...
aws rds wait db-instance-available --db-instance-identifier mynewdbcluster-instance1
```

Le versioni Aurora MySQL 3.x alle quali è possibile aggiornare il cluster dipendono dalla versione 2.x attualmente in esecuzione sul cluster e dalla Regione AWS in cui si trova il cluster. Il primo comando, con `--output text`, mostra soltanto la versione di destinazione disponibile. Il secondo comando mostra l'output JSON completo della risposta. Questa risposta contiene dettagli quali il valore `aurora-mysql` utilizzato per il parametro `engine`. Inoltre, indica che l'aggiornamento a 3.04.0 rappresenta un aggiornamento della versione principale.

```
aws rds describe-db-clusters --db-cluster-identifier mynewdbcluster \
  --query '*[].{EngineVersion:EngineVersion}' --output text
```

```
5.7.mysql_aurora.2.11.2

aws rds describe-db-engine-versions --engine aurora-mysql --engine-version
 5.7.mysql_aurora.2.11.2 \
  --query '*[].[ValidUpgradeTarget]'
...
{
    "Engine": "aurora-mysql",
    "EngineVersion": "8.0.mysql_aurora.3.04.0",
    "Description": "Aurora MySQL 3.04.0 (compatible with MySQL 8.0.28)",
    "AutoUpgrade": false,
    "IsMajorVersionUpgrade": true,
    "SupportedEngineModes": [
        "provisioned"
    ],
    "SupportsParallelQuery": true,
    "SupportsGlobalDatabases": true,
    "SupportsBabelfish": false,
    "SupportsIntegrations": false
},
...
```

Questo esempio mostra che se si immette un numero di versione di destinazione che non è una destinazione di aggiornamento valida per il cluster, Aurora non esegue l'aggiornamento. Aurora non esegue un aggiornamento della versione principale a meno che non si includa il parametro `--allow-major-version-upgrade`. In questo modo, non è possibile eseguire accidentalmente un aggiornamento che potrebbe richiedere modifiche e test estesi al codice dell'applicazione.

```
aws rds modify-db-cluster --db-cluster-identifier mynewdbcluster \
  --engine-version 5.7.mysql_aurora.2.09.2 --apply-immediately
An error occurred (InvalidParameterCombination) when calling the ModifyDBCluster
 operation: Cannot find upgrade target from 5.7.mysql_aurora.2.11.2 with requested
 version 5.7.mysql_aurora.2.09.2.

aws rds modify-db-cluster --db-cluster-identifier mynewdbcluster \
  --engine-version 8.0.mysql_aurora.3.04.0 --region us-east-1 --apply-immediately
An error occurred (InvalidParameterCombination) when calling the ModifyDBCluster
 operation: The AllowMajorVersionUpgrade flag must be present when upgrading to a new
 major version.

aws rds modify-db-cluster --db-cluster-identifier mynewdbcluster \
  --engine-version 8.0.mysql_aurora.3.04.0 --apply-immediately --allow-major-version-
upgrade
```

```
{
  "DBClusterIdentifier": "mynewdbcluster",
  "Status": "available",
  "Engine": "aurora-mysql",
  "EngineVersion": "5.7.mysql_aurora.2.11.2"
}
```

Sono necessari alcuni istanti prima che lo stato del cluster e le istanze database associate vengano modificati in `upgrading`. I numeri di versione per il cluster e le istanze database cambiano solo al termine dell'aggiornamento. Ancora una volta, è possibile utilizzare il comando `wait db-instance-available` perché l'istanza database di scrittura attenda il completamento dell'aggiornamento prima di procedere.

```
aws rds describe-db-clusters --db-cluster-identifier mynewdbcluster \
  --query '*[].[Status,EngineVersion]' --output text
upgrading 5.7.mysql_aurora.2.11.2

aws rds describe-db-instances --db-instance-identifier mynewdbcluster-instance1 \
  --query '*[].
{DBInstanceIdentifier:DBInstanceIdentifier,DBInstanceStatus:DBInstanceStatus} | [0]'
{
    "DBInstanceIdentifier": "mynewdbcluster-instance1",
    "DBInstanceStatus": "upgrading"
}

aws rds wait db-instance-available --db-instance-identifier mynewdbcluster-instance1
```

A questo punto, il numero di versione del cluster corrisponde a quello specificato per l'aggiornamento.

```
aws rds describe-db-clusters --db-cluster-identifier mynewdbcluster \
  --query '*[].[EngineVersion]' --output text

8.0.mysql_aurora.3.04.0
```

L'esempio precedente ha eseguito un aggiornamento immediato specificando il parametro `--apply-immediately`. Per consentire l'aggiornamento in un momento opportuno, quando si prevede che il cluster non sia occupato, è possibile specificare il parametro `--no-apply-immediately`. In questo modo l'aggiornamento viene avviato durante la successiva finestra di manutenzione del cluster. La finestra di manutenzione definisce il periodo durante il quale possono iniziare le operazioni di manutenzione. Un'operazione di lunga durata potrebbe non terminare durante la finestra di

manutenzione. Pertanto, non è necessario definire una finestra di manutenzione più ampia anche se si prevede che l'aggiornamento potrebbe richiedere molto tempo.

Nell'esempio seguente viene aggiornato un cluster che esegue inizialmente Aurora MySQL versione 2.11.2. Nell'output di `describe-db-engine-versions`, i valori `False` e `True` rappresentano la proprietà `IsMajorVersionUpgrade`. Dalla versione 2.11.2, è possibile eseguire l'aggiornamento ad altre versioni 2.*. Questi aggiornamenti non sono considerati aggiornamenti di versione principali e quindi non richiedono un aggiornamento in loco. L'aggiornamento locale è disponibile solo per gli aggiornamenti alle versioni 3.* visualizzate nell'elenco.

```
aws rds describe-db-clusters --db-cluster-identifier mynewdbcluster \
  --query '*[].{EngineVersion:EngineVersion}' --output text
5.7.mysql_aurora.2.11.2

aws rds describe-db-engine-versions --engine aurora-mysql --engine-version
 5.7.mysql_aurora.2.10.2 \
  --query '*[].[ValidUpgradeTarget][0][0]|[*].[EngineVersion,IsMajorVersionUpgrade]'
 --output text

5.7.mysql_aurora.2.11.3 False
5.7.mysql_aurora.2.11.4 False
5.7.mysql_aurora.2.11.5 False
5.7.mysql_aurora.2.11.6 False
5.7.mysql_aurora.2.12.0 False
5.7.mysql_aurora.2.12.1 False
5.7.mysql_aurora.2.12.2 False
5.7.mysql_aurora.2.12.3 False
8.0.mysql_aurora.3.04.0 True
8.0.mysql_aurora.3.04.1 True
8.0.mysql_aurora.3.04.2 True
8.0.mysql_aurora.3.04.3 True
8.0.mysql_aurora.3.05.2 True
8.0.mysql_aurora.3.06.0 True
8.0.mysql_aurora.3.06.1 True
8.0.mysql_aurora.3.07.1 True

aws rds modify-db-cluster --db-cluster-identifier mynewdbcluster \
  --engine-version 8.0.mysql_aurora.3.04.0 --no-apply-immediately --allow-major-
version-upgrade
...
```

Quando un cluster viene creato senza una finestra di manutenzione specificata, Aurora seleziona un giorno casuale della settimana. In questo caso, il comando `modify-db-cluster` viene inviato un lunedì. Pertanto, modifichiamo la finestra di manutenzione affinché sia di martedì mattina. Tutti gli orari sono rappresentati nel fuso orario UTC. La finestra `tue:10:00-tue:10:30` indica dalle 2:00 alle 2:30 ora del Pacifico. La modifica della finestra di manutenzione ha effetto immediato.

```
aws rds describe-db-clusters --db-cluster-identifier mynewdbcluster --query '*[].
[PreferredMaintenanceWindow]'
[
    [
        "sat:08:20-sat:08:50"
    ]
]

aws rds modify-db-cluster --db-cluster-identifier mynewdbcluster --preferred-
maintenance-window tue:10:00-tue:10:30"
aws rds describe-db-clusters --db-cluster-identifier mynewdbcluster --query '*[].
[PreferredMaintenanceWindow]'
[
    [
        "tue:10:00-tue:10:30"
    ]
]
```

Nell'esempio seguente viene illustrato come ottenere un report degli eventi generati dall'aggiornamento. L'argomento `--duration` rappresenta il numero di minuti per recuperare le informazioni sull'evento. Questo argomento è necessario perché, per impostazione predefinita, `describe-events` restituisce solo gli eventi dell'ultima ora.

```
aws rds describe-events --source-type db-cluster --source-identifier mynewdbcluster --
duration 20160
{
    "Events": [
        {
            "SourceIdentifier": "mynewdbcluster",
            "SourceType": "db-cluster",
            "Message": "DB cluster created",
            "EventCategories": [
                "creation"
            ],
            "Date": "2022-11-17T01:24:11.093000+00:00",
            "SourceArn": "arn:aws:rds:us-east-1:123456789012:cluster:mynewdbcluster"
```

```
        },
        {
            "SourceIdentifier": "mynewdbcluster",
            "SourceType": "db-cluster",
            "Message": "Upgrade in progress: Performing online pre-upgrade checks.",
            "EventCategories": [
                "maintenance"
            ],
            "Date": "2022-11-18T22:57:08.450000+00:00",
            "SourceArn": "arn:aws:rds:us-east-1:123456789012:cluster:mynewdbcluster"
        },
        {
            "SourceIdentifier": "mynewdbcluster",
            "SourceType": "db-cluster",
            "Message": "Upgrade in progress: Performing offline pre-upgrade checks.",
            "EventCategories": [
                "maintenance"
            ],
            "Date": "2022-11-18T22:57:59.519000+00:00",
            "SourceArn": "arn:aws:rds:us-east-1:123456789012:cluster:mynewdbcluster"
        },
        {
            "SourceIdentifier": "mynewdbcluster",
            "SourceType": "db-cluster",
            "Message": "Upgrade in progress: Creating pre-upgrade snapshot [preupgrade-
mynewdbcluster-5-7-mysql-aurora-2-10-2-to-8-0-mysql-aurora-3-02-0-2022-11-18-22-55].",
            "EventCategories": [
                "maintenance"
            ],
            "Date": "2022-11-18T23:00:22.318000+00:00",
            "SourceArn": "arn:aws:rds:us-east-1:123456789012:cluster:mynewdbcluster"
        },
        {
            "SourceIdentifier": "mynewdbcluster",
            "SourceType": "db-cluster",
            "Message": "Upgrade in progress: Cloning volume.",
            "EventCategories": [
                "maintenance"
            ],
            "Date": "2022-11-18T23:01:45.428000+00:00",
            "SourceArn": "arn:aws:rds:us-east-1:123456789012:cluster:mynewdbcluster"
        },
        {
            "SourceIdentifier": "mynewdbcluster",
```

```
            "SourceType": "db-cluster",
            "Message": "Upgrade in progress: Purging undo records for old row versions.
 Records remaining: 164",
            "EventCategories": [
                "maintenance"
            ],
            "Date": "2022-11-18T23:02:25.141000+00:00",
            "SourceArn": "arn:aws:rds:us-east-1:123456789012:cluster:mynewdbcluster"
        },
        {
            "SourceIdentifier": "mynewdbcluster",
            "SourceType": "db-cluster",
            "Message": "Upgrade in progress: Purging undo records for old row versions.
 Records remaining: 164",
            "EventCategories": [
                "maintenance"
            ],
            "Date": "2022-11-18T23:06:23.036000+00:00",
            "SourceArn": "arn:aws:rds:us-east-1:123456789012:cluster:mynewdbcluster"
        },
        {
            "SourceIdentifier": "mynewdbcluster",
            "SourceType": "db-cluster",
            "Message": "Upgrade in progress: Upgrading database objects.",
            "EventCategories": [
                "maintenance"
            ],
            "Date": "2022-11-18T23:06:48.208000+00:00",
            "SourceArn": "arn:aws:rds:us-east-1:123456789012:cluster:mynewdbcluster"
        },
        {
            "SourceIdentifier": "mynewdbcluster",
            "SourceType": "db-cluster",
            "Message": "Database cluster major version has been upgraded",
            "EventCategories": [
                "maintenance"
            ],
            "Date": "2022-11-18T23:10:28.999000+00:00",
            "SourceArn": "arn:aws:rds:us-east-1:123456789012:cluster:mynewdbcluster"
        }
    ]
}
```

Individuazione delle cause degli errori di aggiornamento a una versione principale di Aurora MySQL

Nel tutorial, l'aggiornamento dalla versione 2 alla versione 3 di Aurora MySQL è riuscito. Ma se l'aggiornamento non fosse riuscito, vorresti sapere perché.

È possibile iniziare utilizzando il comando `describe-events` dell'interfaccia AWS CLI per esaminare gli eventi del cluster di database. Questo esempio mostra gli eventi delle ultime 10 ore di `mydbcluster`.

```
aws rds describe-events \
    --source-type db-cluster \
    --source-identifier mydbcluster \
    --duration 600
```

In questo caso, si è verificato un errore nei controlli preliminari dell'aggiornamento.

```
{
    "Events": [
        {
            "SourceIdentifier": "mydbcluster",
            "SourceType": "db-cluster",
            "Message": "Database cluster engine version upgrade started.",
            "EventCategories": [
                "maintenance"
            ],
            "Date": "2024-04-11T13:23:22.846000+00:00",
            "SourceArn": "arn:aws:rds:us-east-1:123456789012:cluster:mydbcluster"
        },
        {
            "SourceIdentifier": "mydbcluster",
            "SourceType": "db-cluster",
            "Message": "Database cluster is in a state that cannot be upgraded: Upgrade
 prechecks failed. For more details, see the
             upgrade-prechecks.log file. For more information on troubleshooting the
 cause of the upgrade failure, see
             https://docs.aws.amazon.com/AmazonRDS/latest/AuroraUserGuide/
AuroraMySQL.Upgrading.Troubleshooting.html",
            "EventCategories": [
                "maintenance"
            ],
            "Date": "2024-04-11T13:23:24.373000+00:00",
            "SourceArn": "arn:aws:rds:us-east-1:123456789012:cluster:mydbcluster"
```

```
        }
    ]
}
```

Per diagnosticare la causa esatta del problema, esaminare i log del database per l'istanza database di scrittura. Quando un aggiornamento alla versione 3 di Aurora MySQL non riesce, l'istanza di scrittura contiene un file di log denominato `upgrade-prechecks.log`. Questo esempio mostra come rilevare la presenza di quel log e quindi scaricarlo in un file locale per l'analisi.

```
aws rds describe-db-log-files --db-instance-identifier mydbcluster-instance \
    --query '*[].[LogFileName]' --output text

error/mysql-error-running.log
error/mysql-error-running.log.2024-04-11.20
error/mysql-error-running.log.2024-04-11.21
error/mysql-error.log
external/mysql-external.log
upgrade-prechecks.log

aws rds download-db-log-file-portion --db-instance-identifier mydbcluster-instance \
    --log-file-name upgrade-prechecks.log \
    --starting-token 0 \
    --output text >upgrade_prechecks.log
```

Il file `upgrade-prechecks.log` è in formato JSON. Lo scarichiamo usando l'opzione `--output text` per evitare la codifica dell'output JSON all'interno di un altro wrapper JSON. Per gli aggiornamenti di Aurora MySQL versione 3, questo log include sempre alcuni messaggi informativi e di avviso. Include messaggi di errore solo se l'aggiornamento non riesce. Se l'aggiornamento riesce, il file di log non viene prodotto.

Per riepilogare tutti gli errori e visualizzare i campi oggetto e descrizione associati, è possibile eseguire il comando `grep -A 2 '"level": "Error"'` sul contenuto del file `upgrade-prechecks.log`. In questo modo viene visualizzata ogni riga di errore e le due righe successive. Queste contengono il nome dell'oggetto di database corrispondente e le indicazioni su come correggere il problema.

```
$ cat upgrade-prechecks.log | grep -A 2 '"level": "Error"'

"level": "Error",
"dbObject": "problematic_upgrade.dangling_fulltext_index",
```

```
"description": "Table `problematic_upgrade.dangling_fulltext_index` contains dangling
 FULLTEXT index. Kindly recreate the table before upgrade."
```

In questo esempio, è possibile eseguire il seguente comando SQL sulla tabella che causa il problema per cercare di risolvere il problema oppure è possibile ricreare la tabella senza l'indice inesatto.

```
OPTIMIZE TABLE problematic_upgrade.dangling_fulltext_index;
```

Provare quindi a eseguire nuovamente l'aggiornamento.

Risoluzione dei problemi relativi all'aggiornamento in loco di Aurora MySQL

Utilizza i seguenti suggerimenti per risolvere i problemi relativi agli aggiornamenti in loco di Aurora MySQL. Questi suggerimenti non si applicano a cluster di database Aurora Serverless.

Motivo dell'annullamento o della lentezza dell'aggiornamento in loco	Effetto	Soluzione per consentire il completamento dell'aggiornamento in loco all'interno della finestra di manutenzione
Non sono ancora state applicate patch alla replica tra Regioni Aurora associata	Aurora annulla l'aggiornamento.	Aggiorna la replica tra Regioni Aurora e riprova.
Il cluster ha transazioni XA nello stato preparato	Aurora annulla l'aggiornamento.	Salvare o eseguire il rollback di tutte le transazioni XA preparate.
Il cluster sta elaborando un'istruzione DDL (Data Definition Language)	Aurora annulla l'aggiornamento.	Prendere in considerazione di attendere il completamento di tutte le istruzioni DDL prima di eseguire l'aggiornamento.
Il cluster ha modifiche non salvate per molte righe	L'aggiornamento potrebbe richiedere molto tempo.	Il processo di aggiornamento esegue il rollback delle modifiche non salvate. L'indicatore per questa condizione è il valore di `TRX_ROWS_MODIFIED` nella tabella `INFORMATION_SCHEMA.INNODB_TRX`.

Motivo dell'annullamento o della lentezza dell'aggiornamento in loco	Effetto	Soluzione per consentire il completamento dell'aggiornamento in loco all'interno della finestra di manutenzione
		Prendere in considerazione l'esecuzione dell'aggiornamento solo dopo avere salvato le modifiche o eseguito il rollback di tutte le transazioni di grandi dimensioni.
Il cluster ha un numero elevato di record di annullamento	L'aggiornamento potrebbe richiedere molto tempo.	Anche se le transazioni non salvate non interessano un numero elevato di righe, potrebbero comportare un grande volume di dati. Ad esempio, è possibile che si inserisca no BLOB di grandi dimensioni. Aurora non rileva o genera automaticamente un evento per questo tipo di attività di transazione. L'indicatore per questa condizione è la lunghezza dell'elenco della cronologia (HLL). Il processo di aggiornamento esegue il rollback delle modifiche non salvate. È possibile controllare l'HLL nell'output del comando SQL `SHOW ENGINE INNODB STATUS` o direttamente utilizzando la seguente query SQL: `SELECT count FROM information_schema .innodb_metrics WHERE name = 'trx_rseg_history_len';` Puoi monitorare la metrica `RollbackSegmentHistoryListLength` anche in Amazon CloudWatch. Valuta la possibilità di eseguire l'aggiornamento solo dopo che la lunghezza dell'elenco della cronologia sarà diminuita.

Motivo dell'annullamento o della lentezza dell'aggiornamento in loco	Effetto	Soluzione per consentire il completamento dell'aggiornamento in loco all'interno della finestra di manutenzione
Il cluster è in fase di salvataggio di una transazione di log binario di grandi dimensioni	L'aggiornamento potrebbe richiedere molto tempo.	Il processo di aggiornamento attende fino a quando non vengono applicate le modifiche del log binario. Altre transazioni o istruzioni DDL potrebbero iniziare durante questo periodo, rallentando ulteriormente il processo di aggiornamento. Pianificare il processo di aggiornamento quando il cluster non è occupato con la generazione di modifiche alla replica del log binario. Aurora non rileva o genera automaticamente un evento per questa condizione.

Motivo dell'annullamento o della lentezza dell'aggiornamento in loco	Effetto	Soluzione per consentire il completamento dell'aggiornamento in loco all'interno della finestra di manutenzione
Incongruenze dello schema derivanti dalla rimozione o dal danneggiamento dei file	Aurora annulla l'aggiornamento.	Cambiare il motore di archiviazione predefinito per le tabelle temporanee da MyISAM in InnoDB. Esegui queste fasi: 1. Modificare il parametro database `default_tmp_storage_engine` impostandolo su `InnoDB`. 2. Riavviare il cluster database. 3. Dopo il riavvio, verificare che il parametro database `default_tmp_storage_engine` sia impostato su `InnoDB`. Utilizza il seguente comando: `show global variables like 'default_tmp_storage_engine';` 4. Assicurarsi di non creare tabelle temporanee che utilizzino il motore di archiviazione MyISAM. È consigliabile sospendere qualsiasi carico di lavoro del database e non creare nuove connessioni al database perché l'aggiornamento verrà rilasciato a breve. 5. Provare di nuovo l'aggiornamento sul posto.

Motivo dell'annullamento o della lentezza dell'aggiornamento in loco	Effetto	Soluzione per consentire il completamento dell'aggiornamento in loco all'interno della finestra di manutenzione

L'utente master è stato eliminato	Aurora annulla l'aggiornamento.	**Important**
		Non eliminare l'utente master.
		
		Tuttavia, se per qualche motivo dovessi eliminare l'utente master, ripristinalo utilizzando i seguenti comandi SQL:
		
		`CREATE USER 'master_username '@'% IDENTIFIED BY ' master_user_password ' REQUIRE NONE PASSWORD EXPIRE DEFAULT ACCOUNT UNLOCK;`
		
		`GRANT SELECT, INSERT, UPDATE, DELETE, CREATE, DROP, RELOAD, PROCESS, REFERENCES, INDEX, ALTER, SHOW DATABASES, CREATE TEMPORARY TABLES, LOCK TABLES, EXECUTE, REPLICATION SLAVE, REPLICATION CLIENT, CREATE VIEW, SHOW VIEW, CREATE ROUTINE, ALTER ROUTINE, CREATE USER, EVENT, TRIGGER, LOAD FROM S3, SELECT INTO S3, INVOKE LAMBDA, INVOKE SAGEMAKER, INVOKE COMPREHEND ON *.* TO 'master_username '@'% WITH GRANT OPTION;`

Per ulteriori dettagli sulla risoluzione dei problemi che causano l'esito negativo dei controlli preliminari di aggiornamento, consulta i seguenti blog:

- Amazon Aurora MySQL version 2 (with MySQL 5.7 compatibility) to version 3 (with MySQL 8.0 compatibility) upgrade checklist, Part 1

- Amazon Aurora MySQL version 2 (with MySQL 5.7 compatibility) to version 3 (with MySQL 8.0 compatibility) upgrade checklist, Part 2

È possibile utilizzare la procedura seguente per eseguire controlli personalizzati per alcune delle condizioni della tabella precedente. In questo modo, è possibile pianificare l'aggiornamento in un momento in cui si sa che il database si trova in uno stato in cui l'aggiornamento può essere completato correttamente e con rapidità.

- È possibile verificare la presenza di transazioni XA aperte eseguendo l'istruzione `XA RECOVER`. È quindi possibile salvare le modifiche o eseguire il rollback delle transazioni XA prima di iniziare l'aggiornamento.
- È possibile verificare la presenza di istruzioni DDL eseguendo un'istruzione `SHOW PROCESSLIST` e cercando le istruzioni `CREATE`, `DROP`, `ALTER`, `RENAME` e `TRUNCATE` nell'output. Consentire il completamento di tutte le istruzioni DDL prima di iniziare l'aggiornamento.
- È possibile controllare il numero totale di righe non salvate eseguendo una query sulla tabella `INFORMATION_SCHEMA.INNODB_TRX`. La tabella contiene una riga per ogni transazione. La colonna `TRX_ROWS_MODIFIED` contiene il numero di righe modificate o inserite dalla transazione.
- È possibile controllare la lunghezza della lista della cronologia InnoDB eseguendo l'istruzione `SHOW ENGINE INNODB STATUS SQL` e cercando la `History list length` nell'output. È inoltre possibile controllare direttamente il valore eseguendo la seguente query:

  ```
  SELECT count FROM information_schema.innodb_metrics WHERE name =
   'trx_rseg_history_len';
  ```

  La lunghezza dell'elenco cronologico corrisponde alla quantità di informazioni di annullamento memorizzate dal database per implementare il controllo della concorrenza multiversione (MVCC).

Pulizia post-aggiornamento per Aurora MySQL versione 3

Dopo aver completato l'aggiornamento di un cluster Aurora MySQL versione 2 ad Aurora MySQL versione 3, è possibile eseguire queste altre azioni di pulizia:

- Crea nuove versioni compatibili con MySQL 8.0 di qualsiasi gruppo di parametri personalizzati. Applicare i valori dei parametri personalizzati necessari ai nuovi gruppi di parametri.

- Aggiorna eventuali allarmi di CloudWatch, script di configurazione e così via per utilizzare i nuovi nomi per tutte le metriche i cui nomi sono stati influenzati da modifiche linguistiche inclusive. Per un elenco di parametri, consultare Cambiamenti linguistici inclusivi per Aurora MySQL versione 3.
- Aggiornare qualsiasi modello CloudFormation per utilizzare i nuovi nomi per qualsiasi parametro di configurazione i cui nomi sono stati influenzati da modifiche linguistiche inclusive. Per l'elenco completo dei parametri, consultare Cambiamenti linguistici inclusivi per Aurora MySQL versione 3.

Indici spaziali

Dopo l'aggiornamento a Aurora MySQL versione 3, verificare se è necessario eliminare o ricreare oggetti e indici relativi agli indici spaziali. Prima di MySQL 8.0, Aurora poteva ottimizzare le query spaziali utilizzando indici che non contengono un identificatore delle risorse spaziali (SRID). Aurora MySQL versione 3 utilizza solo indici spaziali contenenti SRID. Durante un aggiornamento, Aurora rilascia automaticamente tutti gli indici spaziali senza SRID e stampa i messaggi di avviso nel registro del database. Se si osservano tali messaggi di avviso, creare nuovi indici spaziali con SRID dopo l'aggiornamento. Per ulteriori informazioni sulle modifiche alle funzioni spaziali e ai tipi di dati in MySQL 8.0, consultare Changes in MySQL 8.0 (Modifiche in MySQL 8.0)nel Manuale di riferimento MySQL.

Aggiornamenti e correzioni per il motore del database per Amazon Aurora MySQL

Puoi trovare le seguenti informazioni nelle note di rilascio di Amazon Aurora edizione compatibile con MySQL:

- Aggiornamenti del motore del database per Amazon Aurora MySQL versione 3
- Aggiornamenti del motore del database Amazon Aurora MySQL versione 2
- Aggiornamenti del motore del database per Amazon Aurora MySQL versione 1 (obsoleta)
- Correzione dei bug di MySQL attraverso gli aggiornamenti del motore del database Aurora MySQL
- Vulnerabilità di sicurezza risolte in Amazon Aurora MySQL

Utilizzo di Amazon Aurora PostgreSQL

Amazon Aurora PostgreSQL è un motore di database relazionale compatibile con ACID e PostgreSQL completamente gestito che combina la velocità, l'affidabilità e la gestibilità di Amazon Aurora con la semplicità e la convenienza dei database open source. Aurora PostgreSQL è un'alternativa a PostgreSQL che semplifica e rende più conveniente dal punto di vista dei costi la configurazione, l'utilizzo e il dimensionamento delle implementazioni PostgreSQL nuove ed esistenti, consentendoti di concentrarti sulle tue attività e applicazioni. Per ulteriori informazioni su Aurora in generale, consulta Che cos'è Amazon Aurora?.

Oltre ai vantaggi di Aurora, Aurora PostgreSQL offre un comodo percorso di migrazione da Amazon RDS ad Aurora, con strumenti di migrazione rapidi che convertono le applicazioni RDS per PostgreSQL esistenti in Aurora PostgreSQL. Anche le attività di database di routine, come il provisioning, l'applicazione di patch, il backup, il ripristino, il rilevamento di guasti e la correzione, sono facili da gestire con Aurora PostgreSQL.

Aurora PostgreSQL può funzionare con molti standard di settore. Ad esempio, è possibile utilizzare i database Aurora PostgreSQL per creare applicazioni conformi allo standard HIPAA e memorizzare le relative informazioni sanitarie, tra cui i dati sanitari protetti (PHI) da un Business Associate Agreement (BAA) stipulato con AWS.

Aurora PostgreSQL è idoneo per FedRAMP HIGH. Per i dettagli AWS e gli sforzi di conformità, consulta la sezione AWS Servizi rientranti nell'ambito del programma di conformità.

Important

Se riscontri un problema con il cluster DB Aurora PostgreSQL, il tuo agente di AWS supporto potrebbe aver bisogno di ulteriori informazioni sullo stato dei tuoi database. L'obiettivo è garantire che AWS Support riceva quante più informazioni richieste possibile nel più breve tempo possibile.
Puoi utilizzare PG Collector per raccogliere informazioni preziose sul database in un file HTML consolidato. Per ulteriori informazioni su PG Collector, su come eseguirlo e su come scaricare il report HTML, consulta PG Collector.
Una volta completato correttamente, lo script restituisce l'output in un formato HTML leggibile, salvo diversa indicazione. Lo script è progettato per escludere dall'output HTML tutti i dati o i dettagli di sicurezza che possono compromettere l'azienda. Inoltre, non apporta modifiche al database o all'ambiente. Tuttavia, se nell'output HTML sono presenti informazioni che

non desideri condividere, rimuovile prima di caricare l'output HTML. Se l'output HTML è accettabile, caricalo utilizzando la sezione degli allegati nei dettagli del caso del tuo caso di supporto.

Argomenti

- Lavorare con l'ambiente di anteprima del database
- Sicurezza con Amazon Aurora PostgreSQL
- Aggiornamento delle applicazioni per la connessione ai cluster DB Aurora PostgreSQL utilizzando nuovi certificati SSL/TLS
- Utilizzo di Autenticazione Kerberos con Aurora PostgreSQL
- Migrazione di dati su Amazon Aurora con compatibilità PostgreSQL
- Ottimizzazione delle prestazioni delle query in Aurora PostgreSQL
- Utilizzo delle tabelle non registrate in Aurora PostgreSQL
- Utilizzo della funzionalità di autovacuum di PostgreSQL in Amazon Aurora PostgreSQL
- Gestione delle controversie relative a TOAST OID in
- Utilizzo di Babelfish per Aurora PostgreSQL
- Prestazioni e dimensionamento per Amazon Aurora PostgreSQL
- Sintonizzazione degli eventi di attesa per Aurora PostgreSQL
- Ottimizzazione di DevOps
- Best practice con Amazon Aurora PostgreSQL
- Replica con Amazon Aurora PostgreSQL
- Inoltro di scrittura locale in Aurora PostgreSQL
- Utilizzo di Aurora PostgreSQL come Knowledge Base per Amazon Bedrock
- Integrazione di Amazon Aurora PostgreSQL con altri servizi AWS
- Monitoraggio dei piani di esecuzione delle query e della memoria di picco per Aurora PostgreSQL
- Gestione dei piani di esecuzione delle query per Aurora PostgreSQL
- Utilizzo di estensioni e wrapper di dati esterni
- Utilizzo di Trusted Language Extensions per PostgreSQL
- Riferimento Amazon Aurora PostgreSQL
- Aggiornamenti del motore di database Amazon Aurora PostgreSQL per 02/06/2020

Lavorare con l'ambiente di anteprima del database

La Community PostgreSQL rilascia ogni anno la nuova versione principale di PostgreSQL, Analogamente, Amazon Aurora rende disponibili alcune versioni principali di PostgreSQL come versioni di anteprima. In tal modo puoi creare cluster di database utilizzando la versione di anteprima e testarne le funzionalità nell'ambiente di anteprima del database.

I cluster Aurora PostgreSQL nell'ambiente di anteprima del database sono funzionalmente simili agli altri cluster Aurora PostgreSQL. Tuttavia, una versione di anteprima non può essere utilizzata in produzione.

Tieni presenti le importanti limitazioni riportate di seguito:

- Tutte le istanze database e tutti i cluster di database vengono eliminati 60 giorni dopo la creazione, insieme a eventuali backup e snapshot.
- Puoi creare un'istanza database solo in un virtual private cloud (VPC) basato sul servizio Amazon VPC.
- Non puoi copiare uno snapshot di un'istanza database in un ambiente di produzione.

Le seguenti opzioni sono supportate dall'anteprima.

- È possibile creare istanze database usando solo i tipi di istanza r5, r6g, r6i, r7g, r7i, r8g, x2g, t3 e t4g. Per ulteriori informazioni sulle classi di istanza, consulta Classi di istanze database Amazon Aurora.
- È possibile utilizzare distribuzioni Single-AZ e Multi-AZ.
- È possibile utilizzare le funzioni di dump e caricamento standard di PostgreSQL per esportare database da o importare database nell'ambiente di anteprima database.

Tipi di classi di istanza database supportate

Amazon Aurora PostgreSQL supporta le seguenti classi di istanza database nella Regione di anteprima:

Classi ottimizzate per la memoria

- db.r5: classi di istanza ottimizzate per la memoria
- db.r6g: classi di istanza ottimizzate per la memoria basate su processori AWS Graviton2

- db.r6i: classi di istanza ottimizzate per la memoria
- db.r7g: classi di istanza ottimizzate per la memoria basate su processori AWS Graviton3
- db.r7i: classi di istanza ottimizzate per la memoria
- db.r8g: classi di istanza ottimizzate per la memoria basate su processori AWS Graviton4
- db.x2g: classi di istanza ottimizzate per la memoria con processori AWS Graviton2

> **Note**
>
> Per ulteriori informazioni sulle classi di istanza, consulta Tipi di classi di istanza database.

Classi espandibili

- db.t3.medium
- db.t3.large
- db.t4g.medium
- db.t4g.large

Funzionalità non supportate nell'ambiente di anteprima

Le seguenti funzionalità non sono disponibili nell'ambiente di anteprima:

- Aurora Serverless v1 e v2
- Aggiornamenti di una versione principale (MVU)
- Non verranno rilasciate nuove versioni secondarie nella Regione di anteprima
- Replica in entrata da RDS per PostgreSQL ad Aurora PostgreSQL
- Implementazioni blu/verdi di Amazon RDS
- Copia di snapshot tra regioni diverse
- Database globale Aurora
- Flussi di attività di database (DAS), proxy RDS e AWS DMS
- Repliche di lettura con dimensionamento automatico
- AWS Bedrock

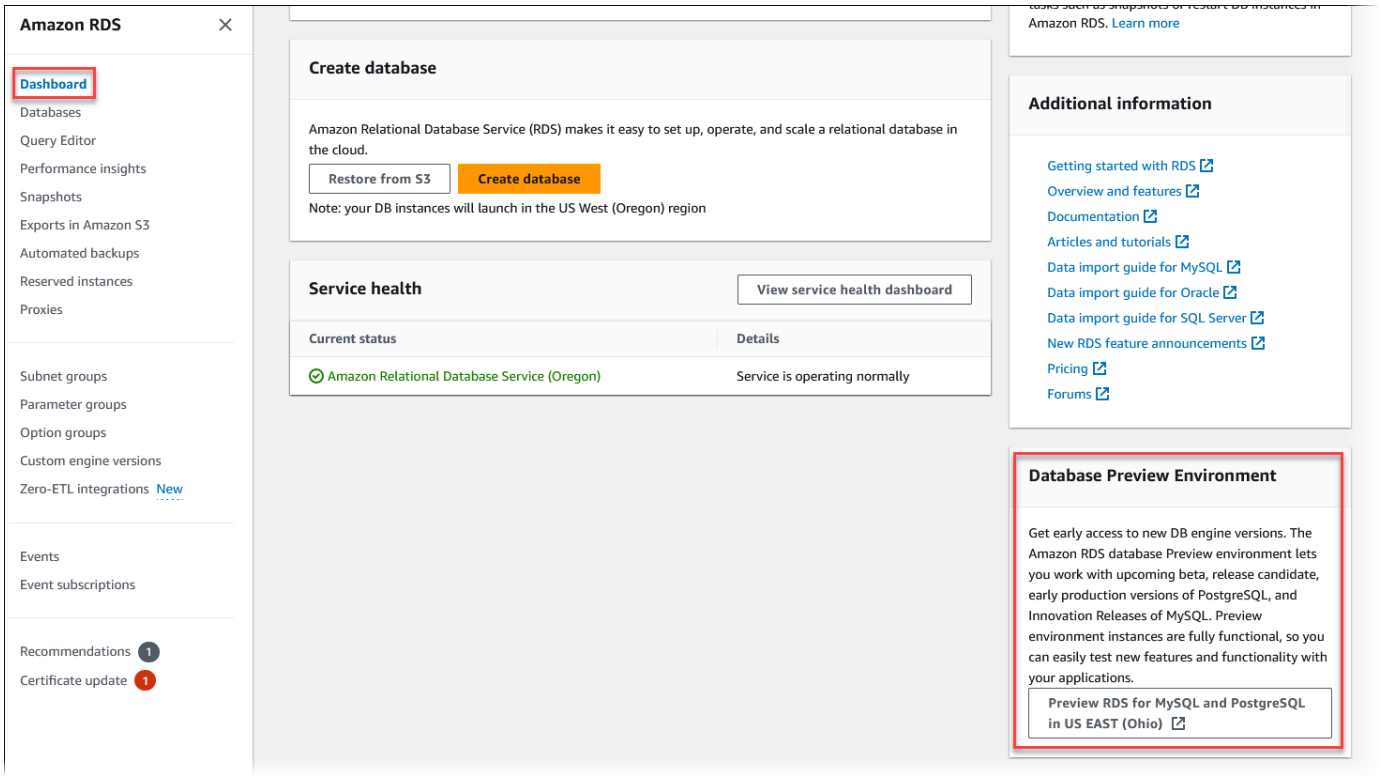
- Esportazione RDS
- Performance Insights
- Endpoint personalizzati
- Copia snapshot
- Zero-ETL
- Babelfish
- Tutte le estensioni

Creazione di un nuovo cluster di database nell'ambiente di anteprima

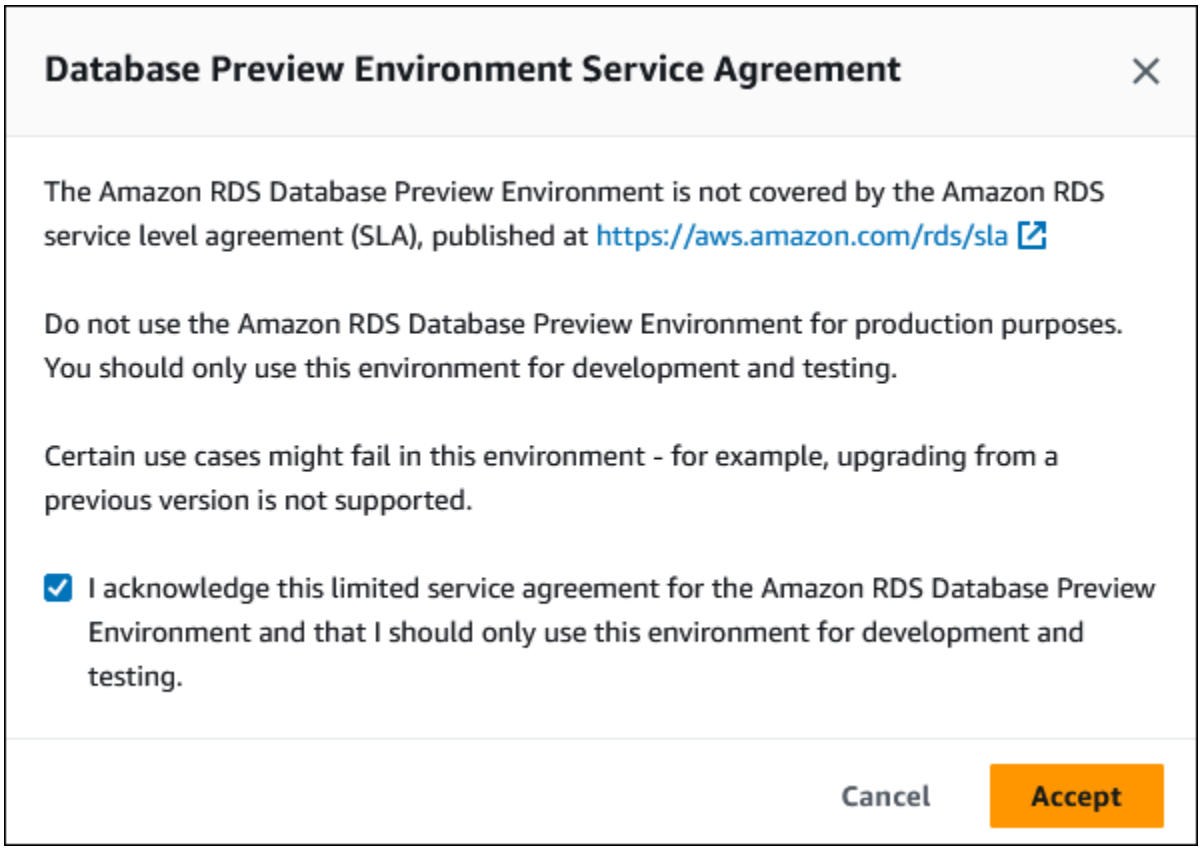
Utilizza la seguente procedura per creare un cluster di database nell'ambiente di anteprima.

Per creare un cluster di database nell'ambiente di anteprima

1. Accedi alla Console di gestione AWS e apri la console Amazon RDS all'indirizzo https://console.aws.amazon.com/rds/.
2. Scegliere Dashboard (Pannello di controllo) nel pannello di navigazione.
3. Nella pagina Dashboard (Pannello di controllo), individua la sezione Database Preview Environment (Ambiente di anteprima del database), come mostrato nell'immagine seguente.



L'ambiente di anteprima del database è accessibile direttamente. Prima di poter procedere, è necessario capire e accettare le limitazioni.



4. Per creare il cluster di database Aurora PostgreSQL, segui la stessa procedura utilizzata per creare qualsiasi cluster di database Aurora. Per ulteriori informazioni, consulta Creazione di un cluster database Amazon Aurora.

Per creare un'istanza nell'ambiente di anteprima del database utilizzando l'API Aurora o AWS CLI, utilizza il seguente endpoint.

```
rds-preview.us-east-2.amazonaws.com
```

PostgreSQL versione 17 nell'ambiente di anteprima del database

⚠ Questa è la documentazione di anteprima per Aurora PostgreSQL versione 17. È soggetta a modifiche.

PostgreSQL versione 17.0 è ora disponibile nell'ambiente di anteprima del database Amazon RDS. PostgreSQL versione 17 include vari miglioramenti, descritti nella seguente documentazione PostgreSQL:

- Rilascio di PostgreSQL 17

Per informazioni sull'ambiente di anteprima del database, consulta Lavorare con l'ambiente di anteprima del database. Per accedere all'ambiente di anteprima dalla console, selezionare https://console.aws.amazon.com/rds-preview/.

Sicurezza con Amazon Aurora PostgreSQL

Per una panoramica generale della sicurezza in Aurora, consulta Sicurezza in Amazon Aurora. Puoi gestire la sicurezza per Amazon Aurora PostgreSQL a diversi livelli:

- Per controllare chi può eseguire azioni di gestione di Amazon RDS su cluster e istanze DB PostgreSQL Aurora, usa (IAM). AWS Identity and Access Management IAM gestisce l'autenticazione dell'identità utente prima che questi possa accedere al servizio. Gestisce inoltre l'autorizzazione, ovvero se l'utente è autorizzato a fare ciò che sta cercando di fare. L'autenticazione del database IAM è un metodo di autenticazione aggiuntivo che è possibile scegliere quando si crea un cluster di database Aurora PostgreSQL. Per ulteriori informazioni, consulta Gestione accessi e identità per Amazon Aurora.

  Se utilizzi IAM con il tuo cluster Aurora PostgreSQL DB, accedi prima con le tue credenziali IAM, prima di aprire la Console di gestione AWS console Amazon RDS all'indirizzo. https://console.aws.amazon.com/rds/
- Assicurati che i cluster di database Aurora vengano creati in un cloud privato virtuale (VPC) utilizzando il servizio Amazon VPC. Per controllare quali dispositivi e EC2 istanze Amazon possono aprire connessioni all'endpoint e alla porta dell'istanza DB per i cluster Aurora DB in un VPC, utilizza un gruppo di sicurezza VPC. Le connessioni a endpoint e porte possono essere stabilite tramite Secure Sockets Layer (SSL). Le regole del firewall aziendale possono inoltre determinare se i dispositivi in esecuzione nell'azienda possono aprire connessioni a un'istanza database. Per ulteriori informazioni su, consulta. VPCs VPC Amazon e Amazon Aurora

  La tenancy VPC supportata dipende dalla classe di istanze database usata dai cluster DB di Aurora PostgresSQL. Con la tenancy VPC `default`, il cluster di database viene eseguito nell'hardware condiviso. Con la tenancy VPC `dedicated`, il cluster di database viene eseguito in un'istanza

hardware dedicata. Le classi di istanza database delle prestazioni in modalità burst supportano solo la tenancy VPC di default. Le classi di istanza database delle prestazioni in modalità burst includono le classi di istanza database db.t3 e db.t4g. Tutte le altre classi di istanza database di Aurora PostgreSQL DB supportano sia il tenancy VPC di default che dedicato.

Per ulteriori informazioni sulle classi di istanza, consulta Classi di istanze database Amazon Aurora. Per ulteriori informazioni sulle tenancy VPC `default` e `dedicated`, consulta Istanze dedicate nella Guida per l'utente di Amazon Elastic Compute Cloud.

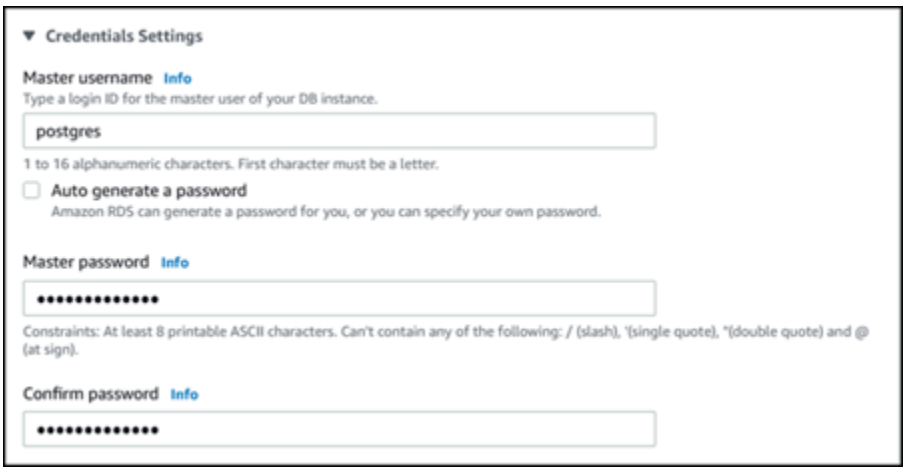
- Per concedere autorizzazioni ai database PostgreSQL eseguiti sul cluster di database Amazon Aurora, puoi adottare lo stesso approccio generale utilizzato con le istanze standalone di PostgreSQL. I comandi come `CREATE ROLE`, `ALTER ROLE`, `GRANT` e `REVOKE` funzionano esattamente come nei database on-premise, ovvero in modo analogo alla modifica diretta delle tabelle dello schema del database.

  PostgreSQL gestisce i privilegi mediante i ruoli. Il ruolo `rds_superuser` è quello più privilegiato a livello di cluster di database Aurora PostgreSQL. Questo ruolo viene creato automaticamente e viene concesso all'utente che crea il cluster di database (l'account utente principale, `postgres` per impostazione predefinita). Per ulteriori informazioni, consulta Informazioni su ruoli e autorizzazioni di PostgreSQL.

Tutte le versioni di Aurora PostgreSQL disponibili, incluse le versioni 10, 11, 12, 13, 14 e versioni successive, supportano il Salted Challenge Response Authentication Mechanism (SCRAM) per le password come alternativa a message digest (). MD5 Ti consigliamo di utilizzare SCRAM perché è più sicuro di. MD5 Per ulteriori informazioni, incluso come migrare le password degli utenti del database da SCRAM MD5 a SCRAM, consulta. Utilizzo delle crittografia password SCRAM per PostgreSQL

Informazioni su ruoli e autorizzazioni di PostgreSQL

Quando crei un cluster di database Aurora PostgreSQL utilizzando la Console di gestione AWS, viene contemporaneamente creato un account amministratore. Per impostazione predefinita, verrà chiamato `postgres`, come mostrato nello screenshot seguente:



Anziché accettare il valore predefinito (`postgres`) è possibile scegliere un nome diverso. In tal caso, il nome scelto deve iniziare con una lettera e contenere da 1 a 16 caratteri alfanumerici. Per semplicità, facciamo riferimento a questo account utente principale utilizzando il suo valore predefinito (`postgres`) in tutta la Guida.

Se utilizzi il comando AWS CLI `create-db-cluster` invece del corrispondente comando della Console di gestione AWS, il nome utente viene creato passandolo con il parametro `master-username`. Per ulteriori informazioni, consulta Fase 2: creazione di un cluster di database Aurora PostgreSQL.

Indipendentemente dal fatto che utilizzi la Console di gestione AWS, la AWS CLI oppure l'API Amazon RDS e che utilizzi il nome `postgres` predefinito o scelga un nome diverso, questo primo account utente del database è membro del gruppo `rds_superuser` e dispone dei privilegi `rds_superuser`.

Argomenti

- Comprendere il ruolo rds_superuser
- Controllo dell'accesso utente al database PostgreSQL
- Delega e controllo della gestione delle password utente
- Utilizzo delle crittografia password SCRAM per PostgreSQL

Comprendere il ruolo rds_superuser

In PostgreSQL, un ruolo può definire un utente, un gruppo o un insieme di autorizzazioni specifiche concesse a un gruppo o a un utente per vari oggetti nel database. I comandi PostgreSQL `CREATE USER` e `CREATE GROUP` sono stati sostituiti dal comando `CREATE ROLE` più generico, ma con

proprietà specifiche per distinguere gli utenti del database. Un utente del database può essere paragonato a un ruolo con il privilegio LOGIN.

> **Note**
>
> È comunque possibile continuare a utilizzare i comandi `CREATE USER` e `CREATE GROUP`. Per ulteriori informazioni, consulta la sezione relativa ai ruoli di database nella documentazione di PostgreSQL.

L'utente `postgres` è l'utente di database più privilegiato nel cluster di database Aurora PostgreSQL. Ha le caratteristiche definite dalla seguente istruzione `CREATE ROLE`.

```
CREATE ROLE postgres WITH LOGIN NOSUPERUSER INHERIT CREATEDB CREATEROLE NOREPLICATION
 VALID UNTIL 'infinity'
```

Le proprietà `NOSUPERUSER`, `NOREPLICATION`, `INHERIT` e `VALID UNTIL 'infinity'` sono le opzioni predefinite per CREATE ROLE, se non diversamente specificato.

Per impostazione predefinita, `postgres` dispone dei privilegi concessi al `rds_superuser` ruolo e delle autorizzazioni per creare ruoli e database. Il ruolo `rds_superuser` consente all'utente `postgres` di eseguire le seguenti operazioni:

- Aggiungere le estensioni che sono disponibili per l'uso con Aurora PostgreSQL. Per ulteriori informazioni, consulta Utilizzo di estensioni e wrapper di dati esterni.
- Creare ruoli per gli utenti e concedere i relativi privilegi. Per ulteriori informazioni, consulta CREATE ROLE e GRANTnella documentazione di PostgreSQL.
- Creare database. Per ulteriori informazioni, consulta CREATE DATABASE nella documentazione di PostgreSQL.
- Concedere privilegi `rds_superuser` a ruoli utente che non dispongono di questi privilegi e revocare i privilegi, se necessario. Si consiglia di concedere questo ruolo solo agli utenti che eseguono attività superuser. In altre parole, è possibile concedere questo ruolo agli amministratori di database (DBA) o agli amministratori di sistema.
- Concedere (e revocare) il ruolo `rds_replication` per gli utenti del database che non hanno il ruolo `rds_superuser`.
- Concedere (e revocare) il ruolo `rds_password` per gli utenti del database che non hanno il ruolo `rds_superuser`.

- Ottenere informazioni sullo stato di tutte le connessioni al database utilizzando la vista `pg_stat_activity`. Quando necessario, il ruolo `rds_superuser` può arrestare qualsiasi connessione utilizzando il comando `pg_terminate_backend` o `pg_cancel_backend`.

Nell'istruzione `CREATE ROLE postgres...`, si può vedere che il ruolo utente `postgres` non concede specificamente autorizzazioni PostgreSQL `superuser`. Aurora PostgreSQL è un servizio gestito e pertanto non è possibile accedere al sistema operativo host, né connettersi utilizzando l'account PostgreSQL`superuser`. Molte delle attività che richiedono l'accesso di tipo `superuser` su un PostgreSQL autonomo viene gestito automaticamente da Aurora.

Per ulteriori informazioni sulla concessione dei privilegi, consulta la sezione relativa al comando GRANT nella documentazione di PostgreSQL.

Il ruolo `rds_superuser` è uno dei diversi ruoli predefinito in uncluster di database Aurora PostgreSQL.

> **Note**
>
> In PostgreSQL 13 e versioni precedenti, i ruoli di default sono conosciuti come ruoli predefiniti.

L'elenco seguente fornisce alcuni degli altri ruoli predefiniti creati automaticamente per un nuovo cluster di database Aurora PostgreSQL. I ruoli predefiniti e i relativi privilegi non possono essere modificati. Non è possibile eliminare, rinominare o modificare i privilegi per questi ruoli predefiniti. Qualsiasi tentativo comporta la generazione di un errore.

- rds_password - Un ruolo in grado di modificare le password e configurare vincoli di password per gli utenti del database. Questo ruolo `rds_superuser` viene assegnato al ruolo per impostazione predefinita e può concedere il ruolo agli utenti del database. Per ulteriori informazioni, consulta Controllo dell'accesso utente al database PostgreSQL.
  - Per le versioni di RDS per PostgreSQL precedenti alla 14, il ruolo `rds_password` può modificare le password e configurare vincoli di password per gli utenti del database e gli utenti con ruolo `rds_superuser`. A partire dalla versione 14 di RDS per PostgreSQL, il ruolo `rds_password` può modificare le password e configurare vincoli di password solo per gli utenti del database. Solo gli utenti con ruolo `rds_superuser` possono eseguire queste azioni su altri utenti con ruolo `rds_superuser`.

- rdsadmin – Un ruolo creato per gestire molte delle attività di gestione che l'amministratore con privilegi `superuser` esegue su un database PostgreSQL autonomo. Questo ruolo viene utilizzato internamente da Aurora PostgreSQL per molte attività di gestione.

Visualizzazione dei ruoli e dei relativi privilegi

È possibile visualizzare i ruoli predefiniti e i relativi privilegi nell'istanza database RDS per PostgreSQL utilizzando comandi diversi a seconda della versione di PostgreSQL in uso. Per visualizzare tutti i ruoli predefiniti, è possibile connettersi a un'istanza database RDS per PostgreSQL ed eseguire i comandi riportati di seguito tramite `psql`.

Per **`psql`** versione 15 e precedenti

Connettiti a un'istanza database RDS per PostgreSQL e utilizza il comando \du in psql:

```
postgres=> \du
                                                    List of roles
    Role name    |                         Attributes                          |
                  Member of
-----------------+-----------------------------------------------------------------
+-----------------------------------------------------
 postgres        | Create role, Create DB                                      +|
 {rds_superuser}
                 | Password valid until infinity                                |
 rds_ad          | Cannot login                                                 | {}
 rds_iam         | Cannot login                                                 | {}
 rds_password    | Cannot login                                                 | {}
 rds_replication | Cannot login                                                 | {}
 rds_superuser   | Cannot login                                                 |
 {pg_monitor,pg_signal_backend,rds_password,rds_replication}
 rdsadmin        | Superuser, Create role, Create DB, Replication, Bypass RLS+| {}
                 | Password valid until infinity                                |
```

Per **`psql`** versione 16 e successive

```
postgres=> \drg+
                            List of role grants
   Role name   |          Member of          |       Options       | Grantor
---------------+-----------------------------+---------------------+----------
 postgres      | rds_superuser               | INHERIT, SET        | rdsadmin
 rds_superuser | pg_checkpoint               | ADMIN, INHERIT, SET | rdsadmin
 rds_superuser | pg_monitor                  | ADMIN, INHERIT, SET | rdsadmin
```

```
 rds_superuser | pg_signal_backend           | ADMIN, INHERIT, SET | rdsadmin
 rds_superuser | pg_use_reserved_connections | ADMIN, INHERIT, SET | rdsadmin
 rds_superuser | rds_password                | ADMIN, INHERIT, SET | rdsadmin
 rds_superuser | rds_replication             | ADMIN, INHERIT, SET | rdsadmin
```

Per verificare l'appartenenza ai ruoli senza dipendenza dalla versione, puoi utilizzare la seguente query SQL:

```
SELECT m.rolname AS "Role name", r.rolname AS "Member of"
FROM pg_catalog.pg_roles m
JOIN pg_catalog.pg_auth_members pam ON (pam.member = m.oid)
LEFT JOIN pg_catalog.pg_roles r ON (pam.roleid = r.oid)
LEFT JOIN pg_catalog.pg_roles g ON (pam.grantor = g.oid)
WHERE m.rolname !~ '^pg_'
ORDER BY 1, 2;
```

Nell'output, si vede che `rds_superuser` non è un ruolo utente del database (non può effettuare il login), ma ha i privilegi di molti altri ruoli. È inoltre possibile vedere che l'utente di database `postgres` è membro del ruolo `rds_superuser`. Come accennato in precedenza, `postgres` è il valore predefinito nella pagina Crea database della console Amazon RDS. Se si sceglie un altro nome, tale nome viene visualizzato nell'elenco dei ruoli.

> **Note**
>
> Aurora PostgreSQL versioni 15.2 e 14.7 hanno introdotto un funzionamento restrittivo del ruolo `rds_superuser`. A un utente Aurora PostgreSQL deve essere concesso il privilegio `CONNECT` per consentire la connessione al database corrispondente anche se tale utente è associato al ruolo `rds_superuser`. Prima delle versioni 14.7 e 15.2 di Aurora PostgreSQL, un utente era in grado di connettersi a qualsiasi database e tabella di sistema se disponeva del ruolo `rds_superuser`. Questo comportamento restrittivo è in linea con l'impegno di AWS e Amazon Aurora per il miglioramento continuo della sicurezza.
> Aggiorna la rispettiva logica nelle tue applicazioni se sono state interessate dal miglioramento precedentemente descritto.

Controllo dell'accesso utente al database PostgreSQL

I nuovi database in PostgreSQL vengono sempre creati con un set predefinito di privilegi nel schema `public` del database, che consente a tutti gli utenti e i ruoli del database di creare oggetti.

I privilegi predefiniti permettono agli utenti del database di connettersi al database e di creare tabelle temporanee durante la connessione.

Per controllare meglio l'accesso degli utenti alle istanze database create sul nodo primario del cluster Aurora PostgreSQL , si consiglia di revocare questi privilegi `public` predefiniti. Dopo averlo fatto, è consigliabile concedere privilegi specifici agli utenti del database su base più granulare, come mostrato nella procedura seguente.

Per impostare ruoli e privilegi per una nuova istanza database

Si supponga di aver configurando un database in un cluster di database Aurora PostgreSQL per poter essere usato da diversi ricercatori, che dovranno avere l'accesso in lettura-scrittura al database.

1. Utilizzare `psql` (o pgAdmin) per connettersi all'istanza database primaria sul cluster di database Aurora PostgreSQL:

   ```
   psql --host=your-cluster-instance-1.666666666666.aws-region.rds.amazonaws.com --
   port=5432 --username=postgres --password
   ```

   Specifica la password, quando richiesto. Il client `psql` si connette e visualizza il database di connessione amministrativa predefinito `postgres=>` come prompt.
2. Per impedire agli utenti del database di creare oggetti nello schema `public`, eseguire le seguenti operazioni:

   ```
   postgres=> REVOKE CREATE ON SCHEMA public FROM PUBLIC;
   REVOKE
   ```
3. Creare quindi una nuova istanza database:

   ```
   postgres=> CREATE DATABASE lab_db;
   CREATE DATABASE
   ```
4. Revocare tutti i privilegi dallo schema `PUBLIC` in questo nuovo database.

   ```
   postgres=> REVOKE ALL ON DATABASE lab_db FROM public;
   REVOKE
   ```
5. Creare un ruolo per gli utenti del database.

   ```
   postgres=> CREATE ROLE lab_tech;
   ```

```
CREATE ROLE
```

6. Concedere agli utenti del database con questo ruolo la possibilità di connettersi al database.

```
postgres=> GRANT CONNECT ON DATABASE lab_db TO lab_tech;
GRANT
```

7. Concedere a tutti gli utenti con il ruolo `lab_tech` tutti i privilegi per questo database.

```
postgres=> GRANT ALL PRIVILEGES ON DATABASE lab_db TO lab_tech;
GRANT
```

8. Creare utenti del database, come segue:

```
postgres=> CREATE ROLE lab_user1 LOGIN PASSWORD 'change_me';
CREATE ROLE
postgres=> CREATE ROLE lab_user2 LOGIN PASSWORD 'change_me';
CREATE ROLE
```

9. Concedere a questi due utenti i privilegi associati al ruolo lab_tech:

```
postgres=> GRANT lab_tech TO lab_user1;
GRANT ROLE
postgres=> GRANT lab_tech TO lab_user2;
GRANT ROLE
```

A questo punto, `lab_user1` e `lab_user2` possono connettersi al database `lab_db`. Questo esempio non segue le best practice per l'utilizzo aziendale, che potrebbero includere la creazione di più istanze database, schemi diversi e la concessione di autorizzazioni limitate. Per informazioni più complete e scenari aggiuntivi, consulta Gestione di utenti e ruoli PostgreSQL.

Per ulteriori informazioni sui privilegi in database PostgreSQL, consulta la sezione relativa al comando GRANT nella documentazione di PostgreSQL.

Delega e controllo della gestione delle password utente

Un amministratore di database (DBA) potrebbe voler delegare la gestione delle password utente. In alternativa, è possibile impedire agli utenti del database di modificare le password o di riconfigurare i vincoli delle password, ad esempio la durata della password. Per garantire che solo gli utenti del database scelti possano modificare le impostazioni della password, è possibile attivare la funzione di

gestione delle password con restrizioni. Quando si attiva questa funzione, solo gli utenti del database a cui è stato concesso il ruolo `rds_password` saranno in grado di gestire le password.

> **Note**
>
> Per utilizzare la gestione delle password limitate, il cluster di database Aurora PostgreSQL deve eseguire Amazon Aurora PostgreSQL 10.6 o superiore.

Per impostazione predefinita, questa funzione è impostata su `off`, come mostrato di seguito:

```
postgres=> SHOW rds.restrict_password_commands;
  rds.restrict_password_commands
--------------------------------
 off
(1 row)
```

Per attivare questa funzione, utilizzare un gruppo di parametri personalizzato e modificare l'impostazione per `rds.restrict_password_commands` su 1. Assicurarsi di riavviare l'istanza database principale Aurora PostgreSQL per implementare l'impostazione.

Con questa funzione attiva, i privilegi `rds_password` sono obbligatori per i seguenti comandi SQL:

```
CREATE ROLE myrole WITH PASSWORD 'mypassword';
CREATE ROLE myrole WITH PASSWORD 'mypassword' VALID UNTIL '2023-01-01';
ALTER ROLE myrole WITH PASSWORD 'mypassword' VALID UNTIL '2023-01-01';
ALTER ROLE myrole WITH PASSWORD 'mypassword';
ALTER ROLE myrole VALID UNTIL '2023-01-01';
ALTER ROLE myrole RENAME TO myrole2;
```

Anche la ridenominazione di un ruolo (`ALTER ROLE myrole RENAME TO newname`) è limitata se la password utilizza l'algoritmo di hashing MD5.

Con questa funzionalità attiva, se si tenta di eseguire uno di questi comandi SQL senza le autorizzazioni di ruolo `rds_password`, viene generato il seguente errore:

```
ERROR: must be a member of rds_password to alter passwords
```

Si consiglia di concedere i privilegi `rds_password` solo a ruoli utilizzati esclusivamente per la gestione delle password. Se si concedono i privilegi `rds_password` agli utenti del database sprovvisti dei privilegi `rds_superuser`, è necessario concedere loro anche l'attributo `CREATEROLE`.

Assicurarsi di verificare i requisiti della password come la scadenza e la complessità necessaria sul lato client. Se si utilizza la propria utilità lato client per le modifiche relative alla password, l'utilità deve essere membro di `rds_password` e avere i privilegi `CREATE ROLE`.

Utilizzo delle crittografia password SCRAM per PostgreSQL

Il meccanismo SCRAM (Salted Challenge Response Authentication Mechanism) è un'alternativa all'algoritmo predefinito MD5 (Message Digest) di PostgreSQL per la crittografia delle password. Il meccanismo di autenticazione SCRAM è considerato più sicuro di MD5. Per ulteriori informazioni su questi due diversi approcci di protezione delle password, consulta la sezione relativa alla autenticazione password nella documentazione di PostgreSQL.

Si consiglia di utilizzare SCRAM anziché MD5 come schema di crittografia password per il cluster database Aurora PostgreSQL. SCRAM è supportato in Aurora PostgreSQL versione 10 e in tutte le versioni principali e secondarie successive. È un meccanismo crittografico di richiesta/risposta che utilizza l'algoritmo scram-sha-256 per l'autenticazione e la crittografia delle password.

Potrebbe essere necessario aggiornare le librerie per le applicazioni client per supportare SCRAM. Ad esempio, le versioni JDBC precedenti alla 42.2.0 non supportano SCRAM. Per ulteriori informazioni, consulta PostgreSQL JDBC Driver nella documentazione di PostgreSQL JDBC Driver. Per un elenco di altri driver PostgreSQL e il supporto SCRAM, consulta Elenco dei driver nella documentazione di PostgreSQL.

Aurora PostgreSQL versione 14 e successive supportano scram-sha-256 per la crittografia password per impostazione predefinita per nuovi cluster database. Per queste versioni, il gruppo di parametri del cluster di database predefinito (`default.aurora-postgresql14`) ha il valore `password_encryption` è impostato su scram-sha-256. SCRAM non è supportato per Aurora Serverless v1.

Configurazione di cluster database Aurora PostgreSQL per richiedere SCRAM

Per Aurora PostgreSQL 14.3 e versioni successive, puoi richiedere che il cluster database Aurora PostgreSQL accetti solo password che utilizzano l'algoritmo scram-sha-256.

> **Important**
>
> Per i proxy RDS esistenti con database PostgreSQL, se si modifica l'autenticazione del database in modo da utilizzare solo `SCRAM`, il proxy diventa non disponibile per un massimo di 60 secondi. Per evitare il problema, procedi in uno dei seguenti modi:
>
> - Assicurati che il database consenta entrambe le autenticazioni `SCRAM` e `MD5`.
> - Per utilizzare solo l'autenticazione `SCRAM`, crea un nuovo proxy, esegui la migrazione del traffico dell'applicazione sul nuovo proxy, quindi elimina il proxy precedentemente associato al database.

Prima di apportare modifiche al sistema, assicurati di comprendere il processo completo, come segue:

- Ottieni informazioni su tutti i ruoli e la crittografia password per tutti gli utenti del database.
- Verifica le impostazioni dei parametri per il cluster database Aurora PostgreSQL per i parametri che controllano la crittografia password.
- Se il cluster database Aurora PostgreSQL utilizza un gruppo di parametri predefinito, devi creare un gruppo di parametri cluster database e applicarlo al cluster database Aurora PostgreSQL in modo da poter modificare i parametri quando necessario. Se il cluster database Aurora PostgreSQL utilizza un gruppo di parametri personalizzati, puoi modificare i parametri necessari in seguito nel processo, in base alle esigenze.
- Modifica il parametro `password_encryption` in `scram-sha-256`.
- Invia una notifica a tutti gli utenti del database per informarli che devono aggiornare le password. Esegui la stessa operazione per l'account `postgres`. Le nuove password sono crittografate e archiviate utilizzando l'algoritmo scram-sha-256.
- Verifica che tutte le password siano crittografate utilizzando come il tipo di crittografia.
- Se tutte le password utilizzano scram-sha-256, puoi modificare il `rds.accepted_password_auth_method` da `md5+scram` a `scram-sha-256`.

> **Warning**
>
> Dopo aver modificato `rds.accepted_password_auth_method` in scram-sha-256, gli eventuali utenti (ruoli) con password crittografate `md5` non potranno connettersi.

Preparazione alla richiesta di SCRAM per il cluster database Aurora PostgreSQL

Prima di apportare modifiche al cluster database Aurora PostgreSQL, controlla tutti gli account utente del database esistenti. Inoltre, controlla il tipo di crittografia utilizzato per le password. Puoi eseguire queste attività utilizzando l'estensione `rds_tools`. Per vedere quali versioni di PostgreSQL supportano `rds_tools`, consulta Versioni delle estensioni per Amazon RDS per PostgreSQL.

Per ottenere un elenco di utenti del database (ruoli) e metodi di crittografia password

1. Utilizza `psql` per connetterti all'istanza database principale del cluster database Aurora PostgreSQL , come mostrato di seguito.

```
psql --host=cluster-name-instance-1.111122223333.aws-region.rds.amazonaws.com --
port=5432 --username=postgres --password
```

2. Installa l'estensione `rds_tools`.

```
postgres=> CREATE EXTENSION rds_tools;
CREATE EXTENSION
```

3. Ottieni un elenco di ruoli e crittografia.

```
postgres=> SELECT * FROM
      rds_tools.role_password_encryption_type();
```

L'output visualizzato è simile al seguente.

```
       rolname        | encryption_type
----------------------+-----------------
 pg_monitor           |
 pg_read_all_settings |
 pg_read_all_stats    |
 pg_stat_scan_tables  |
 pg_signal_backend    |
 lab_tester           | md5
 user_465             | md5
 postgres             | md5
(8 rows)
```

Creazione di un gruppo di parametri del cluster DB personalizzato

> **Note**
>
> Se il cluster database Aurora PostgreSQL utilizza già un gruppo di parametri personalizzati, non è necessario crearne uno nuovo.

Per una panoramica dei gruppi di parametri per Aurora, consulta Creazione di un gruppo di parametri del cluster di database in Amazon Aurora.

Il tipo di crittografia password utilizzato per le password è impostato in un parametro, `password_encryption`. La crittografia consentita dal cluster database Aurora PostgreSQL è impostata in un altro parametro, `rds.accepted_password_auth_method`. La modifica di uno di questi rispetto ai valori predefiniti richiede la creazione di un gruppo di parametri del cluster database personalizzato e l'applicazione al cluster.

Puoi anche utilizzare Console di gestione AWS o l'API RDS per creare un gruppo di parametri del cluster database personalizzato. Per ulteriori informazioni, consulta Creazione di un gruppo di parametri del cluster di database in Amazon Aurora.

Ora puoi associare il gruppo di parametri personalizzati all'istanza database.

Creare un gruppo di parametri del cluster database personalizzato

1. Utilizza il comando CLI `create-db-cluster-parameter-group` per creare il gruppo di parametri personalizzati per il cluster. Il seguente esempio utilizza `aurora-postgresql13` come l'origine per questo gruppo di parametri personalizzati.

   Per Linux, macOS o Unix:

   ```
   aws rds create-db-cluster-parameter-group --db-cluster-parameter-group-name 'docs-lab-scram-passwords' \
     --db-parameter-group-family aurora-postgresql13  --description 'Custom DB cluster parameter group for SCRAM'
   ```

   Per Windows:

   ```
   aws rds create-db-cluster-parameter-group --db-cluster-parameter-group-name "docs-lab-scram-passwords" ^
   ```

```
  --db-parameter-group-family aurora-postgresql13  --description "Custom DB cluster
 parameter group for SCRAM"
```

Ora puoi associare il gruppo di parametri personalizzati al cluster.

2. Utilizza il comando CLI `modify-db-cluster` per applicare questo gruppo di parametri personalizzati al cluster database Aurora PostgreSQL.

   Per Linux, macOS o Unix:

```
aws rds modify-db-cluster --db-cluster-identifier 'your-instance-name' \
        --db-cluster-parameter-group-name "docs-lab-scram-passwords
```

   Per Windows:

```
aws rds modify-db-cluster --db-cluster-identifier "your-instance-name" ^
        --db-cluster-parameter-group-name "docs-lab-scram-passwords
```

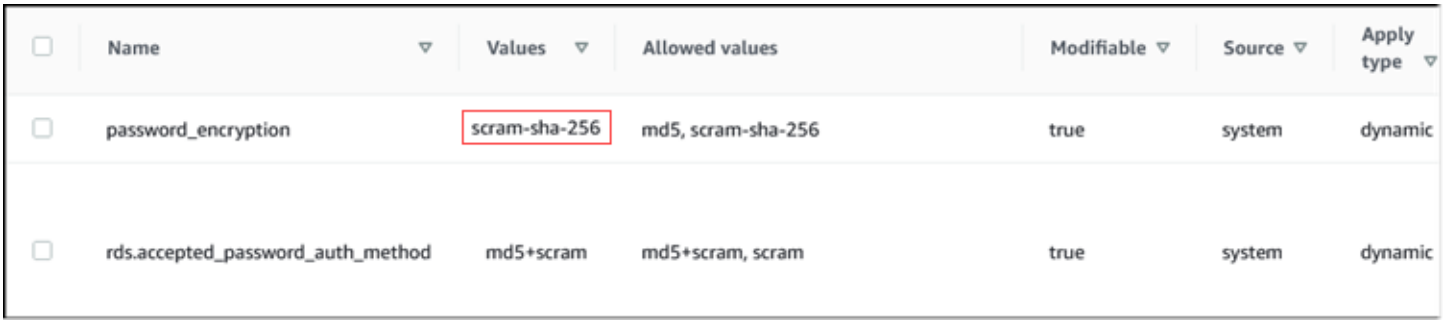
   Per ripetere la sincronizzazione del cluster database Aurora PostgreSQL con il gruppo di parametri del cluster DB personalizzati, è necessario riavviare l'istanza principale e tutte le altre istanze del cluster.

Configurazione della crittografia password per utilizzare SCRAM

Il meccanismo di crittografia password utilizzato da un cluster database Aurora PostgreSQL è impostato nel gruppo di parametri del cluster DB nel parametro `password_encryption`. I valori consentiti sono unset, `md5` o `scram-sha-256`. Il valore predefinito dipende dalla versione di Aurora PostgreSQL come segue:

- Aurora PostgreSQL 14:-l'impostazione predefinita è `scram-sha-256`
- Aurora PostgreSQL 13: l'impostazione predefinita è `md5`

Con un gruppo di parametri del cluster database personalizzato collegato al cluster database Aurora PostgreSQL, puoi modificare i valori per il parametro di crittografia password.



Name	Values	Allowed values	Modifiable	Source	Apply type
password_encryption	scram-sha-256	md5, scram-sha-256	true	system	dynamic
rds.accepted_password_auth_method	md5+scram	md5+scram, scram	true	system	dynamic

Per modificare l'impostazione di crittografia password in scram-sha-256

- Modifica il valore della crittografia password in scram-sha-256, come mostrato di seguito. La modifica può essere applicata immediatamente perché il parametro è dinamico, quindi non è necessario un riavvio per rendere effettiva la modifica.

  Per Linux, macOS o Unix:

```
aws rds modify-db-cluster-parameter-group --db-cluster-parameter-group-name \
  'docs-lab-scram-passwords' --parameters
 'ParameterName=password_encryption,ParameterValue=scram-
sha-256,ApplyMethod=immediate'
```

  Per Windows:

```
aws rds modify-db-parameter-group --db-parameter-group-name ^
  "docs-lab-scram-passwords" --parameters
 "ParameterName=password_encryption,ParameterValue=scram-
sha-256,ApplyMethod=immediate"
```

Migrazione delle password per i ruoli utente in SCRAM

Puoi migrare le password per i ruoli utente a SCRAM come descritto di seguito.

Eseguire la migrazione delle password utente (ruolo) del database da MD5 a SCRAM

1. Accedi come utente amministratore (nome utente predefinito, `postgres`) come mostrato di seguito.

```
psql --host=cluster-name-instance-1.111122223333.aws-region.rds.amazonaws.com --
port=5432 --username=postgres --password
```

2. Controlla l'impostazione del parametro `password_encryption` sull'istanza database RDS per PostgreSQL utilizzando il comando seguente.

```
postgres=> SHOW password_encryption;
 password_encryption
---------------------
 md5
 (1 row)
```

3. Modifica il valore di questo parametro in scram-sha-256. Per ulteriori informazioni, consulta Configurazione della crittografia password per utilizzare SCRAM.
4. Controlla nuovamente il valore per essere certo che ora sia impostato su `scram-sha-256`, come descritto di seguito.

```
postgres=> SHOW password_encryption;
 password_encryption
---------------------
 scram-sha-256
 (1 row)
```

5. Invia una notifica a tutti gli utenti del database con la richiesta di modificare le password. Assicurati di modificare anche la password per l'account `postgres` (l'utente del database con privilegi `rds_superuser`).

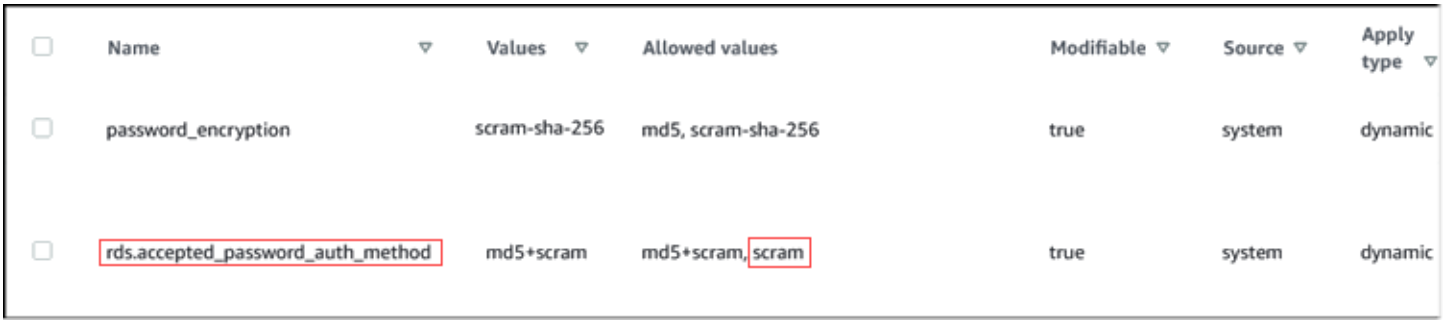
```
labdb=> ALTER ROLE postgres WITH LOGIN PASSWORD 'change_me';
ALTER ROLE
```

6. Ripeti il processo per tutti i database sul cluster database Aurora PostgreSQL.

Modifica del parametro per richiedere SCRAM

Questo è il passaggio finale del processo. Dopo aver apportato la modifica nella procedura seguente, gli eventuali account utente (ruoli) che ancora utilizzano la crittografia `md5` per le password non possono accedere al cluster database Aurora PostgreSQL.

Il `rds.accepted_password_auth_method` specifica il metodo di crittografia accettato dal cluster database Aurora PostgreSQL per una password utente durante il processo di accesso. Il valore predefinito è `md5+scram`, il che significa che entrambi i metodi sono accettati. Nell'immagine seguente, è disponibile l'impostazione predefinita per questo parametro.



Name	Values	Allowed values	Modifiable	Source	Apply type
password_encryption	scram-sha-256	md5, scram-sha-256	true	system	dynamic
rds.accepted_password_auth_method	md5+scram	md5+scram, scram	true	system	dynamic

I valori consentiti per questo parametro sono `md5+scram` o `scram`. La modifica del valore di questo parametro in `scram` lo rende un requisito.

Modificare il valore del parametro per richiedere l'autenticazione SCRAM per le password

1. Verifica che tutte le password degli utenti del database per tutti i database sul cluster database Aurora PostgreSQL utilizzino `scram-sha-256` per la crittografia password. A questo proposito, esegui la query su `rds_tools` per il ruolo (utente) e il tipo di crittografia, come segue.

```
postgres=> SELECT * FROM rds_tools.role_password_encryption_type();
  rolname        | encryption_type
  ---------------------+-----------------
  pg_monitor           |
  pg_read_all_settings |
  pg_read_all_stats    |
  pg_stat_scan_tables  |
  pg_signal_backend    |
  lab_tester           | scram-sha-256
  user_465             | scram-sha-256
  postgres             | scram-sha-256
  ( rows)
```

2. Ripeti la query su tutte le istanze database nel cluster database Aurora PostgreSQL.

   Se tutte le password utilizzano scram-sha-256, puoi procedere.

3. Modifica il valore dell'autenticazione password accettata in scram-sha-256, come riportato di seguito.

   Per Linux, macOS o Unix:

```
aws rds modify-db-cluster-parameter-group --db-cluster-parameter-group-name 'docs-
lab-scram-passwords' \
```

```
  --parameters
 'ParameterName=rds.accepted_password_auth_method,ParameterValue=scram,ApplyMethod=immediat
```

Per Windows:

```
aws rds modify-db-cluster-parameter-group --db-cluster-parameter-group-name "docs-
lab-scram-passwords" ^
  --parameters
 "ParameterName=rds.accepted_password_auth_method,ParameterValue=scram,ApplyMethod=immediat
```

Sicurezza dei dati Aurora PostgreSQL con SSL/TLS

Amazon RDS supporta la crittografia SSecure Socket Layer (SSL) e Transport Layer Security (TLS) per i cluster DB Aurora PostgreSQL. Utilizzo dell'SSL/TLS, you can encrypt a connection between your applications and your Aurora PostgreSQL DB clusters. You can also force all connections to your Aurora PostgreSQL DB cluster to use SSL/TLS. Amazon Aurora PostgreSQL supporta Transport Layer Security (TLS) versioni 1.1 e 1.2. Si consiglia di utilizzare TLS 1.2 per connessioni crittografate. Abbiamo aggiunto il supporto per la versione TLSv1 2.3 delle seguenti versioni di Aurora PostgreSQL:

- 15.3 e tutte le versioni successive
- 14.8 o versioni successive alla 14
- 13.11 o versioni successive alla 13
- 12.15 e versioni successive alla 12
- 11.20 e versioni successive alla 11

Per informazioni generali sul SSL/TLS supporto e sui database PostgreSQL, consulta il supporto SSL nella documentazione di PostgreSQL. Per informazioni sull'utilizzo di una SSL/TLS connessione tramite JDBC, vedere Configurazione del client nella documentazione di PostgreSQL.

Argomenti

- Richiesta di una connessione SSL/TLS a un cluster DB Aurora PostgreSQL
- Determinazione dello stato di connessione SSL/TLS
- Configurazione di suite di cifratura per connessioni ai cluster di database Aurora PostgreSQL

Il supporto SSL/TLS è disponibile in tutte le regioni per AWS Aurora PostgreSQL. Amazon RDS crea un SSL/TLS certificato per il cluster DB Aurora PostgreSQL al momento della creazione del cluster DB. Se abiliti la verifica del SSL/TLS certificato, il SSL/TLS certificato include l'endpoint del cluster DB come nome comune (CN) del SSL/TLS certificato per proteggerlo dagli attacchi di spoofing.

Per effettuare la connessione a un cluster DB Aurora PostgreSQL tramite SSL/TLS

1. Scaricare il certificato.

   Per ulteriori informazioni sul download dei certificati, consultare Utilizzo SSL/TLS per crittografare una connessione a un'.
2. Importare il certificato nel proprio sistema operativo.
3. Effettua la connessione al tuo cluster DB Aurora PostgreSQL su SSL/TLS.

   Quando ti connetti tramite SSL/TLS, il tuo client può scegliere di verificare o meno la catena di certificati. Se i parametri di connessione specificano `sslmode=verify-ca` o `sslmode=verify-full`, il client richiede che i certificati CA RDS siano nell'archivio attendibilità o facciano riferimento all'URL della connessione. Questo requisito è verificare la catena di certificati che firma il certificato del database.

   Quando un client, come psql o JDBC, è configurato con il SSL/TLS supporto, per impostazione predefinita il client tenta innanzitutto di connettersi al database. SSL/TLS Se il client non riesce a connettersi con. SSL/TLS, it reverts to connecting without SSL/TLS Per impostazione predefinita, l'opzione `sslmode` per i client JDBC e basati su libpq è impostata su `prefer`.

   Utilizzare il parametro `sslrootcert` come riferimento per il certificato, ad esempio `sslrootcert=rds-ssl-ca-cert.pem`.

Di seguito è riportato un esempio di utilizzo di psql per la connessione a un cluster DB Aurora PostgreSQL.

```
$ psql -h testpg.cdhmuqifdpib.us-east-1.rds.amazonaws.com -p 5432 \
    "dbname=testpg user=testuser sslrootcert=rds-ca-2015-root.pem sslmode=verify-full"
```

Richiesta di una connessione SSL/TLS a un cluster DB Aurora PostgreSQL

Per richiedere SSL/TLS connessioni al cluster Aurora PostgreSQL DB, usa il parametro. `rds.force_ssl`

- Per richiedere SSL/TLS connessioni, imposta il valore del `rds.force_ssl` parametro su 1 (attivo).
- Per disattivare SSL/TLS le connessioni richieste, impostate il valore del `rds.force_ssl` parametro su 0 (disattivato).

Il valore predefinito di questo parametro dipende dalla versione di Aurora PostgreSQL:

- Per Aurora PostgreSQL versioni 17 e successive: il valore predefinito è 1 (on).
- Per Aurora PostgreSQL versioni 16 e precedenti: il valore predefinito è 0 (off).

> **Note**
>
> Quando si esegue un aggiornamento di versione principale da Aurora PostgreSQL versione 16 o precedente alla versione 17 o successiva, il valore predefinito del parametro cambia da 0 (off) a 1 (on). Questa modifica può causare errori di connettività per le applicazioni non configurate per SSL. È possibile ripristinare il comportamento predefinito precedente impostando questo parametro su 0 (disattivato).

Per ulteriori informazioni sulla gestione dei parametri, consultare Gruppi di parametri per Amazon Aurora.

L'aggiornamento del `rds.force_ssl` parametro imposta anche il parametro `ssl` PostgreSQL su 1 (on) e modifica il file `pg_hba.conf` del cluster DB per supportare la nuova configurazione. SSL/TLS

Quando il `rds.force_ssl` parametro è impostato su 1 per un cluster DB, quando ci si connette viene visualizzato un output simile al seguente, a indicare che SSL/TLS ora è necessario:

```
$ psql postgres -h SOMEHOST.amazonaws.com -p 8192 -U someuser
psql (9.3.12, server 9.4.4)
WARNING: psql major version 9.3, server major version 9.4.
Some psql features might not work.
SSL connection (cipher: DHE-RSA-AES256-SHA, bits: 256)
Type "help" for help.

postgres=>
```

Determinazione dello stato di connessione SSL/TLS

Lo stato crittografato della tua connessione è indicato nel banner di accesso quando ti connetti al cluster DB:

```
Password for user master:
psql (9.3.12)
SSL connection (cipher: DHE-RSA-AES256-SHA, bits: 256)
Type "help" for help.

postgres=>
```

È inoltre possibile caricare l'`sslinfo`estensione e quindi richiamare la `ssl_is_used()` funzione per determinare se SSL/TLS viene utilizzata. La funzione restituisce `t` se la connessione utilizza la crittografia SSL/TLS, altrimenti restituisce `f`.

```

postgres=> create extension sslinfo;
CREATE EXTENSION

postgres=> select ssl_is_used();
 ssl_is_used
---------
t
(1 row)
```

Puoi utilizzare il comando `select ssl_cipher()` per determinare la crittografia SSL/TLS:

```

postgres=> select ssl_cipher();
ssl_cipher
--------------------
DHE-RSA-AES256-SHA
(1 row)

```

Se abiliti `set rds.force_ssl` e riavvi il tuo cluster DB, le connessioni non SSL vengono rifiutate con il seguente messaggio:

```

$ export PGSSLMODE=disable
```

```
$ psql postgres -h SOMEHOST.amazonaws.com -p 8192 -U someuser
psql: FATAL: no pg_hba.conf entry for host "host.ip", user "someuser", database
 "postgres", SSL off
$
```

Per informazioni sull'opzione `sslmode`, consulta l'argomento relativo alle funzioni di controllo della connessione del database nella documentazione di PostgreSQL.

Configurazione di suite di cifratura per connessioni ai cluster di database Aurora PostgreSQL

Utilizzando suite di cifratura configurabili, è possibile avere maggiore controllo sulla sicurezza delle connessioni al database. È possibile specificare un elenco di suite di crittografia che si desidera consentire per proteggere le SSL/TLS connessioni dei client al database. Con le suite di cifratura, è possibile controllare la crittografia di connessione accettata dal server di database. Ciò aiuta a prevenire l'uso di crittografie obsolete o non sicure.

Le suite di cifratura configurabili sono supportate nelle versioni 11.8 e successive di Aurora PostgreSQL.

Per specificare l'elenco di cifrature consentite per la crittografia delle connessioni, modifica il parametro del cluster `ssl_ciphers`. Imposta il `ssl_ciphers` parametro su una stringa di valori di cifratura separati da virgole in un gruppo di parametri del cluster utilizzando l'API Console di gestione AWS, the o RDS. AWS CLI Per impostare i parametri del cluster, consulta Modifica dei parametri in un gruppo di parametri del cluster di database in Amazon Aurora.

La tabella seguente mostra le cifrature supportate per le versioni valide del motore Aurora PostgreSQL.

Versioni del motore Aurora PostgreSQL	Cifrature supportate	TLS 1.1	TLS 1.2	TLS 1.3
9.6, 10.20 e versioni precedenti, 11.15 e versioni precedenti, 12.10 e versioni precedenti, 13.6 e versioni precedenti	DHE-RSA- -SHA AES128	Sì	No	No
	DHE-RSA- AES128 -SHA256	No	Sì	No
	DHE-RSA- -GCM- AES128 SHA256	No	Sì	No
		No	Sì	No

Versioni del motore Aurora PostgreSQL	Cifrature supportate	TLS 1.1	TLS 1.2	TLS 1.3
	DHE-RSA- AES256 -SHA	No	Sì	No
	DHE-RSA- AES256 -SHA256	No	Sì	No
	DHE-RSA- -GCM- AES256 SHA384	Sì	Sì	No
	ECDHE-ECDSA- AES256 -SHA	No	Sì	No
	ECDHE-ECDSA- -GCM- AES256 SHA384	Sì	Sì	No
	ECDH-RSA- - AES256 SHA384	No	Sì	No
	ECDHE-RSA- AES128 -SHA	No	Sì	No
	ECDHE-RSA- - AES128 SHA256	No	Sì	No
	ECDHE-RSA- -GCM AES128 - SHA256	Sì	Sì	No
	ECDHE-RSA- AES256 -SHA	No	Sì	No
	ECDHE-RSA- AES256 -GCM- SHA384	No	Sì	No

Versioni del motore Aurora PostgreSQL	Cifrature supportate	TLS 1.1	TLS 1.2	TLS 1.3
10.21, 11.16, 12.11, 13.7, 14.3 e 14.4	ECDHE-RSA-AES128 -SHATLS_ECDHE_RSA_CON_AES_128_CBC_SHA	Sì	Sì	No
	TLS_ECDHE_RSA_CON_AES_128_GCM_SHA256	No	Sì	No
	TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA	Sì	Sì	No
	TLS_ECDHE_RSA_CON_AES_256_GCM_SHA384	No	Sì	No
	TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA	Sì	Sì	No
	TLS_ECDHE_RSA_CON_AES_128_GCM_SHA256	No	Sì	No
	TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA	Sì	Sì	No
	TLS_ECDHE_RSA_CON_	No	Sì	No

Versioni del motore Aurora PostgreSQL	Cifrature supportate	TLS 1.1	TLS 1.2	TLS 1.3
	AES_256_GCM_ SHA384 TLS_RSA_C ON_AES_256_GCM_ SHA384 TLS_RSA_W ITH_AES_2 56_CBC_SHA TLS_RSA_C ON_AES_128_GCM_ SHA256 TLS_RSA_W ITH_AES_1 28_CBC_SHA TLS_ECDHE _RSA_CON_ 0_ CHACHA2 05_ POLY13 SHA256			

Versioni del motore Aurora PostgreSQL	Cifrature supportate	TLS 1.1	TLS 1.2	TLS 1.3
10.22, 11.17, 12.12, 13.8, 14.5 e 15.2	TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA	Sì	Sì	No
		No	Sì	No
	TLS_ECDHE_RSA_CON_AES_128_CBC_SHA256	No	Sì	No
		Sì	Sì	No
		No	Sì	No
	TLS_ECDHE_RSA_CON_AES_128_GCM_SHA256	Sì	Sì	No
		No	Sì	No
		No	Sì	No
	TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA	Sì	Sì	No
		No	Sì	No
	TLS_ECDHE_RSA_CON_AES_256_GCM_SHA384	No	Sì	No
		Sì	Sì	No
		No	Sì	No
	TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA	Sì	Sì	No
		Sì	Sì	No
	TLS_ECDHE_RSA_CON_AES_128_CBC_SHA256	No	Sì	No
	TLS_ECDHE_RSA_CON_AES_128_GCM_SHA256			

Versioni del motore Aurora PostgreSQL	Cifrature supportate	TLS 1.1	TLS 1.2	TLS 1.3
	TLS_ECDHE _RSA_WITH _AES_256_CBC_SHA TLS_ECDHE _RSA_CON_ AES_256_GCM_ SHA384 TLS_RSA_C ON_AES_256_GCM_ SHA384 TLS_RSA_W ITH_AES_2 56_CBC_SHA TLS_RSA_C ON_AES_128_GCM_ SHA256 TLS_RSA_C ON_AES_128_CBC_ SHA256 TLS_RSA_W ITH_AES_1 28_CBC_SHA TLS_ECDHE _RSA_CON_ 0_ CHACHA2 05_ POLY13 SHA256			

Versioni del motore Aurora PostgreSQL	Cifrature supportate	TLS 1.1	TLS 1.2	TLS 1.3
11.20, 12.15, 13.11, 14.8, 15.3, 16.1 e versioni successive	TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA	Sì	Sì	No
		No	Sì	No
	TLS_ECDHE_RSA_CON_AES_128_CBC_SHA256	No	Sì	No
		Sì	Sì	No
		No	Sì	No
	TLS_ECDHE_RSA_CON_AES_128_GCM_SHA256	Sì	Sì	No
		No	Sì	No
		No	Sì	No
	TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA	Sì	Sì	No
		No	Sì	No
	TLS_ECDHE_RSA_CON_AES_256_GCM_SHA384	No	Sì	No
		Sì	Sì	No
		No	Sì	No
	TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA	Sì	Sì	No
		Sì	Sì	No
	TLS_ECDHE_RSA_CON_AES_128_CBC_SHA256	No	Sì	No
		No	No	Sì
		No	No	Sì
	TLS_ECDHE_RSA_CON_AES_128_GCM_SHA256			

Versioni del motore Aurora PostgreSQL	Cifrature supportate	TLS 1.1	TLS 1.2	TLS 1.3
	TLS_ECDHE _RSA_WITH _AES_256_CBC_SHA TLS_ECDHE _RSA_CON_ AES_256_GCM_ SHA384 TLS_RSA_C ON_AES_256_GCM_ SHA384 TLS_RSA_W ITH_AES_2 56_CBC_SHA TLS_RSA_C ON_AES_128_GCM_ SHA256 TLS_RSA_C ON_AES_128_CBC_ SHA256 TLS_RSA_W ITH_AES_1 28_CBC_SHA TLS_ECDHE _RSA_CON_ 0_ CHACHA2 05_ POLY13 SHA256 TLS_AES_128_GCM_ SHA256			

Versioni del motore Aurora PostgreSQL	Cifrature supportate	TLS 1.1	TLS 1.2	TLS 1.3
	TLS_AES_256_GCM_SHA384			

È inoltre possibile utilizzare il comando CLI describe-engine-default-cluster-parameters per determinare quali suite di crittografia sono attualmente supportate per una famiglia di gruppi di parametri specifica. L'esempio seguente mostra come ottenere i valori consentiti per il parametro del cluster `ssl_cipher` per Aurora PostgreSQL 11.

```
aws rds describe-engine-default-cluster-parameters --db-parameter-group-family aurora-
postgresql11

    ...some output truncated...
 {
  "ParameterName": "ssl_ciphers",
  "Description": "Sets the list of allowed TLS ciphers to be used on secure
 connections.",
  "Source": "engine-default",
  "ApplyType": "dynamic",
  "DataType": "list",
  "AllowedValues": "DHE-RSA-AES128-SHA,DHE-RSA-AES128-SHA256,DHE-RSA-AES128-GCM-
SHA256,DHE-RSA-AES256-SHA,DHE-RSA-AES256-SHA256,DHE-RSA-AES256-GCM-SHA384,
  ECDHE-RSA-AES128-SHA,ECDHE-RSA-AES128-SHA256,ECDHE-RSA-AES128-GCM-
SHA256,ECDHE-RSA-AES256-SHA,ECDHE-RSA-AES256-SHA384,ECDHE-RSA-AES256-GCM-
SHA384,TLS_RSA_WITH_AES_256_GCM_SHA384,

 TLS_RSA_WITH_AES_256_CBC_SHA,TLS_RSA_WITH_AES_128_GCM_SHA256,TLS_RSA_WITH_AES_128_CBC_SHA256,T

 TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384,TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA,TLS_ECDHE_RSA_WITH_AE
  "IsModifiable": true,
  "MinimumEngineVersion": "11.8",
  "SupportedEngineModes": [
   "provisioned"
  ]
 },
    ...some output truncated...
```

Il parametro `ssl_ciphers` è impostato per tutte le suite di cifrature consentite. Per ulteriori informazioni sulla crittografia, consulta la variabile ssl_ciphers nella documentazione di PostgreSQL.

Utilizzo del mascheramento dinamico con Aurora PostgreSQL

Il mascheramento dinamico dei dati è una funzionalità di sicurezza che protegge i dati sensibili nei database Aurora PostgreSQL controllando il modo in cui i dati vengono visualizzati dagli utenti al momento della query. Aurora lo implementa tramite l'estensione. `pg_columnmask` `pg_columnmask`offre una protezione dei dati a livello di colonna che integra la sicurezza nativa a livello di riga di PostgreSQL e i meccanismi di controllo granulare degli accessi.

Con`pg_columnmask`, crei policy di mascheramento che determinano la visibilità dei dati in base ai ruoli degli utenti. Quando gli utenti interrogano tabelle con policy di mascheramento, Aurora PostgreSQL applica la funzione di mascheramento appropriata al momento della query in base al ruolo e al peso delle policy dell'utente. I dati sottostanti rimangono invariati nell'archiviazione.

`pg_columnmask`supporta le seguenti funzionalità:

- Funzioni di mascheramento integrate e personalizzate: utilizza funzioni predefinite per modelli comuni come il mascheramento di e-mail e testo oppure crea funzioni personalizzate per proteggere i dati sensibili (PII) tramite politiche di mascheramento basate su SQL.
- Strategie di mascheramento multiple: nascondi completamente le informazioni, sostituisci i valori parziali con caratteri jolly o definisci approcci di mascheramento personalizzati.
- Assegnazione di priorità alle politiche: definisci più politiche per una singola colonna. Utilizza i pesi per determinare quale politica di mascheramento deve essere utilizzata quando più politiche si applicano a una colonna. Aurora PostgreSQL applica politiche basate sul peso e sull'appartenenza al ruolo dell'utente.

`pg_columnmask`è disponibile su Aurora PostgreSQL versione 16.10 e successive e versione 17.6 e successive. È disponibile senza costi aggiuntivi.

Guida introduttiva al mascheramento dinamico

Per mascherare dinamicamente i dati, installate l'`pg_columnmask`estensione nel database e create politiche di mascheramento per le tabelle. Il processo di configurazione prevede la verifica dei prerequisiti, l'installazione dell'estensione, la configurazione dei ruoli, la creazione di policy e il test di convalida.

Installazione e configurazione delle estensioni

Connettiti al tuo cluster Aurora PostgreSQL utilizzando RDS Console Query Editor o un client PostgreSQL come psql con credenziali rds_superuser (utente principale).

Esegui `pg_columnmask` il comando di creazione dell'estensione per abilitare la funzionalità:

```
CREATE EXTENSION pg_columnmask;
```

Questo comando installa l'`pg_columnmask`estensione, crea le tabelle di catalogo necessarie e registra le funzioni di mascheramento integrate. L'installazione dell'estensione è specifica del database, il che significa che è necessario installarla separatamente in ogni database in cui è richiesta la funzionalità.

> **Note**
>
> Le connessioni effettuate prima dell'installazione di questa estensione mostreranno comunque i dati non mascherati. Chiudi e riconnettiti per risolvere il problema.

Verifica l'installazione dell'estensione controllando le funzioni di mascheramento disponibili:

```
SELECT proname FROM pg_proc
    WHERE pronamespace = 'pgcolumnmask'::regnamespace AND proname LIKE 'mask_%';
    proname
--------Output --------
 mask_email
 mask_text
 mask_timestamp
(3 rows)
```

Procedure per la gestione delle politiche di mascheramento dei dati

È possibile gestire le politiche di mascheramento utilizzando le procedure fornite dall'`pg_columnmask`estensione. Per creare, modificare o eliminare le politiche di mascheramento, è necessario disporre di uno dei seguenti privilegi:

- Proprietario della tabella su cui si sta creando la `pg_columnmask` politica.
- Membro di`rds_superuser`.
- Membro del ruolo di `pg_columnmask` policy manager impostato dal `pgcolumnmask.policy_admin_rolname` parametro.

Il comando seguente crea una tabella che viene utilizzata nelle sezioni successive:

```
CREATE TABLE public.customers (
    id SERIAL PRIMARY KEY,
    name TEXT,
    phone TEXT,
    address TEXT,
    email TEXT
);
```

CREATE_MASKING_POLICY

La procedura seguente crea una nuova politica di mascheramento per una tabella utente:

Sintassi

```
create_masking_policy(
    policy_name,
    table_name,
    masking_expressions,
    roles,
    weight)
```

Arguments (Argomenti)

Parametro	DataType	Description
`policy_name`	NAME	Nome della policy di mascheramento. Deve essere unico per tabella.
`table_name`	REGCLASS	Il qualified/unqualified nome o l'ID della tabella a cui applicare la politica di mascheramento.
`masking_expressions`	JSONB	Oggetto JSON contenent e il nome della colonna e le coppie di funzioni di mascheramento. Ogni chiave è un nome di colonna e il

Parametro	DataType	Description
		suo valore è l'espressione di mascheramento da applicare su quella colonna.
`roles`	NOME []	I ruoli a cui si applica questa politica di mascheramento. L'impostazione predefinita è PUBLIC.
`weight`	INT	Peso della politica di mascheramento. Quando sono applicabili più criteri a una determinata query dell'utente, il criterio con il peso più elevato (numero intero più alto) verrà applicato a ciascuna colonna mascherata. Il valore predefinito è 0. Non esistono due politiche di mascheramento sulla tabella che possono avere lo stesso peso.

Tipo restituito

Nessuno

Example di creare una politica di mascheramento che maschera la colonna di posta elettronica per il ruolo: **`test_user`**

```
CALL pgcolumnmask.create_masking_policy(
    'customer_mask',
    'public.customers',
    JSON_OBJECT('{
        "email", "pgcolumnmask.mask_email(email)"
```

```
    }')::JSONB,
    ARRAY['test_user'],
    100
);
```

ALTER_MASKING_POLICY

Questa procedura modifica una politica di mascheramento esistente. `ALTER_MASKING_POLICY`può modificare le espressioni di mascheramento delle policy, l'insieme di ruoli a cui si applica la policy e il peso della policy di mascheramento. Quando uno di questi parametri viene omesso, la parte corrispondente della politica rimane invariata.

Sintassi

```
alter_masking_policy(
    policy_name,
    table_name,
    masking_expressions,
    roles,
    weight)
```

Arguments (Argomenti)

Parametro	DataType	Description
`policy_name`	NAME	Nome esistente della politica di mascheramento.
`table_name`	REGCLASS	Il qualified/unqualified nome id della tabella contenente la politica di mascheramento.
`masking_expressions`	JSONB	Nuovo oggetto JSON contenente il nome della colonna e le coppie di funzioni di mascheramento o NULL in caso contrario.
`roles`	NOME []	L'elenco dei nuovi ruoli a cui si applica questa politica di

Parametro	DataType	Description
		mascheramento o NULL in caso contrario.
`weight`	INT	Nuovo peso per la politica di mascheramento o NULL in caso contrario.

Tipo restituito

Nessuno

Example di aggiungere il ruolo di analista a una politica di mascheramento esistente senza modificare altri attributi della politica.

```
CALL pgcolumnmask.alter_masking_policy(
    'customer_mask',
    'public.customers',
    NULL,
    ARRAY['test_user', 'analyst'],
    NULL
);

-- Alter the weight of the policy without altering other details
CALL pgcolumnmask.alter_masking_policy(
    'customer_mask',
    'customers',
    NULL,
    NULL,
    4
);
```

DROP_MASKING_POLICY

Questa procedura rimuove una politica di mascheramento esistente.

Sintassi

```
drop_masking_policy(
        policy_name,
```

```
        table_name)
```

Arguments (Argomenti)

Parametro	DataType	Description
`policy_name`	NAME	Nome esistente della politica di mascheramento.
`table_name`	REGCLASS	Il qualified/unqualified nome id della tabella contenente la politica di mascheramento.

Tipo restituito

Nessuno

Example di eliminare la politica di mascheramento customer_mask

```
-- Drop a masking policy
    CALL pgcolumnmask.drop_masking_policy(
        'customer_mask',
        'public.customers',
    );
```

Identificatori di escape nella procedura DDL della politica di mascheramento

Quando si creano policy di mascheramento dei dati con identificatori tra virgolette, è necessario un corretto escape per garantire i riferimenti agli oggetti e l'applicazione delle policy corretti. Per utilizzare gli identificatori tra virgolette all'interno delle procedure di gestione delle policy di mascheramento: `pg_columnmask`

- Nome della politica: deve essere racchiuso tra virgolette doppie.
- Nome tabella: sia il nome dello schema che il nome della tabella devono essere racchiusi tra virgolette doppie singolarmente, se necessario.
- Espressioni di mascheramento: i nomi delle colonne e delle funzioni nelle espressioni di mascheramento devono essere racchiusi tra virgolette doppie e le virgolette stesse devono essere eliminate utilizzando una barra rovesciata.

- Ruoli: la matrice dei nomi dei ruoli viene citata automaticamente. Il nome del ruolo deve corrispondere esattamente al nome indicato, `pg_roles` inclusa la distinzione tra maiuscole e minuscole.

Example della sintassi in escape e tra virgolette

Questo esempio mostra la sintassi corretta di escape e virgolette quando si creano policy di mascheramento per tabelle, colonne, funzioni e ruoli che utilizzano nomi composti da maiuscole e minuscole o richiedono identificatori tra virgolette in Aurora PostgreSQL.

```
-- Create a table and columns with mixed case name
CREATE TABLE public."Employees" (
    "Name" TEXT,
    "Email" TEXT,
    ssn VARCHAR(20)
);

-- Create a role with mixed case name
CREATE ROLE "Masked_user";

-- Create a function with mixed case name
CREATE OR REPLACE FUNCTION public."MaskEmail"(text)
    RETURNS character varying
    LANGUAGE plpgsql
    IMMUTABLE PARALLEL SAFE
    AS $$ BEGIN
        RETURN 'XXXXXXXX'::text;
    END $$;

-- Now use these objects with mixed case names in
-- masking policy management procedures
CALL pgcolumnmask.create_masking_policy(
    '"Policy1"',  -- policy name should be surrounded with double quotes for quoting
    'public."Employees"', -- table and schema name should be individually
                          -- surrounded with double quotes for quoting
    JSON_OBJECT('{
        "\"Email\"", "\"MaskEmail\"(\"Email\")"
    }')::JSONB, -- masking expression should have double quotes around function names
                -- and columns names etc when needed. Also the double quotes itself
                -- should be escaped using \ (backslash) since this is a JSON string
    ARRAY['Masked_user'], -- Rolename do not need quoting
                          -- (this behaviour may change in future release)
```

```
    100
);

SELECT * FROM pgcolumnmask.pg_columnmask_policies
    WHERE tablename = 'Employees';
-[ RECORD 1 ]-----+-------------------------------------
schemaname        | public
tablename         | Employees
policyname        | Policy1
roles             | {Masked_user}
masked_columns    | {Email}
masking_functions | {"(\"MaskEmail\"(\"Email\")):text"}
weight            | 100
```

Viste amministrative

È possibile rivedere tutta la `pg_columnmask` politica utilizzando la visualizzazione `pgcolumnmask.pg_columnmask_policies` amministrativa accessibile al pubblico. Le seguenti informazioni sono disponibili utilizzando questa visualizzazione. La vista restituisce solo le politiche di mascheramento di proprietà dell'utente corrente.

Nome della colonna	Tipo di dati	Description
schemaname	NAME	Schema della relazione a cui è allegata la politica
tablename	NAME	Nome della relazione a cui è allegata la politica
nome della politica	NAME	Nome della politica di mascheramento, tutte le politiche di mascheramento hanno nomi univoci
roles	TESTO []	Ruolo a cui si applica la politica.
masked_columns	TESTO []	Colonne mascherate
funzioni_mascheramento	TESTO []	Funzioni di mascheramento

Nome della colonna	Tipo di dati	Description
peso	INT	Peso della politica allegata

Funzioni di mascheramento dei dati predefinite

`pg_columnmask`l'estensione fornisce funzioni di utilità integrate scritte in linguaggio C (per un'esecuzione più rapida) che possono essere utilizzate come espressione di mascheramento per le `pg_columnmask` politiche.

mask_text

Una funzione per mascherare i dati di testo con opzioni di visibilità configurabili.

Arguments (Argomenti)

Parametro	DataType	Description
`input`	TEXT	La stringa di testo originale da mascherare
`mask_char`	CARATTERE (1)	Carattere usato per mascherare (predefinito: 'X')
`visible_prefix`	INT	Numero di caratteri all'inizio del testo di input che rimarranno non mascherati (impostazione predefinita: 0)
`visible_suffix`	INT	Numero di caratteri alla fine del testo di input che rimarranno non mascherati (impostazione predefinita: 0)
`use_hash_mask`	BOOLEAN	Se TRUE, utilizza un mascheramento basato su hash anziché mask_char

Parametro	DataType	Description
		(impostazione predefinita: FALSE)

Example di utilizzare diverse opzioni di mascheramento

Maschera l'intera stringa di input con il carattere 'X' predefinito

```
postgres=> SELECT pgcolumnmask.mask_text('Hello World');
  mask_text
-------------
 XXXXXXXXXXX
```

Usa l'`mask_char`argomento per mascherare l'immissione di testo utilizzando un carattere diverso

```
postgres=> SELECT pgcolumnmask.mask_text('Hello World', '*');
  mask_text
-------------
 ***********
```

Usa `visible_prefix` i `visible_suffix` parametri per controllare quanti caratteri rimangono non mascherati all'inizio e alla fine del testo

```
postgres=> SELECT pgcolumnmask.mask_text('Hello World', '*', 5, 1);
  mask_text
-------------
 Hello*****d
```

Quando `use_hash_mask` è vero, la stringa di input viene mascherata utilizzando caratteri casuali, l'`mask_char`argomento viene ignorato ma viene comunque `visible_prefix` rispettato `visible_suffix`

```
postgres=> SELECT pgcolumnmask.mask_text('Hello World', '*', 2, 2, true);
  mask_text
-------------
 Hex36dOHild
```

mask_timestamp

Parametro	DataType	Description
`ts_to_mask`	TIMESTAMP	Il timestamp originale da mascherare
`mask_part`	TEXT	Speciifica quale parte del timestamp mascherare (impostazione predefinita: 'all') Valori validi: 'anno', 'mese', 'giorno', 'ora', 'minuto', 'secondo', 'tutto'
`mask_value`	TIMESTAMP	Il valore del timestamp da usare per il mascheramento (predefinito: '1900-01-01 00:00:00 ')

Example di utilizzo di **`mask_timestamps`**

Questi esempi mostrano il mascheramento completo del timestamp su un valore predefinito, il mascheramento parziale di componenti del timestamp specifici (solo anno) e il mascheramento con un valore sostitutivo personalizzato.

Maschera completamente il valore di input al timestamp predefinito

```
postgres=> SELECT pgcolumnmask.mask_timestamp('2023-06-15 14:30:00');
   mask_timestamp
---------------------
 1900-01-01 00:00:00
```

Per mascherare solo una parte del timestamp, ad esempio solo l'anno

```
postgres=> SELECT pgcolumnmask.mask_timestamp('2023-06-15 14:30:00', 'year');
   mask_timestamp
---------------------
 1900-06-15 14:30:00
```

Per modificare il valore mascherato per il timestamp usa l'argomento `mask_value`

```
postgres=> SELECT pgcolumnmask.mask_timestamp('2023-06-15 14:30:00', 'all', '2012-12-12
 12:12:12');
   mask_timestamp
---------------------
 2012-12-12 12:12:12
```

mask_timestamp

Una funzione per mascherare gli indirizzi e-mail preservando la struttura delle e-mail.

Parametro	DataType	Description
`input`	TEXT	L'indirizzo email originale da mascherare
`mask_char`	CARATTERE (1)	Carattere usato per mascherare (predefinito: 'X')
`mask_local`	BOOLEAN	Se TRUE, maschera la parte locale dell'email (prima di @) (impostazione predefinita: TRUE)
`mask_domain`	BOOLEAN	Se TRUE, maschera la parte del dominio dell'email (dopo @) (impostazione predefinita: TRUE)

Example di utilizzo di **`mask_email`**

Questi esempi illustrano il mascheramento completo delle e-mail, i caratteri di maschera personalizzati e il mascheramento selettivo della parte locale o della parte del dominio dell'indirizzo e-mail.

Mascheratura completa

```
postgres=> SELECT pgcolumnmask.mask_email('user@example.com');
    mask_email
------------------
```

```
 XXXX@XXXXXXX.com
```

Si usa `mask_char` per cambiare il carattere usato per mascherare

```
postgres=> SELECT pgcolumnmask.mask_email('user@example.com', '*');
    mask_email
------------------
 ****@*******.com
```

Usa `mask_local` e controlla `mask_domain` il mascheramento a livello locale e di dominio

```
postgres=> SELECT pgcolumnmask.mask_email('user@example.com', '*', true, false);
    mask_email
------------------
 ****@example.com

postgres=> SELECT pgcolumnmask.mask_email('user@example.com', '*', false, true);
    mask_email
------------------
 user@*******.com
```

Implementazione di pg_columnmask in un flusso di lavoro end-to-end

Questa sezione illustra un'implementazione completa dell'`pg_columnmask`utilizzo di una tabella dei dipendenti di esempio con dati sensibili. Imparerai come creare funzioni di mascheramento personalizzate, definire più politiche di mascheramento con diversi livelli di peso per vari ruoli (stagista, supporto, analista) e osserverai come gli utenti con appartenenza a uno o più ruoli vedono diversi livelli di dati mascherati. Gli esempi riguardano anche il comportamento di mascheramento nelle istruzioni DML con clausole RETURNING, i trigger nelle tabelle rispetto alle viste e le operazioni di gestione delle politiche, tra cui la ridenominazione, la modifica dei pesi e la pulizia.

1. Crea una tabella di esempio con alcuni dati sensibili:

```
CREATE SCHEMA hr;

CREATE TABLE hr.employees (
    id INT PRIMARY KEY,
    name TEXT NOT NULL,
    email TEXT,
    ssn TEXT,
    salary NUMERIC(10,2)
```

```
 );

INSERT INTO hr.employees VALUES
    (1, 'John Doe', 'john.doe@example.com', '123-45-6789', 50000.00),
    (2, 'Jane Smith', 'jane.smith@example.com', '987-65-4321', 60000.00);
```

2. Crea funzioni di mascheramento personalizzate:

```
CREATE OR REPLACE FUNCTION public.mask_ssn(ssn TEXT)
    RETURNS TEXT AS $$
    BEGIN
        RETURN 'XXX-XX-' || RIGHT(ssn, 4);
    END;
    $$ LANGUAGE plpgsql;

CREATE OR REPLACE FUNCTION public.mask_salary(salary NUMERIC, multiplier NUMERIC
 DEFAULT 0.0)
    RETURNS NUMERIC AS $$
    BEGIN
        RETURN salary * multiplier;
    END;
    $$ LANGUAGE plpgsql;
```

3. Crea più politiche con diversi livelli di mascheramento in base ai ruoli degli utenti:

```
-- Create different roles
CREATE ROLE analyst_role;
CREATE ROLE support_role;
CREATE ROLE intern_role;

GRANT USAGE ON SCHEMA hr TO analyst_role, support_role, intern_role;
GRANT SELECT ON hr.employees TO analyst_role, support_role, intern_role;
---------------------------------------------------------------------

-- Low-Weight Policy (Intern)
CALL pgcolumnmask.create_masking_policy(
    'employee_mask_strict',
    'hr.employees',
    JSON_BUILD_OBJECT('name', 'pgcolumnmask.mask_text(name, ''*'')',
                      'email', 'pgcolumnmask.mask_email(email)',
                      'ssn', 'pgcolumnmask.mask_text(ssn, ''*'')',
                      'salary', 'public.mask_salary(salary)')::JSONB,
    ARRAY['intern_role'],
```

```
    10  -- Lowest weight
);

-------------------------------------------------------------------
-- Medium-Weight Policy (Support)
CALL pgcolumnmask.create_masking_policy(
    'employee_mask_moderate',
    'hr.employees',
    JSON_BUILD_OBJECT('email', 'pgcolumnmask.mask_email(email, ''#'')',
                      'ssn', 'public.mask_ssn(ssn)',
                      'salary', 'public.mask_salary(salary)')::JSONB,
    ARRAY['support_role'],
    50   -- Medium weight
);

-------------------------------------------------------------------
-- High-Weight Policy (Analyst)
CALL pgcolumnmask.create_masking_policy(
    'employee_mask_light',
    'hr.employees',
    JSON_BUILD_OBJECT('ssn', 'public.mask_ssn(ssn)',
                      'salary', 'public.mask_salary(salary, 0.9)')::JSONB,
    ARRAY['analyst_role'],
    100   -- Highest weight
);
```

4. Gli esempi seguenti mostrano come i diversi utenti visualizzano i dati in base all'appartenenza al ruolo e al peso delle politiche.

```
-- Create users
CREATE USER sarah_intern;
GRANT intern_role TO sarah_intern;

CREATE USER lisa_support;
GRANT support_role TO lisa_support;

CREATE USER mike_analyst;
GRANT analyst_role TO mike_analyst;

CREATE USER ethan_support_intern;
GRANT support_role, intern_role TO ethan_support_intern;

CREATE USER john_analyst_intern;
```

```
GRANT analyst_role, intern_role TO john_analyst_intern;
```

Come tirocinante (mascheramento più rigoroso):

```
SET ROLE sarah_intern;

SELECT * FROM hr.employees;
 id |    name    |         email          |     ssn     | salary
----+------------+------------------------+-------------+--------
  1 | ********   | XXXXXXXX@XXXXXXX.com   | ********** |   0.00
  2 | ********** | XXXXXXXXXX@XXXXXXX.com | ********** |   0.00
```

Come utente di supporto (mascheramento moderato):

```
SET ROLE lisa_support;

SELECT * FROM hr.employees;
 id |    name    |         email          |     ssn     | salary
----+------------+------------------------+-------------+--------
  1 | John Doe   | ########@#######.com   | XXX-XX-6789 |   0.00
  2 | Jane Smith | ##########@#######.com | XXX-XX-4321 |   0.00
```

In qualità di analista (mascheramento più leggero):

```
SET ROLE mike_analyst;

SELECT * FROM hr.employees;
 id |    name    |         email          |     ssn     |  salary
----+------------+------------------------+-------------+----------
  1 | John Doe   | john.doe@example.com   | XXX-XX-6789 | 45000.00
  2 | Jane Smith | jane.smith@example.com | XXX-XX-4321 | 54000.00
```

Come utente ethan_support_intern che è sia stagista che utente di supporto:

```
SET ROLE ethan_support_intern;

-- masking policies appliable to this user: employee_mask_strict and
 employee_mask_moderate
-- id : unmasked because no masking policy appliable on ethan_support_intern
--            masks these columns
-- name : masked because of employee_mask_strict policy
```

```
-- email, ssn, salary : both employee_mask_strict and employee_mask_moderate mask
 these columns
--                      but employee_mask_moderate will be use because of higher
 weight

SELECT * FROM hr.employees;
 id |    name    |         email          |     ssn     | salary
----+------------+------------------------+-------------+--------
  1 | ********   | ########@#######.com   | XXX-XX-6789 |   0.00
  2 | ********** | ##########@#######.com | XXX-XX-4321 |   0.00
```

Come john_analyst_intern che è sia stagista che analista:

```
SET ROLE john_analyst_intern;

-- masking policies appliable to this user: employee_mask_strict and
 employee_mask_light
-- id : unmasked because no masking policy appliable on john_analyst_intern
--            masks these columns
-- name, email : masked because of employee_mask_strict
-- ssn, salary : both employee_mask_strict and employee_mask_light mask these
 columns
--               but employee_mask_light will be use because of higher weight

SELECT * FROM hr.employees;
 id |    name    |         email          |     ssn     |  salary
----+------------+------------------------+-------------+----------
  1 | ********   | XXXXXXXX@XXXXXXX.com   | XXX-XX-6789 | 45000.00
  2 | ********** | XXXXXXXXXX@XXXXXXX.com | XXX-XX-4321 | 54000.00
```

Comprensione del comportamento di mascheramento nelle operazioni DML

`pg_columnmasks`si applica in modo coerente a tutte le operazioni DML, incluse le istruzioni INSERT, UPDATE, DELETE e MERGE. Quando si eseguono queste operazioni, Aurora PostgreSQL maschera i dati secondo un principio fondamentale: tutti i dati letti dallo storage vengono mascherati in base alle policy applicabili dell'utente corrente.

Il mascheramento influisce su alcuni dei seguenti componenti di interrogazione, come:

- Clausole WHERE
- Condizioni JOIN

- Sottoquery
- Clausole di RESTITUZIONE

Tutti questi componenti funzionano su valori mascherati, non sui dati originali. Sebbene i dati vengano scritti nello storage senza mascherare, gli utenti vedono la loro visualizzazione mascherata solo quando li rileggono.

Aurora PostgreSQL applica tutti i vincoli del database (NOT NULL, UNIQUE, CHECK, FOREIGN KEY) sui valori effettivi memorizzati, non sui valori mascherati. Ciò può occasionalmente creare apparenti incongruenze se le funzioni di mascheramento non sono progettate con cura.

Il mascheramento funziona insieme alle autorizzazioni a livello di colonna:

- Gli utenti senza privilegi SELECT non possono leggere le colonne
- Gli utenti con privilegi SELECT vedono i valori mascherati in base alle politiche applicabili

Comprensione del comportamento di mascheramento nelle funzioni di attivazione

Quando `pg_columnmask` le policy vengono applicate alle tabelle, è importante capire come il mascheramento interagisce con le funzioni di attivazione. I trigger sono funzioni di database che vengono eseguite automaticamente in risposta a determinati eventi su una tabella, come le operazioni INSERT, UPDATE o DELETE.

Per impostazione predefinita, DDM applica regole di mascheramento diverse a seconda del tipo di trigger:

Trigger della tabella

Le tabelle di transizione sono smascherate: le funzioni di attivazione sulle tabelle hanno accesso ai dati non mascherati nelle relative tabelle di transizione sia per le vecchie che per le nuove versioni di riga

I proprietari delle tabelle creano i trigger e sono proprietari dei dati, in modo da avere pieno accesso per gestire le proprie tabelle in modo efficace

Visualizza i trigger (ANZICHÉ i trigger)

Le tabelle di transizione sono mascherate: le funzioni di attivazione nelle viste visualizzano i dati mascherati in base alle autorizzazioni dell'utente corrente

I proprietari delle viste possono differire dai proprietari delle tabelle di base e devono rispettare le politiche di mascheramento sulle tabelle sottostanti

Due parametri di configurazione a livello di server controllano il comportamento dei trigger con tabelle mascherate. Questi possono essere impostati solo da: `rds_superuser`

- Limita i trigger sulle tabelle mascherate: impedisce l'esecuzione dei trigger quando un utente mascherato esegue operazioni DML su tabelle con politiche di mascheramento applicabili.
- Limita i trigger sulle viste con tabelle mascherate: — Impedisce l'esecuzione dei trigger sulle viste quando la definizione della vista include tabelle con politiche di mascheramento applicabili all'utente corrente.

Example delle differenze tra la funzione, l'applicazione, la tabella e la visualizzazione

L'esempio seguente crea una funzione di attivazione che stampa valori di riga vecchi e nuovi, quindi dimostra come la stessa funzione si comporti in modo diverso quando è collegata a una tabella rispetto a una vista.

```
-- Create trigger function
CREATE OR REPLACE FUNCTION print_changes()
    RETURNS TRIGGER AS
    $$
        BEGIN
        RAISE NOTICE 'Old row: name=%, email=%, ssn=%, salary=%',
            OLD.name, OLD.email, OLD.ssn, OLD.salary;

        RAISE NOTICE 'New row: name=%, email=%, ssn=%, salary=%',
            NEW.name, NEW.email, NEW.ssn, NEW.salary;

        RETURN NEW;
        END;
    $$ LANGUAGE plpgsql;

-- Create trigger
CREATE TRIGGER print_changes_trigger
    BEFORE UPDATE ON hr.employees
    FOR EACH ROW
    EXECUTE FUNCTION print_changes();

-- Grant update to analyst role
```

```
GRANT UPDATE ON hr.employees TO analyst_role;

-- Unmasked data must be seen inside trigger even for masked user for the OLD and NEW
-- row passed to trigger function
BEGIN;
SET ROLE mike_analyst;
UPDATE hr.employees SET id = id + 10 RETURNING *;
NOTICE:  Old row: name=John Doe, email=john.doe@example.com, ssn=123-45-6789,
 salary=50000.00
NOTICE:  New row: name=John Doe, email=john.doe@example.com, ssn=123-45-6789,
 salary=50000.00
NOTICE:  Old row: name=Jane Smith, email=jane.smith@example.com, ssn=987-65-4321,
 salary=60000.00
NOTICE:  New row: name=Jane Smith, email=jane.smith@example.com, ssn=987-65-4321,
 salary=60000.00
 id |    name    |         email          |     ssn     |  salary
----+------------+------------------------+-------------+----------
 11 | John Doe   | john.doe@example.com   | XXX-XX-6789 | 45000.00
 12 | Jane Smith | jane.smith@example.com | XXX-XX-4321 | 54000.00
(2 rows)

ROLLBACK;


-- Triggers on views (which are supposed to see masked data for new/old row)
CREATE VIEW hr.view_over_employees AS SELECT * FROM hr.employees;
GRANT UPDATE, SELECT ON hr.view_over_employees TO analyst_role;

-- Create trigger for this view
CREATE TRIGGER print_changes_trigger
    INSTEAD OF UPDATE ON hr.view_over_employees
    FOR EACH ROW
    EXECUTE FUNCTION print_changes();

-- Masked new and old rows should be passed to trigger if trigger is on view
BEGIN;
SET ROLE mike_analyst;
UPDATE hr.view_over_employees SET id = id + 10 RETURNING *;
NOTICE:  Old row: name=John Doe, email=john.doe@example.com, ssn=XXX-XX-6789,
 salary=45000.00
NOTICE:  New row: name=John Doe, email=john.doe@example.com, ssn=XXX-XX-6789,
 salary=45000.00
NOTICE:  Old row: name=Jane Smith, email=jane.smith@example.com, ssn=XXX-XX-4321,
 salary=54000.00
```

```
NOTICE:  New row: name=Jane Smith, email=jane.smith@example.com, ssn=XXX-XX-4321,
 salary=54000.00
 id |    name    |         email          |     ssn     |  salary
----+------------+------------------------+-------------+----------
 11 | John Doe   | john.doe@example.com   | XXX-XX-6789 | 45000.00
 12 | Jane Smith | jane.smith@example.com | XXX-XX-4321 | 54000.00
(2 rows)
ROLLBACK;
```

Consigliamo di esaminare il comportamento dei trigger prima di implementarli nelle tabelle mascherate. I trigger delle tabelle hanno accesso ai dati non mascherati nelle tabelle di transizione, mentre i trigger di visualizzazione visualizzano i dati mascherati.

Example della politica di mascheramento della ridenominazione

L'esempio seguente mostra come rinominare le politiche esistenti utilizzando la procedura. `rename_masking_policy`

```
-- Rename the strict policy
CALL pgcolumnmask.rename_masking_policy(
    'employee_mask_strict',
    'hr.employees',
    'intern_protection_policy'
);

-- Verify the rename
SELECT policyname, roles, weight
    FROM pgcolumnmask.pg_columnmask_policies
    WHERE tablename = 'employees'
    ORDER BY weight DESC;

        policyname        |     roles      | weight
--------------------------+----------------+--------
 employee_mask_light      | {analyst_role} |    100
 employee_mask_moderate   | {support_role} |     50
 intern_protection_policy | {intern_role}  |     10
```

Example di alterare il peso delle politiche

L'esempio seguente mostra come modificare i pesi delle politiche per modificarne il peso.

```
-- Change weight of moderate policy
CALL pgcolumnmask.alter_masking_policy(
```

```
    'employee_mask_moderate'::NAME,
    'hr.employees'::REGCLASS,
    NULL,    -- Keep existing masking expressions
    NULL,    -- Keep existing roles
    75       -- New weight
);

-- Verify the changes
SELECT policyname, roles, weight
    FROM pgcolumnmask.pg_columnmask_policies
    WHERE tablename = 'employees'
    ORDER BY weight DESC;
        policyname        |     roles      | weight
--------------------------+----------------+--------
 employee_mask_light      | {analyst_role} |    100
 employee_mask_moderate   | {support_role} |     75
 intern_protection_policy | {intern_role}  |     10
```

Example di pulizia

L'esempio seguente mostra come eliminare tutte le politiche, le tabelle e gli utenti.

```
-- Drop policies
CALL pgcolumnmask.drop_masking_policy(
    'intern_protection_policy',
    'hr.employees'
);

CALL pgcolumnmask.drop_masking_policy(
    'employee_mask_moderate',
    'hr.employees'
);

CALL pgcolumnmask.drop_masking_policy(
    'employee_mask_light',
    'hr.employees'
);

-- Drop table and functions
DROP VIEW IF EXISTS hr.view_over_employees;
DROP TABLE IF EXISTS hr.employees;
DROP SCHEMA IF EXISTS hr;
DROP FUNCTION IF EXISTS public.mask_ssn(text);
DROP FUNCTION IF EXISTS public.mask_salary(numeric, numeric);
```

```
-- Drop users
DROP USER sarah_intern, lisa_support, mike_analyst,
    ethan_support_intern, john_analyst_intern;
DROP ROLE intern_role, support_role, analyst_role;
```

Configurazione del ruolo di gestione delle policy di mascheramento

L'estensione `pg_columnmask` PostgreSQL per il mascheramento delle colonne consente di delegare la gestione delle policy di mascheramento a un ruolo specifico, anziché richiedere i privilegi del proprietario della tabella. `rds_superuser` Ciò fornisce un controllo più granulare su chi può creare, modificare ed eliminare le politiche di mascheramento.

Per configurare il ruolo che disporrà dei privilegi di gestione delle policy di mascheramento, procedi nel seguente modo:

1. Crea il ruolo di amministratore delle politiche: come utente`rds_superuser`, crea un nuovo ruolo responsabile della gestione delle politiche di mascheramento:

   ```
   CREATE ROLE mask_admin NOLOGIN;
   ```

2. Configura il parametro PostgreSQL: nel gruppo di parametri del cluster DB personalizzato, imposta il parametro di configurazione `pgcolumnmask.policy_admin_rolname` del motore sul nome del ruolo che hai creato:

   ```
   pgcolumnmask.policy_admin_rolname = mask_admin
   ```

   Questi parametri di configurazione del motore possono essere impostati in un gruppo di parametri del cluster DB e non richiede il riavvio dell'istanza. Per i dettagli sull'aggiornamento dei parametri, vedereModifica dei parametri in un gruppo di parametri del cluster di database in Amazon Aurora.

3. Concedi il ruolo agli utenti In quanto utente`rds_superuser`, concedi il `mask_admin` ruolo agli utenti che dovrebbero essere in grado di gestire le politiche di mascheramento:

   ```
   CREATE USER alice LOGIN;
   CREATE USER bob LOGIN;
   GRANT mask_admin TO alice, bob;
   ```

   Inoltre, assicurati che gli utenti abbiano il privilegio USAGE sugli schemi in cui gestiranno le politiche di mascheramento:

```
GRANT USAGE ON SCHEMA hr TO alice, bob;
```

Ora, quando gli utenti `alice` si `bob` connettono al database, possono utilizzare le funzioni di `pg_columnmask` estensione standard per creare, modificare ed eliminare le politiche di mascheramento su tutte le tabelle in tutti gli schemi in cui dispongono `USAGE` dei privilegi sullo schema.

Le migliori pratiche per l'implementazione sicura di pg_columnmask

La sezione seguente fornisce le best practice di sicurezza per l'implementazione `pg_columnmask` nell'ambiente Aurora PostgreSQL. Segui questi consigli per:

- Stabilisci un'architettura di controllo degli accessi sicura basata sui ruoli
- Sviluppa funzioni di mascheramento che impediscano le vulnerabilità di sicurezza
- Comprendi e controlla il comportamento dei trigger con dati mascherati

Architettura di sicurezza basata sui ruoli

Definisci una gerarchia di ruoli per implementare i controlli di accesso nel tuo database. Aurora PostgreSQL `pg_columnmask` aumenta questi controlli fornendo un livello aggiuntivo per il mascheramento granulare dei dati all'interno di tali ruoli.

Crea ruoli dedicati in linea con le funzioni organizzative anziché concedere autorizzazioni ai singoli utenti. Questo approccio offre una migliore verificabilità e semplifica la gestione delle autorizzazioni man mano che la struttura organizzativa si evolve.

Example di creare una gerarchia dei ruoli organizzativi

L'esempio seguente crea una gerarchia di ruoli organizzativi con ruoli dedicati per diverse funzioni, quindi assegna i singoli utenti ai ruoli appropriati. In questo esempio, i ruoli organizzativi (analyst_role, support_role) vengono creati per primi, quindi ai singoli utenti viene concessa l'appartenenza a questi ruoli. Questa struttura consente di gestire le autorizzazioni a livello di ruolo anziché per ogni singolo utente.

```
-- Create organizational role hierarchy
CREATE ROLE data_admin_role;
CREATE ROLE security_admin_role;
```

```
CREATE ROLE analyst_role;
CREATE ROLE support_role;
CREATE ROLE developer_role;

-- Specify security_admin_role as masking policy manager in the DB cluster parameter
-- group pgcolumnmask.policy_admin_rolname = 'security_admin_role'

-- Create specific users and assign to appropriate roles
CREATE USER security_manager;
CREATE USER data_analyst1, data_analyst2;
CREATE USER support_agent1, support_agent2;

GRANT security_admin_role TO security_manager;
GRANT analyst_role TO data_analyst1, data_analyst2;
GRANT support_role TO support_agent1, support_agent2;
```

Implementa il principio del privilegio minimo concedendo solo le autorizzazioni minime necessarie per ogni ruolo. Evita di concedere autorizzazioni ampie che potrebbero essere sfruttate se le credenziali vengono compromesse.

```
-- Grant specific table permissions rather than schema-wide access
GRANT SELECT ON sensitive_data.customers TO analyst_role;
GRANT SELECT ON sensitive_data.transactions TO analyst_role;
-- Do not grant: GRANT ALL ON SCHEMA sensitive_data TO analyst_role;
```

Gli amministratori delle politiche richiedono USAGE privilegi sugli schemi in cui gestiscono le politiche di mascheramento. Concedi questi privilegi in modo selettivo, seguendo il principio del privilegio minimo. Effettua revisioni periodiche delle autorizzazioni di accesso allo schema per garantire che solo il personale autorizzato mantenga le capacità di gestione delle politiche.

La configurazione dei parametri del ruolo di amministratore delle politiche è riservata ai soli amministratori del database. Questo parametro non può essere modificato a livello di database o di sessione, per evitare che gli utenti senza privilegi abbiano la precedenza sulle assegnazioni degli amministratori delle policy. Questa restrizione garantisce che il controllo delle policy di mascheramento rimanga centralizzato e sicuro.

Assegna il ruolo di amministratore delle politiche a individui specifici anziché a gruppi. Questo approccio mirato garantisce un accesso selettivo alla gestione delle policy di mascheramento, in quanto gli amministratori delle policy hanno la possibilità di mascherare tutte le tabelle all'interno del database.

Sviluppo sicuro della funzione di mascheramento

Sviluppa funzioni di mascheramento utilizzando la semantica di associazione precoce per garantire il corretto tracciamento delle dipendenze e prevenire vulnerabilità di associazione tardiva, come la modifica del percorso di ricerca durante il runtime. Si consiglia di utilizzare la `BEGIN ATOMIC` sintassi per le funzioni SQL per abilitare la convalida in fase di compilazione (ovvero l'associazione anticipata) e la gestione delle dipendenze.

```
-- Example - Secure masking function with early binding
CREATE OR REPLACE FUNCTION secure_mask_ssn(input_ssn TEXT)
    RETURNS TEXT
    LANGUAGE SQL
    IMMUTABLE PARALLEL SAFE STRICT
    BEGIN ATOMIC
        SELECT CASE
            WHEN input_ssn IS NULL THEN NULL
            WHEN length(input_ssn) < 4 THEN repeat('X', length(input_ssn))
            ELSE repeat('X', length(input_ssn) - 4) || right(input_ssn, 4)
        END;
    END;
```

In alternativa, è possibile creare funzioni immuni alle modifiche del percorso di ricerca qualificando in modo esplicito lo schema di tutti i riferimenti agli oggetti, garantendo un comportamento coerente tra le diverse sessioni utente.

```
-- Function immune to search path changes
CREATE OR REPLACE FUNCTION data_masking.secure_phone_mask(phone_number TEXT)
    RETURNS TEXT
    LANGUAGE SQL
    IMMUTABLE PARALLEL SAFE STRICT
    AS $$
    SELECT CASE
        WHEN phone_number IS NULL THEN NULL
        WHEN public.length(public.regexp_replace(phone_number, '[^0-9]', '', 'g')) < 10
 THEN 'XXX-XXX-XXXX'
        ELSE public.regexp_replace(
            phone_number,
            '([0-9]{3})[0-9]{3}([0-9]{4})',
            public.concat('\1-XXX-\2')
        )
    END;
    $$;
```

Implementa la convalida degli input all'interno delle funzioni di mascheramento per gestire i casi limite e prevenire comportamenti imprevisti. Includi sempre la gestione dei NULL e convalida i formati di input per garantire un comportamento di mascheramento coerente.

```
-- Robust masking function with comprehensive input validation
CREATE OR REPLACE FUNCTION secure_mask_phone(phone_number TEXT)
    RETURNS TEXT
    LANGUAGE SQL
    IMMUTABLE PARALLEL SAFE STRICT
    BEGIN ATOMIC
        SELECT CASE
            WHEN phone_number IS NULL THEN NULL
            WHEN length(trim(phone_number)) = 0 THEN phone_number
            WHEN length(regexp_replace(phone_number, '[^0-9]', '', 'g')) < 10 THEN
 'XXX-XXX-XXXX'
            ELSE regexp_replace(phone_number, '([0-9]{3})[0-9]{3}([0-9]{4})', '\1-XXX-
\2')
        END;
    END;
```

DML attiva il comportamento con pg_columnmask

Per i trigger delle tabelle, le tabelle di transizione verranno completamente smascherate. Per i trigger di visualizzazione (IOT), le tabelle di transizione verranno mascherate in base alle autorizzazioni di visualizzazione dell'utente corrente.

Trigger di tabella con pg_columnmask

Ai trigger viene passata una tabella di transizione che contiene la vecchia e la nuova versione delle righe modificate dalla query DML di avvio. A seconda di quando viene attivato il trigger, Aurora PostgreSQL popola le righe vecchie e nuove. Ad esempio, un `BEFORE INSERT` trigger contiene solo nuove versioni delle righe e le vecchie versioni vuote perché non esiste una vecchia versione a cui fare riferimento.

`pg_columnmask`non maschera le tabelle di transizione all'interno dei trigger delle tabelle. I trigger possono utilizzare colonne mascherate all'interno del loro corpo e vedono i dati non mascherati. Il creatore del trigger deve assicurarsi che il trigger venga eseguito per un utente. L'esempio seguente funziona correttamente in questo caso.

```
-- Example for table trigger uses masked column in its definition
-- Create a table and insert some rows
```

```
CREATE TABLE public.credit_card_table (
    name TEXT,
    credit_card_no VARCHAR(16),
    is_fraud BOOL
);

INSERT INTO public.credit_card_table (name, credit_card_no, is_fraud)
    VALUES
    ('John Doe', '4532015112830366', false),
    ('Jane Smith', '5410000000000000', true),
    ('Brad Smith', '1234567891234567', true);

-- Create a role which will see masked data and grant it privileges
CREATE ROLE intern_user;
GRANT SELECT, DELETE ON public.credit_card_table TO intern_user;

-- Trigger which will silenty skip delete of non fraudelent credit cards
CREATE OR REPLACE FUNCTION prevent_non_fraud_delete()
    RETURNS TRIGGER AS
    $$
    BEGIN
        IF OLD.is_fraud = false THEN
            RETURN NULL;
        END IF;
        RETURN OLD;
    END;
    $$ LANGUAGE plpgsql;

CREATE TRIGGER prevent_non_fraud_delete
    BEFORE DELETE ON credit_card_table
    FOR EACH ROW
    EXECUTE FUNCTION prevent_non_fraud_delete();

CREATE OR REPLACE FUNCTION public.return_false()
    RETURNS BOOLEAN
    LANGUAGE SQL
    IMMUTABLE PARALLEL SAFE STRICT
    BEGIN ATOMIC
      SELECT false;
    END;

-- A masking policy that masks both credit card number and is_fraud column.
-- If we apply masking inside trigger then prevent_non_fraud_delete trigger will
-- allow deleting more rows to masked user (even non fraud ones).
```

```
CALL pgcolumnmask.create_masking_policy(
    'mask_credit_card_no_&_is_fraud'::NAME,
    'public.credit_card_table'::REGCLASS,
    JSON_BUILD_OBJECT('credit_card_no', 'pgcolumnmask.mask_text(credit_card_no)',
                      'is_fraud', 'public.return_false()')::JSONB,
    ARRAY['intern_user']::NAME[],
    10::INT
);

-- Test trigger behaviour using intern_user
BEGIN;
SET ROLE intern_user;
-- credit card number & is_fraud is completely masked from intern_user
SELECT * FROM public.credit_card_table;
    name    |  credit_card_no  | is_fraud
------------+------------------+----------
 John Doe   | XXXXXXXXXXXXXXXX | f
 Jane Smith | XXXXXXXXXXXXXXXX | f
 Brad Smith | XXXXXXXXXXXXXXXX | f
(3 rows)

-- The delete trigger lets the intern user delete rows for Jane and Brad even though
-- intern_user sees their is_fraud = false, but the table trigger works with
 original
-- unmasked value
DELETE FROM public.credit_card_table RETURNING *;
    name    |  credit_card_no  | is_fraud
------------+------------------+----------
 Jane Smith | XXXXXXXXXXXXXXXX | f
 Brad Smith | XXXXXXXXXXXXXXXX | f
(2 rows)

COMMIT;
```

Il creatore di Trigger comunica i dati non mascherati all'utente se non presta attenzione alle istruzioni che utilizza nel corpo del trigger. Ad esempio, l'utilizzo di a `RAISE NOTICE '%', masked_column;` stampa la colonna per l'utente corrente.

```
-- Example showing table trigger leaking column value to current user
CREATE OR REPLACE FUNCTION leaky_trigger_func()
    RETURNS TRIGGER AS
    $$
```

```
    BEGIN
        RAISE NOTICE 'Old credit card number was: %', OLD.credit_card_no;
        RAISE NOTICE 'New credit card number is %', NEW.credit_card_no;
        RETURN NEW;
    END;
    $$ LANGUAGE plpgsql;

CREATE TRIGGER leaky_trigger
    AFTER UPDATE ON public.credit_card_table
    FOR EACH ROW
    EXECUTE FUNCTION leaky_trigger_func();

-- Grant update on column is_fraud to auditor role
-- auditor will NOT HAVE PERMISSION TO READ DATA
CREATE ROLE auditor;
GRANT UPDATE (is_fraud) ON public.credit_card_table TO auditor;

-- Also add auditor role to existing masking policy on credit card table
CALL pgcolumnmask.alter_masking_policy(
    'mask_credit_card_no_&_is_fraud'::NAME,
    'public.credit_card_table'::REGCLASS,
    NULL::JSONB,
    ARRAY['intern_user', 'auditor']::NAME[],
    NULL::INT
);

-- Log in as auditor
-- [auditor]
-- Update will fail if trying to read data from the table
UPDATE public.credit_card_table
    SET is_fraud = true
    WHERE credit_card_no = '4532015112830366';
ERROR:  permission denied for table cc_table

-- [auditor]
-- But leaky update trigger will still print the entire row even though
-- current user does not have permission to select from public.credit_card_table
UPDATE public.credit_card_table SET is_fraud = true;
NOTICE:  Old credit_card_no was: 4532015112830366
NOTICE:  New credit_card_no is 4532015112830366
```

Trigger sulle viste con pg_columnmask (anziché trigger)

I trigger possono essere creati solo nelle viste in PostgreSQL. Vengono utilizzati per eseguire istruzioni DML su viste che non sono aggiornabili. Le tabelle di transito sono sempre mascherate all'interno anziché al trigger (IOT), poiché la vista e le tabelle di base utilizzate all'interno della query di visualizzazione potrebbero avere proprietari diversi. In tal caso, le tabelle di base potrebbero avere alcune politiche di mascheramento applicabili al proprietario della vista e il proprietario della vista deve sempre visualizzare i dati mascherati delle tabelle di base all'interno dei suoi trigger. Questo è diverso dai trigger sulle tabelle perché in tal caso il creatore del trigger e i dati all'interno delle tabelle sono di proprietà dello stesso utente, cosa che non avviene in questo caso.

```
-- Create a view over credit card table
CREATE OR REPLACE VIEW public.credit_card_view
    AS
    SELECT * FROM public.credit_card_table;

-- Truncate credit card table and insert fresh data
TRUNCATE TABLE public.credit_card_table;
INSERT INTO public.credit_card_table (name, credit_card_no, is_fraud)
    VALUES
    ('John Doe', '4532015112830366', false),
    ('Jane Smith', '5410000000000000', true),
    ('Brad Smith', '1234567891234567', true);

CREATE OR REPLACE FUNCTION public.print_changes()
    RETURNS TRIGGER AS
    $$
    BEGIN
        RAISE NOTICE 'Old row: name=%, credit card number=%, is fraud=%',
            OLD.name, OLD.credit_card_no, OLD.is_fraud;

        RAISE NOTICE 'New row: name=%, credit card number=%, is fraud=%',
            NEW.name, NEW.credit_card_no, NEW.is_fraud;

    RETURN NEW;
   END;
   $$ LANGUAGE plpgsql;

CREATE TRIGGER print_changes_trigger
    INSTEAD OF UPDATE ON public.credit_card_view
    FOR EACH ROW
```

```
    EXECUTE FUNCTION public.print_changes();

GRANT SELECT, UPDATE ON public.credit_card_view TO auditor;

-- [auditor]
-- Login as auditor role
BEGIN;

-- Any data coming out from the table will be masked in instead of triggers
-- according to masking policies applicable to current user
UPDATE public.credit_card_view
    SET name = CONCAT(name, '_new_name')
    RETURNING *;
NOTICE:  Old row: name=John Doe, credit card number=XXXXXXXXXXXXXXXX, is fraud=f
NOTICE:  New row: name=John Doe_new_name, credit card number=XXXXXXXXXXXXXXXX, is
 fraud=f
NOTICE:  Old row: name=Jane Smith, credit card number=XXXXXXXXXXXXXXXX, is fraud=f
NOTICE:  New row: name=Jane Smith_new_name, credit card number=XXXXXXXXXXXXXXXX, is
 fraud=f
NOTICE:  Old row: name=Brad Smith, credit card number=XXXXXXXXXXXXXXXX, is fraud=f
NOTICE:  New row: name=Brad Smith_new_name, credit card number=XXXXXXXXXXXXXXXX, is
 fraud=f
        name         |  credit_card_no  | is_fraud
---------------------+------------------+----------
 John Doe_new_name   | XXXXXXXXXXXXXXXX | f
 Jane Smith_new_name | XXXXXXXXXXXXXXXX | f
 Brad Smith_new_name | XXXXXXXXXXXXXXXX | f

 -- Any new data going into the table using INSERT or UPDATE command will be
 unmasked
 UPDATE public.credit_card_view
    SET credit_card_no = '9876987698769876'
    RETURNING *;
NOTICE:  Old row: name=John Doe, credit card number=XXXXXXXXXXXXXXXX, is fraud=f
NOTICE:  New row: name=John Doe, credit card number=9876987698769876, is fraud=f
NOTICE:  Old row: name=Jane Smith, credit card number=XXXXXXXXXXXXXXXX, is fraud=f
NOTICE:  New row: name=Jane Smith, credit card number=9876987698769876, is fraud=f
NOTICE:  Old row: name=Brad Smith, credit card number=XXXXXXXXXXXXXXXX, is fraud=f
NOTICE:  New row: name=Brad Smith, credit card number=9876987698769876, is fraud=f
    name    |  credit_card_no  | is_fraud
------------+------------------+----------
 John Doe   | 9876987698769876 | f
 Jane Smith | 9876987698769876 | f
 Brad Smith | 9876987698769876 | f
```

```
COMMIT;
```

Livello GuCs di database/utente per controllare il comportamento dei trigger

Due parametri di configurazione controllano il comportamento di esecuzione dei trigger per gli utenti con politiche di mascheramento applicabili. Utilizzate questi parametri per impedire l'esecuzione dei trigger su tabelle o viste mascherate quando sono necessarie restrizioni di sicurezza aggiuntive. Entrambi i parametri sono disabilitati per impostazione predefinita, per consentire ai trigger di essere eseguiti normalmente.

Primo GUC: restrizione di attivazione dei trigger sulle tabelle mascherate

Specifiche:

- Valore: `pgcolumnmask.restrict_dml_triggers_for_masked_users`
- Tipo: `boolean`
- Impostazione predefinita: `false` (i trigger possono essere eseguiti)

Impedisce l'esecuzione dei trigger sulle tabelle mascherate per gli utenti mascherati se impostata su TRUE. `pg_columnmask`esegue l'errore.

Secondo GUC: attiva la restrizione di attivazione delle viste con tabelle mascherate

Specifiche:

- Valore: `pgcolumnmask.restrict_iot_triggers_for_masked_users`
- Tipo: `boolean`
- Impostazione predefinita: `false` (i trigger possono essere eseguiti)

Impedisce l'esecuzione dei trigger nelle viste che includono tabelle mascherate nella loro definizione per gli utenti mascherati se impostata su TRUE.

Questi parametri funzionano in modo indipendente e sono configurabili come i parametri di configurazione standard del database.

Scenari di spostamento dei dati in Aurora PostgreSQL pg_columnmask

`pg_columnmask`il comportamento varia tra le diverse operazioni di spostamento dei dati a seconda che l'operazione avvenga a livello di archiviazione, logico o applicativo. Le operazioni a livello di storage (come la clonazione) si comportano in modo diverso dalle operazioni logiche

(ad esempiopg_dump) e dalle operazioni a livello di applicazione (come le query FDW). Questa sezione descrive il comportamento di mascheramento per scenari comuni, tra cui replica, backup, esportazioni e migrazioni, e spiega le implicazioni di sicurezza per ciascuno di essi.

Argomenti

- Database globale Aurora e repliche di lettura
- Duplicazione del database e ripristino delle istantanee
- Replica logica
- Implementazioni blu/verdi
- Stream Zero-ETL e CDC
- AWS Database Migration Service
- Esportazioni di dati
- Viste e viste materializzate
- Dump e ripristino dei dati
- Wrapper di dati esterni

Database globale Aurora e repliche di lettura

Le `pg_columnmask` policy Aurora sono archiviate nelle tabelle del sistema di database all'interno del volume del cluster. Tutte le repliche accedono alle stesse policy e restituiscono risultati mascherati in modo coerente. Per le implementazioni di Aurora Global Database, `pg_columnmask` le policy vengono replicate su tabelle secondarie Regioni AWS insieme ad altre tabelle di sistema di database, garantendo una protezione dei dati coerente in tutte le regioni. Durante gli scenari di failover, tutte le `pg_columnmask` policy rimangono intatte e funzionali.

Duplicazione del database e ripristino delle istantanee

Le operazioni di clonazione rapida e ripristino delle istantanee di Aurora preservano tutte le `pg_columnmask` policy, i ruoli e le configurazioni come parte delle tabelle del sistema di database. Il database clonato o ripristinato eredita tutte le politiche esistenti dal cluster di origine. Dopo la clonazione o il ripristino, ogni cluster di database mantiene politiche indipendenti. `pg_columnmask`

Replica logica

Durante la sincronizzazione iniziale, la replica logica utilizza operazioni SQL COPY standard e le `pg_columnmask` policy vengono applicate in base alle autorizzazioni dell'utente di replica. Durante il CDC (change data capture) in corso, le politiche di mascheramento non vengono applicate e i dati

non mascherati vengono replicati tramite i record WAL. Gli utenti con `pg_create_subscription` privilegi possono potenzialmente esfiltrare i dati non mascherati impostando la replica su un sistema che controllano.

Implementazioni blu/verdi

Durante il ripristino delle istantanee, `pg_columnmask` le policy vengono incluse automaticamente. L'ambiente verde inizia con una copia identica di tutte le politiche dell'ambiente blu. Durante la replica dal blu al verde, i dati non vengono mascherati. Le successive modifiche alle policy di mascheramento (comandi DDL) sul cluster blu non vengono replicate sul cluster verde e invalidano le distribuzioni RDS. blue/green

Stream Zero-ETL e CDC

La replica dei dati non è influenzata dalle policy. `pg_columnmask` Zero-ETL supporta la replica DDL ma non replica o le policy RLS. `pg_columnmask` Nessuna policy di mascheramento viene applicata ai dati replicati in Zero-ETL.

AWS Database Migration Service

La sincronizzazione iniziale dei dati viene mascherata o smascherata in base all'utente selezionato per l'attività DMS. I dati CDC vengono sempre smascherati. Sebbene le `pg_columnmask` relative politiche RLS interne possano essere migrate, non funzioneranno su target non abilitati per pg_columnmask.

Esportazioni di dati

`pg_columnmask`tratta le esportazioni come qualsiasi altra operazione di interrogazione: il mascheramento viene applicato in base alle autorizzazioni dell'utente che esegue l'esecuzione. Questo vale per comandi SQL come COPY, SELECT INTO, CREATE TABLE AS e la funzionalità di esportazione S3 di Aurora PostgreSQL.

> **Note**
>
> Quando gli utenti mascherati esportano dati, i file risultanti contengono valori mascherati che possono violare i vincoli del database una volta ripristinati.

Viste e viste materializzate

Tieni a mente le seguenti considerazioni quando utilizzi le viste:

- Visualizzazioni normali: utilizza sempre la `INVOKER` semantica. Le politiche di mascheramento dell'utente corrente si applicano quando si esegue una query sulla vista, indipendentemente da chi ha creato la vista.
- Viste materializzate: quando vengono aggiornate, si applicano le politiche di mascheramento del proprietario della vista materializzata, non le politiche dell'utente che esegue l'aggiornamento. Se il proprietario ha delle politiche di mascheramento, la vista materializzata contiene sempre dati mascherati.

Dump e ripristino dei dati

`pg_dump`funziona come un normale utente del database e applica politiche di mascheramento basate sulle autorizzazioni dell'utente che si connette. Se un utente mascherato esegue un dump, il file di backup contiene dati mascherati. `pg_columnmask`le politiche sono incluse nel dump come parte dello schema del database. Il ripristino riuscito richiede che tutti i ruoli di riferimento siano presenti nel database di destinazione e che sulla destinazione sia installata l'`pg_columnmask`estensione.

> **Note**
>
> A partire da PostgreSQL 18`pg_dump`, supporta l'opzione che esclude sia `–no-policies` la Row Level Security (RLS) che le policy di mascheramento dai dump del database. `pg_columnmask` Per ulteriori informazioni, consulta pg_dump.

Wrapper di dati esterni

Quando si utilizzano wrapper di dati esterni, i criteri di mascheramento sulle tabelle remote vengono applicati in base alle autorizzazioni dell'utente mappato sul server di origine, non alle autorizzazioni dell'utente che esegue le query locali e, sebbene sia possibile accedere ai dati remoti mascherati tramite FDW, non è possibile creare politiche DDM o RLS direttamente su tabelle esterne nel database locale.

Aggiornamento delle applicazioni per la connessione ai cluster DB Aurora PostgreSQL utilizzando nuovi certificati SSL/TLS

A partire dal 13 gennaio 2023, sono stati pubblicati da Amazon RDS nuovi certificati dell'autorità di certificazione (CA) per la connessione ai cluster database Aurora utilizzando Secure Socket Layer o

Transport Layer Security (SSL/TLS). Di seguito sono disponibili le informazioni sull'aggiornamento delle applicazioni per utilizzare i nuovi certificati.

Le informazioni in questo argomento possono aiutarti a stabilire se le applicazioni client utilizzano SSL/TLS per connettersi ai cluster DB. In caso affermativo, puoi determinare anche se le applicazioni richiedono la verifica del certificato per la connessione.

> **Note**
>
> Alcune applicazioni sono configurate per connettersi ai cluster DB Aurora PostgreSQL solo se sono in grado di verificare il certificato del server.
> Per queste applicazioni, è necessario aggiornare gli archivi di trust delle applicazioni client per includere i nuovi certificati CA.

Dopo aver aggiornato i certificati CA negli archivi attendibilità delle applicazioni client, puoi ruotare i certificati nei cluster DB. Consigliamo vivamente di testare queste procedure in un ambiente di sviluppo o di gestione temporanea prima di implementarle negli ambienti di produzione.

Per ulteriori informazioni sulla rotazione dei certificati, consulta Rotazione del certificato SSL/TLS. Per ulteriori informazioni sul download, consulta Utilizzo SSL/TLS per crittografare una connessione a un'. Per informazioni sull'utilizzo di SSL/TLS con i cluster DB PostgreSQL, consulta Sicurezza dei dati Aurora PostgreSQL con SSL/TLS.

Argomenti

- Determinare se un'applicazione si connette al cluster DB Aurora PostgreSQL utilizzando SSL
- Determinare se un client richiede la verifica del certificato per la connessione
- Aggiornare l'archivio di trust delle applicazioni
- Utilizzo delle connessioni SSL/TLS per diversi tipi di applicazioni

Determinare se un'applicazione si connette al cluster DB Aurora PostgreSQL utilizzando SSL

Controlla la configurazione del cluster DB per individuare il valore del parametro `rds.force_ssl`. Per impostazione predefinita, il parametro `rds.force_ssl` è impostato su `0` (disattivato). Se il parametro `rds.force_ssl` è impostato su `1` (attivato), i client devono utilizzare SSL/TLS per le

connessioni. Per ulteriori informazioni sui gruppi di parametri, consultare Gruppi di parametri per Amazon Aurora.

Se `rds.force_ssl` non è impostato su 1 (attivato), esegui una query sulla visualizzazione `pg_stat_ssl` per ottenere le connessioni che utilizzano SSL. Ad esempio, la query seguente restituisce solo le connessioni SSL e le informazioni sui client che utilizzano SSL.

```
select datname, usename, ssl, client_addr from pg_stat_ssl inner join pg_stat_activity
 on pg_stat_ssl.pid = pg_stat_activity.pid where ssl is true and usename<>'rdsadmin';
```

Vengono visualizzate solo le righe che utilizzano connessioni SSL/TLS con informazioni sulla connessione. Di seguito è riportato un output di esempio.

```
 datname  | usename | ssl | client_addr
----------+---------+-----+-------------
 benchdb  | pgadmin | t   | 53.95.6.13
 postgres | pgadmin | t   | 53.95.6.13
(2 rows)
```

La query precedente visualizza solo le connessioni in uso al momento della query. L'assenza di risultati non indica che nessuna applicazione stia utilizzando connessioni SSL. Altre connessioni SSL potrebbero essere stabilite in un momento diverso.

Determinare se un client richiede la verifica del certificato per la connessione

Quando un client, come psql o JDBC, è configurato con il supporto SSL, per impostazione predefinita tenta innanzitutto di connettersi al database con SSL. Se il client non riesce a connettersi con SSL, torna a connettersi senza SSL. La modalità `sslmode` predefinita utilizzata per i client basati su libpq (come psql) e JDBC è impostata su `prefer`. Il certificato del server viene verificato solo quando `sslrootcert` viene fornito con `sslmode` impostato su `verify-ca` o `verify-full`. Se il certificato non è valido viene generato un errore.

Utilizza `PGSSLROOTCERT` per verificare il certificato con la variabile di ambiente `PGSSLMODE`, con `PGSSLMODE` impostato su `verify-ca` o `verify-full`.

```
PGSSLMODE=verify-full PGSSLROOTCERT=/fullpath/ssl-cert.pem psql -h
 pgdbidentifier.cxxxxxxxx.us-east-2.rds.amazonaws.com -U primaryuser -d postgres
```

Usa l'argomento `sslrootcert` per verificare il certificato con `sslmode` nel formato stringa di connessione, con `sslmode` impostato su `verify-ca` o `verify-full`.

```
psql "host=pgdbidentifier.cxxxxxxxx.us-east-2.rds.amazonaws.com sslmode=verify-full
 sslrootcert=/full/path/ssl-cert.pem user=primaryuser dbname=postgres"
```

Ad esempio, nel caso precedente, se utilizzi un certificato root non valido, sul client viene visualizzato un errore simile al seguente.

```
psql: SSL error: certificate verify failed
```

Aggiornare l'archivio di trust delle applicazioni

Per informazioni sull'aggiornamento dell'archivio attendibilità per le applicazioni PostgreSQL, consulta l'argomento relativo alle connessioni TCP/IP sicure con SSL nella documentazione di PostgreSQL.

> **Note**
>
> Quando aggiorni l'archivio di trust puoi conservare i certificati meno recenti oltre ad aggiungere i nuovi certificati.

Aggiornare l'archivio di trust delle applicazioni per JDBC

Puoi aggiornare l'archivio di trust delle applicazioni che utilizzano JDBC per le connessioni SSL/TLS.

Per ulteriori informazioni sul download del certificato root, consulta Utilizzo SSL/TLS per crittografare una connessione a un'.

Per gli script di esempio che importano i certificati, consulta Script di esempio per l'importazione di certificati nel tuo archivio di trust.

Utilizzo delle connessioni SSL/TLS per diversi tipi di applicazioni

Di seguito vengono fornite le informazioni sull'utilizzo delle connessioni SSL/TLS per diversi tipi di applicazioni.

- psql

Il client viene invocato dalla riga di comando specificando le opzioni come stringa di connessione o variabili di ambiente. Le opzioni rilevanti per le connessioni SSL/TLS sono `sslmode` (variabile di ambiente `PGSSLMODE`), `sslrootcert` (variabile di ambiente `PGSSLROOTCERT`).

Per l'elenco completo delle opzioni, consulta l'argomento relativo alle parole chiave dei parametri nella documentazione di PostgreSQL. Per l'elenco completo delle variabili di ambiente, consulta l'argomento relativo alle variabili di ambiente nella documentazione di PostgreSQL.

- pgAdmin

  Questo client basato su browser fornisce un'interfaccia più intuitiva per la connessione a un database PostgreSQL.

  Per informazioni sulla configurazione delle connessioni, consulta la documentazione di pgAdmin.

- JDBC

  JDBC abilita le connessioni al database con le applicazioni Java.

  Per informazioni generali sulla connessione a un database PostgreSQL con JDBC, consulta l'argomento relativo alla connessione al database nella documentazione di PostgreSQL. Per informazioni sulla connessione con SSL/TLS, consulta l'argomento relativo alla configurazione del client nella documentazione di PostgreSQL.

- Python

  Una popolare libreria Python per la connessione ai database PostgreSQL è `psycopg2`.

  Per informazioni sull'utilizzo di `psycopg2`, consulta la documentazione psycopg2. Per un breve tutorial su come connettersi a un database PostgreSQL, consulta il tutorial di Psycopg2. Le informazioni sulle opzioni accettate dal comando di connessione sono disponibili nell'argomento relativo al contenuto del modulo Psycopg2.

> **Important**
>
> Dopo aver determinato che le connessioni al database utilizzano SSL/TLS e aver aggiornato l'archivio di trust delle applicazioni, puoi aggiornare il database per utilizzare i certificati rds-ca-rsa2048-g1. Per istruzioni, consulta la fase 3 in Aggiornamento del certificato CA modificando l'istanza database .

Utilizzo di Autenticazione Kerberos con Aurora PostgreSQL

Puoi utilizzare Kerberos per autenticare gli utenti quando si connettonoal cluster di database che esegue PostgreSQL. A tale scopo, configura l' del cluster DB da utilizzare AWS Directory Service for Microsoft Active Directory per l'autenticazione Kerberos. AWS Directory Service for Microsoft Active Directory viene anche chiamato. AWS Managed Microsoft ADÈ una funzionalità disponibile con Directory Service. Per ulteriori informazioni, vedi Cos'è Directory Service? nella Guida all'AWS Directory Service amministrazione.

Per iniziare, crea una AWS Managed Microsoft AD directory per memorizzare le credenziali dell'utente. Per il cluster di database PostgreSQL, specifica quindi il dominio di Active Directory e altre informazioni. Quando gli utenti eseguono l'autenticazione con il cluster database PostgreSQL, le richieste di autenticazione vengono inoltrate alla directory AWS Managed Microsoft AD .

Mantenere tutte le credenziali nella stessa directory consente di ridurre il tempo e l'impegno. È disponibile una posizione centralizzata per archiviare e gestire le credenziali per più cluster di database. L'uso di una directory può inoltre migliorare il profilo di sicurezza complessivo.

Puoi inoltre accedere alle credenziali da Microsoft Active Directory on-premise. A tale scopo, crea una relazione di dominio trusting in modo che la directory AWS Managed Microsoft AD consideri attendibile Microsoft Active Directory on-premise. In questo modo, gli utenti possono accedere ai cluster PostgreSQL con la stessa esperienza SSO (Single Sign-On) Windows dei carichi di lavoro nella rete locale.

Un database può utilizzare Kerberos AWS Identity and Access Management (IAM) o entrambe le procedure di autenticazione Kerberos e IAM. Tuttavia, poiché l'autenticazione Kerberos e IAM forniscono metodi di autenticazione diversi, un utente del database specifico può accedere a un database utilizzando solo uno o l'altro metodo di autenticazione, ma non entrambi. Per ulteriori informazioni sull’autenticazione IAM, consulta Autenticazione del database IAM .

> **Note**
>
> RDS per PostgreSQL non supporta l’autenticazione Kerberos per gruppi di Active Directory.

Argomenti

- Disponibilità di regioni e versioni
- Panoramica di Autenticazione Kerberos per cluster di database di PostgreSQL

- Configurazione dell'autenticazione Kerberos per cluster di database di PostgreSQL
- Gestione di un cluster di database Aurora PostgreSQL in un dominio Active Directory
- Connessione a PostgreSQL con Autenticazione Kerberos
- Utilizzo dei gruppi di sicurezza AD per il controllo degli accessi di Aurora PostgreSQL

Disponibilità di regioni e versioni

Il supporto varia a seconda delle versioni specifiche di ciascun motore di database e a seconda delle Regioni AWS. Per ulteriori informazioni sulla disponibilità di versioni e regioni di Aurora PostgreSQL con autenticazione Kerberos, consultare Autenticazione Kerberos con Aurora PostgreSQL.

Panoramica di Autenticazione Kerberos per cluster di database di PostgreSQL

Per configurare l'autenticazione Kerberos per un cluster di database di PostgreSQL, segui queste fasi, descritte dettagliatamente più avanti:

1. Utilizzare AWS Managed Microsoft AD per creare una AWS Managed Microsoft AD directory. È possibile utilizzare l' Console di gestione AWS AWS CLI, la o l' Directory Service API per creare la directory. Assicurati di aprire le porte in uscita rilevanti nel gruppo di sicurezza della directory in modo che la directory possa comunicare con l'del cluster.
2. Crea un ruolo che fornisca ad Amazon Aurora l'accesso ad Amazon per effettuare chiamate alla tua directory. AWS Managed Microsoft AD A tale scopo, crea un ruolo AWS Identity and Access Management (IAM) che utilizzi la policy IAM gestita. `AmazonRDSDirectoryServiceAccess`

   Affinché il ruolo IAM consenta l'accesso, l'endpoint AWS Security Token Service (AWS STS) deve essere attivato nella AWS regione corretta per il tuo AWS account. AWS STS Gli endpoint sono tutti Regioni AWS attivi per impostazione predefinita e puoi utilizzarli senza ulteriori azioni. Per ulteriori informazioni, consulta Attivazione e disattivazione AWS STS in una AWS regione nella Guida per l'utente IAM.
3. Crea e configura gli utenti nella AWS Managed Microsoft AD directory utilizzando gli strumenti di Microsoft Active Directory. Per ulteriori informazioni sulla creazione di utenti in Active Directory, consulta Gestire utenti e gruppi in AWS Managed Microsoft AD nella Guida all'Directory Service amministrazione.

4. Se prevedi di localizzare la directory e l'istanza DB in diversi AWS account o cloud privati virtuali (VPCs), configura il peering VPC. Per ulteriori informazioni, consulta Che cos'è il peering di VPC? nella Amazon VPC Peering Guide.
5. Creare o modificare un cluster di database di PostgreSQL dalla console, da CLI o dall'API di RDS utilizzando uno dei seguenti metodi:
   - Creazione e connessione di un cluster di database Aurora PostgreSQL
   - Modifica di un cluster database Amazon Aurora
   - Ripristino da uno snapshot cluster database
   - Ripristino di un cluster di database a un determinato momento

   Puoi localizzare l' del cluster nello stesso Amazon Virtual Private Cloud (VPC) della directory o in un account AWS o VPC diverso. Quando crei o modifichi l' cluster database PostgreSQL, completa le seguenti operazioni:
   - Specifica l'identificativo del dominio (identificativo `d-*`) generato al momento della creazione della directory.
   - Specifica anche il nome del ruolo IAM creato.
   - Assicurati che il gruppo di sicurezza dell'istanza database possa ricevere traffico in entrata dal gruppo di sicurezza della directory.
6. Utilizzare le credenziali dell'utente master RDS per connettersi al cluster di database di PostgreSQL. Creare l'utente in PostgreSQL per l'identificazione esterna. Gli utenti identificati esternamente possono accedere al cluster database di PostgreSQL con l'autenticazione Kerberos.

Configurazione dell'autenticazione Kerberos per cluster di database di PostgreSQL

Per configurare l'autenticazione Kerberos, completa la procedura seguente.

Argomenti

- Passaggio 1: creare una directory utilizzando AWS Managed Microsoft AD
- Fase 2: (facoltativa) creazione di una relazione di trust tra Active Directory on-premise e Directory Service
- Fase 3: creare un ruolo IAM per Amazon Aurora Amazon per accedere a Directory Service
- Fase 4: creazione e configurazione di utenti
- Fase 5: abilitazione del traffico tra VPC tra la directory e l'istanza database

- Fase 6: creazione o modifica di un cluster database PostgreSQL
- Fase 7: creazione di utenti PostgreSQL per i principali Kerberos
- Fase 8: configurazione di un client PostgreSQL

Passaggio 1: creare una directory utilizzando AWS Managed Microsoft AD

Directory Servicecrea una Active Directory completamente gestita nel AWS cloud. Quando crei una AWS Managed Microsoft AD directory, Directory Service crea due controller di dominio e server DNS per te. I server di directory vengono creati in sottoreti diverse in un VPC. Questa ridondanza assicura che la directory rimanga accessibile anche se si verifica un errore.

Quando si crea una AWS Managed Microsoft AD AWS directory, Directory Service esegue le seguenti attività per conto dell'utente:

- Configura una Active Directory all'interno del VPC.
- Crea un account amministratore della directory con il nome utente `Admin` e la password specificata. Puoi utilizzare questo account per gestire le directory.

  > **Important**
  >
  > Assicurati di salvare questa password. Directory Servicenon memorizza questa password e non può essere recuperata o reimpostata.

- Crea un gruppo di sicurezza per i controller della directory. Il gruppo di sicurezza deve consentire la comunicazione con il cluster database PostgreSQL.

All'avvioAWS Directory Service for Microsoft Active Directory, AWS crea un'unità organizzativa (OU) che contiene tutti gli oggetti della directory. Questa unità organizzativa, che ha lo stesso nome NetBIOS che hai immesso al momento della creazione della directory, si trova nella radice del dominio. La radice del dominio è di proprietà e gestita daAWS.

L'`Admin`account creato con la AWS Managed Microsoft AD directory dispone delle autorizzazioni per le attività amministrative più comuni dell'unità organizzativa:

- Creazione, aggiornamento o eliminazione di utenti
- Aggiungi risorse al dominio, come file server o server di stampa, e assegna le autorizzazioni per tali risorse a utenti dell'unità organizzativa

- Crea contenitori aggiuntivi OUs
- Delega dell'autorità
- Ripristino degli oggetti eliminati dal cestino di Active Directory
- Esegui i moduli Active Directory e Domain Name Service (DNS) per Windows PowerShell sul servizio Web Active Directory

L'account `Admin` dispone anche dei diritti per eseguire queste attività in tutto il dominio:

- gestione delle configurazioni DNS (aggiunta, eliminazione o aggiornamento di record, zone e server d'inoltro);
- visualizzazione di log di eventi DNS;
- visualizzazione di log di eventi di sicurezza.

Per creare una directory con AWS Managed Microsoft AD

1. Nel riquadro di navigazione della console Directory Service, scegliere Directory, quindi selezionare Configurazione della directory.
2. Scegli AWS Managed Microsoft AD. AWS Managed Microsoft ADè l'unica opzione attualmente supportata per l'uso con Amazon Aurora RDS.
3. Scegli Next (Successivo).
4. Nella pagina Enter directory information (Inserisci le informazioni sulla directory) inserisci le seguenti informazioni:

   Edizione

   Scegliere l'edizione più adatta alle proprie esigenze.

   Nome DNS directory

   Il nome completo della directory, ad esempio **`corp.example.com`**.

   Nome NetBIOS della directory

   Nome breve opzionale della directory, ad esempio `CORP`.

   Descrizione della directory

   Descrizione opzionale della directory.

Password amministratore

La password dell'amministratore della directory. Con il processo di creazione della directory viene generato un account amministratore con nome utente `Admin` e questa password.

La password dell'amministratore della directory non può includere il termine "admin". La password distingue tra maiuscole e minuscole e la lunghezza deve essere compresa tra 8 e 64 caratteri. Deve anche contenere un carattere di almeno tre delle seguenti quattro categorie:

- Lettere minuscole (a–z)
- Lettere maiuscole (A–Z)
- Numeri (0–9)
- Caratteri non alfanumerici (~!@#$%^&*_-+=`|\(){}[]:;"'<>,.?/)

Confirm password (Conferma password)

Digitare di nuovo la password dell'amministratore.

> **Important**
>
> Assicurati di salvare questa password. Directory Servicenon memorizza questa password e non può essere recuperata o reimpostata.

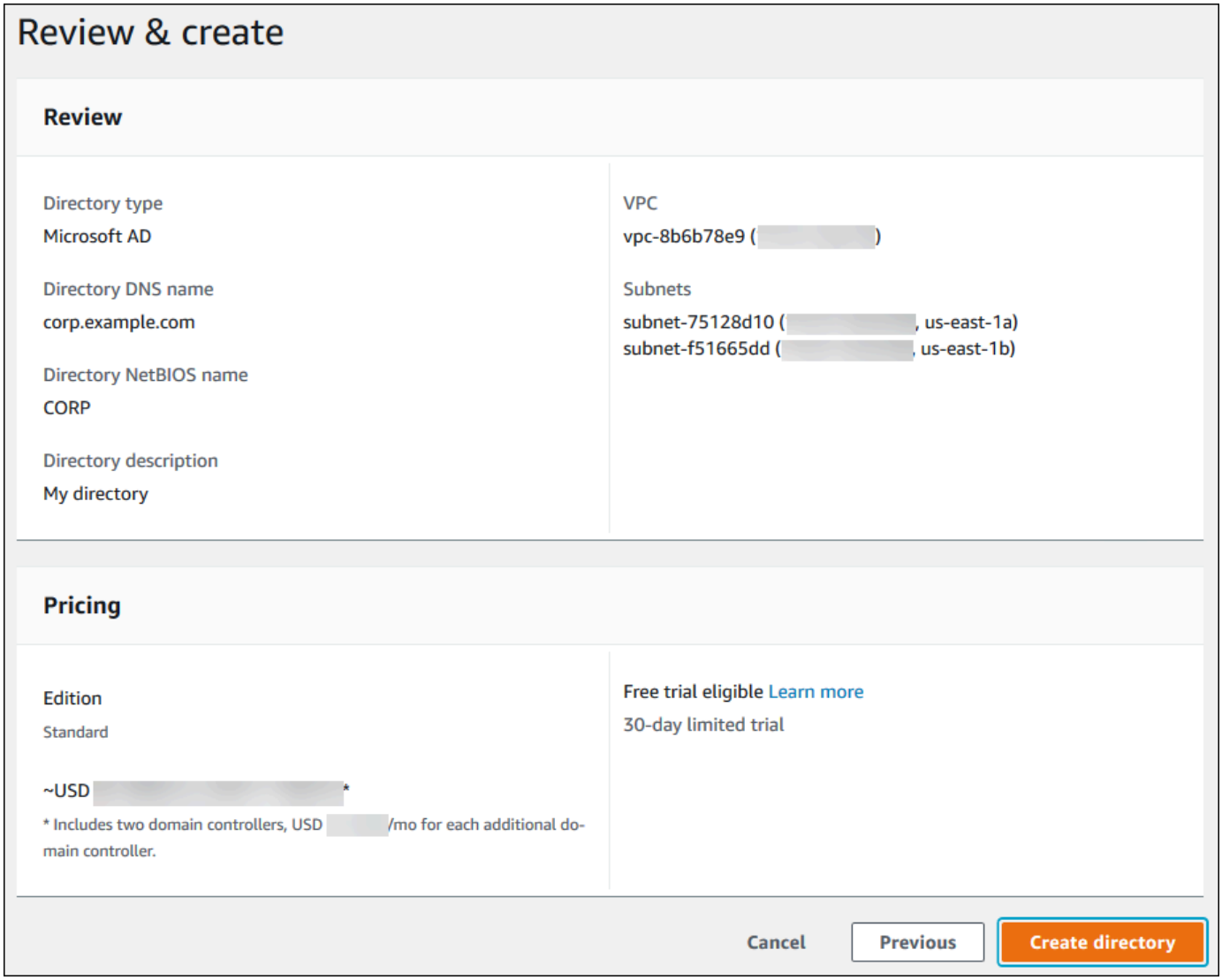
5. Scegli Next (Successivo).
6. Nella pagina Choose VPC and subnets (Scegli VPC e sottoreti) fornire le seguenti informazioni:

   VPC

   Scegliere il VPC per la directory. È possibile creare il cluster di database di PostgreSQL in questo VPC o in uno diverso.

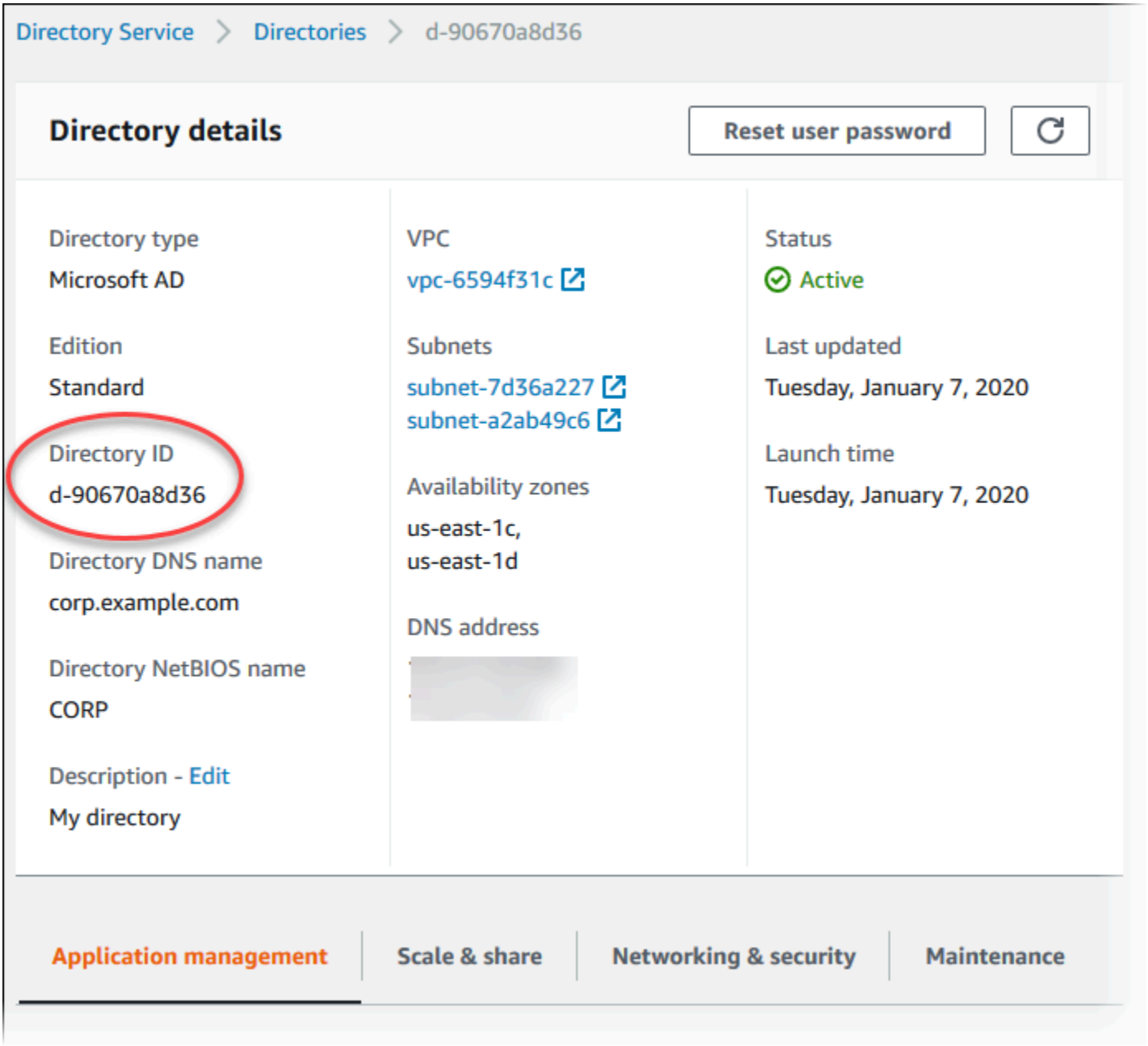
   Sottoreti

   Seleziona le sottoreti per i server di directory. Le due sottoreti devono trovarsi in diverse zone di disponibilità.
7. Seleziona Next (Successivo).
8. Verificare le informazioni della directory Se sono necessarie modifiche, seleziona Previous (Precedente) e apporta le modifiche. Quando le informazioni sono corrette, scegli Create Directory (Crea directory).



Per creare la directory sono necessari alcuni minuti. Una volta creata correttamente la directory, il valore Status (Stato) viene modificato in Active (Attivo).

Per consultare le informazioni sulla directory, selezionare l'ID della directory nell'elenco di directory. Prendere nota del valore Directory ID (ID directory). Questo valore è necessario per creare o modificare l'istanza database PostgreSQL.



Fase 2: (facoltativa) creazione di una relazione di trust tra Active Directory on-premise e Directory Service

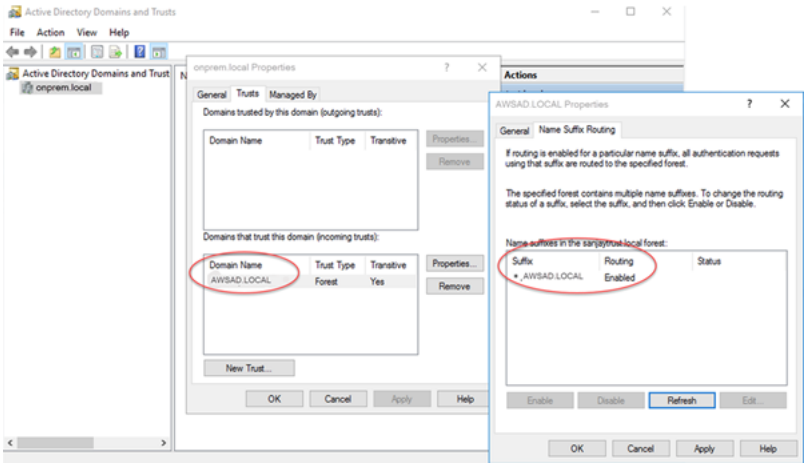
Se non prevedi di utilizzare Microsoft Active Directory locale, passa a Fase 3: creare un ruolo IAM per Amazon Aurora Amazon per accedere a Directory Service.

Per ottenere l'autenticazione Kerberos utilizzando l'Active Directory locale, è necessario creare una relazione di dominio affidabile utilizzando un trust di foresta tra Microsoft Active Directory locale e la AWS Managed Microsoft AD directory (creata in). Passaggio 1: creare una directory utilizzando AWS Managed Microsoft AD L'attendibilità può essere unidirezionale, in cui la AWS Managed

Microsoft AD directory considera attendibile Microsoft Active Directory locale. Il trust può anche essere bidirezionale, in cui entrambe le Active Directory si considerano reciprocamente attendibili. Per ulteriori informazioni sulla configurazione dei trust utilizzandoDirectory Service, vedere Quando creare una relazione di trust nella Administration Guide. AWS Directory Service

> **Note**
>
> Se utilizzi una Microsoft Active Directory locale:
>
> - I client Windows devono connettersi utilizzando il nome di dominio di nell'endpoint anziché rds.amazonaws.comDirectory Service. Per ulteriori informazioni, consulta Connessione a PostgreSQL con Autenticazione Kerberos.
> - I client Windows non possono connettersi con endpoint personalizzati Aurora. Per ulteriori informazioni, consulta Connessioni degli endpoint Amazon Aurora.
> - Per i database globali:
>   - I client Windows possono connettersi utilizzando endpoint di istanza o endpoint di cluster solo nel database primario del database globale. Regione AWS
>   - I client Windows non possono connettersi utilizzando gli endpoint del cluster in modalità secondaria. Regioni AWS

Assicurati che il nome di dominio di Microsoft Active Directory locale includa un routing del suffisso DNS che corrisponde alla nuova relazione di trust creata. Il risultato è mostrato nella screenshot seguente.



Fase 3: creare un ruolo IAM per Amazon Aurora Amazon per accedere a Directory Service

Affinché Amazon Aurora Amazon possa Directory Service chiamarti, il tuo AWS account deve avere un ruolo IAM che utilizzi la policy IAM gestita. `AmazonRDSDirectoryServiceAccess` Questo ruolo permette ad Amazon Aurora di effettuare chiamate ad Directory Service. (Tieni presente che questo ruolo IAM a cui accedere Directory Service è diverso dal ruolo IAM per Autenticazione del database IAM cui viene utilizzato.)

Quando crei un'istanza DB utilizzando l'account utente della console Console di gestione AWS e l'account utente della console dispone dell'`iam:CreateRole`autorizzazione, la console crea automaticamente il ruolo IAM necessario. In questo caso, il nome del ruolo è `rds-directoryservice-kerberos-access-role`. In caso contrario, è necessario creare manualmente il ruolo IAM. Quando crei questo ruolo IAM`Directory Service`, scegli e allega la policy AWS gestita `AmazonRDSDirectoryServiceAccess` ad esso.

Per ulteriori informazioni sulla creazione di ruoli IAM per un servizio, consulta Creating a role to delegate permissions to an AWS service nella IAM User Guide.

> **Note**
>
> Il ruolo IAM utilizzato per l'autenticazione Windows per RDS per Microsoft SQL Server non può essere utilizzato per Amazon Aurora.

Facoltativamente, puoi creare policy con le autorizzazioni richieste anziché utilizzare la policy `AmazonRDSDirectoryServiceAccess` gestita. In questo caso, il ruolo IAM deve avere la seguente policy di attendibilità IAM.

JSON

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Sid": "",
      "Effect": "Allow",
      "Principal": {
        "Service": [
```

```
          "directoryservice.rds.amazonaws.com",
          "rds.amazonaws.com"
        ]
      },
      "Action": "sts:AssumeRole"
    }
  ]
}
```

Il ruolo deve anche disporre della seguente policy del ruolo IAM.

JSON

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Action": [
        "ds:DescribeDirectories",
        "ds:AuthorizeApplication",
        "ds:UnauthorizeApplication",
        "ds:GetAuthorizedApplicationDetails"
      ],
    "Effect": "Allow",
    "Resource": "*"
    }
  ]
}
```

Per l'attivazioneRegioni AWS, utilizza i principali di servizio specifici della regione nelle policy di fiducia dei ruoli IAM. Quando si crea crei una policy di attendibilità per i servizi in queste Regioni, specificare il codice regionale nel principale di servizio.

L'esempio seguente mostra una policy di attendibilità che include i principali del servizio specifici della Regione:

JSON

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": [
          "directoryservice.rds.REGION-CODE.amazonaws.com",
          "rds.REGION-CODE.amazonaws.com"
        ]
      },
      "Action": "sts:AssumeRole"
    }
  ]
}
```

Sostituire REGION-CODE con il codice della Regione specifica. Ad esempio, utilizzare i seguenti principali del servizio per la Regione Asia Pacifico (Melbourne):

```
"Service": [
  "directoryservice.rds.ap-southeast-4.amazonaws.com",
  "rds.ap-southeast-4.amazonaws.com"
]
```

Fase 4: creazione e configurazione di utenti

Puoi creare gli utenti con lo strumento Utenti Active Directory e computer. Questo è uno degli strumenti di Active Directory Domain Services e Active Directory Lightweight Directory Services. Per ulteriori informazioni, consulta Add Users and Computers to the Active Directory domain (Aggiungere utenti e computer al dominio di Active Directory) nella documentazione di Microsoft. In questo caso, gli utenti sono individui o altre entità, ad esempio i computer, che fanno parte del dominio e le cui identità vengono conservate nella directory.

Per creare utenti in una Directory Service directory, devi essere connesso a un' EC2 istanza Amazon basata su Windows che fa parte della Directory Service directory. Allo stesso tempo, devi essere connesso come un utente che dispone di privilegi per creare utenti. Per ulteriori informazioni, consulta Creazione di un utente nella Guida di amministrazione di AWS Directory Service.

Fase 5: abilitazione del traffico tra VPC tra la directory e l'istanza database

Se prevedi di individuare la directory e il cluster di database nello stesso VPC, ignora questa fase e passa a Fase 6: creazione o modifica di un cluster database PostgreSQL.

Se prevedi di localizzare la directory e l'istanza DB in modo diverso VPCs, configura il traffico cross-VPC utilizzando il peering VPC o il Transit Gateway. AWS

La procedura seguente abilita il traffico tra un VPCs utilizzo e l'altro del peering VPC. Segui le istruzioni in Che cos'è il peering di VPC? nella Amazon Virtual Private Cloud Peering Guide.

Per abilitare il traffico tra VPC utilizzando il peering di VPC

1. Configurare le regole di routing VPC appropriate per garantire che il traffico di rete possa scorrere in entrambe le direzioni.
2. Assicurati che il gruppo di sicurezza dell'istanza database possa ricevere traffico in entrata dal gruppo di sicurezza della directory.
3. Assicurati che non sia presente una regola della lista di controllo accessi (ACL) di rete per bloccare il traffico.

Se la directory è di proprietà di un altro AWS account, è necessario condividerla.

Per condividere la cartella tra AWS account

1. Inizia a condividere la directory con l'AWSaccount in cui verrà creata l'istanza DB seguendo le istruzioni in Tutorial: Sharing your AWS Managed Microsoft AD directory for seamless EC2 Domain-join nella Administration Guide. Directory Service
2. Accedi alla Directory Service console utilizzando l'account per l'istanza DB e assicurati che il dominio abbia lo stato prima di procedere. `SHARED`
3. Dopo aver effettuato l'accesso alla Directory Service console utilizzando l'account per l'istanza DB, annota il valore Directory ID. Utilizzare questo ID directory per aggiungere l'istanza database al dominio.

Fase 6: creazione o modifica di un cluster database PostgreSQL

Crea o modifica un cluster di database di PostgreSQL da usare con la directory. Puoi utilizzare la console, CLI o l'API di RDS per associare un cluster di database a una directory. Questa operazione può essere eseguita in uno dei seguenti modi:

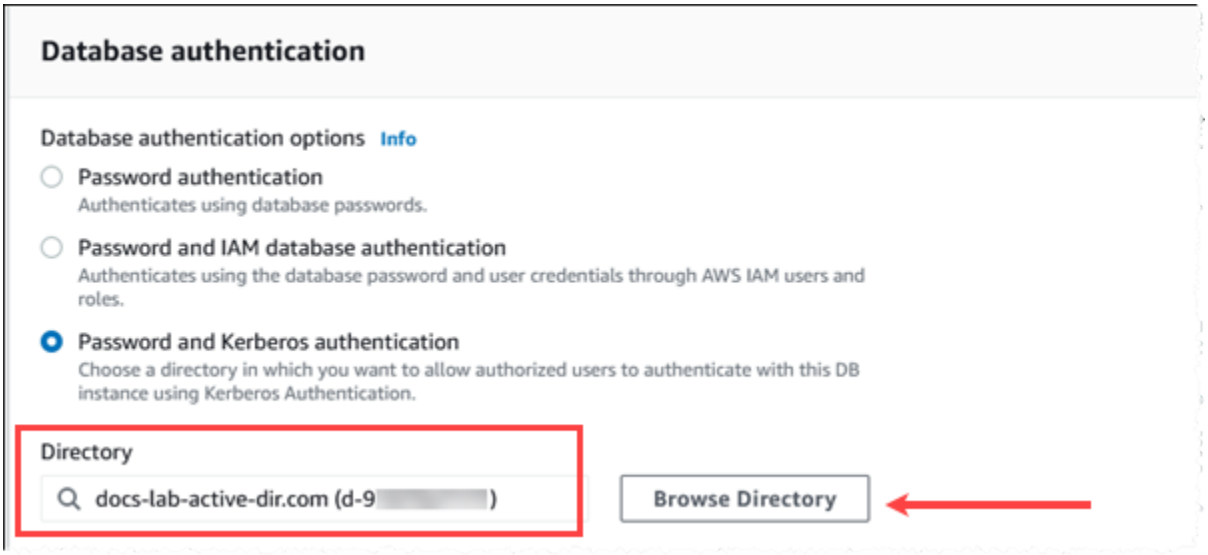
- Crea un nuovo cluster DB PostgreSQL utilizzando la console, il comando create-db-clusterCLI o l'operazione Create RDS API. DBCluster Per istruzioni, consulta Creazione e connessione di un cluster di database Aurora PostgreSQL.
- Modifica un cluster PostgreSQL DB esistente utilizzando la console, il comando modify-db-clusterCLI o l'operazione Modify RDS API. DBCluster Per istruzioni, consulta Modifica di un cluster database Amazon Aurora.
- Ripristina un cluster DB PostgreSQL da un'istantanea DB utilizzando la console, il comando CLI restore-db-cluster-from-db-snapshot o l'operazione dell'API Restore From RDS. DBCluster DBSnapshot Per istruzioni, consulta Ripristino da uno snapshot cluster database.
- Ripristina un cluster DB PostgreSQL utilizzando la console, il comando restore-db-instance-to-point-in-time CLI o l'operazione Restore RDS API. point-in-time DBCluster ToPointInTime Per istruzioni, consulta Ripristino di un cluster di database a un determinato momento.

L'autenticazione Kerberos è supportata solo per cluster di PostgreSQL DB in un VPC. Il cluster di database può trovarsi nello stesso VPC della directory o in un VPC diverso. Il cluster database deve utilizzare un gruppo di sicurezza che accetta traffico in ingresso e in uscita all'interno del VPC della directory, in modo che il cluster database possa comunicare con la directory.

> **Note**
>
> L'abilitazione dell'autenticazione Kerberos non è attualmente supportata sul cluster di database Aurora PostgreSQL durante la migrazione da RDS per PostgreSQL. È possibile abilitare l'autenticazione Kerberos solo su un cluster di database Aurora PostgreSQL autonomo.

Console

Quando utilizzi la console per creare, modificare o ripristinare un cluster di database, scegli Autenticazione Kerberos nella sezione Autenticazione database. Quindi scegli Sfoglia directory. Seleziona la directory o seleziona Crea una nuova directory per utilizzare il servizio directory.



AWS CLI

Quando si utilizza ilAWS CLI, sono necessari i seguenti parametri affinché l' del cluster DB possa utilizzare la directory creata:

- Per il parametro `--domain`, utilizza l'identificatore di dominio (identificatore "d-*") generato durante la creazione della directory.
- Per il parametro `--domain-iam-role-name`, utilizza il ruolo creato che utilizza la policy IAM gestita `AmazonRDSDirectoryServiceAccess`.

Ad esempio, il comando CLI seguente modifica un cluster di database per utilizzare una directory.

```
aws rds  modify-db-cluster --db-cluster-identifier mydbinstance --domain d-Directory-ID
 --domain-iam-role-name role-name
```

> **Important**
>
> Se modifichi un cluster database per abilitare l'autenticazione Kerberos, riavvia il cluster database dopo aver apportato la modifica.

Fase 7: creazione di utenti PostgreSQL per i principali Kerberos

A questo punto, il cluster di database Aurora PostgreSQL viene aggiunto al dominio AWS Managed Microsoft AD. Gli utenti che hai creato nella directory in Fase 4: creazione e configurazione di utenti

devono essere impostati come utenti del database PostgreSQL e devono essere concessi loro i privilegi per accedere al database. Puoi farlo accedendo come utente del database con privilegi `rds_superuser`. Ad esempio, se hai accettato i valori predefiniti quando hai creato il cluster database Aurora PostgreSQL, utilizzi `postgres`, come illustrato nei passaggi seguenti.

Per creare utenti del database PostgreSQL per i principali Kerberos

1. Usa `psql` per la connessione all'endpoint dell'istanza database del cluster database Aurora PostgreSQL utilizzando `psql`. L'esempio seguente utilizza l'account `postgres` predefinito per il ruolo `rds_superuser`.

   ```
   psql --host=cluster-instance-1.111122223333.aws-region.rds.amazonaws.com --
   port=5432 --username=postgres --password
   ```

2. Crea un nome utente del database per ogni principale Kerberos (nome utente di Active Directory) a cui desideri fornire l'accesso al database. Utilizza il nome utente canonico (identità) come definito nell'istanza Active Directory, ovvero un `alias` minuscolo (nome utente in Active Directory) e il nome maiuscolo del dominio Active Directory per quel nome utente. Il nome utente di Active Directory è un utente autenticato esternamente, quindi usa le virgolette intorno al nome come mostrato di seguito.

   ```
   postgres=> CREATE USER "username@CORP.EXAMPLE.COM" WITH LOGIN;
   CREATE ROLE
   ```

3. Autorizza il ruolo `rds_ad` per l'utente del database.

   ```
   postgres=> GRANT rds_ad TO "username@CORP.EXAMPLE.COM";
   GRANT ROLE
   ```

Dopo aver completato la creazione di tutti gli utenti PostgreSQL per le identità utente Active Directory, gli utenti possono accedere al cluster database Aurora PostgreSQL utilizzando le proprie credenziali Kerberos.

È necessario che gli utenti del database che eseguono l'autenticazione tramite Kerberos la effettuino attraverso computer client membri del dominio Active Directory.

Gli utenti del database a cui è stato concesso il ruolo `rds_ad` non possono avere anche il ruolo `rds_iam`. Questo vale anche per le appartenenze nidificate. Per ulteriori informazioni, consulta Autenticazione del database IAM .

Configurazione del cluster database Aurora PostgreSQL per nomi utente senza distinzione tra maiuscole e minuscole

Le versioni di Aurora PostgreSQL 14.5, 13.8, 12.12 e 11.17 supportano il parametro PostgreSQL `krb_caseins_users`. Questo parametro supporta i nomi utente di Active Directory senza distinzione tra maiuscole e minuscole. Per impostazione predefinita, questo parametro è impostato su false, quindi i nomi utente vengono interpretati applicando la distinzione tra maiuscole e minuscole da Aurora PostgreSQL. Questo è il comportamento predefinito in tutte le precedenti versioni di Aurora PostgreSQL. Tuttavia, puoi impostare questo parametro su `true` nel gruppo di parametri del·cluster database personalizzato e consentire al cluster database Aurora PostgreSQL di interpretare i nomi utente, senza distinzione tra maiuscole e minuscole. Questa operazione può essere utile per gli utenti del database, che a volte potrebbero digitare in modo errato le maiuscole e le minuscole del proprio nome utente durante l'autenticazione tramite Active Directory.

Per modificare il parametro `krb_caseins_users`, il cluster database Aurora PostgreSQL deve usare un gruppo di parametri del cluster database personalizzato. Per informazioni sull'utilizzo di un gruppo di parametri del·cluster database personalizzato, consulta Gruppi di parametri per Amazon Aurora.

È possibile utilizzare AWS CLI o the Console di gestione AWS per modificare l'impostazione. Per ulteriori informazioni, consulta Modifica dei parametri in un gruppo di parametri del cluster di database in Amazon Aurora.

Fase 8: configurazione di un client PostgreSQL

Per configurare un client PostgreSQL, procedi come indicato di seguito:

- Crea un file krb5.conf (o equivalente) che punti al dominio.
- Verifica che il traffico possa fluire tra l'host del client eDirectory Service. Utilizza un'utilità di rete come Netcat per le operazioni seguenti:
  - Verifica il traffico su DNS per la porta 53.
  - Verifica il traffico verso TCP/UDP la porta 53 e per Kerberos, che include le porte 88 e 464 per. Directory Service
- Verifica che il traffico scorra senza problemi tra l'host client e l'istanza database sulla porta del database. Ad esempio, utilizza psql per connetterti e accedere al database.

Di seguito è riportato un esempio di contenuto krb5.conf per. AWS Managed Microsoft AD

```
[libdefaults]
 default_realm = EXAMPLE.COM
[realms]
 EXAMPLE.COM = {
  kdc = example.com
  admin_server = example.com
 }
[domain_realm]
 .example.com = EXAMPLE.COM
 example.com = EXAMPLE.COM
```

Di seguito è riportato un esempio di contenuto krb5.conf per una Microsoft Active Directory locale.

```
[libdefaults]
 default_realm = EXAMPLE.COM
[realms]
 EXAMPLE.COM = {
  kdc = example.com
  admin_server = example.com
 }
 ONPREM.COM = {
  kdc = onprem.com
  admin_server = onprem.com
 }
[domain_realm]
 .example.com = EXAMPLE.COM
 example.com = EXAMPLE.COM
 .onprem.com = ONPREM.COM
 onprem.com = ONPREM.COM
 .rds.amazonaws.com = EXAMPLE.COM
 .amazonaws.com.cn = EXAMPLE.COM
 .amazon.com = EXAMPLE.COM
```

Gestione di un cluster di database Aurora PostgreSQL in un dominio Active Directory

È possibile utilizzare la console, l’interfaccia a riga di comando (CLI) o l’API RDS per gestire il cluster di database e la relativa relazione con Microsoft Active Directory. Ad esempio, puoi associare una Active Directory per abilitare l'autenticazione Kerberos. Puoi anche annullare l'associazione ad Active Directory per disabilitare l'autenticazione Kerberos. Puoi anche spostare un cluster di database affinché venga autenticata esternamente da una Microsoft Active Directory a un'altra.

Ad esempio, utilizzando la CLI, puoi effettuare quanto segue:

- Per provare ad abilitare di nuovo l'autenticazione Kerberos per un'appartenenza non riuscita, utilizza il comando CLI modify-db-cluster. Specifica l'ID della directory dell'appartenenza corrente per l'opzione `--domain`.
- Per disabilitare l'autenticazione Kerberos su un'istanza database, utilizza il comando CLI modify-db-cluster. Specifica `none` per l'opzione `--domain`.
- Per spostare un'istanza database da un dominio all'altro, utilizza il comando CLI modify-db-cluster. Specifica l'identificatore del nuovo dominio per l'opzione `--domain`.

Appartenenza al dominio

Dopo avere creato o modificato il cluster database, le istanze database diventano membri del dominio. Puoi visualizzare lo stato dell'appartenenza al dominio nella console o eseguendo il comando CLI describe-db-instances. Lo stato dell'istanza di database può essere uno dei seguenti:

- `kerberos-enabled`: l'autenticazione Kerberos è abilitata nell'istanza database.
- `enabling-kerberos`: AWS si trova nella fase di abilitazione dell'autenticazione Kerberos su questa istanza database.
- `pending-enable-kerberos`: l'abilitazione dell'autenticazione Kerberos è in corso su questa istanza database.
- `pending-maintenance-enable-kerberos`: AWS proverà ad abilitare l'autenticazione Kerberos sull'istanza database durante la prossima finestra di manutenzione pianificata.
- `pending-disable-kerberos`: la disabilitazione dell'autenticazione Kerberos è in corso su questa istanza database.
- `pending-maintenance-disable-kerberos`: AWS proverà a disabilitare l'autenticazione Kerberos sull'istanza database durante la prossima finestra di manutenzione pianificata.
- `enable-kerberos-failed`: un problema di configurazione ha impedito ad AWS di abilitare l'autenticazione Kerberos sull'istanza database. Correggi il problema di configurazione prima di inviare nuovamente il comando per modificare l'istanza database.
- `disabling-kerberos`: AWS si trova nella fase di disabilitazione dell'autenticazione Kerberos su questa istanza database.

Una richiesta per abilitare l'autenticazione Kerberos potrebbe non andare a buon fine a causa di un problema di connettività di rete o un ruolo IAM non corretto. In alcuni casi, il tentativo di abilitare

l'autenticazione Kerberos potrebbe non riuscire quando crei o modifichi un cluster di database. In questo caso, verifica di utilizzare il ruolo IAM corretto, quindi modifica il cluster di database per l'aggiunta al dominio.

Connessione a PostgreSQL con Autenticazione Kerberos

Puoi connetterti a PostgreSQL con l'autenticazione Kerberos tramite l'interfaccia pgAdmin o un'interfaccia a riga di comando come psql. Per ulteriori informazioni sulla connessione, consulta Connessione a un cluster di database Amazon Aurora PostgreSQL. Per informazioni su come ottenere l'endpoint, il numero di porta e altri dettagli necessari per la connessione, consulta Visualizzazione degli endpoint per un cluster Aurora.

> **Note**
>
> L'autenticazione e la crittografia GSSAPI in PostgreSQL sono implementate dalla libreria Kerberos `libkrb5.so`. Funzionalità come `postgres_fdw` e `dblink` si basano anche su questa stessa libreria per le connessioni in uscita con autenticazione o crittografia Kerberos.

pgAdmin

Per utilizzare pgAdmin per connetterti a PostgreSQL con l'autenticazione Kerberos, completa la procedura seguente:

1. Avviare l'applicazione pgAdmin sul computer client.
2. Nella scheda Dashboard (Pannello di controllo) selezionare Add New Server (Aggiungi nuovo server).
3. Nella finestra di dialogo Crea - Server, immettere un nome nella scheda Generale per identificare il server in pgAdmin.
4. Nella scheda Connection (Connessione), immetti le informazioni seguenti relative al database Aurora PostgreSQL.
   - In Host, immetti l'endpoint per l'istanza di scrittura del cluster database Aurora PostgreSQL. Un endpoint è simile al seguente:

     ```
     AUR-cluster-instance.111122223333.aws-region.rds.amazonaws.com
     ```

     Per connetterti a Microsoft Active Directory on-premise da un client Windows, usa il nome di dominio di Active Directory gestito da AWS anziché `rds.amazonaws.com` nell'endpoint host.

Si supponga, ad esempio, che il nome di dominio per la Active Directory gestita da AWS sia `corp.example.com`. In Host, l'endpoint viene quindi specificato come segue:

```
AUR-cluster-instance.111122223333.aws-region.corp.example.com
```

- Per Porta, immettere la porta assegnata.
- In Database di manutenzione immettere il nome del database iniziale a cui si connetterà il client.
- In Nome utente, immettere il nome utente immesso per l'autenticazione Kerberos in Fase 7: creazione di utenti PostgreSQL per i principali Kerberos .

5. Scegli Save (Salva).

Psql

Per utilizzare psql per connetterti a PostgreSQL con l'autenticazione Kerberos, completare la procedura seguente:

1. Al prompt dei comandi, eseguire questo comando.

```
kinit username
```

Sostituire *username* con il nome utente. Al prompt, immettere la password per l'utente memorizzata in Microsoft Active Directory.

2. Se il cluster database PostgreSQL utilizza un VPC accessibile pubblicamente, inserire un indirizzo IP per l'endpoint del cluster database nel file `/etc/hosts` nelclient EC2. Ad esempio, i comandi seguenti ottengono l'indirizzo IP e lo inseriscono nel file `/etc/hosts`.

```
% dig +short PostgreSQL-endpoint.AWS-Region.rds.amazonaws.com
;; Truncated, retrying in TCP mode.
ec2-34-210-197-118.AWS-Region.compute.amazonaws.com.
34.210.197.118

% echo " 34.210.197.118  PostgreSQL-endpoint.AWS-Region.rds.amazonaws.com" >> /etc/
hosts
```

Se utilizzi una Microsoft Active Directory locale da un client Windows, dovrai connetterti utilizzando un endpoint speciale. Anziché utilizzare il dominio Amazon `rds.amazonaws.com` nell'endpoint host, utilizzare il nome di dominio della Active Directory gestita da AWS.

Si supponga, ad esempio, che il nome di dominio per la Active Directory gestita da AWS sia `corp.example.com`. Quindi utilizzare il formato *`PostgreSQL-endpoint.AWS-Region.corp.example.com`* per l'endpoint e inserirlo nel file `/etc/hosts`.

```
% echo " 34.210.197.118  PostgreSQL-endpoint.AWS-Region.corp.example.com" >> /etc/
hosts
```

3. Utilizzare il comando psql seguente per accedere a un cluster di database PostgreSQL integrata con Active Directory. Utilizzare un cluster o un endpoint dell'istanza.

```
psql -U username@CORP.EXAMPLE.COM -p 5432 -h PostgreSQL-endpoint.AWS-
Region.rds.amazonaws.com postgres
```

Per accedere al cluster di database PostgreSQL da un client Windows utilizzando una Active Directory locale, utilizzare il comando psql seguente con il nome di dominio del passaggio precedente (`corp.example.com`):

```
psql -U username@CORP.EXAMPLE.COM -p 5432 -h PostgreSQL-endpoint.AWS-
Region.corp.example.com postgres
```

Utilizzo dei gruppi di sicurezza AD per il controllo degli accessi di Aurora PostgreSQL

A partire dalle versioni 14.10 e 15.5 di Aurora PostgreSQL, il controllo degli accessi di Aurora PostgreSQL può essere gestito utilizzando AWS Directory Service per i gruppi di sicurezza Microsoft Active Directory (AD). Le versioni precedenti di Aurora PostgreSQL supportano l'autenticazione basata su Kerberos con AD solo per singoli utenti. Ogni utente AD doveva essere assegnato in modo esplicito al cluster di database per ottenere l'accesso.

Invece di effettuare in modo esplicito il provisioning di ciascun utente AD al cluster di database in base alle esigenze aziendali, è possibile sfruttare i gruppi di sicurezza AD come spiegato di seguito:

- Gli utenti AD sono membri di vari gruppi di sicurezza AD in Active Directory. Questi non sono dettati dall'amministratore del cluster di database, ma si basano su requisiti aziendali e sono gestiti da un amministratore AD.
- Gli amministratori dei cluster di database creano ruoli di database nelle istanze database in base ai requisiti aziendali. Questi ruoli di database possono avere autorizzazioni o privilegi diversi.
- Gli amministratori dei cluster di database configurano una mappatura dai gruppi di sicurezza AD ai ruoli di database per ogni cluster di database.
- Gli utenti di database possono accedere ai cluster di database utilizzando le proprie credenziali AD. L'accesso si basa sull'appartenenza al gruppo di sicurezza AD. Gli utenti AD ottengono o perdono automaticamente l'accesso in base all'appartenenza ai gruppi AD.

Prerequisiti

Assicurarsi di disporre di quanto segue prima di configurare l'estensione per i gruppi di sicurezza AD:

- Configurazione dell'autenticazione Kerberos per cluster di database di PostgreSQL. Per ulteriori informazioni, consulta Configurazione dell'autenticazione Kerberos per cluster di database di PostgreSQL.

> **Note**
>
> Per i gruppi di sicurezza AD, saltare il passaggio 7: Creare utenti PostgreSQL per i principali Kerberos in questa procedura di configurazione.

> **Important**
>
> Se si abilitano i gruppi di sicurezza AD su un cluster Aurora PostgreSQL in cui è già abilitata l'autenticazione Kerberos, potrebbero verificarsi problemi di autenticazione. Ciò si verifica quando si aggiunge `pg_ad_mapping` al parametro `shared_preload_libraries` e si riavvia il database. Quando si utilizzano gli endpoint del cluster, i tentativi di accesso con un utente AD che non è un utente di database con il ruolo `rds_ad` possono avere esito negativo. Ciò può anche causare potenziali arresti anomali del motore. Per risolvere questo problema, disattivare e riattivare l'autenticazione Kerberos sul cluster. Questa soluzione alternativa è necessaria per le istanze esistenti ma non influisce sulle istanze create dopo aprile 2025.

- Gestione di un cluster di database in un dominio. Per ulteriori informazioni, consulta Gestione di un cluster di database in un dominio.

Configurazione dell'estensione pg_ad_mapping

Aurora PostgreSQL fornisce ora un'estensione `pg_ad_mapping` per gestire la mappatura tra gruppi di sicurezza AD e ruoli di database nel cluster Aurora PostgreSQL. Per ulteriori informazioni sulle funzioni fornite da `pg_ad_mapping`, consulta Utilizzo delle funzioni dall'estensione `pg_ad_mapping`.

Per configurare l'estensione `pg_ad_mapping` sul cluster di database Aurora PostgreSQL, aggiungere innanzitutto `pg_ad_mapping` alle librerie condivise nel gruppo di parametri del cluster di database personalizzato per il cluster di database Aurora PostgreSQL. Per informazioni sulla creazione di un gruppo di parametri personalizzato del cluster di database, consulta Gruppi di parametri per Amazon Aurora. Quindi, installare l'estensione `pg_ad_mapping`. Le procedure in questa sezione mostrano come fare. Puoi utilizzare la Console di gestione AWS o l'AWS CLI.

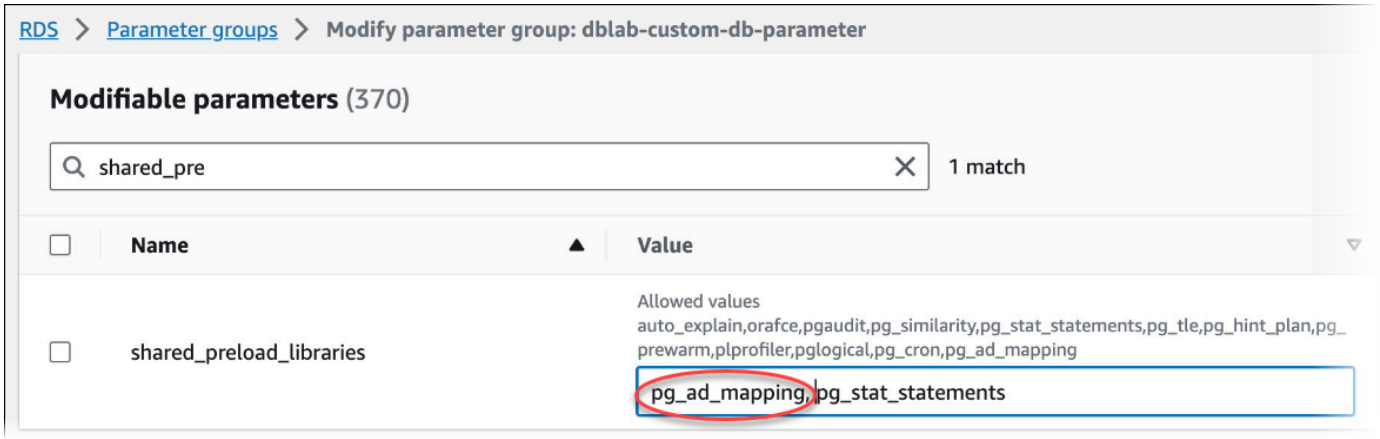
Per eseguire tutte queste attività, sono richieste autorizzazioni come il ruolo `rds_superuser`.

I passaggi seguenti si basano sull'ipotesi che il cluster di database Aurora PostgreSQL sia associato a un gruppo di parametri personalizzato del cluster di database.

Console

Per configurare l'estensione **`pg_ad_mapping`**

1. Accedi alla Console di gestione AWS e apri la console Amazon RDS all'indirizzo https://console.aws.amazon.com/rds/.
2. Nel riquadro di navigazione, scegliere l'istanza di scrittura del cluster di database Aurora PostgreSQL.
3. Aprire la scheda Configurazione per l'istanza di scrittura del cluster di database Aurora PostgreSQL. Tra i dettagli dell'istanza, individua il collegamento Gruppo di parametri.
4. Scegli il collegamento per aprire i parametri personalizzati associati al cluster database Aurora PostgreSQL.
5. Nel campo di ricerca Parametri, digita `shared_pre` per trovare il parametro `shared_preload_libraries`.
6. Scegli Edit parameters (Modifica parametri) per accedere ai valori delle proprietà.
7. Aggiungi `pg_ad_mapping` all'elenco nel campo Values (Valori). Utilizza una virgola per separare gli elementi nell'elenco di valori.



8. Riavviare l'istanza di scrittura del cluster di database Aurora PostgreSQL in modo da rendere effettiva la modifica al parametro `shared_preload_libraries`.
9. Quando l'istanza è disponibile, verifica che `pg_ad_mapping` sia stato inizializzato. Utilizzare `psql` per connettersi all'istanza di scrittura del cluster di database Aurora PostgreSQL, quindi eseguire il comando seguente.

```
SHOW shared_preload_libraries;
shared_preload_libraries
--------------------------
rdsutils,pg_ad_mapping
```

```
(1 row)
```

10. Con `pg_ad_mapping` inizializzato, puoi ora creare l'estensione. L'estensione deve essere creata dopo aver inizializzato la libreria per iniziare a utilizzare le funzioni fornite da questa estensione.

```
CREATE EXTENSION pg_ad_mapping;
```

11. Chiudi la sessione `psql`.

```
labdb=> \q
```

AWS CLI

Per configurare pg_ad_mapping

Per configurare pg_ad_mapping utilizzando AWS CLI, chiamare l'operazione modify-db-parameter-group per aggiungere questo parametro al gruppo di parametri personalizzato, come mostrato nella procedura seguente.

1. Utilizza il seguente comando AWS CLI per aggiungere `pg_ad_mapping` al parametro `shared_preload_libraries`.

```
aws rds modify-db-parameter-group \
   --db-parameter-group-name custom-param-group-name \
   --parameters
 "ParameterName=shared_preload_libraries,ParameterValue=pg_ad_mapping,ApplyMethod=pending-
reboot" \
   --region aws-region
```

2. Utilizzare il comando AWS CLI seguente per riavviare l'istanza di scrittura del cluster di database Aurora PostgreSQL in modo che pg_ad_mapping venga inizializzato.

```
aws rds reboot-db-instance \
    --db-instance-identifier writer-instance \
    --region aws-region
```

3. Quando l'istanza è disponibile, verifica che `pg_ad_mapping` sia stato inizializzato. Utilizzare `psql` per connettersi all'istanza di scrittura del cluster di database Aurora PostgreSQL, quindi eseguire il comando seguente.

```
SHOW shared_preload_libraries;
shared_preload_libraries
-------------------------
rdsutils,pg_ad_mapping
(1 row)
```

Con pg_ad_mapping inizializzato, puoi ora creare l'estensione.

```
CREATE EXTENSION pg_ad_mapping;
```

4. Chiudi la sessione `psql` in modo da poter utilizzare AWS CLI.

```
labdb=> \q
```

Recupero del SID del gruppo Active Directory in PowerShell

Un identificatore di sicurezza (SID) viene utilizzato per identificare in modo univoco un principale di sicurezza o un gruppo di sicurezza. Ogni volta che un gruppo di sicurezza o un account viene creato in Active Directory, gli viene assegnato un SID. Per recuperare il SID del gruppo di sicurezza AD da Active Directory, utilizzare il cmdlet Get-ADGroup in un computer client Windows che fa parte del dominio Active Directory. Il parametro Identity specifica il nome del gruppo Active Directory per ottenere il SID corrispondente.

L'esempio seguente restituisce il SID del gruppo AD *adgroup1*.

```
C:\Users\Admin> Get-ADGroup -Identity adgroup1 | select SID

             SID
-----------------------------------------------
S-1-5-21-3168537779-1985441202-1799118680-1612
```

Mappatura del ruolo di database con il gruppo di sicurezza AD

È necessario eseguire il provisioning in modo esplicito dei gruppi di sicurezza AD nel database come ruolo di database PostgreSQL. Un utente AD, che fa parte di almeno un gruppo di sicurezza AD di cui è stato eseguito il provisioning, avrà accesso al database. Non è consigliabile concedere

`rds_ad role` al ruolo di database basato sulla sicurezza del gruppo AD. L'autenticazione Kerberos per il gruppo di sicurezza verrà attivata utilizzando il suffisso del nome di dominio come *`user1@example.com`*. Questo ruolo di database non può utilizzare l'autenticazione tramite password o IAM per accedere al database.

> **Note**
>
> Gli utenti AD che hanno un ruolo di database corrispondente nel database a cui è stato concesso il ruolo `rds_ad` non possono accedere come parte del gruppo di sicurezza AD. Avranno accesso tramite il ruolo di database come utente individuale.

Ad esempio, accounts-group è un gruppo di sicurezza in AD in cui si desidera eseguire il provisioning di questo gruppo di sicurezza in Aurora PostgreSQL come accounts-role.

Gruppo di sicurezza AD	Ruolo di database di PostgreSQL
accounts-group	accounts-role

Quando si esegue la mappatura del ruolo di database con il gruppo di sicurezza AD, è necessario assicurarsi che il ruolo di database abbia l'attributo LOGIN impostato e che disponga del privilegio CONNECT sul database di accesso richiesto.

```
postgres => alter role accounts-role login;

ALTER ROLE
postgres => grant connect on database accounts-db to accounts-role;
```

L'amministratore può ora procedere alla creazione della mappatura tra il gruppo di sicurezza AD e il ruolo di database di PostgreSQL.

```
admin=>select pgadmap_set_mapping('accounts-group', 'accounts-role', <SID>, <Weight>);
```

Per informazioni sul recupero del SID del gruppo di sicurezza AD, consulta Recupero del SID del gruppo Active Directory in PowerShell.

Potrebbero verificarsi casi in cui un utente AD appartiene a più gruppi; in tal caso, l'utente AD erediterà i privilegi del ruolo di database, di cui è stato eseguito il provisioning con il peso più

elevato. Se i due ruoli hanno lo stesso peso, l'utente AD erediterà i privilegi del ruolo di database corrispondente alla mappatura aggiunta di recente. È consigliabile specificare pesi che riflettano le autorizzazioni/privilegi relativi dei singoli ruoli di database. Maggiore è il livello di autorizzazioni o privilegi di un ruolo di database, maggiore sarà il peso da associare alla voce di mappatura. Ciò eviterà l'ambiguità di due mappature aventi lo stesso peso.

La tabella seguente mostra un esempio di mappatura dai gruppi di sicurezza AD ai ruoli di database di Aurora PostgreSQL.

Gruppo di sicurezza AD	Ruolo di database di PostgreSQL	Weight
accounts-group	accounts-role	7
sales-group	sales-role	10
dev-group	dev-role	7

Nell'esempio seguente, `user1` erediterà i privilegi di sales-role poiché ha il peso maggiore, mentre `user2` erediterà i privilegi di `dev-role` poiché la mappatura per questo ruolo è stata creata dopo `accounts-role`, che ha lo stesso peso di `accounts-role`.

Username	Appartenenza al gruppo di sicurezza
user1	accounts-group sales-group
user2	accounts-group dev-group

I comandi psql per stabilire, elencare e cancellare le mappature sono mostrati di seguito. Al momento non è possibile modificare una singola voce di mappatura. La voce esistente deve essere eliminata e la mappatura deve essere ricreata.

```
admin=>select pgadmap_set_mapping('accounts-group', 'accounts-role', 'S-1-5-67-890',
 7);
admin=>select pgadmap_set_mapping('sales-group', 'sales-role', 'S-1-2-34-560', 10);
admin=>select pgadmap_set_mapping('dev-group', 'dev-role', 'S-1-8-43-612', 7);
```

```
admin=>select * from pgadmap_read_mapping();

ad_sid       | pg_role        | weight | ad_grp
-------------+----------------+--------+--------------
S-1-5-67-890 | accounts-role | 7      | accounts-group
S-1-2-34-560 | sales-role    | 10     | sales-group
S-1-8-43-612 | dev-role      | 7      | dev-group
(3 rows)
```

Registrazione di log/audit dell'identità dell'utente AD

Utilizzare il comando seguente per determinare il ruolo di database ereditato dall'utente corrente o di sessione:

```
postgres=>select session_user, current_user;

session_user | current_user
-------------+--------------
dev-role     | dev-role

(1 row)
```

Per determinare l'identità principale di sicurezza AD, utilizzare il seguente comando:

```
postgres=>select principal from pg_stat_gssapi where pid = pg_backend_pid();

 principal
-------------------------
 user1@example.com

(1 row)
```

Attualmente, l'identità dell'utente AD non è visibile nei log di audit. È possibile abilitare il parametro `log_connections` per registrare la creazione della sessione di database. Per ulteriori informazioni, consulta log_connections. L'output di ciò include l'identità dell'utente AD, come illustrato di seguito. Il

PID di backend associato a questo output può quindi aiutare ad attribuire le operazioni all'utente AD effettivo.

```
pgrole1@postgres:[615]:LOG: connection authorized: user=pgrole1
 database=postgres application_name=psql GSS (authenticated=yes, encrypted=yes,
 principal=Admin@EXAMPLE.COM)
```

Limitazioni

- Microsoft Entra ID noto come Azure Active Directory non è supportato.

Utilizzo delle funzioni dall'estensione **pg_ad_mapping**

L'estensione pg_ad_mapping ha fornito supporto alle seguenti funzioni:

pgadmap_set_mapping

Questa funzione stabilisce la mappatura tra il gruppo di sicurezza AD e il ruolo di database con un peso associato.

Sintassi

```
pgadmap_set_mapping(
ad_group,
db_role,
ad_group_sid,
weight)
```

Argomenti

Parametro	Descrizione
ad_group	Nome del gruppo AD. Il valore non può essere null o una stringa vuota.
db_role	Ruolo di database da mappare al gruppo AD specificato. Il valore non può essere null o una stringa vuota.
ad_group_sid	Identificatore di sicurezza utilizzato per identificare in modo univoco il gruppo AD. Il valore inizia con 'S-1-' e non può essere

Parametro	Descrizione
	null o una stringa vuota. Per ulteriori informazioni, consulta Recupero del SID del gruppo Active Directory in PowerShell.
weight	Peso associato al ruolo di database. Il ruolo con il peso più elevato ha la precedenza quando l'utente è membro di più gruppi. Il valore predefinito del peso è 1.

Tipo restituito

None

Note per l'utilizzo

Questa funzione aggiunge una nuova mappatura dal gruppo di sicurezza AD al ruolo di database. Può essere eseguito solo sull'istanza database primaria del cluster di database da un utente con il privilegio rds_superuser.

Esempi

```
postgres=> select pgadmap_set_mapping('accounts-group','accounts-
role','S-1-2-33-12345-67890-12345-678',10);

pgadmap_set_mapping

(1 row)
```

pgadmap_read_mapping

Questa funzione elenca le mappature tra il gruppo di sicurezza AD e il ruolo di database che sono state impostate utilizzando la funzione `pgadmap_set_mapping`.

Sintassi

```
pgadmap_read_mapping()
```

Argomenti

None

Tipo restituito

Parametro	Descrizione
ad_group_sid	Identificatore di sicurezza utilizzato per identificare in modo univoco il gruppo AD. Il valore inizia con 'S-1-' e non può essere null o una stringa vuota. Per ulteriori informazioni, consulta Recupero del SID del gruppo Active Directory in PowerShel l.accounts-role@example.com
db_role	Ruolo di database da mappare al gruppo AD specificato. Il valore non può essere null o una stringa vuota.
weight	Peso associato al ruolo di database. Il ruolo con il peso più elevato ha la precedenza quando l'utente è membro di più gruppi. Il valore predefinito del peso è 1.
ad_group	Nome del gruppo AD. Il valore non può essere null o una stringa vuota.

Note per l'utilizzo

Chiamare questa funzione per elencare tutte le mappature disponibili tra il gruppo di sicurezza AD e il ruolo di database.

Esempi

```
postgres=> select * from pgadmap_read_mapping();

ad_sid                              | pg_role       | weight | ad_grp
------------------------------------+---------------+--------+-----------------
S-1-2-33-12345-67890-12345-678      | accounts-role | 10     | accounts-group
(1 row)

(1 row)
```

pgadmap_reset_mapping

Questa funzione ripristina una o tutte le mappature impostate con la funzione `pgadmap_set_mapping`.

Sintassi

```
pgadmap_reset_mapping(
ad_group_sid,
db_role,
weight)
```

Argomenti

Parametro	Descrizione
ad_group_sid	Identificatore di sicurezza utilizzato per identificare in modo univoco il gruppo AD.
db_role	Ruolo di database da mappare al gruppo AD specificato.
weight	Peso associato al ruolo di database.

Se non vengono forniti argomenti, tutte le mappature dei gruppi AD ai ruoli di database vengono reimpostate. È necessario fornire tutti gli argomenti o nessuno.

Tipo restituito

None

Note per l'utilizzo

Chiamare questa funzione per eliminare uno specifico gruppo AD nella mappatura dei ruoli di database o per ripristinare tutte le mappature. Questa funzione può essere eseguita solo sull'istanza database primaria del cluster di database da un utente con il privilegio `rds_superuser`.

Esempi

```
postgres=> select * from pgadmap_read_mapping();

   ad_sid                        | pg_role      | weight      | ad_grp
---------------------------------+--------------+-------------+-------------------
 S-1-2-33-12345-67890-12345-678 | accounts-role| 10          | accounts-group
 S-1-2-33-12345-67890-12345-666 | sales-role   | 10          | sales-group

(2 rows)
```

```
postgres=> select pgadmap_reset_mapping('S-1-2-33-12345-67890-12345-678', 'accounts-
role', 10);

pgadmap_reset_mapping
(1 row)

postgres=> select * from pgadmap_read_mapping();

   ad_sid                        | pg_role      | weight      | ad_grp
---------------------------------+--------------+-------------+---------------
 S-1-2-33-12345-67890-12345-666 | sales-role   | 10          | sales-group

(1 row)
postgres=> select pgadmap_reset_mapping();

pgadmap_reset_mapping
(1 row)

postgres=> select * from pgadmap_read_mapping();

   ad_sid                        | pg_role      | weight      | ad_grp
---------------------------------+--------------+-------------+--------------
 (0 rows)
```

Migrazione di dati su Amazon Aurora con compatibilità PostgreSQL

Per la migrazione dei dati da un database esistente a un cluster DB Amazon Aurora edizione compatibile con PostgreSQL sono disponibili diverse opzioni. Le opzioni di migrazione dipendono anche dal database da cui esegui la migrazione e dalle dimensioni dei dati sottoposti a migrazione. Le opzioni sono:

Migrazione di un'istanza database RDS for PostgreSQL tramite uno snapshot

È possibile eseguire la migrazione dei dati direttamente da uno snapshot DB RDS for PostgreSQL a un cluster di database Aurora PostgreSQL.

Migrazione di un'istanza database RDS for PostgreSQL tramite una replica di lettura Aurora

È anche possibile eseguire la migrazione da un'istanza database RDS for PostgreSQL creando una replica di lettura Aurora PostgreSQL di un'istanza database RDS for PostgreSQL. Quando il ritardo di replica tra l'istanza database RDS for PostgreSQL e la replica di lettura Aurora

PostgreSQL è zero, puoi fermare la replica. A questo punto, è possibile trasformare la replica di lettura Aurora in un cluster DB Aurora PostgreSQL autonomo per la lettura e la scrittura.

Importazione di dati da Amazon S3 in Aurora PostgreSQL

È possibile eseguire la migrazione dei dati importandoli da Amazon S3 in una tabella appartenente a un cluster DB Aurora PostgreSQL.

Migrazione da un database non compatibile con PostgreSQL

Puoi utilizzare AWS Database Migration Service (AWS DMS) per eseguire la migrazione dei dati da un database non compatibile con PostgreSQL. Per ulteriori informazioni su AWS DMS, consulta Che cos'è AWS Database Migration Service? nella Guida per l'utente di AWS Database Migration Service.

> **Note**
>
> L'abilitazione dell'autenticazione Kerberos non è attualmente supportata sul cluster di database Aurora PostgreSQL durante la migrazione da RDS per PostgreSQL. È possibile abilitare l'autenticazione Kerberos solo su un cluster di database Aurora PostgreSQL autonomo.

Per un elenco di Regioni AWS in cui è disponibile Aurora, consulta Amazon Aurora in Riferimenti generali di AWS.

> **Important**
>
> Se si prevede di eseguire la migrazione di un'istanza database di RDS for PostgreSQL a un cluster di database Aurora PostgreSQL nel prossimo futuro, è consigliabile disattivare gli aggiornamenti automatici delle versioni secondarie per l'istanza database all'inizio della fase di pianificazione della migrazione. La migrazione a Aurora PostgreSQL potrebbe subire un ritardo se la versione RDS per PostgreSQL non è ancora supportata da Aurora PostgreSQL. Per informazioni sulle versioni di Aurora PostgreSQL, consulta Versioni del motore per Amazon Aurora PostgreSQL.

Migrazione di uno snapshot per una istanza database RDS for PostgreSQL a un cluster di database Aurora PostgreSQL

Per creare un cluster di database Aurora PostgreSQL, è possibile migrare uno snapshot DB di un'istanza database RDS for PostgreSQL. Il nuovo cluster di database Aurora PostgreSQL viene popolato con dati da un'istanza database RDS for PostgreSQL. Per informazioni sulla creazione di uno snapshot DB, consulta Creazione di uno snapshot DB.

In alcuni casi, lo snapshot DB potrebbe non trovarsi nella Regione AWS dove desideri posizionare i dati. In questo caso, utilizza la console Amazon RDS per copiare lo snapshot DB in quella Regione AWS. Per informazioni sulla copia di una snapshot DB, consulta Copia di una snapshot DB.

Puoi eseguire la migrazione di snapshot RDS for PostgreSQL compatibili con le versioni di Aurora PostgreSQL disponibili nella Regione AWS specificata. Ad esempio, è possibile eseguire la migrazione di uno snapshot da una istanza RDS for PostgreSQL 11.1 ad Aurora PostgreSQL versione 11.4, 11.7, 11.8 o 11.9 nella regione Stati Uniti occidentali (California settentrionale). È possibile eseguire la migrazione di uno snapshot RDS PostgreSQL 10.11 ad Aurora PostgreSQL 10.11, 10.12, 10.13 e 10.14. In altre parole, lo snapshot RDS for PostgreSQL deve utilizzare la stessa o una versione inferiore di Aurora PostgreSQL.

Puoi anche scegliere che il nuovo cluster di database Aurora PostgreSQL sia crittografato a riposo utilizzando una AWS KMS key. Questa opzione è disponibile solo per gli snapshot DB non crittografati.

Per eseguire la migrazione di uno snapshot DB RDS for PostgreSQL in un cluster di database di Aurora PostgreSQL, è possibile utilizzare la Console di gestione AWS, la AWS CLI o l'API RDS. Quando utilizzi la Console di gestione AWS, la console esegue le operazioni necessarie per creare sia il cluster di database che l'istanza primaria.

Console

Come eseguire la migrazione di uno snapshot DB PostgreSQL utilizzando la console RDS

1. Accedi alla Console di gestione AWS e apri la console Amazon RDS all'indirizzo https://console.aws.amazon.com/rds/.
2. SelezionareSnapshots (Snapshot).
3. Nella pagina Snapshot selezionare lo snapshot RDS for PostgreSQL che si desidera migrare in un cluster di database Aurora PostgreSQL.

4. Per Actions (Operazioni) scegliere Migrate snapshot (Migra snapshot).
5. Impostare i valori seguenti nella pagina Migrate database (Migra database):

   - DB engine version (Versione del motore database): scegli la versione del motore database che desideri utilizzare per la nuova istanza migrata.
   - DB instance identifier (Identificatore istanze DB): inserire un nome per il cluster di database univoco per l'account nella Regione AWS selezionata. Questo identificatore viene utilizzato negli indirizzi degli endpoint per le istanze nel cluster di database. Puoi scegliere di aggiungere informazioni utili al nome, ad esempio includendo la Regione AWS e il motore di database selezionato, come **`aurora-cluster1`**.

     L'identificatore istanze database presenta i seguenti vincoli:

     - Deve contenere da 1 –a 63 caratteri alfanumerici o trattini.
     - Il primo carattere deve essere una lettera.
     - Non può terminare con un trattino o contenere due trattini consecutivi.
     - Deve essere univoco per tutte le istanze database, per ogni account AWS e in ogni Regione AWS.
   - Classe di istanza database: scegli una classe dell'istanza database che abbia lo spazio di archiviazione e la capacità richiesti per il database, ad esempio `db.r6g.large`. I volumi dei cluster Aurora aumentano automaticamente quando aumenta la quantità di dati nel database. È quindi sufficiente scegliere una classe di istanza database che soddisfi i requisiti di storage correnti. Per ulteriori informazioni, consulta Panoramica dell'archiviazione di Amazon Aurora.
   - Virtual Private Cloud (VPC): se esiste già un VPC, è possibile usarlo con il cluster di database Aurora PostgreSQL scegliendo l'identificatore del VPC, ad esempio `vpc-a464d1c1`. Per informazioni sulla creazione di un VPC, consulta Tutorial: Creazione di un Amazon VPC da utilizzare con un cluster database (solo IPv4).

     In alternativa, è possibile consentire a Amazon RDS di creare un VPC scegliendo Create a new VPC (Crea un nuovo VPC).
   - DB subnet group (Gruppo di sottoreti DB): se un gruppo di sottoreti esiste già, è possibile utilizzarlo con il cluster di database Aurora PostgreSQL scegliendo l'identificatore del gruppo di sottoreti, ad esempio `gs-subnet-group1`.
   - Public access (Accesso pubblico): scegliere No per specificare che le istanze nel cluster di database sono accessibili solo dalle risorse all'interno del VPC. Scegliere Yes (Si) per specificare che le istanze nel cluster DB sono accessibili dalle risorse nella rete pubblica.

> **Note**
>
> Il cluster DB di produzione potrebbe non trovarsi in una sottorete pubblica, poiché solo i server applicativi richiedono l'accesso al cluster DB. Se il cluster di database non deve necessariamente trovarsi in una sottorete pubblica, impostare Public access (Accesso pubblico) su No.

- VPC security group (Gruppo di sicurezza VPC): scegliere un gruppo di sicurezza VPC per consentire l'accesso al database.
- Availability Zone (Zona di disponibilità): scegliere la zona di disponibilità per ospitare l'istanza primaria per il cluster DB Aurora PostgreSQL. Per fare in modo che Amazon RDS imposti automaticamente una zona di disponibilità, scegliere No Preference (Nessuna preferenza).
- Database port (Porta database): inserire la porta predefinita da usare quando si effettua la connessione a istanze nel cluster di database Aurora PostgreSQL. Il valore predefinito è `5432`.

> **Note**
>
> Potrebbe essere presente un firewall aziendale che non permette l'accesso alle porte predefinite, come la porta predefinita di PostgreSQL, 5432. In questo caso, fornire un valore di porta permesso dal firewall aziendale. Ricordare quel valore di porta più avanti, al momento della connessione al cluster DB Aurora PostgreSQL.

- Encryption (Crittografia): seleziona Enable Encryption (Abilita crittografia) affinché il nuovo cluster di database Aurora PostgreSQL venga crittografato mentre è inattivo. Scegliere anche una chiave KMS come valore AWS KMS key.
- Auto minor version upgrade (Aggiornamento automatico della versione secondaria): scegli Enable auto minor version upgrade (Abilita l'aggiornamento automatico della versione secondaria) per abilitare il cluster di database Aurora PostgreSQL a ricevere automaticamente gli aggiornamenti della versione secondaria del motore database PostgreSQL quando diventano disponibili.

  L'opzione Auto minor version upgrade (Aggiornamento automatico della versione secondaria) si applica solo agli aggiornamenti alle versioni secondarie del motore PostgreSQL per il cluster di database Aurora PostgreSQL. Non riguarda le patch normali applicate per mantenere la stabilità del sistema.

6. Selezionare Migrate (Migra) per effettuare la migrazione della snapshot DB.

7. Scegliere Databases (Database) per visualizzare il nuovo cluster di database. Scegliere il nuovo cluster di database per monitorare l'avanzamento della migrazione. Una volta completata la migrazione, lo stato del cluster è Disponibile. Nella scheda Connectivity & security (Connettività e sicurezza) è possibile trovare l'endpoint del cluster da utilizzare per la connessione all'istanza di scrittura principale del cluster di database. Per ulteriori informazioni sulla connessione a un cluster di database Aurora PostgreSQL, consulta Connessione a un cluster database Amazon Aurora.

AWS CLI

L'utilizzo di AWS CLI per migrare uno snapshot DB RDS for PostgreSQL in Aurora PostgreSQL implica due comandi AWS CLI diversi. Innanzitutto, viene utilizzato il comando `restore-db-cluster-from-snapshot` della AWS CLI per creare un nuovo cluster di database di Aurora PostgreSQL. Quindi viene utilizzato il comando `create-db-instance` per creare l'istanza database primaria nel nuovo cluster per completare la migrazione. La procedura seguente crea un cluster di database di Aurora PostgreSQL con istanza database primaria con la stessa configurazione dell'istanza database utilizzata per creare lo snapshot.

Per migrare uno snapshot DB RDS for PostgreSQL in un cluster di database di Aurora PostgreSQL

1. Utilizzare il comando describe-db-snapshots per ottenere informazioni sullo snapshot DB da migrare. È possibile specificare il parametro `--db-instance-identifier` o il parametro `--db-snapshot-identifier` nel comando. Se non si specifica uno di questi parametri, si ottengono tutti gli snapshot.

   ```
   aws rds describe-db-snapshots --db-instance-identifier <your-db-instance-name>
   ```

2. Il comando restituisce tutti i dettagli di configurazione per gli snapshot creati dall'istanza database specificata. Nell'output, trovare lo snapshot che si desidera migrare e individuare il relativo Amazon Resource Name (ARN). Per ulteriori informazioni sugli ARN di Amazon RDS, consultare Amazon Relational Database Service (Amazon RDS). Un ARN è simile al seguente output.

   ```
   “DBSnapshotArn": "arn:aws:rds:aws-region:111122223333:snapshot:<snapshot_name>"
   ```

   Inoltre, nell'output è possibile trovare i dettagli di configurazione per l'istanza database RDS for PostgreSQL, come la versione del motore, l'archiviazione allocata, se l'istanza database è crittografata o meno e così via.

3. Utilizzare il comando restore-db-cluster-from-snapshot per avviare la migrazione. Specifica i seguenti parametri:

   - `--db-cluster-identifier`: il nome da assegnare al cluster di database di Aurora PostgreSQL. Questo cluster di database di Aurora è la destinazione per la migrazione dello snapshot DB.
   - `--snapshot-identifier`: l'Amazon Resource Name (ARN) dello snapshot DB da migrare.
   - `--engine`: specificare `aurora-postgresql` per il cluster motore di database Aurora.
   - `--kms-key-id`: questo parametro facoltativo consente di creare un cluster di database di Aurora PostgreSQL crittografato da uno snapshot DB non crittografato. Consente inoltre di scegliere una chiave di crittografia diversa per il cluster di database rispetto alla chiave utilizzata per lo snapshot DB.

   > **Note**
   >
   > Non puoi creare un cluster di database di Aurora PostgreSQL non crittografato da uno snapshot DB crittografato.

   Senza il parametro `--kms-key-id` specificato come illustrato di seguito, il comando restore-db-cluster-from-snapshot della AWS CLI crea un cluster di database di Aurora PostgreSQL vuoto che viene crittografato utilizzando la stessa chiave dello snapshot DB o viene decrittografato se lo snapshot DB di origine non è crittografato.

   Per Linux, macOS o Unix:

   ```
   aws rds restore-db-cluster-from-snapshot \
       --db-cluster-identifier cluster-name \
       --snapshot-identifier arn:aws:rds:aws-region:111122223333:snapshot:your-
   snapshot-name \
       --engine aurora-postgresql
   ```

   Per Windows:

   ```
   aws rds restore-db-cluster-from-snapshot ^
       --db-cluster-identifier new_cluster ^
       --snapshot-identifier arn:aws:rds:aws-region:111122223333:snapshot:your-
   snapshot-name ^
   ```

```
    --engine aurora-postgresql
```

4. Il comando restituisce dettagli sul cluster di database di Aurora PostgreSQL creato per la migrazione. Puoi verificare lo stato del cluster di database di Aurora PostgreSQL utilizzando il comando describe-db-clusters della AWS CLI.

```
aws rds describe-db-clusters --db-cluster-identifier cluster-name
```

5. Quando il cluster di database diventa "disponibile", si utilizza il comando create-db-instance per popolare il cluster di database di Aurora PostgreSQL con l'istanza database basata sullo snapshot DB Amazon RDS. Specifica i seguenti parametri:

   - `--db-cluster-identifier`: il nome del nuovo cluster di database di Aurora PostgreSQL creato nel passaggio precedente.
   - `--db-instance-identifier`: il nome da assegnare all'istanza database. Questa istanza diventa il nodo primario del cluster di database di Aurora PostgreSQL.
   - `----db-instance-class` : specificare la classe di istanza database da utilizzare. Scegliere tra le classi di istanze database supportate dalla versione Aurora PostgreSQL a cui si sta eseguendo la migrazione. Per ulteriori informazioni, consultare Tipi di classi di istanza database e Motori di database supportati per classi di istanza database.
   - `--engine`: specificare `aurora-postgresql` per l'istanza database.

   È inoltre possibile creare l'istanza database con una configurazione diversa rispetto allo snapshot DB di origine, passando le opzioni appropriate nel comando `create-db-instance` della AWS CLI. Per ulteriori informazioni, vedi il comando create-db-instance.

   Per Linux, macOS o Unix:

```
aws rds create-db-instance \
    --db-cluster-identifier cluster-name \
    --db-instance-identifier --db-instance-class db.instance.class \
    --engine aurora-postgresql
```

   Per Windows:

```
aws rds create-db-instance ^
    --db-cluster-identifier cluster-name ^
    --db-instance-identifier --db-instance-class db.instance.class ^
```

```
    --engine aurora-postgresql
```

Al termine del processo di migrazione, il cluster Aurora PostgreSQL ha un'istanza database principale popolata.

Migrazione di dati da un'istanza database RDS for PostgreSQL a un cluster di database Aurora PostgreSQL tramite una replica di lettura Aurora

Puoi utilizzare un'istanza database RDS for PostgreSQL come base per un nuovo cluster di database Aurora PostgreSQL utilizzando una replica di lettura Aurora per il processo di migrazione. L'opzione di replica di lettura Aurora è disponibile solo per la migrazione all'interno dello stesso account Regione AWS ed è disponibile solo se la regione offre una versione compatibile di Aurora PostgreSQL per l'istanza DB RDS per PostgreSQL. Per compatibile si intende che la versione di Aurora PostgreSQL è la stessa utilizzata da RDS for PostgreSQL o che è una versione secondaria successiva della stessa famiglia di versioni principali.

Ad esempio, per utilizzare questa tecnica per migrare un'istanza database RDS for PostgreSQL 11.14, la regione deve offrire Aurora PostgreSQL versione 11.14 o una versione secondaria superiore nella famiglia di PostgreSQL versione 11.

Argomenti

- Panoramica della migrazione dei dati tramite una replica di lettura Aurora
- Preparazione alla migrazione dei dati tramite una replica di lettura Aurora
- Creazione di una replica di lettura Aurora
- Promozione di una replica di lettura Aurora

Panoramica della migrazione dei dati tramite una replica di lettura Aurora

La migrazione da un'istanza database RDS for PostgreSQL a un cluster di database Aurora PostgreSQL richiede una procedura in più fasi. In primo luogo, si crea una replica di lettura Aurora dell'istanza database RDS for PostgreSQL di origine. Questo avvia un processo di replica dall'istanza database RDS for PostgreSQL a un cluster di database per finalità speciali noto come cluster di replica. Il cluster di replica è costituito esclusivamente da una replica di lettura Aurora (un'istanza di lettura).

> **Note**
>
> Per completare la migrazione possono essere necessarie diverse ore per ogni terabyte di dati.

Promozione di una replica di Aurora PostgreSQL

Dopo aver creato un cluster di database Aurora PostgreSQL, seguire questi passaggi per promuovere la replica di Aurora:

1. Arrestare tutto il carico di lavoro di scrittura del database sull'istanza database RDS per PostgreSQL di origine.
2. Ottenere l'oggetto `WAL LSN` corrente dall'istanza database RDS per PostgreSQL di origine:

```
SELECT pg_current_wal_lsn();
pg_current_wal_lsn
--------------------
0/F0000318
(1 row)
```

3. Nel cluster della replica di Aurora PostgreSQL, verificare che l'LSN riprodotto sia maggiore dell'LSN del passaggio 2:

```
SELECT pg_last_wal_replay_lsn();
pg_last_wal_replay_lsn
------------------------
0/F0000400
(1 row)
```

   In alternativa, è possibile utilizzare la seguente query sull'istanza database RDS per PostgreSQL di origine:

```
SELECT restart_lsn FROM pg_replication_slots;
```

4. Promuovere il cluster di replica di Aurora PostgreSQL.

   Quando la replica si interrompe, il cluster di replica viene promosso a cluster di database Aurora PostgreSQL autonomo e l'istanza di lettura viene promossa a istanza di scrittura per il cluster. A questo punto, è possibile aggiungere le istanze al cluster di database Aurora PostgreSQL per dimensionarlo in base al caso d'uso. Se non è più necessaria l'istanza database RDS per PostgreSQL originale, è possibile eliminarla.

Non è possibile creare una replica di lettura Aurora se l'istanza database RDS for PostgreSQL ha già una replica di lettura Aurora o se ha una replica di lettura tra regioni.

Preparazione alla migrazione dei dati tramite una replica di lettura Aurora

> **Note**
>
> Quando si prepara la migrazione dei dati ad Aurora PostgreSQL, è importante identificare e gestire in modo appropriato le tabelle non registrate. Per ulteriori informazioni, consulta Gestione delle tabelle non registrate durante la migrazione.

Durante il processo di migrazione tramite la replica di lettura Aurora, gli aggiornamenti apportati all'istanza database RDS for PostgreSQL di origine vengono replicati in modo asincrono nella replica di lettura Aurora del cluster di replica. Il processo utilizza la funzionalità di replica dello streaming nativa di PostgreSQL che memorizza i segmenti WAL (write ahead log) sull'istanza di origine. Prima di iniziare questo processo di migrazione, assicurati che l'istanza abbia una capacità di archiviazione sufficiente controllando i valori per i parametri elencati nella tabella.

Metrica	Descrizione
`FreeStorageSpace`	Spazio di storage disponibile. Unità: byte
`OldestReplicationSlotLag`	Entità del ritardo per i dati WAL nella replica con il maggiore ritardo. Unità: megabyte
`RDSToAuroraPostgreSQLReplicaLag`	Periodo di tempo, in secondi, di ritardo di un cluster DB Aurora PostgreSQL rispetto all'istan za database RDS di origine.
`TransactionLogsDiskUsage`	Spazio su disco utilizzato dai log delle transazio ni. Unità: megabyte

Per ulteriori informazioni sul monitoraggio dell'istanza, consulta Monitoraggio nella Guida per l'utente di Amazon RDS.

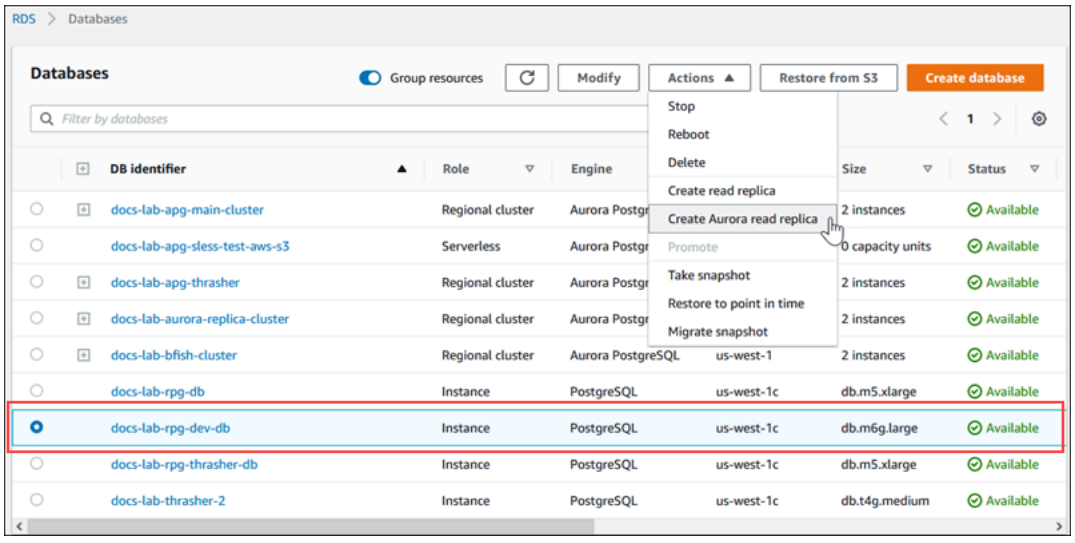
Creazione di una replica di lettura Aurora

È possibile creare una replica di lettura Aurora per un'istanza DB RDS per PostgreSQL utilizzando o il. Console di gestione AWS AWS CLI L'opzione per creare una replica di lettura di Aurora utilizzando Console di gestione AWS è disponibile solo se Regione AWS offre una versione Aurora PostgreSQL compatibile. Ovvero, è disponibile solo se è presente una versione di Aurora PostgreSQL che è la stessa della versione RDS for PostgreSQL o una versione secondaria superiore nella stessa famiglia di versioni principali.

Console

Per creare una replica di lettura Aurora da un'istanza database PostgreSQL di origine

1. Accedi a Console di gestione AWS e apri la console Amazon RDS all'indirizzo https://console.aws.amazon.com/rds/.
2. Nel pannello di navigazione, seleziona Database.
3. Scegliere l'istanza database RDS for PostgreSQL da usare come origine per la replica di lettura Aurora. Per Actions (Operazioni), scegli Create Aurora read replica (Crea replica di lettura Aurora). Se questa scelta non viene visualizzata, significa che una versione compatibile di Aurora PostgreSQL non è disponibile nella regione.



4. Nella pagina delle impostazioni per la creazione della replica di lettura Aurora è possibile configurare le proprietà per il cluster di database Aurora PostgreSQL come illustrato nella tabella riportata di seguito. Il cluster di database di replica viene creato da uno snapshot dell'istanza

database di origine utilizzando lo stesso nome utente e password “master” dell'origine, quindi non è possibile modificarli in questo momento.

Opzione	Descrizione
DB instance class (Classe istanza database)	Scegliere una classe di istanza database che soddisfi i requisiti di elaborazione e di memoria per l'istanza primaria nel cluster di database. Per ulteriori informazioni, consulta Classi di istanze database Amazon Aurora.
Multi-AZ deployment (Implementazione Multi-AZ)	Non disponibile durante la migrazione
DB instance identifier (Identificatore istanze DB)	Inserire il nome da assegnare all'istanza database. Questo identificatore viene utilizzato nell'indirizzo dell'endpoint per l'istanza primaria del nuovo cluster di database. L’identificatore istanze database presenta i seguenti vincoli: • Deve contenere da 1 –a 63 caratteri alfanumerici o trattini. • Il primo carattere deve essere una lettera. • Non può terminare con un trattino o contenere due trattini consecutivi. • Deve essere unica per tutte le istanze DB di ogni AWS account, per ciascuno. Regione AWS
Virtual Private Cloud (VPC)	Scegliere il VPC che ospita il cluster DB. Scegliere Create a new VPC (Crea un nuovo VPC) per fare in modo che Amazon RDS crei un VPC. Per ulteriori informazioni, consulta Prerequisiti per i cluster di database.

Opzione	Descrizione
DB subnet group (Gruppo di sottoreti DB)	Scegliere il gruppo di sottoreti di database da utilizzare per il cluster DB. Scegliere Create new DB subnet group (Crea nuovo gruppo di sottoreti DB) per fare in modo che Amazon RDS crei automaticamente un gruppo di sottoreti di database. Per ulteriori informazioni, consulta Prerequisiti per i cluster di database.
Public accessibility (Accesso pubblico)	Scegliere Sì per assegnare al cluster di database un indirizzo IP pubblico, altrimenti, selezionare No. Le istanze nel cluster di database possono essere una combinazione di istanze database pubbliche e private. Per ulteriori informazioni su come non consentire l'accesso pubblico per le istanze, consulta Nascondere cluster database in un VPC da Internet.
Availability zone (Zona di disponibilità)	Stabilire se si desidera specificare una zona di disponibilità particolare. Per ulteriori informazioni sulle zone di disponibilità, consultare Regioni e zone di disponibilità.
Gruppi di sicurezza VPC	Scegliere uno o più gruppi di sicurezza VPC per proteggere l'accesso di rete al cluster DB. Scegliere Create a new VPC security group (Crea nuovo gruppo di sicurezza VPC) per fare in modo che Amazon RDS crei un gruppo di sicurezza VPC. Per ulteriori informazioni, consulta Prerequisiti per i cluster di database.
Database port (Porta del database)	Specifica la porta per applicazioni e utilità da utilizzare per accedere al database. I cluster DB PostgreSQL Aurora utilizzano per impostazione predefinita la porta predefinita PostgreSQL, 5432. I firewall presso alcune aziende bloccano le connessioni a questa porta. Se il firewall dell'azienda blocca la porta predefinita, scegliere un'altra porta per il nuovo cluster di database.

Opzione	Descrizione
DB parameter group (Gruppo di parametri database)	Scegliere un gruppo di parametri del database per il cluster di database di Aurora PostgreSQL. Puoi utilizzare il gruppo parametri del database predefinito di Aurora oppure puoi creare il tuo gruppo parametri del database personalizzato. Per ulteriori informazioni sui gruppi di parametri database, consulta Gruppi di parametri per Amazon Aurora.
DB cluster parameter group (Gruppo di parametri del cluster di database)	Scegliere un gruppo di parametri del cluster di database per il cluster di database di Aurora PostgreSQL. Puoi utilizzare il gruppo di parametri del cluster di database predefinito di Aurora oppure puoi creare il tuo gruppo di parametri. Per ulteriori informazi oni sui gruppi di parametri del cluster di database, consulta Gruppi di parametri per Amazon Aurora.
Encryption (Crittografia)	Seleziona Enable Encryption (Abilita crittografia) affinché il nuovo cluster di database Aurora sia crittografato mentre è inattivo. Se si sceglie Enable encryption (Abilita crittografia), scegliere una chiave KMS come valore AWS KMS key.
Priority (Priorità)	Scegliere una priorità di failover per il cluster di database. Se non specifichi alcun valore, l'imposta zione predefinita è tier-1. Questa priorità determina l'ordine di promozione delle repliche di Aurora durante il recupero da un errore dell'istanza principale. Per ulteriori informazioni, consulta Tolleranza ai guasti di un cluster DB Aurora.
Backup retention period (Periodo di retention dei backup)	Scegliere l'intervallo di tempo, da 1 –a 35 giorni, nel quale Aurora conserverà le copie di backup del database. Le copie di Backup possono essere utilizzat e per il point-in-time ripristino (PITR) del database fino al secondo.

Opzione	Descrizione
Enhanced Monitoring (Monitoraggio avanzato)	Scegliere Enable enhanced monitoring (Abilita monitoraggio avanzato) per abilitare la raccolta di parametri in tempo reale per il sistema operativo su cui viene eseguito il cluster DB. Per ulteriori informazioni, consulta Monitoraggio dei parametri del sistema operativo con il monitoraggio avanzato.
Monitoring Role (Ruolo monitoraggio)	Disponibile solo se hai scelto Enable enhanced monitoring (Abilita monitoraggio avanzato). Il ruolo AWS Identity and Access Management (IAM) da utilizzare per il monitoraggio avanzato. Per ulteriori informazioni, consulta Configurare e abilitare Monitoraggio avanzato.
Granularity (Granularità)	Disponibile solo se hai scelto Enable enhanced monitoring (Abilita monitoraggio avanzato). Impostare l'intervallo, in secondi, tra le operazioni di raccolta dei parametri per il cluster DB.
Auto minor version upgrade (Aggiornamento automatico della versione secondaria)	Scegliere Yes (Sì) per abilitare il cluster DB Aurora PostgreSQL in modo che riceva automaticamente gli aggiornamenti delle versioni minori del motore di database PostgreSQL non appena diventano disponibili. L'opzione Auto minor version upgrade (Aggiornamento automatico della versione secondaria) si applica solo agli aggiornamenti alle versioni secondarie del motore PostgreSQL per il cluster di database Aurora PostgreSQL. Non riguarda le patch normali applicate per mantenere la stabilità del sistema.
Maintenance window (Finestra di manutenzione)	Scegliere l'intervallo temporale settimanale durante il quale può avvenire la manutenzione dei sistemi.

5. Scegli Create read replica (Crea replica di lettura).

AWS CLI

Per creare una replica di lettura Aurora da un'istanza DB RDS for PostgreSQL di origine utilizzando il AWS CLI, è innanzitutto necessario utilizzare il comando CLI create-db-clusterper creare un cluster Aurora DB vuoto. Una volta creato il cluster di database, dovrai creare l'istanza primaria per il cluster di database utilizzando il comando create-db-instance. L'istanza primaria è la prima istanza creata in un cluster di database Aurora. In questo caso, viene creata inizialmente come replica di lettura Aurora dell'istanza database RDS for PostgreSQL. Al termine del processo, la migrazione dell'istanza database RDS for PostgreSQL in un cluster di database Aurora PostgreSQL è completa.

Non è necessario specificare l'account utente principale (in genere, `postgres`), la password o il nome del database. La replica di lettura Aurora li ottiene automaticamente dall'istanza DB RDS di origine per PostgreSQL che identifichi quando richiami i comandi. AWS CLI

È necessario specificare la versione del motore da utilizzare per il cluster di database Aurora PostgreSQL e l'istanza database. La versione specificata deve corrispondere all'istanza database RDS for PostgreSQL di origine. Se l'istanza database RDS for PostgreSQL di origine è crittografata, è necessario specificare anche la crittografia per l'istanza primaria del cluster di database Aurora PostgreSQL. La migrazione di un'istanza crittografata in un cluster di database Aurora non crittografato non è supportata.

Gli esempi seguenti mostrano come creare un cluster di database Aurora PostgreSQL denominato `my-new-aurora-cluster`, che utilizza un'istanza database RDS di origine non crittografata. In primo luogo, crei il cluster di database Aurora PostgreSQL chiamando il comando CLI `create-db-cluster`. L'esempio mostra come utilizzare il parametro facoltativo `--storage-encrypted` per specificare che il cluster di database deve essere crittografato. Poiché il database di origine non è crittografato, è necessario specificare una chiave utilizzando `--kms-key-id`. Per ulteriori informazioni sui parametri obbligatori e facoltativi, vedi l'elenco dell'esempio seguente.

Per Linux, macOS o Unix:

```
aws rds create-db-cluster  \
    --db-cluster-identifier my-new-aurora-cluster \
    --db-subnet-group-name my-db-subnet
    --vpc-security-group-ids sg-11111111
    --engine aurora-postgresql \
    --engine-version same-as-your-rds-instance-version \
    --replication-source-identifier arn:aws:rds:aws-region:111122223333:db/rpg-source-
db \
    --storage-encrypted \
```

```
    --kms-key-id arn:aws:kms:aws-
region:111122223333:key/11111111-2222-3333-444444444444
```

Per Windows:

```
aws rds create-db-cluster  ^
    --db-cluster-identifier my-new-aurora-cluster ^
    --db-subnet-group-name my-db-subnet ^
    --vpc-security-group-ids sg-11111111 ^
    --engine aurora-postgresql ^
    --engine-version same-as-your-rds-instance-version ^
    --replication-source-identifier arn:aws:rds:aws-region:111122223333:db/rpg-source-
db ^
    --storage-encrypted ^
    --kms-key-id arn:aws:kms:aws-
region:111122223333:key/11111111-2222-3333-444444444444
```

Nell'elenco seguente è possibile trovare ulteriori informazioni su alcune delle opzioni mostrate nell'esempio. Se non diversamente specificato, questi parametri sono obbligatori.

- `--db-cluster-identifier`. Assegna un nome al nuovo cluster di database Aurora PostgreSQL.
- `--db-subnet-group-name`. Crea il cluster di database Aurora PostgreSQL nella stessa sottorete database dell'istanza database di origine.
- `--vpc-security-group-ids`. Specifica il gruppo di sicurezza per il cluster di database Aurora PostgreSQL.
- `--engine-version`. Specifica la versione da utilizzare per il cluster di database Aurora PostgreSQL. Questa dovrebbe essere uguale alla versione utilizzata dall'istanza database RDS per PostgreSQL di origine o una versione secondaria superiore.
- `--replication-source-identifier`. Identifica l'istanza database RDS for PostgreSQL utilizzando il suo Amazon Resource Name (ARN). Per ulteriori informazioni su Amazon RDS ARNs, consulta Amazon Relational Database Service (Amazon RDS) nel. Riferimenti generali di AWSdel tuo cluster DB.
- `--storage-encrypted` : Opzionale. Utilizzalo solo quando devi specificare la crittografia, come spiegato di seguito:
  - Utilizza questo parametro quando l'istanza database di origine ha un'archiviazione crittografata. Se non utilizzi questo parametro con un'istanza database di origine con archiviazione

crittografata, la chiamata al `create-db-cluster` restituisce un errore. Se desideri utilizzare una chiave diversa per il cluster di database Aurora PostgreSQL rispetto alla chiave utilizzata dall'istanza database di origine, è necessario specificare anche la `--kms-key-id`.

- Utilizzala se l'archiviazione dell'istanza database di origine non è crittografata, ma desideri che il cluster di database Aurora PostgreSQL lo sia. In tal caso, devi anche identificare la chiave di crittografia necessaria utilizzando il parametro `--kms-key-id`.

- `--kms-key-id` : Opzionale. Permette di specificare la chiave da utilizzare per la crittografia dell'archiviazione (`--storage-encrypted`), utilizzando l'ARN, l'ID, l'alias ARN della chiave o il relativo nome alias. Questo parametro è necessario solo nelle situazioni seguenti:
  - Per scegliere una chiave diversa per il cluster di database Aurora PostgreSQL rispetto a quella utilizzata dall'istanza database di origine.
  - Per creare un cluster crittografato da un'origine non crittografata. In questo caso, è necessario specificare la chiave che Aurora PostgreSQL deve utilizzare per la crittografia.

Una volta creato il cluster di database Aurora PostgreSQL, dovrai creare l'istanza principale utilizzando il comando CLI `create-db-instance`, come mostrato di seguito:

Per Linux, macOS o Unix:

```
aws rds create-db-instance \
    --db-cluster-identifier my-new-aurora-cluster \
    --db-instance-class db.x2g.16xlarge \
    --db-instance-identifier rpg-for-migration \
    --engine aurora-postgresql
```

Per Windows:

```
aws rds create-db-instance ^
    --db-cluster-identifier my-new-aurora-cluster ^
    --db-instance-class db.x2g.16xlarge ^
    --db-instance-identifier rpg-for-migration ^
    --engine aurora-postgresql
```

L'elenco seguente fornisce ulteriori informazioni su alcune delle opzioni mostrate nell'esempio.

- `--db-cluster-identifier`: specifica il nome del nuovo cluster di database Aurora PostgreSQL creato nel passaggio precedente con il comando `create-db-instance`.

- `--db-instance-class`: nome della classe dell'istanza database da utilizzare per l'istanza primaria, come `db.r4.xlarge`, `db.t4g.medium`, `db.x2g.16xlarge`, ecc. Per un elenco delle classi di istanza database disponibili, vedi Tipi di classi di istanza database.
- `--db-instance-identifier`: specifica il nome da assegnare all'istanza primaria.
- `--engine aurora-postgresql`: specifica il `aurora-postgresql` per il motore.

API RDS

Per creare una replica di lettura Aurora da un'istanza database RDS for PostgreSQL di origine, per prima cosa crea un nuovo cluster di database Aurora utilizzando l'operazione API RDS `CreateDBCluster`, che verrà utilizzato per la replica. Quando il cluster di database Aurora PostgreSQL è disponibile, utilizza `CreateDBInstance`per creare l'istanza primaria per il cluster di database Aurora.

Non è necessario specificare l'account utente principale (in genere, `postgres`), la password o il nome del database. La replica di lettura Aurora li ottiene automaticamente dall'istanza database RDS for PostgreSQL di origine specificata con `ReplicationSourceIdentifier`.

È necessario specificare la versione del motore da utilizzare per il cluster di database Aurora PostgreSQL e l'istanza database. La versione specificata deve corrispondere all'istanza database RDS for PostgreSQL di origine. Se l'istanza database RDS for PostgreSQL di origine è crittografata, è necessario specificare anche la crittografia per l'istanza primaria del cluster di database Aurora PostgreSQL. La migrazione di un'istanza crittografata in un cluster di database Aurora non crittografato non è supportata.

Per creare il cluster di database Aurora per la replica di lettura Aurora, utilizza l'operazione API RDS `CreateDBCluster` con i seguenti parametri:

- `DBClusterIdentifier`: nome del cluster di database da creare.
- `DBSubnetGroupName`: nome del gruppo di sottoreti database da associare al cluster di database.
- `Engine=aurora-postgresql`: nome del motore da utilizzare.
- `ReplicationSourceIdentifier`: l'Amazon Resource Name (ARN) per l'istanza database PostgreSQL di origine. Per ulteriori informazioni su Amazon RDS ARNs, consulta Amazon Relational Database Service (Amazon RDS) nel.Riferimenti generali di Amazon Web Services Se `ReplicationSourceIdentifier` riconosce la fonte come crittografata, Amazon RDS utilizza la tua chiave KMS di default, a meno che tu non specifichi una chiave diversa utilizzando l'opzione `KmsKeyId`.

- `VpcSecurityGroupIds`— L'elenco dei gruppi di sicurezza Amazon EC2 VPC da associare a questo cluster DB.
- `StorageEncrypted`: indica che il cluster di database è crittografato. Quando utilizzi questo parametro senza specificare anche il `ReplicationSourceIdentifier`, Amazon RDS utilizza la chiave KMS di default.
- `KmsKeyId`: la chiave per un cluster crittografato. Permette di specificare la chiave da utilizzare per la crittografia dell'archiviazione utilizzando l'ARN, l'ID, l'alias ARN della chiave o il relativo nome alias.

Per ulteriori informazioni, consulta `CreateDBCluster` nella Guida di riferimento delle API di Amazon RDS.

Quando il cluster di database Aurora è disponibile, crea un'istanza primaria per il cluster di database utilizzando l'operazione API RDS `CreateDBInstance` con i seguenti parametri:

- `DBClusterIdentifier`: nome del cluster di database.
- `DBInstanceClass`: nome della classe dell'istanza database da utilizzare per l'istanza primaria.
- `DBInstanceIdentifier`: nome dell'istanza primaria.
- `Engine=aurora-postgresql`: nome del motore da utilizzare.

Per ulteriori informazioni, consulta `CreateDBInstance` nella Guida di riferimento delle API di Amazon RDS.

Promozione di una replica di lettura Aurora

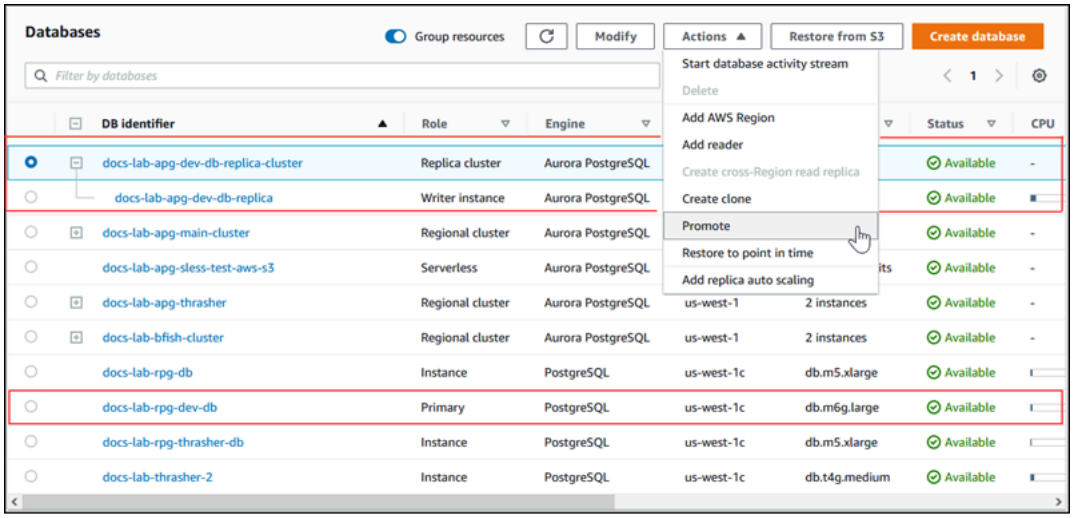
La migrazione ad Aurora PostgreSQL non è completa fino a quando non si promuove il cluster di replica, quindi non eliminare ancora l'istanza database RDS for PostgreSQL di origine.

Prima di promuovere il cluster di replica, assicurati che l'istanza database RDS for PostgreSQL non abbia transazioni in corso o altre attività di scrittura sul database. Quando il ritardo di replica sulla replica di lettura Aurora è zero (0), è possibile promuovere il cluster di replica. Per ulteriori informazioni sul monitoraggio del ritardo di replica, consulta Monitoraggio della replica Aurora PostgreSQL. e Parametri a livello di istanza per Amazon Aurora.

Console

Come promuovere una replica di lettura Aurora a cluster di database Aurora

1. Accedi a Console di gestione AWS e apri la console Amazon RDS all'indirizzo https://console.aws.amazon.com/rds/.
2. Nel pannello di navigazione, seleziona Database.
3. Scegli il cluster di replica.



4. In Actions (Operazioni), seleziona Promote (Promuovi). L'operazione può richiedere alcuni minuti e comportare tempi di inattività.

Al termine del processo, il cluster di replica Aurora è un cluster di database regionale Aurora PostgreSQL, con un'istanza di scrittura contenente i dati dall'istanza database RDS for PostgreSQL.

AWS CLI

Per promuovere una replica di lettura Aurora in un cluster DB autonomo, usa il comando. promote-read-replica-db-cluster AWS CLI

Example

Per Linux, macOS o Unix:

```
aws rds promote-read-replica-db-cluster \
    --db-cluster-identifier myreadreplicacluster
```

Per Windows:

```
aws rds promote-read-replica-db-cluster ^
    --db-cluster-identifier myreadreplicacluster
```

API RDS

Per promuovere una replica di lettura Aurora in un cluster DB autonomo, utilizza l'operazione API RDS. PromoteReadReplicaDBCluster

Dopo aver promosso il cluster di replica, puoi verificare che la promozione sia stata completata controllando il registro eventi, come segue.

Per confermare che il cluster di replica Aurora sia stato promosso

1. Accedi a Console di gestione AWS e apri la console Amazon RDS all'indirizzo https://console.aws.amazon.com/rds/.
2. Nel pannello di navigazione seleziona Events (Eventi).
3. Nella pagina Events (Eventi), trova il nome del cluster nell'elenco Source (Origine). Ogni evento ha origine, tipo, ora e messaggio. Puoi vedere tutti gli eventi che si sono verificati nella tua Regione AWS per il tuo account. Una promozione avvenuta con successo genera il seguente messaggio.

   ```
   Promoted Read Replica cluster to a stand-alone database cluster.
   ```

Al completamento della promozione, l'istanza database RDS for PostgreSQL di origine e il cluster di replica Aurora vengono scollegati. Puoi indirizzare le applicazioni client all'endpoint per la replica di lettura Aurora. Per ulteriori informazioni sugli endpoint Aurora, consulta Connessioni degli endpoint Amazon Aurora. A questo punto, puoi eliminare l'istanza database.

Ottimizzazione delle prestazioni delle query in Aurora PostgreSQL

L'ottimizzazione delle prestazioni delle query è un passaggio fondamentale perché consente ai database di funzionare in modo più rapido ed efficiente utilizzando meno risorse, il che si traduce in una migliore esperienza utente e in una riduzione dei costi operativi. Amazon Aurora PostgreSQL offre diverse funzionalità che aiutano a ottimizzare le prestazioni delle query per i carichi di lavoro PostgreSQL.

Argomenti

- Prestazioni delle query migliorate per Aurora PostgreSQL con Letture ottimizzate per Aurora
- Ottimizzazione delle sottoquery correlate in Aurora PostgreSQL
- Miglioramento delle prestazioni delle query utilizzando join adattivo

Prestazioni delle query migliorate per Aurora PostgreSQL con Letture ottimizzate per Aurora

Con Letture ottimizzate per Aurora è possibile velocizzare l'elaborazione delle query per Aurora PostgreSQL. Un'istanza database Aurora PostgreSQL che utilizza Letture ottimizzate per Aurora offre una latenza delle query fino a 8 volte migliorata e risparmi sui costi fino al 30% per applicazioni con set di dati di grandi dimensioni, che superano la capacità di memoria di un'istanza database.

Argomenti

- Panoramica di Letture ottimizzate per Aurora in PostgreSQL
- Utilizzo di Letture ottimizzate per Aurora
- Casi d'uso per letture ottimizzate per Aurora
- Monitoraggio delle istanze database che utilizzano Letture ottimizzate per Aurora
- Best practice per Letture ottimizzate per Aurora

Panoramica di Letture ottimizzate per Aurora in PostgreSQL

Aurora Optimized Reads è disponibile per impostazione predefinita quando si crea un cluster DB che utilizza istanze R6gd, R8gd e R6id basate su Graviton e R6id basate su Intel con storage express () in memoria non volatile. NVMe È disponibile a partire dalle versioni di PostgreSQL seguenti:

- 14.12 e versioni successive, 15.7 e versioni successive, 16.3 e versioni successive, 17.4 e versioni successive per istanze R8gd
- 14.9 e versioni successive, 15.4 e versioni successive, 16.1 e tutte le versioni successive per istanze R6gd e R6id

Letture ottimizzate per Aurora supporta due funzionalità: cache a più livelli e oggetti temporanei.

Cache a più livelli abilitata per Letture ottimizzate: con una cache a più livelli puoi ottenere una capacità di caching dell'istanza database fino a 5 volte superiore la memoria dell'istanza. Ciò assicura

che la cache contenga automaticamente i dati più recenti e coerenti dal punto di vista transazionale, sollevando le applicazioni dall'onere di gestire la valuta dei dati delle soluzioni di caching basate su set di risultati esterni. Offre una latenza fino a 8 volte migliore per le query che in precedenza recuperavano dati dall'archiviazione di Aurora.

In Aurora, il valore per `shared_buffers` nel gruppo di parametri predefinito è in genere impostato su circa il 75% della memoria disponibile. Tuttavia, per i tipi di istanza r8gd, r6gd e r6id, Aurora riduce `shared_buffers` lo spazio del 4,5% per ospitare i metadati per la cache delle letture ottimizzate.

Oggetti temporanei ottimizzati per la lettura: utilizzando oggetti temporanei, è possibile ottenere un'elaborazione delle query più rapida posizionando i file temporanei generati da PostgreSQL nella memoria locale. NVMe Ciò riduce il traffico verso Elastic Block Storage (EBS) attraverso la rete. Offre una latenza e un throughput fino a 2 volte migliori per le query avanzate che ordinano, uniscono o integrano grandi volumi di dati che non rientrano nella capacità di memoria disponibile su un'istanza database.

In un cluster Aurora I/O ottimizzato, Optimized Reads utilizza sia la cache a più livelli che gli oggetti temporanei sullo storage. NVMe Con la funzionalità di cache a più livelli abilitata per Letture ottimizzate, Aurora alloca il doppio della memoria dell'istanza per gli oggetti temporanei, circa il 10% dell'archiviazione per le operazioni interne e l'archiviazione rimanente come cache a più livelli. In un cluster Aurora standard, Letture ottimizzate utilizza solo oggetti temporanei.

I cluster ottimizzati per l'I/O Aurora consentono di ridimensionare lo spazio allocato per gli oggetti temporanei abilitati a Letture ottimizzate utilizzando il parametro dinamico `aurora_temp_space_size` a livello di istanza. La funzionalità di ridimensionamento è disponibile a partire dalle versioni di PostgreSQL seguenti:

- 16.8 e tutte le versioni successive
- 15.12 e versioni successive alla 15
- 14.17 e versioni successive alla 14

Con questo parametro, è possibile ridimensionare la capacità da 2 a 6 volte la memoria dell'istanza senza richiedere il riavvio del motore di database. Quando si espande lo spazio per gli oggetti temporanei, la modifica diventa immediatamente effettiva, indipendentemente dai carichi di lavoro simultanei. Tuttavia, quando si riduce lo spazio, la modifica viene completata solo se lo spazio inutilizzato per gli oggetti temporanei è sufficiente per soddisfare la nuova richiesta di dimensione. Dopo aver ridimensionato gli oggetti temporanei abilitati a Letture ottimizzate, la cache a più livelli si regola automaticamente per utilizzare lo spazio disponibile.

Motore	Configurazione dell'archiviazione del cluster	Oggetti temporanei abilitati per Letture ottimizzate	Cache a più livelli abilitata per Letture ottimizzate	Versioni supportate
Amazon Aurora edizione compatibile con PostgreSQL	Standard	Sì	No	• 14.9 e versioni successive, 15.4 e versioni successive, 16.1 e tutte le versioni successive per le istanze r6gd e r6id • 14.12 e versioni successive, 15.7 e versioni successive, 16.3 e versioni successive, 17.4 e versioni successive per istanze r8gd
	Ottimizzata per l'I/O	Sì	Sì	

> **Note**
>
> Il passaggio tra cluster ottimizzati per IO e standard su una classe di istanze DB basata comporta un NVMe riavvio immediato del motore di database.

In Aurora PostgreSQL, utilizzare il parametro `temp_tablespaces` per configurare lo spazio della tabella in cui vengono archiviati gli oggetti temporanei.

Per verificare se gli oggetti temporanei sono configurati, utilizzare il seguente comando:

```
postgres=> show temp_tablespaces;
temp_tablespaces
---------------------
aurora_temp_tablespace
(1 row)
```

`aurora_temp_tablespace`È un tablespace configurato da Aurora che punta all'archiviazione locale. NVMe Non puoi modificare questo parametro o tornare all'archiviazione Amazon EBS.

Per verificare se la cache per le letture ottimizzate è attiva, usa il seguente comando:

```
postgres=> show shared_preload_libraries;
                 shared_preload_libraries
-----------------------------------------------------------
rdsutils,pg_stat_statements,aurora_optimized_reads_cache
```

Utilizzo di Letture ottimizzate per Aurora

Quando si esegue il provisioning di un'istanza DB Aurora PostgreSQL con una delle istanze DB NVMe basate, l'istanza DB utilizza automaticamente Aurora Optimized Reads.

Per attivare Letture ottimizzate per Aurora, procedi in uno dei seguenti modi:

- Crea un cluster Aurora PostgreSQL DB utilizzando una delle classi di istanze DB basate. NVMe Per ulteriori informazioni, consulta Creazione di un cluster database Amazon Aurora.
- Modifica un cluster Aurora PostgreSQL DB esistente per utilizzare una delle classi di istanze DB basate. NVMe Per ulteriori informazioni, consulta Modifica di un cluster database Amazon Aurora.

Aurora Optimized Reads è disponibile Regioni AWS ovunque siano supportate una o più classi di istanze DB con archiviazione NVMe SSD locale. Per ulteriori informazioni, consulta Classi di istanze database Amazon Aurora.

Per tornare a un'istanza Aurora di lettura non ottimizzata, modifica la classe di istanza DB dell'istanza Aurora con la classe di istanza simile senza spazio di archiviazione NVMe effimero per i carichi di lavoro del database. Ad esempio, se la classe di istanza database corrente è db.r6gd.4xlarge, scegli db.r6g.4xlarge per tornare indietro. Per ulteriori informazioni, consulta la pagina relativa alla modifica di un'istanza database Aurora.

Casi d'uso per letture ottimizzate per Aurora

Cache a più livelli abilitata per Letture ottimizzate

Di seguito sono riportati alcuni casi d'uso in cui è possibile trarre vantaggio dalla funzionalità Letture ottimizzate con cache a più livelli:

- Applicazioni su scala Internet come elaborazione dei pagamenti, fatturazione, e-commerce con prestazioni rigorose. SLAs
- Dashboard di reporting in tempo reale che eseguono centinaia di interrogazioni mirate a fini di raccolta. metrics/data
- Applicazioni di intelligenza artificiale generativa con l'estensione pgvector per la ricerca di vicini esatti o più prossimi tra milioni di incorporamenti vettoriali.

Oggetti temporanei abilitati per Letture ottimizzate

Di seguito sono riportati alcuni casi d'uso in cui è possibile trarre vantaggio dalla funzionalità Letture ottimizzate con oggetti temporanei:

- Query analitiche che includono Common Table Expressions (CTEs), tabelle derivate e operazioni di raggruppamento.
- Repliche di lettura che gestiscono le query non ottimizzate per un'applicazione.
- Query di reporting on demand o dinamiche con operazioni complesse come GROUP BY e ORDER BY che non sempre possono utilizzare indici appropriati.
- Operazioni `CREATE INDEX` o `REINDEX` per l'ordinamento.
- Altri carichi di lavoro che utilizzano tabelle temporanee interne.

Monitoraggio delle istanze database che utilizzano Letture ottimizzate per Aurora

Per monitorare le query che utilizzano la cache a più livelli abilitata per Letture ottimizzate utilizza il comando EXPLAIN, come mostrato nell'esempio seguente:

```
Postgres=> EXPLAIN (ANALYZE, BUFFERS) SELECT c FROM sbtest15 WHERE id=100000000


QUERY PLAN
------------------------------------------------------------------------------
```

```
 Index Scan using sbtest15_pkey on sbtest15  (cost=0.57..8.59 rows=1 width=121) (actual
 time=0.287..0.288 rows=1 loops=1)
   Index Cond: (id = 100000000)
   Buffers: shared hit=3 read=2 aurora_orcache_hit=2
   I/O Timings: shared/local read=0.264
 Planning:
   Buffers: shared hit=33 read=6 aurora_orcache_hit=6
   I/O Timings: shared/local read=0.607
 Planning Time: 0.929 ms
 Execution Time: 0.303 ms
(9 rows)
Time: 2.028 ms
```

> **Note**
>
> I campi `aurora_orcache_hit` e `aurora_storage_read` della sezione `Buffers` del piano di spiegazione vengono visualizzati solo quando la funzionalità Letture ottimizzate è attivata e i valori sono maggiori di zero. Il campo Lettura indica il totale dei campi `aurora_orcache_hit` e `aurora_storage_read`.

È possibile monitorare le istanze DB che utilizzano Aurora Optimized Reads utilizzando le seguenti metriche: CloudWatch

- `AuroraOptimizedReadsCacheHitRatio`
- `FreeEphemeralStorage`
- `ReadIOPSEphemeralStorage`
- `ReadLatencyEphemeralStorage`
- `ReadThroughputEphemeralStorage`
- `WriteIOPSEphemeralStorage`
- `WriteLatencyEphemeralStorage`
- `WriteThroughputEphemeralStorage`

Queste metriche forniscono dati sullo spazio di archiviazione dell'archivio dell'istanza, sulle operazioni IOPS e sulla velocità di trasmissione effettiva disponibili. Per ulteriori informazioni su questi parametri, consulta Parametri a livello di istanza per Amazon Aurora.

È inoltre possibile utilizzare l'`pg_proctab`estensione per monitorare l'archiviazione. NVMe

```
postgres=>select * from pg_diskusage();

major | minor |       devname       | reads_completed | reads_merged | sectors_read |
 readtime | writes_completed | writes_merged | sectors_written | writetime | current_io
 | iotime  | totaliotime
------+-------+---------------------+-----------------+--------------+--------------
+----------+------------------+---------------+-----------------+-----------
+------------+---------+-------------
      |       | rdstemp             |           23264 |            0 |       191450 |
  11670 |          1750892 |             0 |        24540576 |    819350 |          0 |
 3847580 |      831020
      |       | rdsephemeralstorage |           23271 |            0 |       193098 |
   2620 |           114961 |             0 |        13845120 |    130770 |          0 |
  215010 |      133410
(2 rows)
```

Best practice per Letture ottimizzate per Aurora

Usa le seguenti best practice per Letture ottimizzate per Aurora:

- Monitora lo spazio di archiviazione disponibile sull'instance store con la CloudWatch metrica`FreeEphemeralStorage`. Se l'archivio dell'istanza sta raggiungendo il limite a causa del carico di lavoro sull'istanza database, ottimizzare la concorrenza e le query che utilizzano grandi volumi di oggetti temporanei oppure modificarlo affinché utilizzi una classe di istanza database più grande.
- Monitora la CloudWatch metrica per la frequenza di accesso alla cache di Optimized Reads. Operazioni come VACUUM permettono di modificare blocchi numerosi molto rapidamente. Ciò può causare un calo temporaneo della percentuale di riscontri. Utilizza l'estensione `pg_prewarm` per caricare dati nella cache del buffer che consente ad Aurora di scrivere in modo proattivo alcuni di questi blocchi nella cache delle Letture ottimizzate.
- È possibile abilitare Gestione della cache del cluster (CCM) per preparare la cache del buffer e la cache a più livelli su un lettore di livello 0, che verrà utilizzato come destinazione di failover. Quando la funzionalità CCM è abilitata, la cache del buffer viene scansionata periodicamente per

scrivere pagine idonee all'espulsione nella cache a più livelli. Per ulteriori informazioni sulla CCM, consulta Ripristino rapido dopo il failover con Cluster Cache Management per Aurora PostgreSQL.

Ottimizzazione delle sottoquery correlate in Aurora PostgreSQL

Una sottoquery correlata fa riferimento alle colonne della tabella dalla query esterna. Viene valutata una volta per ogni riga restituita dalla query esterna. Nell'esempio seguente, la sottoquery fa riferimento a una colonna della tabella ot. Questa tabella non è inclusa nella clausola FROM della sottoquery, ma vi viene fatto riferimento nella clausola FROM della query esterna. Se la tabella ot ha 1 milione di righe, la sottoquery deve essere valutata 1 milione di volte.

```
SELECT ot.a, ot.b FROM ot WHERE ot.b < (SELECT AVG(it.b) FROM it WHERE it.a = ot.a);
```

Note

- La trasformazione delle sottoquery e la cache delle sottoquery sono disponibili in Aurora PostgreSQL a partire dalla versione 16.8, mentre Babelfish per Aurora PostgreSQL supporta queste funzionalità dalla versione 4.2.0.
- A partire dalle versioni 4.6.0 e 5.2.0 di Babelfish per Aurora PostgreSQL, queste funzionalità sono controllate dai seguenti parametri:
  - babelfishpg_tsql.apg_enable_correlated_scalar_transform
  - babelfishpg_tsql.apg_enable_subquery_cache

  Per impostazione predefinita, entrambi i parametri sono attivati.

Miglioramento delle prestazioni delle query di Aurora PostgreSQL con la trasformazione delle sottoquery

Aurora PostgreSQL può accelerare le sottoquery correlate trasformandole in outer join equivalenti. Questa ottimizzazione si applica ai seguenti due tipi di sottoquery correlate:

- Sottoquery che restituiscono un singolo valore aggregato e vengono visualizzate nell'elenco SELECT.

```
SELECT ot.a, ot.b, (SELECT AVG(it.b) FROM it WHERE it.a = ot.a) FROM ot;
```

- Sottoquery che restituiscono un singolo valore aggregato e vengono visualizzate in una clausola WHERE.

```
SELECT ot.a, ot.b FROM ot WHERE ot.b < (SELECT AVG(it.b) FROM it WHERE it.a = ot.a);
```

Abilitazione della trasformazione nella sottoquery

Per abilitare la trasformazione di sottoquery correlate in outer join equivalenti, impostare il parametro `apg_enable_correlated_scalar_transform` su `ON`. Il valore predefinito di questo parametro è `OFF`.

È possibile modificare il gruppo di parametri del cluster o dell'istanza per impostare i parametri. Per ulteriori informazioni, consulta Gruppi di parametri per Amazon Aurora.

In alternativa, è possibile configurare l'impostazione solo per la sessione corrente con il seguente comando:

```
SET apg_enable_correlated_scalar_transform TO ON;
```

Verifica della trasformazione

Utilizzare il comando EXPLAIN per verificare se la sottoquery correlata è stata trasformata in un outer join nel piano di query.

Quando la trasformazione è abilitata, la parte di sottoquery correlata applicabile verrà trasformata in outer join. Per esempio:

```
postgres=> CREATE TABLE ot (a INT, b INT);
CREATE TABLE
postgres=> CREATE TABLE it (a INT, b INT);
CREATE TABLE

postgres=> SET apg_enable_correlated_scalar_transform TO ON;
SET
postgres=> EXPLAIN (COSTS FALSE) SELECT ot.a, ot.b FROM ot WHERE ot.b < (SELECT
 AVG(it.b) FROM it WHERE it.a = ot.a);

                        QUERY PLAN
------------------------------------------------------------
```

```
 Hash Join
   Hash Cond: (ot.a = apg_scalar_subquery.scalar_output)
   Join Filter: ((ot.b)::numeric < apg_scalar_subquery.avg)
   ->  Seq Scan on ot
   ->  Hash
         ->  Subquery Scan on apg_scalar_subquery
               ->  HashAggregate
                     Group Key: it.a
                     ->  Seq Scan on it
```

La stessa query non viene trasformata quando il parametro GUC viene impostato su OFF. Il piano non avrà un outer join ma un sottopiano.

```
postgres=> SET apg_enable_correlated_scalar_transform TO OFF;
SET
postgres=> EXPLAIN (COSTS FALSE) SELECT ot.a, ot.b FROM ot WHERE ot.b < (SELECT
 AVG(it.b) FROM it WHERE it.a = ot.a);
                QUERY PLAN
---------------------------------------
 Seq Scan on ot
   Filter: ((b)::numeric < (SubPlan 1))
   SubPlan 1
     ->  Aggregate
           ->  Seq Scan on it
                 Filter: (a = ot.a)
```

Limitazioni

- La sottoquery deve trovarsi nell'elenco SELECT o in una delle condizioni della clausola where. In caso contrario, non verrà trasformata.
- La sottoquery deve restituire una funzione di aggregazione. Le funzioni di aggregazione definite dall'utente non sono supportate per la trasformazione.
- Una sottoquery la cui espressione restituita non è una semplice funzione di aggregazione non verrà trasformata.
- La condizione correlata nelle clausole WHERE della sottoquery deve essere un semplice riferimento di colonna. In caso contrario, non verrà trasformata.
- La condizione correlata nella sottoquery in cui le clausole devono essere un semplice predicato di uguaglianza.
- La sottoquery non può contenere né una clausola HAVING né una clausola GROUP BY.

- La clausola where nella sottoquery può contenere uno o più predicati combinati con AND.

> **Note**
>
> L'impatto della trasformazione sulle prestazioni varia a seconda dello schema, dei dati e del carico di lavoro. L'esecuzione di sottoquery correlate con la trasformazione può migliorare significativamente le prestazioni all'aumentare del numero di righe prodotte dalla query esterna. Si consiglia vivamente di testare questa funzionalità in un ambiente non di produzione con lo schema, i dati e il carico di lavoro effettivi prima di abilitarla in un ambiente di produzione.

Utilizzo della cache delle sottoquery per migliorare le prestazioni delle query di Aurora PostgreSQL

Aurora PostgreSQL supporta la cache delle sottoquery per archiviare i risultati delle sottoquery correlate. Questa funzionalità salta le esecuzioni ripetute delle sottoquery correlate quando i risultati delle sottoquery sono già presenti nella cache.

Informazioni sulla cache delle sottoquery

Il nodo Memoize di PostgreSQL è la parte fondamentale della cache delle sottoquery. Il nodo Memoize mantiene una tabella hash nella cache locale per mappare i valori dei parametri di input alle righe dei risultati delle query. Il limite di memoria per la tabella hash è il prodotto di work_mem e hash_mem_multiplier. Per ulteriori informazioni, consulta Consumo di risorse.

Durante l'esecuzione delle query, la cache delle sottoquery utilizza la percentuale di riscontri nella cache (CHR, Cache Hit Rate) per stimare se la cache sta migliorando le prestazioni delle query e per decidere in fase di esecuzione della query se continuare a utilizzare la cache. CHR è il rapporto tra il numero di riscontri nella cache e il numero totale di richieste. Ad esempio, se una sottoquery correlata deve essere eseguita 100 volte e 70 di tali risultati di esecuzione possono essere recuperati dalla cache, il CHR è 0,7.

Per ogni numero di mancati riscontri nella cache di apg_subquery_cache_check_interval, il vantaggio della cache delle sottoquery viene valutato controllando se il CHR è maggiore di apg_subquery_cache_hit_rate_threshold. In caso contrario, la cache verrà eliminata dalla memoria e l'esecuzione della query tornerà alla riesecuzione della sottoquery originale, non memorizzata nella cache.

Parametri che controllano il comportamento della cache delle sottoquery

La tabella seguente elenca i parametri che controllano il comportamento della cache delle sottoquery.

Parametro	Descrizione	Default	Consentito
apg_enable_subquery_cache	Abilita l'uso della cache per sottoquery scalari correlate.	OFF	ON, OFF
apg_subquery_cache_check_interval	Imposta la frequenza, in numero di mancati riscontri nella cache, per valutare la percentuale di riscontri nella cache delle sottoquery.	500	0–2147483647
apg_subquery_cache_hit_rate_threshold	Imposta la soglia per la percentuale di riscontri nella cache delle sottoquery.	0.3	0.0 - 1.0

Note

- Valori maggiori di `apg_subquery_cache_check_interval` possono migliorare l'accuratezza della stima dei vantaggi della cache basata su CHR, ma aumenteranno il sovraccarico della cache, poiché CHR non verrà valutato finché la tabella della cache non avrà righe `apg_subquery_cache_check_interval`.
- Valori più elevati di `apg_subquery_cache_hit_rate_threshold` tendono ad abbandonare la cache delle sottoquery e a tornare alla riesecuzione originale delle sottoquery non memorizzate nella cache.

È possibile modificare il gruppo di parametri del cluster o dell'istanza per impostare i parametri. Per ulteriori informazioni, consulta Gruppi di parametri per Amazon Aurora.

In alternativa, è possibile configurare l'impostazione solo per la sessione corrente con il seguente comando:

```
SET apg_enable_subquery_cache TO ON;
```

Attivazione della cache delle sottoquery in Aurora PostgreSQL

Quando la cache delle sottoquery è abilitata, Aurora PostgreSQL applica la cache per salvare i risultati delle sottoquery. Il piano di query avrà quindi un nodo Memoize in SubPlan.

Ad esempio, la sequenza di comandi seguente mostra il piano di esecuzione delle query stimato di una sottoquery correlata semplice senza cache delle sottoquery.

```
postgres=> SET apg_enable_subquery_cache TO OFF;
SET
postgres=> EXPLAIN (COSTS FALSE) SELECT ot.a, ot.b FROM ot WHERE ot.b < (SELECT it.b
 FROM it WHERE it.a = ot.a);

             QUERY PLAN
-----------------------------------
 Seq Scan on ot
   Filter: (b < (SubPlan 1))
   SubPlan 1
     ->  Seq Scan on it
           Filter: (a = ot.a)
```

Dopo l'attivazione di `apg_enable_subquery_cache`, il piano di query conterrà un nodo Memoize sotto il nodo SubPlan, a indicare che la sottoquery prevede di utilizzare la cache.

```
postgres=> SET apg_enable_subquery_cache TO ON;
SET
postgres=> EXPLAIN (COSTS FALSE) SELECT ot.a, ot.b FROM ot WHERE ot.b < (SELECT it.b
 FROM it WHERE it.a = ot.a);

             QUERY PLAN
-----------------------------------
 Seq Scan on ot
   Filter: (b < (SubPlan 1))
   SubPlan 1
     ->  Memoize
           Cache Key: ot.a
```

```
           Cache Mode: binary
           ->  Seq Scan on it
                 Filter: (a = ot.a)
```

Il piano di esecuzione delle query effettivo contiene maggiori dettagli sulla cache delle sottoquery, inclusi i riscontri nella cache e i mancati riscontri nella cache. L'output seguente mostra il piano di esecuzione delle query effettivo della query di esempio precedente dopo l'inserimento di alcuni valori nelle tabelle.

```
postgres=> EXPLAIN (COSTS FALSE, TIMING FALSE, ANALYZE TRUE) SELECT ot.a, ot.b FROM ot
 WHERE ot.b < (SELECT it.b FROM it WHERE it.a = ot.a);
            QUERY PLAN
-------------------------------------------------------------------------------
 Seq Scan on ot (actual rows=2 loops=1)
   Filter: (b < (SubPlan 1))
   Rows Removed by Filter: 8
   SubPlan 1
     ->  Memoize (actual rows=0 loops=10)
           Cache Key: ot.a
           Cache Mode: binary
           Hits: 4  Misses: 6  Evictions: 0  Overflows: 0  Memory Usage: 1kB
           ->  Seq Scan on it (actual rows=0 loops=6)
                 Filter: (a = ot.a)
                 Rows Removed by Filter: 4
```

Il numero totale di riscontri nella cache è 4 e il numero totale di mancati riscontri nella cache è 6. Se il numero totale di riscontri e mancati riscontri è inferiore al numero di loop nel nodo Memoize, significa che la valutazione CHR non ha avuto esito positivo e che la cache è stata ripulita e abbandonata a un certo punto. L'esecuzione della sottoquery è quindi tornata alla riesecuzione originale non memorizzata nella cache.

Limitazioni

La cache delle sottoquery non supporta determinati modelli di sottoquery correlate. Questi tipi di query verranno eseguiti senza cache, anche se la cache delle sottoquery è attivata:

- Sottoquery correlate IN/EXISTS/ANY/ALL
- Sottoquery correlate contenenti funzioni non deterministiche.
- Sottoquery correlate che fanno riferimento a colonne esterne della tabella con tipi di dati che non supportano operazioni di hashing o uguaglianza.

Miglioramento delle prestazioni delle query utilizzando join adattivo

Panoramica

Join adattivo è una funzionalità di anteprima di Aurora PostgreSQL 17.4 che aiuta a migliorare le prestazioni delle query. Questa funzionalità è disabilitata per impostazione predefinita, ma è possibile abilitarla utilizzando i parametri GUC (Global User Configuration). Poiché si tratta di una funzionalità di anteprima, i valori dei parametri predefiniti potrebbero cambiare. Se abilitato, il join adattivo aiuta a ottimizzare le prestazioni delle query passando dinamicamente da un loop join nidificato a un hash join in runtime. Questo passaggio si verifica quando lo strumento di ottimizzazione di PostgreSQL ha scelto erroneamente un loop join nidificato a causa di stime di cardinalità imprecise.

Configurazione del join adattivo

È possibile controllare il join adattivo utilizzando questi tre parametri GUC:

Parametri di configurazione di join adattivo

Parametro GUC	Descrizione	Opzioni predefinite e di configurazione
apg_adaptive_join_crossover_multiplier	Questo moltiplicatore funziona con il punto di incrocio delle righe per determinare quando passare da un loop nidificato a un hash join. Il punto di incrocio delle righe è il punto in cui lo strumento di ottimizzazione SQL stima che le operazioni di loop nidificato e hash join abbiano lo stesso costo. Un valore di moltiplicatore più elevato riduce la probabilità che il join adattivo passi a un hash join.	Controlla se Join adattivo è abilitato • Valore predefinito: -1 (disabilitato) • Intervallo valido: da -1 a DBL_MAX • Imposta su >= 1 per attivarla
apg_adaptive_join_cost_threshold	Questo parametro imposta una soglia minima di costo delle query. Join adattivo si disattiva automaticamente per le query al di sotto di questa soglia. In questo modo si	Imposta la soglia di costo minima per la query • Valore predefinito: 100

Parametro GUC	Descrizione	Opzioni predefinite e di configurazione
	evita un sovraccarico prestazionale nelle query semplici in cui il costo della pianificazione di un join adattivo potrebbe superare i vantaggi del passaggio dal nested loop all'hash join.	• Intervallo valido: da 0 a DBL_MAX

apg_enable_paramet erized_adaptive_join	Se abilitato, questo parametro estende la funzionalità join adattivo ai loop join nidificati parametrizzati. Per impostazione predefinita, il join adattivo funziona solo con loop join nidificati non parametrizzati, poiché è più probabile che questi traggano vantaggio dal passaggio all'hash join. I loop join nidificati parametri zzati in genere offrono prestazioni migliori, rendendo meno importante il passaggio all'hash join.	Controlla il comportamento del join adattivo per loop join nidificati
		• Valore predefinito: false
		• Valori validi: true/false
		  • Se falso: funziona solo con loop join nidificati non parametrizzati
		  • Se vero: funziona con loop join nidificat i parametrizzati e non parametrizzati
		
		Note
		Deve essere prima abilitato `apg_adapt ive_join_ crossover _multiplier`

Utilizzo delle tabelle non registrate in Aurora PostgreSQL

Amazon Aurora PostgreSQL supporta tabelle non registrate che proteggono dagli arresti anomali e mantengono l'integrità dei dati anche dopo errori o failover delle istanze di scrittura. In PostgreSQL standard, le tabelle non registrate ignorano il log WAL (Write-Ahead Log) durante le operazioni di scrittura, aumentando di conseguenza la velocità di scrittura. Tuttavia, questo si traduce in una ridotta durabilità, perché le tabelle non registrate non sono protette dagli arresti anomali e possono perdere dati dopo un guasto del sistema o un arresto non corretto. Queste tabelle non registrate vengono troncate automaticamente dopo un arresto anomalo o non corretto. Inoltre, i relativi contenuti e indici non vengono replicati sui server di standby.

Aurora PostgreSQL, invece, gestisce diversamente le tabelle non registrate grazie alla sua architettura di archiviazione distribuita. Questo è dovuto al sistema di archiviazione Aurora, che non si basa sul tradizionale log WAL PostgreSQL per la durabilità. Tuttavia, i vantaggi in termini di prestazioni tipicamente associati alle tabelle non registrate in PostgreSQL standard potrebbero non essere così significativi in Aurora. Questo è dovuto all'architettura di archiviazione distribuita di Aurora, che può introdurre un sovraccarico aggiuntivo rispetto all'archiviazione locale utilizzata in PostgreSQL standard.

Durante l'utilizzo delle tabelle non registrate in Aurora PostgreSQL, considerare quando segue:

- È possibile accedere alle tabelle non registrate solo dal nodo di scrittura nel cluster Aurora database.
- I nodi di lettura possono accedere alle tabelle non registrate solo quando vengono promossi allo stato di scrittura.
- Se si prova ad accedere alle tabelle non registrate da un nodo di lettura, si verifica il seguente errore:

  ```
  cannot access temporary or unlogged relations during recovery.
  ```

Creazione di tabelle non registrate

Per creare una tabella non registrata in Aurora PostgreSQL, aggiungere la parola chiave UNLOGGED nell'istruzione CREATE TABLE:

```
CREATE UNLOGGED TABLE staging_sales_data (
    transaction_id bigint,
```

```
    customer_id bigint,
    product_id bigint,
    transaction_date date,
    amount NUMERIC
);
```

Gestione delle tabelle non registrate durante la migrazione

Quando si prepara la migrazione dei dati ad Aurora PostgreSQL, è importante identificare e gestire in modo appropriato le tabelle non registrate. Le tabelle non registrate sono escluse sia dal log WAL che dal flusso di replica, il che significa che non verranno replicate su una replica di lettura Aurora.

Convertire le tabelle non registrate in tabelle registrate o eliminarle se non vengono utilizzate prima di creare una replica di lettura Aurora.

Per verificare la presenza di tabelle non registrate in ogni database all'interno dell'istanza, utilizzare il comando seguente:

```
SELECT oid, relfilenode, relname, relpersistence, relkind
FROM pg_class
WHERE relpersistence ='u';
```

Per riconvertire una tabella non registrata in tabella registrata, utilizzare il seguente comando:

```
ALTER TABLE table_name SET LOGGED;
```

Questa operazione riscrive l'intera tabella e la blocca in modo esclusivo fino al completamento. Per le tabelle di grandi dimensioni, questo può tradursi in tempi di inattività significativi.

Conversione di tabelle non registrate in tabelle registrate

Quando è necessario riconvertire una tabella non registrata in tabella registrata, è possibile utilizzare il seguente comando:

```
ALTER TABLE table_name SET LOGGED;
```

Questa operazione riscrive l'intera tabella e la blocca in modo esclusivo fino al completamento dell'operazione. Per le tabelle di grandi dimensioni, questo può tradursi in tempi di inattività significativi.

Tabelle non registrate e replica logica

Le tabelle non registrate in genere non sono incluse nella replica logica perché questa si basa sul log WAL per acquisire e trasferire le modifiche. Per impostazione predefinita, le modifiche alle tabelle non registrate sono escluse sia dal log WAL che dal flusso di replica, il che le rende inadatte ai casi d'uso in cui è richiesta la replica logica. Tuttavia, Aurora PostgreSQL fornisce un parametro chiamato `rds.logically_replicate_unlogged_tables` che consente di controllare questo comportamento:

- Quando `rds.logically_replicate_unlogged_tables` è impostato su 0 (disattivato), le tabelle non registrate vengono escluse dalla replica logica.
- Quando `rds.logically_replicate_unlogged_tables` è impostato su 1 (attivato), le tabelle non registrate vengono incluse nella replica logica.

> **Note**
>
> In Aurora PostgreSQL, il parametro predefinito per `rds.logically_replicate_unlogged_tables` è impostato su 1 (attivato) nella versione 14 e precedenti e su 0 (disattivato) nella versione 15 e successive.

Utilizzo della funzionalità di autovacuum di PostgreSQL in Amazon Aurora PostgreSQL

Consigliamo vivamente di usare la caratteristica di autovacuum per mantenere l'integrità dell'istanza database di PostgreSQL. La funzione di autovaacuum automatizza l'esecuzione del comando VACUUM e ANALYZE. Verifica la presenza di tabelle con un numero elevato di tuple inserite, aggiornate o eliminate. Dopo questa verifica, recupera lo storage rimuovendo i dati obsoleti o le tuple da database PostgreSQL.

Per impostazione predefinita, la funzionalità di autovacuum è abilitata per le istanze database di Aurora PostgreSQL create utilizzando uno dei gruppi di parametri di database PostgreSQL predefiniti. Per impostazione predefinita vengono impostati anche altri parametri di configurazione associati alla caratteristica di autovacuum. Poiché questi valori di default sono in qualche modo generici, è possibile trarre vantaggio dalla regolazione di alcuni parametri associati alla caratteristica di autovacuum per il carico di lavoro specifico.

Di seguito, puoi trovare ulteriori informazioni sulla funzionalità di autovacuum e su come regolare alcuni dei relativi parametri sulla tua istanza database di Aurora PostgreSQL.

Argomenti

- Allocazione di memoria per il vacuum
- Riduzione della probabilità che si verifichi il wraparound dell'ID delle transazioni
- Determinare se le tabelle nel database devono essere sottoposte a vacuum
- Determinare quali tabelle sono attualmente idonee per l'Autovacuum
- Determinare se l'Auotvacuum è attualmente in esecuzione e per quanto tempo
- Esecuzione di un congelamento manuale del vacuum
- Indicizzare di nuovo una tabella quando l'autovacuum è in esecuzione
- Gestione di autovacuum con indici di grandi dimensioni
- Altri parametri che influenzano l'autovacuum
- Impostazione dei parametri di autovacuum a livello tabella
- Registrazione delle attività di autovacuum e vacuum
- Comprensione del comportamento della funzionalità di autovacuum con database non validi
- Identificazione e risoluzione dei blocchi per i processi di vacuum aggressivi in Aurora PostgreSQL

Allocazione di memoria per il vacuum

Uno dei parametri più importanti che influenzano le prestazioni dell'autovacuum è il parametro `autovacuum_work_mem`. Nelle versioni 14 e precedenti di Aurora PostgreSQL, il parametro `autovacuum_work_mem` è impostato su -1, a indicare che al suo posto viene utilizzata l'impostazione `maintenance_work_mem`. Per tutte le altre versioni, `autovacuum_work_mem` è determinato da GREATEST({DBInstanceClassMemory/32768}, 65536).

Le operazioni manuali di vacuum utilizzano sempre l'impostazione `maintenance_work_mem`, con un'impostazione predefinita di GREATEST({DBInstanceClassMemory/63963136*1024}, 65536), che

può anche essere regolata a livello di sessione utilizzando il comando SET per operazioni VACUUM manuali più mirate.

autovacuum_work_mem determina la memoria di autovacuum per contenere gli identificatori delle tuple inattive (pg_stat_all_tables.n_dead_tup) per gli indici dell'attività di vacuum.

Quando esegui i calcoli per determinare il valore del parametro autovacuum_work_mem, tieni presente quanto segue:

- Se imposti il parametro su un valore troppo basso, il processo di vacuum potrebbe dover eseguire la scansione della tabella più volte per completare il lavoro. Queste scansioni multiple possono avere un impatto negativo sulle prestazioni Per le istanze più grandi, impostare maintenance_work_mem o autovacuum_work_mem su almeno 1 GB può migliorare le prestazioni delle tabelle di vacuum con un numero elevato di tuple morte. Tuttavia, nelle versioni 16 e precedenti di PostgreSQL, l'utilizzo della memoria di vacuum è limitato a 1 GB, il che è sufficiente per elaborare circa 179 milioni di tuple morte in un unico passaggio. Se una tabella ha una quantità superiore di tuple morte, il vacuum dovrà effettuare più passaggi tra gli indici della tabella, facendo aumentare notevolmente il tempo necessario. A partire dalla versione 17 di PostgreSQL, non esiste un limite di 1 GB e la funzionalità di autovacuum può elaborare più di 179 milioni di tuple utilizzando strutture radice.

  Un identificatore di tuple ha dimensioni di 6 byte. Per stimare la memoria necessaria per il processo di vacuum di un indice di una tabella, esegui una query pg_stat_all_tables.n_dead_tup per trovare il numero di tuple morte, quindi moltiplica questo numero per 6 per determinare la memoria richiesta per svuotare l'indice in un unico passaggio. Puoi utilizzare la seguente query.

  ```
  SELECT
      relname AS table_name,
      n_dead_tup,
      pg_size_pretty(n_dead_tup * 6) AS estimated_memory
  FROM
      pg_stat_all_tables
  WHERE
      relname = 'name_of_the_table';
  ```

- Il parametro autovacuum_work_mem funziona insieme al parametro autovacuum_max_workers. Ogni nodo worker tra autovacuum_max_workers può utilizzare la memoria allocata. Se si dispone di molte tabelle di piccole dimensioni, assegna più autovacuum_max_workers e meno autovacuum_work_mem. Se disponi di tabelle di grandi dimensioni (ad esempio, dimensioni superiori a 100 GB), alloca più memoria e meno processi

worker. Si deve avere abbastanza memoria allocata affinché si abbia esito positivo sulle tabelle più grandi. Quindi assicurati che la combinazione di processi worker e memoria sia uguale alla memoria totale che desideri allocare.

Riduzione della probabilità che si verifichi il wraparound dell'ID delle transazioni

In alcuni casi, le impostazioni del gruppo di parametri correlate all'autovacuum potrebbero non essere abbastanza aggressive da prevenire il wraparound dell'ID delle transazioni. Per risolvere questo problema, Aurora PostgreSQL offre un meccanismo che adatta automaticamente i valori del parametro di autovacuum. Ottimizzazione adattiva del parametro di autovacuum è una funzionalità di Aurora PostgreSQL. Una spiegazione dettagliata di Wraparound della transazione si trova nella documentazione di PostgreSQL.

La regolazione adattiva del parametro di autovacuum è attivata per impostazione predefinita per le istanze Aurora PostgreSQL con il parametro dinamico `rds.adaptive_autovacuum` impostato su ON. Si consiglia di tenere questa opzione attivata. Tuttavia, per disattivare l'ottimizzazione adattiva del parametro di autovacuum, impostare il parametro `rds.adaptive_autovacuum` su 0 o OFF.

Il wraparound degli ID di transazione è ancora possibile quando Aurora Amazon RDS regola i parametri di autovacuum. Si consiglia di implementare un allarme Amazon CloudWatch per il wraparound dell'ID della transazione. Per ulteriori informazioni, consulta il post Implementazione di un sistema di avvisi rapidi per il wraparound degli ID transazione in RDS for PostgreSQL del blog sui database di AWS.

Con la regolazione adattiva del parametro di autovacuum attivata, Amazon RDS inizia a regolare i parametri di autovacuum quando il parametro di CloudWatch `MaximumUsedTransactionIDs` raggiunge il valore del parametro `autovacuum_freeze_max_age` o 500.000.000, a seconda di quale sia quello maggiore.

Amazon RDS continua ad adattare i parametri per l'autovacuum se una tabella continua a tendere verso i wraparound dell'ID della transazione. Ognuno di questi aggiustamenti dedica più risorse all'autovacuum per evitare il wraparound. Amazon RDS aggiorna i seguenti parametri correlati all'autovacuum:

- autovacuum_vacuum_cost_delay
- autovacuum_vacuum_cost_limit
- `autovacuum_work_mem`

- autovacuum_naptime

RDS modifica questi parametri solo se il nuovo valore rende l'autovacuum più aggressivo. I parametri vengono modificati nella memoria sull'istanza database. I valori nel gruppo di parametri non vengono modificati. Per visualizzare le impostazioni in memoria correnti, utilizzare il comando SQL PostgreSQL SHOW.

Quando Amazon RDS modifica uno qualsiasi dei parametri autovacuum, genera un evento per l'istanza database interessata. Questo evento è visibile sulla Console di gestione AWS e tramite l'API Amazon RDS. Quando il parametro CloudWatch `MaximumUsedTransactionIDs` restituisce un valore a di sotto della soglia, Amazon RDS ripristina i parametri relativi all'autovacuum in memoria riportandoli ai valori specificati nel gruppo di parametri. Quindi genera un altro evento corrispondente a questa modifica.

Determinare se le tabelle nel database devono essere sottoposte a vacuum

La seguente query può essere utilizzata per mostrare il numero di transazioni sbloccate in un database. La colonna `datfrozenxid` di una riga di database `pg_database` è il margine inferiore sui normali ID di transazione che appaiono nel database. Questa colonna è il minimo dei valori `relfrozenxid` per tabella all'interno del database.

```
SELECT datname, age(datfrozenxid) FROM pg_database ORDER BY age(datfrozenxid) desc
 limit 20;
```

Ad esempio, i risultati dell'esecuzione della query precedente potrebbero essere i seguenti.

```
datname    | age
mydb       | 1771757888
template0  | 1721757888
template1  | 1721757888
rdsadmin   | 1694008527
postgres   | 1693881061
(5 rows)
```

Quando l'età di un database raggiunge i due miliardi di ID di transazioni, si verifica il wraparound ddell'ID della transazione (XID) e il database entra in modalità di sola lettura. Puoi utilizzare questa query per produrre un parametro ed eseguirla alcune volte al giorno. Per impostazione predefinita, l'autovacuum è impostato per mantenere l'età delle transazioni a non più di 200,000,000 (`autovacuum_freeze_max_age`).

Un esempio di strategia di monitoraggio potrebbe avere questo aspetto:

- Impostare il valore `autovacuum_freeze_max_age` su 200 milioni di transazioni.
- Se una tabella raggiunge 500 milioni di transazioni sbloccate, viene attivato un allarme a bassa gravità. Questo non è un valore irragionevole, ma può indicare che l'autovacuum non riesce a mantenere il passo.
- Se una tabella invecchia a un miliardo, questo dovrebbe essere trattato come un allarme per cui intervenire. In generale, si desidera mantenere le età più vicine a `autovacuum_freeze_max_age` per motivi di prestazioni. Si consiglia di investigare utilizzando le raccomandazioni che seguono.
- Se una tabella raggiunge 1,5 milioni di transazioni senza vacuum, viene attivato un allarme a gravità elevata. A seconda della velocità con cui il database utilizza gli ID delle transazioni, questo allarme può indicare che il tempo del sistema per eseguire l'autovacuum sta per scadere. In questo caso, consigliamo di risolvere il problema immediatamente.

Se una tabella superando costantemente queste soglie, modifica ulteriormente i parametri dell'autovacuum. Per impostazione predefinita, l'utilizzo manuale di VACUUM (che ha disabilitato i ritardi basati sui costi) è più aggressivo dell'autovacuum predefinito, ma è anche più intrusivo per il sistema nel suo complesso.

Consigliamo quanto segue:

- Attiva un meccanismo di monitoraggio in modo da essere consapevole dell'età delle transazioni più vecchie.

  Per informazioni sulla creazione di un processo che fornisce avvisi sul wraparound degli ID transazione, consulta il post nel blog di AWS Database Implementazione di un sistema di avvisi rapidi per il wraparound degli ID transazione in Amazon RDS for PostgreSQL.
- Per le tabelle più occupate, eseguire regolarmente un congelamento manuale del vacuum durante una finestra di manutenzione, oltre a fare affidamento sull'autovacuum. Per informazioni sull'esecuzione di un congelamento manuale del vacuum, consulta Esecuzione di un congelamento manuale del vacuum.

Determinare quali tabelle sono attualmente idonee per l'Autovacuum

Spesso, una o due tabelle hanno bisogno del vacuum. Le tabelle il cui valore `relfrozenxid` sia maggiore del numero di transazioni in `autovacuum_freeze_max_age` sono sempre destinate

all'autovacuum. Altrimenti, se il numero di tuple reso obsoleto dall'ultimo VACUUM supera la soglia del vacuum, la tabella viene sottoposta a vacuum.

La autovacuum threshold (soglia di autovacuum) viene definita come:

```
Vacuum-threshold = vacuum-base-threshold + vacuum-scale-factor * number-of-tuples
```

dove `vacuum base threshold` è `autovacuum_vacuum_threshold`, `vacuum scale factor` è `autovacuum_vacuum_scale_factor` e `number of tuples` è `pg_class.reltuples`.

Mentre sei connesso al database, esegui la seguente query per visualizzare un elenco di tabelle che Autovacuum vede come idonee per il vacuuming.

```
WITH vbt AS (SELECT setting AS autovacuum_vacuum_threshold FROM
pg_settings WHERE name = 'autovacuum_vacuum_threshold'),
vsf AS (SELECT setting AS autovacuum_vacuum_scale_factor FROM
pg_settings WHERE name = 'autovacuum_vacuum_scale_factor'),
fma AS (SELECT setting AS autovacuum_freeze_max_age FROM pg_settings WHERE name =
 'autovacuum_freeze_max_age'),
sto AS (select opt_oid, split_part(setting, '=', 1) as param,
split_part(setting, '=', 2) as value from (select oid opt_oid, unnest(reloptions)
 setting from pg_class) opt)
SELECT '"'||ns.nspname||'"."'||c.relname||'"' as relation,
pg_size_pretty(pg_table_size(c.oid)) as table_size,
age(relfrozenxid) as xid_age,
coalesce(cfma.value::float, autovacuum_freeze_max_age::float)
 autovacuum_freeze_max_age,
(coalesce(cvbt.value::float, autovacuum_vacuum_threshold::float) +
coalesce(cvsf.value::float,autovacuum_vacuum_scale_factor::float) * c.reltuples)
AS autovacuum_vacuum_tuples, n_dead_tup as dead_tuples FROM
pg_class c join pg_namespace ns on ns.oid = c.relnamespace
join pg_stat_all_tables stat on stat.relid = c.oid join vbt on (1=1) join vsf on (1=1)
 join fma on (1=1)
left join sto cvbt on cvbt.param = 'autovacuum_vacuum_threshold' and c.oid =
 cvbt.opt_oid
left join sto cvsf on cvsf.param = 'autovacuum_vacuum_scale_factor' and c.oid =
 cvsf.opt_oid
left join sto cfma on cfma.param = 'autovacuum_freeze_max_age' and c.oid = cfma.opt_oid
WHERE c.relkind = 'r' and nspname <> 'pg_catalog'
AND (age(relfrozenxid) >= coalesce(cfma.value::float, autovacuum_freeze_max_age::float)
OR coalesce(cvbt.value::float, autovacuum_vacuum_threshold::float) +
coalesce(cvsf.value::float,autovacuum_vacuum_scale_factor::float) *
```

```
c.reltuples <= n_dead_tup)
ORDER BY age(relfrozenxid) DESC LIMIT 50;
```

Determinare se l'Auotvacuum è attualmente in esecuzione e per quanto tempo

Se è necessario procedere manualmente con il vacuum in una tabella, devi determinare se l'autovacuum è attualmente in esecuzione. In tal caso, potrebbe essere necessario regolare i parametri per farlo eseguire in modo più efficiente oppure disattivare l'autovacuum temporaneamente in modo da poter eseguire manualmente il VACUUM.

Utilizzare la seguente query per determinare se l'autovacuum è in esecuzione, da quanto tempo è in esecuzione e se è in attesa su un'altra sessione.

```
SELECT datname, usename, pid, state, wait_event, current_timestamp - xact_start AS
 xact_runtime, query
FROM pg_stat_activity
WHERE upper(query) LIKE '%VACUUM%'
ORDER BY xact_start;
```

Dopo l'esecuzione della query, si dovrebbe visualizzare un output simile a quello riportato di seguito.

```
 datname | usename  |  pid  | state  | wait_event |      xact_runtime       | query
 --------+----------+-------+--------+------------+-------------------------
+-----------------------------------------------------------------------------------------------
 mydb    | rdsadmin | 16473 | active |            | 33 days 16:32:11.600656 |
 autovacuum: VACUUM ANALYZE public.mytable1 (to prevent wraparound)
 mydb    | rdsadmin | 22553 | active |            | 14 days 09:15:34.073141 |
 autovacuum: VACUUM ANALYZE public.mytable2 (to prevent wraparound)
 mydb    | rdsadmin | 41909 | active |            | 3 days 02:43:54.203349  |
 autovacuum: VACUUM ANALYZE public.mytable3
 mydb    | rdsadmin |   618 | active |            | 00:00:00                |
 SELECT datname, usename, pid, state, wait_event, current_timestamp - xact_start AS
 xact_runtime, query+
         |          |       |        |            |                         | FROM
 pg_stat_activity
           +
         |          |       |        |            |                         | WHERE
 query like '%VACUUM%'
          +
```

```
        |          |       |       |           |                         | ORDER BY
xact_start;
      +
```

Diversi problemi possono provocare sessioni di autovacuum di lunga esecuzione (che durano più giorni). Il problema più comune è che il valore del parametro `maintenance_work_mem` è impostato come troppo basso per la dimensione della tabella o la frequenza degli aggiornamenti.

Consigliamo di utilizzare la seguente formula per impostare il valore del parametro `maintenance_work_mem`.

```
GREATEST({DBInstanceClassMemory/63963136*1024},65536)
```

Le sessioni autovacuum a breve esecuzioni possono anche indicare dei problemi:

- Può indicare che non ci sono abbastanza `autovacuum_max_workers` per il carico di lavoro. In questo caso, sarà necessario indicare il numero di lavoratori.
- Può indicare che esiste un danneggiamento dell'indice (l'autovacuum si blocca e si riavvia sulla stessa relazione ma non ci sono progressi). In questo caso, esegui un `vacuum freeze verbose` *`table`* manuale per vedere la causa esatta.

Esecuzione di un congelamento manuale del vacuum

Si potrebbe voler eseguire un vacuum manuale su una tabella che ha già un processo di vacuum in esecuzione. Ciò è utile se hai identificato una tabella con un'età che si avvicina a 2 miliardi (o al di sopra di qualsiasi soglia monitorata).

I seguenti passaggi sono delle linea guida, con diverse varianti del processo. Ad esempio, durante la verifica, supporre che il valore del parametro `maintenance_work_mem` sia stato impostato come troppo piccolo e che sia necessario agire immediatamente su una tabella. Tuttavia, probabilmente al momento non desideri che l'istanza non venga recapitata. Utilizzando le query delle sezioni precedenti, si determina quale tabella rappresenta il problema e si nota una sessione autovacuum a lunga esecuzione. Si sa che è necessario cambiare l'impostazione del parametro `maintenance_work_mem`, ma è necessario anche agire immediatamente ed eseguire il vacuum della tabella in questione. La procedura seguente mostra cosa fare in questa situazione.

Per eseguire manualmente un congelamento del vacuum

1. Aprire due sessioni nel database che contiene la tabella che si desidera sottoporre a vacuum. Per la seconda sessione, utilizzare "screen" o un'altra utility che mantiene la sessione se la connessione viene interrotta.
2. Nella sessione uno, ottieni l'ID di processo (PID) della sessione di autovacuum in esecuzione sulla tabella.

   Eseguire la query seguente per ottenere il PID della sessione di autovacuum.

   ```
   SELECT datname, usename, pid, current_timestamp - xact_start
   AS xact_runtime, query
   FROM pg_stat_activity WHERE upper(query) LIKE '%VACUUM%' ORDER BY
   xact_start;
   ```
3. Nella sessione due, calcoli la quantità di memoria necessaria per questa operazione. In questo esempio, stabiliamo che è possibile permettersi di utilizzare fino a 2 GB di memoria per questa operazione, pertanto abbiamo impostato `maintenance_work_mem` per la sessione corrente su 2 GB.

   ```
   SET maintenance_work_mem='2 GB';
   SET
   ```
4. Nella sessione 2, inviare un comando `vacuum freeze verbose` per la tabella. L'impostazione di verbose è utile perché, anche se al momento non vi è alcun rapporto sullo stato di avanzamento in PostgreSQL, è possibile visualizzare l'attività.

   ```
   \timing on
   Timing is on.
   vacuum freeze verbose pgbench_branches;
   ```

   ```
   INFO:  vacuuming "public.pgbench_branches"
   INFO:  index "pgbench_branches_pkey" now contains 50 row versions in 2 pages
   DETAIL:  0 index row versions were removed.
   0 index pages have been deleted, 0 are currently reusable.
   CPU 0.00s/0.00u sec elapsed 0.00 sec.
   INFO:  index "pgbench_branches_test_index" now contains 50 row versions in 2 pages
   DETAIL:  0 index row versions were removed.
   0 index pages have been deleted, 0 are currently reusable.
   CPU 0.00s/0.00u sec elapsed 0.00 sec.
   ```

```
INFO:  "pgbench_branches": found 0 removable, 50 nonremovable row versions
     in 43 out of 43 pages
DETAIL:  0 dead row versions cannot be removed yet.
There were 9347 unused item pointers.
0 pages are entirely empty.
CPU 0.00s/0.00u sec elapsed 0.00 sec.
VACUUM
Time: 2.765 ms
```

5. Nella sessione uno, se il processo di autovacuum bloccava la sessione di vacuum, `pg_stat_activity` mostra che l'attesa è T per la sessione di vacuum. In questo caso, termina il processo di autovacuum come segue.

```
SELECT pg_terminate_backend('the_pid');
```

> **Note**
>
> Alcune versioni precedenti di Amazon Aurora non possono terminare un processo di autovacuum utilizzando il comando precedente e viene generato il seguente errore: `ERROR: 42501: must be a superuser to terminate superuser process LOCATION: pg_terminate_backend, signalfuncs.c:227`. Per trovare le versioni di PostgreSQL a cui è stata applicata la patch, cerca il seguente punto negli aggiornamenti di PostgreSQL di Amazon Aurora:
>
> ```
> Allow rds_superuser to terminate backends which are not explicitly
>  associated with a role
> ```

   In questo momento, inizia la sessione. La funzionalità di autovacuum si riavvia immediatamente poiché questa tabella ha probabilmente la posizione superiore nella lista di lavoro.

6. Avvia il comando `vacuum freeze verbose` nella sessione due, quindi termina il processo di autovacuum nella sessione uno.

Indicizzare di nuovo una tabella quando l'autovacuum è in esecuzione

Se un indice diventa corrotto, l'autovacuum continua a elaborare la tabella e avrà esito negativo. Setenti di eseguire un vacuum manuale in questa situazione, riceverai un messaggio di errore come il seguente.

```
postgres=>  vacuum freeze pgbench_branches;
ERROR: index "pgbench_branches_test_index" contains unexpected
   zero page at block 30521
HINT: Please REINDEX it.
```

Quando l'indice è corrotto e l'autovacuum sta tentando l'esecuzione sulla tabella, ci sarà una contesa con una sessione di autovacuum già in esecuzione. Quando si immette un comando REINDEX, si richiede un blocco esclusivo sulla tabella. Le operazioni in scrittura sono bloccate e anche quelle in lettura che utilizzano l'indice specifico.

Per indicizzare di nuovo una tabella quando l'autovacuum è in esecuzione sulla tabella

1. Apri due sessioni nel database che contiene la tabella da sottoporre a vacuum. Per la seconda sessione, utilizzare "screen" o un'altra utility che mantiene la sessione se la connessione viene interrotta.
2. Nella sessione numero uno, ottenere il PID della sessione di autovacuum in esecuzione sulla tabella.

   Eseguire la query seguente per ottenere il PID della sessione di autovacuum.

   ```
   SELECT datname, usename, pid, current_timestamp - xact_start
   AS xact_runtime, query
   FROM pg_stat_activity WHERE upper(query) like '%VACUUM%' ORDER BY
   xact_start;
   ```

3. Nella sessione due, rilasciare il comando di reindicizzazione.

   ```
   \timing on
   Timing is on.
   reindex index pgbench_branches_test_index;
   REINDEX
     Time: 9.966 ms
   ```

4. Nella sessione uno, se l'autovacuum bloccava il processo, in `pg_stat_activity` vedi che l'attesa è "T" per la sessione di vacuum. In questo caso, termina il processo di autovacuum.

   ```
   SELECT pg_terminate_backend('the_pid');
   ```

   In questo momento, inizia la sessione. È importante notare che l'autovacuum si riavvia immediatamente poiché questa tabella è probabilmente la più alta nella lista di lavori.

5. Avvia il comando nella sessione due, quindi termina il processo di autovacuum nella sessione 1.

Gestione di autovacuum con indici di grandi dimensioni

Come parte del funzionamento, autovacuum esegue diverse fasi di vacuum mentre viene eseguito su una tabella. Prima che la tabella venga pulita, tutti i suoi indici vengono prima sottoposti al vacuum. Quando si rimuovono più indici di grandi dimensioni, questa fase richiede una notevole quantità di tempo e risorse. Pertanto, come best practice, assicurati di controllare il numero di indici in una tabella ed eliminare gli indici non utilizzati.

Per questo processo, controlla innanzitutto la dimensione complessiva degli indici. Quindi, determina se ci sono indici potenzialmente inutilizzati da rimuovere come mostrato negli esempi seguenti.

Per verificare la dimensione della tabella e dei relativi indici

```
postgres=> select pg_size_pretty(pg_relation_size('pgbench_accounts'));
pg_size_pretty
6404 MB
(1 row)
```

```
postgres=> select pg_size_pretty(pg_indexes_size('pgbench_accounts'));
pg_size_pretty
11 GB
(1 row)
```

In questo esempio, la dimensione degli indici è maggiore della tabella. Questa differenza può causare problemi di prestazioni perché gli indici sono aumentati in dimensioni o inutilizzati, il che influisce sull'autovacuum e sulle operazioni di inserimento.

Per verificare la presenza di indici non utilizzati

Utilizzando la visualizzazione `pg_stat_user_indexes`, è possibile verificare la frequenza con cui viene utilizzato un indice con la colonna `idx_scan`. Nell'esempio seguente, gli indici non utilizzati hanno `idx_scan` con il valore `0`.

```
postgres=> select * from pg_stat_user_indexes where relname = 'pgbench_accounts' order
 by idx_scan desc;

relid  | indexrelid | schemaname | relname          | indexrelname          | idx_scan
 | idx_tup_read | idx_tup_fetch
```

```
-------+------------+------------+-----------------+-----------------------+----------
+--------------+---------------
16433  | 16454      | public     | pgbench_accounts | index_f               | 6
 | 6            | 0
16433  | 16450      | public     | pgbench_accounts | index_b               | 3
 | 199999       | 0
16433  | 16447      | public     | pgbench_accounts | pgbench_accounts_pkey | 0
 | 0            | 0
16433  | 16452      | public     | pgbench_accounts | index_d               | 0
 | 0            | 0
16433  | 16453      | public     | pgbench_accounts | index_e               | 0
 | 0            | 0
16433  | 16451      | public     | pgbench_accounts | index_c               | 0
 | 0            | 0
16433  | 16449      | public     | pgbench_accounts | index_a               | 0
 | 0            | 0
(7 rows)
```

```
postgres=> select schemaname, relname, indexrelname, idx_scan from pg_stat_user_indexes
 where relname = 'pgbench_accounts' order by idx_scan desc;

schemaname  | relname          | indexrelname          | idx_scan
------------+------------------+-----------------------+----------
public      | pgbench_accounts | index_f               | 6
public      | pgbench_accounts | index_b               | 3
public      | pgbench_accounts | pgbench_accounts_pkey | 0
public      | pgbench_accounts | index_d               | 0
public      | pgbench_accounts | index_e               | 0
public      | pgbench_accounts | index_c               | 0
public      | pgbench_accounts | index_a               | 0
(7 rows)
```

Note

Queste statistiche sono incrementali dal momento in cui vengono ripristinate. Supponi di avere un indice utilizzato solo alla fine di un trimestre lavorativo o solo per un report specifico. È possibile che questo indice non sia stato utilizzato da quando le statistiche sono state ripristinate. Per ulteriori informazioni, consulta Funzioni statistiche. Gli indici utilizzati per garantire l'univocità non vengono sottoposti ad analisi e non devono essere identificati come

indici non utilizzati. Per identificare gli indici non utilizzati, è necessario avere una conoscenza approfondita dell'applicazione e delle relative query.

Per verificare quando le statistiche sono state ripristinate l'ultima volta per un database, usa pg_stat_database

```
postgres=> select datname, stats_reset from pg_stat_database where datname =
 'postgres';

datname   | stats_reset
----------+-------------------------------
postgres  | 2022-11-17 08:58:11.427224+00
(1 row)
```

Vacuum di una tabella il più rapidamente possibile

RDS per PostgreSQL 12 e versioni successive

Se sono presenti troppi indici in una tabella di grandi dimensioni, l'istanza database potrebbe essere vicina al wraparound dell'ID di transazione (XID), ovvero quando il contatore XID arriva a zero. Se non controllata, questa situazione potrebbe causare la perdita di dati. Tuttavia, è possibile eseguire rapidamente il vacuum della tabella senza ripulire gli indici. In RDS per PostgreSQL 12 e versioni successive, puoi usare VACUUM con la clausola INDEX_CLEANUP.

```
postgres=> VACUUM (INDEX_CLEANUP FALSE, VERBOSE TRUE) pgbench_accounts;

INFO: vacuuming "public.pgbench_accounts"
INFO: table "pgbench_accounts": found 0 removable, 8 nonremovable row versions in 1 out
 of 819673 pages
DETAIL: 0 dead row versions cannot be removed yet, oldest xmin: 7517
Skipped 0 pages due to buffer pins, 0 frozen pages.
CPU: user: 0.01 s, system: 0.00 s, elapsed: 0.01 s.
```

Se è già in esecuzione una sessione di autovacuum, è necessario interromperla per iniziare il VACUUM manuale. Per informazioni sull'esecuzione di un congelamento manuale del vacuum, consulta Esecuzione di un congelamento manuale del vacuum.

> **Note**
>
> Se si evita di eseguire regolarmente la pulizia, si verifica il bloat dell'indice, con ripercussioni sulle prestazioni della scansione. L'indice mantiene le righe morte e la tabella mantiene i puntatori delle linee morte. Di conseguenza, `pg_stat_all_tables.n_dead_tup` aumenta finché non viene eseguito un processo di autovacuum o un VACUUM manuale con pulizia dell'indice. Come best practice, utilizza questa procedura solo per evitare il wraparound degli ID di transazione.

RDS per PostgreSQL 11 e versioni precedenti

Tuttavia, in RDS per PostgreSQL 11 e versioni precedenti, l'unico modo per eseguire il vacuum più rapidamente è riducendo il numero di indici su una tabella. L'eliminazione di un indice può influire sui piani di query. Ti consigliamo di eliminare prima gli indici inutilizzati, quindi quelli che hanno il wraparound XID molto vicino. Una volta completato il processo di vacuum, è possibile ricreare questi indici.

Altri parametri che influenzano l'autovacuum

La query seguente mostra i valori di alcuni dei parametri che influenzano direttamente l'autovacuum e il suo comportamento. I parametri di autovacuum vengono descritti in forma completa nella documentazione di PostgreSQL.

```
SELECT name, setting, unit, short_desc
FROM pg_settings
WHERE name IN (
'autovacuum_max_workers',
'autovacuum_analyze_scale_factor',
'autovacuum_naptime',
'autovacuum_analyze_threshold',
'autovacuum_analyze_scale_factor',
'autovacuum_vacuum_threshold',
'autovacuum_vacuum_scale_factor',
'autovacuum_vacuum_threshold',
'autovacuum_vacuum_cost_delay',
'autovacuum_vacuum_cost_limit',
'vacuum_cost_limit',
'autovacuum_freeze_max_age',
'maintenance_work_mem',
```

```
'vacuum_freeze_min_age');
```

Mentre questi riguardano tutti l'autovacuum, alcuni dei più importanti sono:

- maintenance_work_mem
- autovacuum_freeze_max_age
- autovacuum_max_workers
- autovacuum_vacuum_cost_delay
- autovacuum_vacuum_cost_limit

Impostazione dei parametri di autovacuum a livello tabella

Puoi impostare i parametri di archiviazione correlati all'autovacuum a livello di tabella, che può essere meglio di alterare il comportamento dell'intero database. Per le tabelle di grandi dimensioni, potrebbe essere necessario regolare impostazioni aggressive e si potrebbe non desiderare di eseguire l'autovacuum in questo modo per tutte le tabelle.

La query seguente mostra quali tabelle attualmente dispongono di opzioni a livello di tabella.

```
SELECT relname, reloptions
FROM pg_class
WHERE reloptions IS NOT null;
```

Un esempio in cui ciò potrebbe essere utile è per tabelle che sono molto più grandi rispetto al resto delle tabelle. Supponi di disporre di una tabella da 300 GB e di altre 30 tabelle da meno di un GB. Se disponi di una tabella da 300 GB e di altre 30 tabelle da meno di 1 GB, puoi impostare alcuni parametri specifici per la tabella di grandi dimensioni in modo da non alterare il comportamento dell'intero sistema.

```
ALTER TABLE mytable set (autovacuum_vacuum_cost_delay=0);
```

In questo modo si disattiva il ritardo dell'autovacuum basato sul costo per questa tabella a spese di un maggiore utilizzo delle risorse sul sistema. Normalmente, l'autovacuum si ferma per `autovacuum_vacuum_cost_delay` ogni volta che viene raggiunto `autovacuum_cost_limit`. Per ulteriori dettagli, consulta la documentazione di PostegreSQL relativa al vacuuming basato sul costo.

Registrazione delle attività di autovacuum e vacuum

Le informazioni sulle attività dell'autovacuum vengono inviate a `postgresql.log` in base al livello specificato nel parametro `rds.force_autovacuum_logging_level`. Di seguito sono riportati i valori consentiti per questo parametro e le versioni di PostgreSQL per le quali tale valore è l'impostazione predefinita:

- `disabled` (PostgreSQL 10, PostgreSQL 9.6)
- `debug5`, `debug4`, `debug3`, `debug2`, `debug1`
- `info` (PostgreSQL 12, PostgreSQL 11)
- `notice`
- `warning` (PostgreSQL 13 e versioni successive)
- `error`, log, `fatal`, `panic`

`rds.force_autovacuum_logging_level` funziona con il parametro `log_autovacuum_min_duration`. Il valore del parametro `log_autovacuum_min_duration` è la soglia (in millisecondi) al di sopra della quale vengono registrate le azioni dell'autovacuum. Un ambiente di `-1` non registra nulla, mentre un'impostazione di 0 registra tutte le azioni. Come con `rds.force_autovacuum_logging_level`, i valori predefiniti per `log_autovacuum_min_duration` dipendono dalla versione, come segue:

- `10000 ms` – PostgreSQL 14, PostgreSQL 13, PostgreSQL 12 e PostgreSQL 11
- `(empty)` – Nessun valore predefinito per PostgreSQL 10 e PostgreSQL 9.6

Consigliamo di impostare `rds.force_autovacuum_logging_level` su `WARNING`. Consigliamo anche di impostare `log_autovacuum_min_duration` su un valore compreso tra 1000 e 5000. Un'impostazione di 5000 registri di attività che richiede più di 5000 millisecondi. Qualsiasi impostazione diversa da -1 registra anche i messaggi se l'azione dell'autovacuum viene ignorata a causa di un blocco in conflitto o di relazioni interrotte simultaneamente. Per ulteriori informazioni, consulta la pagina relativa al vacuum automatico nella documentazione di PostgreSQL.

Per risolvere i problemi, è possibile modificare il parametro `rds.force_autovacuum_logging_level` in uno dei livelli di debug, da `debug1` fino a `debug5` per le informazioni più dettagliate. Si consiglia di utilizzare le impostazioni di debug per brevi periodi di tempo e solo per la risoluzione dei problemi. Per ulteriori informazioni, consulta Quando registrare nella documentazione di PostgreSQL.

> **Note**
>
> PostgreSQL consente all'account `rds_superuser` di visualizzare le sessioni di autovacuum in `pg_stat_activity`. Ad esempio, è possibile identificare e terminare una sessione di autovacuum che blocca l'esecuzione di un comando o l'esecuzione più lenta di un comando vacuum emesso manualmente.

Comprensione del comportamento della funzionalità di autovacuum con database non validi

Viene introdotto un nuovo valore `-2` nella colonna `datconnlimit` del catalogo `pg_database` per indicare come non validi i database che sono stati interrotti durante l'operazione DROP DATABASE.

Questo nuovo valore è disponibile nelle seguenti versioni di Aurora PostgreSQL:

- 15.4 e tutte le versioni successive
- 14.9 e versioni successive
- 13.12 e versioni successive
- 12.16 e versioni successive
- 11.21 e versioni successive

I database non validi non influiscono sulla capacità della funzionalità di autovacuum di bloccare le funzionalità dei database validi. La funzionalità autovacuum ignora i database non validi. Di conseguenza, le normali operazioni di vacuum continueranno a funzionare correttamente ed efficacemente per tutti i database validi nell'ambiente PostgreSQL.

Argomenti

- Monitoraggio dell'ID di transazione
- Modifica della query di monitoraggio
- Risoluzione dei problemi relativi a database non validi

Monitoraggio dell'ID di transazione

La funzione `age(datfrozenxid)` viene comunemente utilizzata per monitorare l'età degli ID di transazione (XID) dei database per evitare che si verifichi il wraparound degli ID.

Poiché i database non validi vengono esclusi dall'autovacuum, il relativo contatore dell'ID di transazione (XID) può raggiungere il valore massimo di `2 billion`, eseguire il wraparound a `- 2 billion` e continuare questo ciclo all'infinito. Una tipica query per monitorare il wraparound degli ID di transazione potrebbe essere simile a:

```
SELECT max(age(datfrozenxid)) FROM pg_database;
```

Tuttavia, con l'introduzione del valore -2 per `datconnlimit`, i database non validi possono alterare i risultati di questa query. Poiché questi database non sono validi e non devono essere sottoposti a regolari controlli di manutenzione, possono causare falsi positivi, inducendo a credere che il valore `age(datfrozenxid)` sia superiore a quello reale.

Modifica della query di monitoraggio

Per garantire l'accuratezza del monitoraggio, è necessario modificare la query di monitoraggio per escludere i database non validi. Segui questa query consigliata:

```
SELECT
    max(age(datfrozenxid))
FROM
    pg_database
WHERE
    datconnlimit <> -2;
```

Questa query garantisce che nel calcolo di `age(datfrozenxid)` vengano presi in considerazione solo i database validi, fornendo un riflesso fedele dell'età degli ID di transazione nell'ambiente PostgreSQL.

Risoluzione dei problemi relativi a database non validi

Quando tenti di connetterti a un database non valido, potrebbe essere visualizzato un messaggio di errore simile al seguente:

```
postgres=> \c db1
connection to server at "mydb.xxxxxxxxxx.us-west-2.rds.amazonaws.com" (xx.xx.xx.xxx),
 port xxxx failed: FATAL:  cannot connect to invalid database "db1"
```

```
HINT:  Use DROP DATABASE to drop invalid databases.
Previous connection kept
```

Inoltre, se il parametro `log_min_messages` è impostato su `DEBUG2` o successivo, è possibile che vengano visualizzate le seguenti voci di log che indicano che il processo autovacuum sta ignorando il database non valido:

```
2024-07-30 05:59:00 UTC::@:[32000]:DEBUG:  autovacuum: skipping invalid database "db6"
2024-07-30 05:59:00 UTC::@:[32000]:DEBUG:  autovacuum: skipping invalid database "db1"
```

Per risolvere il problema, segui il valore `HINT` fornito durante il tentativo di connessione. Connettiti a qualsiasi database valido utilizzando l'account master RDS o un account di database con il ruolo `rds_superuser` ed elimina i database non validi.

```
SELECT
    'DROP DATABASE ' || quote_ident(datname) || ';'
FROM
    pg_database
WHERE
    datconnlimit = -2 \gexec
```

Identificazione e risoluzione dei blocchi per i processi di vacuum aggressivi in Aurora PostgreSQL

In PostgreSQL, il processo di vacuum è fondamentale per garantire l'integrità del database poiché consente di recuperare spazio di archiviazione e previene eventuali problemi di wraparound degli ID transazione. Tuttavia, in alcune situazioni è possibile che il processo di vacuum non funzioni come desiderato; le conseguenze possono essere: riduzione delle prestazioni, bloat dello spazio di archiviazione e impatto sulla disponibilità dell'istanza database in uso a causa del wraparound degli ID transazione. Pertanto, l'identificazione e la risoluzione dei suddetti problemi sono essenziali per garantire prestazioni e disponibilità ottimali del database. Per ulteriori informazioni sul processo di autovacuum, consulta Understanding autovacuum in Amazon RDS for PostgreSQL environments.

La funzione `postgres_get_av_diag()` aiuta a identificare i problemi che impediscono o ritardano l'avanzamento dei processi di vacuum aggressivi. Vengono forniti dei suggerimenti, che possono includere comandi per risolvere il problema laddove esso sia identificabile o indicazioni per ulteriori operazioni di diagnosi laddove il problema non sia identificabile. Eventuali elementi bloccanti per i processi di vacuum aggressivi vengono segnalati quando l'età supera la soglia di autovacuum adattivo di RDS, che corrisponde a 500 milioni di ID transazione.

Qual è l'età dell'ID transazione?

La funzione `age()` per gli ID transazione calcola il numero di transazioni che sono state eseguite a partire dall'ID transazione non bloccato meno recente per un database (`pg_database.datfrozenxid`) o una tabella (`pg_class.relfrozenxid`). Questo valore indica l'attività del database dall'ultima operazione di vacuum aggressiva ed evidenzia il probabile carico di lavoro per i successivi processi VACUUM.

In cosa consiste un processo di vacuum aggressivo?

Un'operazione VACUUM aggressiva esegue una scansione completa di tutte le pagine di una tabella, comprese quelle che solitamente vengono ignorate durante i normali processi VACUUM. Questa scansione approfondita mira a bloccare gli ID transazione che si avvicinano alla loro età massima, prevenendo in modo efficace un problema noto come wraparound degli ID transazione.

Affinché `postgres_get_av_diag()` li segnali, gli elementi bloccanti devono avere un'età di almeno 500 milioni di transazioni.

Argomenti

- Installazione di strumenti di diagnostica e monitoraggio dei processi di autovacuum in Aurora PostgreSQL
- Funzioni di postgres_get_av_diag() in Aurora PostgreSQL
- Risoluzione dei blocchi identificabili per i processi di vacuum in Aurora PostgreSQL
- Risoluzione dei blocchi non identificabili per i processi di vacuum in Aurora PostgreSQL
- Risoluzione dei problemi relativi alle prestazioni di vacuum in Aurora PostgreSQL
- Spiegazione dei messaggi NOTICE in Aurora PostgreSQL

Installazione di strumenti di diagnostica e monitoraggio dei processi di autovacuum in Aurora PostgreSQL

La funzione `postgres_get_av_diag()` è attualmente disponibile nelle seguenti versioni di Aurora PostgreSQL:

- 17.4 e versioni 17 successive
- 16.7 e versioni 16 successive
- 15.11 e versioni 15 successive
- 14.16 e versioni 14 successive
- 13.19 e versioni 13 successive

Per utilizzare `postgres_get_av_diag()`, crea l'estensione `rds_tools`.

```
postgres=> CREATE EXTENSION rds_tools ;
CREATE EXTENSION
```

Verifica che l'estensione sia stata installata.

```
postgres=> \dx rds_tools
             List of installed extensions
   Name    | Version |  Schema   |                    Description
 ----------+---------+-----------
+-------------------------------------------------------------
 rds_tools |   1.9   | rds_tools | miscellaneous administrative functions for RDS
 PostgreSQL
 1 row
```

Verifica che la funzione sia stata creata.

```
postgres=> SELECT
    proname function_name,
    pronamespace::regnamespace function_schema,
    proowner::regrole function_owner
FROM
    pg_proc
WHERE
    proname = 'postgres_get_av_diag';
    function_name     | function_schema | function_owner
```

```
----------------------+-----------------+----------------
 postgres_get_av_diag | rds_tools       | rds_superuser
(1 row)
```

Funzioni di postgres_get_av_diag() in Aurora PostgreSQL

La funzione `postgres_get_av_diag()` recupera informazioni diagnostiche sui processi di autovacuum che si bloccano o sono in ritardo in un database Aurora PostgreSQL. Per ottenere risultati accurati, la query deve essere eseguita nel database con l'ID transazione meno recente. Per ulteriori informazioni sull'utilizzo del database con l'ID transazione meno recente, consulta Mancata connessione al database con l'età dell'ID transazione meno recente.

```
SELECT
    blocker,
    DATABASE,
    blocker_identifier,
    wait_event,
    TO_CHAR(autovacuum_lagging_by, 'FM9,999,999,999') AS autovacuum_lagging_by,
    suggestion,
    suggested_action
FROM (
    SELECT
        *
    FROM
        rds_tools.postgres_get_av_diag ()
    ORDER BY
        autovacuum_lagging_by DESC) q;
```

La funzione `postgres_get_av_diag()` restituisce una tabella con le seguenti informazioni:

elemento bloccante

Specifica la categoria di attività del database che blocca il processo di vacuum.

- Istruzione attiva
- Inattivo in transazione
- Transazione preparata
- Slot di replica logica
- Istanze di lettura
- Tabelle temporanee

database

Specifica il nome del database, ove pertinente e supportato. Si tratta del database in cui l'attività è in corso e che blocca o bloccherà il processo di autovacuum. È il database a cui devi connetterti e sul quale devi agire.

blocker_identifier

Specifica l'identificatore dell'attività che blocca o bloccherà il processo di autovacuum. L'identificatore può essere un ID processo insieme a un'istruzione SQL, una transazione preparata, un indirizzo IP di una replica di lettura e il nome dello slot di replica (logica o fisica).

wait_event

Specifica l'evento di attesa della sessione di blocco ed è applicabile ai seguenti elementi bloccanti:

- Istruzione attiva
- Inattivo in transazione

autovacum_lagging_by

Specifica il numero di transazioni di cui il processo di autovacuum è in ritardo nel lavoro di backlog per categoria.

suggestion

Specifica i suggerimenti per risolvere il blocco. Queste istruzioni includono il nome del database in cui esistono l'attività (ove applicabile), l'ID processo (PID) della sessione (ove applicabile) e l'azione da intraprendere.

suggested_action

Suggerisci l'azione che è necessario intraprendere per risolvere il blocco.

Risoluzione dei blocchi identificabili per i processi di vacuum in Aurora PostgreSQL

La funzione di autovacuum esegue processi di vacuum aggressivi e porta l'età degli ID transazione al di sotto della soglia specificata dal parametro `autovacuum_freeze_max_age` dell'istanza RDS in uso. Puoi tenere traccia del valore di età utilizzando la metrica `MaximumUsedTransactionIDs` di Amazon CloudWatch.

Per trovare l'impostazione di `autovacuum_freeze_max_age` (che ha un valore predefinito di 200 milioni di ID transazione) per la tua istanza Amazon RDS, puoi utilizzare la seguente query:

```
SELECT
    TO_CHAR(setting::bigint, 'FM9,999,999,999') autovacuum_freeze_max_age
FROM
    pg_settings
WHERE
    name = 'autovacuum_freeze_max_age';
```

`postgres_get_av_diag()` verifica la presenza di eventuali elementi bloccanti per i processi di vacuum aggressivi solo quando l'età supera la soglia di autovacuum adattivo di Amazon RDS, che corrisponde a 500 milioni di ID transazione. Affinché `postgres_get_av_diag()` li rilevi, gli elementi bloccanti devono avere un'età di almeno 500 milioni di transazioni.

La funzione `postgres_get_av_diag()` identifica i seguenti tipi di elementi bloccanti:

Argomenti

- Istruzione attiva
- Inattivo in transazione
- Transazione preparata
- Slot di replica logica
- Istanze di lettura
- Tabelle temporanee

Istruzione attiva

In PostgreSQL, un'istruzione attiva è un'istruzione SQL attualmente che è in fase di esecuzione da parte del database. Sono incluse query, transazioni o eventuali operazione in corso. Durante il monitoraggio tramite `pg_stat_activity`, la colonna dello stato indica che il processo con il PID corrispondente è attivo.

La funzione `postgres_get_av_diag()` visualizza un output simile a quanto segue quando identifica un'istruzione che è attiva.

```
blocker               | Active statement
database              | my_database
blocker_identifier    | SELECT pg_sleep(20000);
wait_event            | Timeout:PgSleep
autovacuum_lagging_by | 568,600,871
```

```
suggestion            | Connect to database "my_database", review carefully and you
 may consider terminating the process using suggested_action. For more information, see
 Working with PostgreSQL autovacuum in the Amazon RDS User Guide.
suggested_action      | {"SELECT pg_terminate_backend (29621);"}
```

Operazione suggerita

Seguendo le indicazioni riportate nella colonna `suggestion`, l'utente può connettersi al database in cui è presente l'istruzione attiva e, come specificato nella colonna `suggested_action`, è consigliabile esaminare attentamente l'opzione per terminare la sessione. Se è sicuro eseguire la terminazione, è possibile utilizzare la funzione `pg_terminate_backend()` per terminare la sessione. Questa azione può essere eseguita da un utente amministratore (come l'account principale RDS) o da un utente con il privilegio `pg_terminate_backend()` richiesto.

> **Warning**
>
> Una sessione terminata annullerà (`ROLLBACK`) le modifiche apportate. A seconda delle esigenze del caso, potresti voler eseguire nuovamente l'istruzione. Tuttavia, si consiglia di eseguire tale operazione solo dopo che il processo di autovacuum ha terminato il processo di vacuum aggressivo.

Inattivo in transazione

Un'istruzione di inattività in transazione si riferisce a una sessione che ha aperto una transazione esplicita (ad esempio, emettendo un'istruzione `BEGIN`), ha eseguito del lavoro e ora è in attesa che il client passi altro lavoro o segnali la fine della transazione emettendo un'istruzione `COMMIT`, `ROLLBACK` o `END` (che darebbe luogo a un'operazione `COMMIT` implicita).

La funzione `postgres_get_av_diag()` visualizza un output simile a quanto segue quando identifica un'istruzione `idle in transaction` come elemento bloccante.

```
blocker               | idle in transaction
database              | my_database
blocker_identifier    | INSERT INTO tt SELECT * FROM tt;
wait_event            | Client:ClientRead
autovacuum_lagging_by | 1,237,201,759
suggestion            | Connect to database "my_database", review carefully and you
 may consider terminating the process using suggested_action. For more information, see
 Working with PostgreSQL autovacuum in the Amazon RDS User Guide.
```

```
suggested_action      | {"SELECT pg_terminate_backend (28438);"}
```

Operazione suggerita

Come indicato nella colonna `suggestion`, è possibile connettersi al database in cui è presente una sessione di inattività in transazione e terminare la sessione utilizzando la funzione `pg_terminate_backend()`. L'utente può essere l'utente amministratore (account principale RDS) o un utente con il privilegio `pg_terminate_backend()`.

> **Warning**
>
> Una sessione terminata annullerà (`ROLLBACK`) le modifiche apportate. A seconda delle esigenze del caso, potresti voler eseguire nuovamente l'istruzione. Tuttavia, si consiglia di eseguire tale operazione solo dopo che il processo di autovacuum ha terminato il processo di vacuum aggressivo.

Transazione preparata

PostgreSQL consente le transazioni che fanno parte di una strategia di commit in due fasi, denominate transazioni preparate. Queste vengono abilitate impostando il parametro `max_prepared_transactions` su un valore diverso da zero. Le transazioni preparate sono concepite per garantire che una transazione sia duratura e rimanga disponibile anche dopo crash del database, riavvii o disconnessioni del client. Così come avviene con le transazioni standard, anche le transazioni preparate ricevono in assegnazione un ID transazione e possono influire sul processo di autovacuum. Se lasciato in stato preparato, il processo di autovacuum non può eseguire il blocco e può portare al wraparound degli ID transazione.

Quando le transazioni vengono lasciate in stato preparato indefinitamente senza essere risolte da un gestore di transazioni, diventano transazioni preparate orfane. L'unica soluzione consiste nell'eseguire il commit o il rollback della transazione utilizzando rispettivamente i comandi `COMMIT PREPARED` o `ROLLBACK PREPARED`.

> **Note**
>
> Tieni presente che un backup eseguito durante una transazione preparata conterrà comunque tale transazione dopo il ripristino. Consulta le seguenti informazioni su come individuare e chiudere tali transazioni.

La funzione `postgres_get_av_diag()` visualizza il seguente output quando identifica un elemento bloccante che è una transazione preparata.

```
blocker               | Prepared transaction
database              | my_database
blocker_identifier    | myptx
wait_event            | Not applicable
autovacuum_lagging_by | 1,805,802,632
suggestion            | Connect to database "my_database" and consider either COMMIT
 or ROLLBACK the prepared transaction using suggested_action. For more information, see
 Working with PostgreSQL autovacuum in the Amazon RDS User Guide.
suggested_action      | {"COMMIT PREPARED 'myptx';",[OR],"ROLLBACK PREPARED 'myptx';"}
```

Operazione suggerita

Come indicato nella colonna dei suggerimenti, connettiti al database in cui si trova la transazione preparata. In base alla colonna `suggested_action`, valuta attentamente se eseguire un'operazione di `COMMIT` o `ROLLBACK` e poi l'azione appropriata.

Per monitorare le transazioni preparate in generale, PostgreSQL offre una vista in stile catalogo denominata `pg_prepared_xacts`. La seguente query può essere utilizzata per trovare le transazioni preparate.

```
SELECT
    gid,
    prepared,
    owner,
    database,
    transaction AS oldest_xmin
FROM
    pg_prepared_xacts
ORDER BY
    age(transaction) DESC;
```

Slot di replica logica

Lo scopo di uno slot di replica è conservare le modifiche non utilizzate finché non vengono replicate su un server di destinazione. Per ulteriori informazioni, consulta la pagina relativa alla replica logica di PostgreSQL.

Esistono due tipi di slot di replica logica.

Slot di replica logica inattivi

Quando la replica viene terminata, i log delle transazioni non utilizzati non possono essere rimossi e lo slot di replica diventa inattivo. Gli slot di replica logica inattivi non vengono utilizzati dai nodi subscriber, ma rimangono comunque sul server, il che comporta la conservazione dei file WAL e impedisce la rimozione dei log delle transazioni meno recenti. Ciò può aumentare l'utilizzo del disco e, nello specifico, impedire al processo di autovacuum di eseguire la pulizia delle tabelle interne del catalogo, poiché il sistema deve impedire che le informazioni LSN vengano sovrascritte. Se il problema non viene risolto, possono verificarsi conseguenze come bloat del catalogo, riduzione delle prestazioni, aumento del rischio di wraparound nel processo di vacuum e quindi potenziali tempi di inattività per le transazioni.

Slot di replica logica attivi ma lenti

Talvolta la rimozione delle tuple morte del catalogo viene ritardata a causa della riduzione delle prestazioni della replica logica. Questo ritardo nella replica, che rallenta l'aggiornamento di `catalog_xmin`, può causare bloat del catalogo e wraparound nel processo di vacuum.

La funzione `postgres_get_av_diag()` visualizza un output simile a quanto segue quando individua in uno slot di replica logica un elemento bloccante.

```
blocker               | Logical replication slot
database              | my_database
blocker_identifier    | slot1
wait_event            | Not applicable
autovacuum_lagging_by | 1,940,103,068
suggestion            | Ensure replication is active and resolve any lag for the slot
 if active. If inactive, consider dropping it using the command in suggested_action.
 For more information, see Working with PostgreSQL autovacuum in the Amazon RDS User
 Guide.
suggested_action      | {"SELECT pg_drop_replication_slot('slot1') FROM
 pg_replication_slots WHERE active = 'f';"}
```

Operazione suggerita

Per risolvere la situazione, controlla la configurazione di replica per individuare eventuali problemi con i dati o lo schema di destinazione che potrebbero determinare la terminazione del processo di applicazione. I problemi più comuni sono i seguenti:

- Colonne mancanti
- Tipi di dati incompatibili

- Mancata corrispondenza dei dati
- Tabella mancante

Se il problema è legato a problemi relativi all'infrastruttura:

- Problemi di rete: consulta Come posso risolvere i problemi relativi a un database Amazon RDS in uno stato di rete incompatibile?
- Il database o l'istanza database non è disponibile per i seguenti motivi:
  - L'istanza di replica ha esaurito lo spazio di archiviazione: per informazioni su come aggiungere spazio di archiviazione, consulta la pagina che spiega cosa fare quando le istanze database Amazon RDS esauriscono lo spazio di archiviazione.
  - Parametri incompatibili: per ulteriori informazioni su cosa fare, consulta Come posso risolvere i problemi di un'istanza database Amazon RDS bloccata nello stato parametri incompatibili?

Se l'istanza si trova al di fuori della rete AWS oppure su AWS EC2, contatta l'utente amministratore per sapere come risolvere i problemi relativi alla disponibilità o all'infrastruttura.

Eliminazione dello slot inattivo

> **Warning**
>
> Attenzione: prima di eliminare uno slot di replica, verifica con attenzione che non vi sia alcun processo di replica in esecuzione, e che lo slot sia inattivo e sia in uno stato irreversibile. Se uno slot viene eliminato troppo presto, la replica potrebbe interrompersi oppure potrebbe verificarsi una perdita di dati.

Dopo aver verificato che lo slot di replica non è più necessario, eliminalo per far sì che il processo di autovacuum possa continuare. La condizione `active = 'f'` garantisce che uno slot venga eliminato solo se inattivo.

```
SELECT pg_drop_replication_slot('slot1') WHERE active ='f'
```

Istanze di lettura

Quando l'impostazione hot_standby_feedback è abilitata, impedisce che il processo di autovacuum eseguito sull'istanza di scrittura rimuova le righe morte che potrebbero essere ancora necessarie per

le query in esecuzione sull'istanza di lettura. Questo comportamento è necessario perché le query in esecuzione sull'istanza di lettura (lo stesso vale anche per le istanze di lettura nel Database globale Aurora) richiedono che tali righe rimangano disponibili sull'istanza di scrittura, prevenendo conflitti e annullamenti delle query.

> **Note**
>
> La funzione `hot_standby_feedback` è abilitata per impostazione predefinita e non è modificabile in Aurora PostgreSQL.

La funzione `postgres_get_av_diag()` visualizza un output simile a quando segue quando individua in una replica di lettura con slot di replica fisica un elemento bloccante.

```
blocker               | Oldest query running on aurora reader
database              | Not applicable
blocker_identifier    | my-aurora-reader-2
wait_event            | Not applicable
autovacuum_lagging_by | 540,122,859
suggestion            | Run the following query on the reader "my-aurora-reader-2" to
 find the long running query:

                      | SELECT * FROM pg_catalog.pg_stat_activity WHERE
 backend_xmin::text::bigint = 523476310;

                      | Review carefully and you may consider terminating the query on
 reader using suggested_action.
suggested_action      | {"SELECT pg_terminate_backend(pid) FROM
 pg_catalog.pg_stat_activity WHERE backend_xmin::text::bigint = 523476310;","

                      | [OR]


                      | ","Delete the reader if not needed"}
```

Come consigliato nella colonna `suggested_action`, esamina attentamente queste opzioni per sbloccare il processo di autovacuum.

- Terminare la query: seguendo le indicazioni riportate nella colonna dei suggerimenti, puoi connetterti alla replica di lettura, come specificato nella colonna suggested_action; è consigliabile esaminare attentamente l'opzione per terminare la sessione. Se è sicuro eseguire la terminazione,

è possibile utilizzare la funzione `pg_terminate_backend()` per terminare la sessione. Questa azione può essere eseguita da un utente amministratore (come l'account principale RDS) o da un utente con il privilegio pg_terminate_backend() richiesto.

È possibile eseguire il seguente comando SQL sulla replica di lettura per terminare la query che impedisce al processo di autovacuum sul nodo primario di eseguire la pulizia delle righe meno recenti. Il valore di `backend_xmin` è riportato nell'output della funzione:

```
SELECT
    pg_terminate_backend(pid)
FROM
    pg_catalog.pg_stat_activity
WHERE
    backend_xmin::text::bigint = backend_xmin;
```

- Eliminare le istanze di lettura se non sono più necessarie: se l'istanza di lettura non è più necessaria, è possibile eliminarla. In questo modo, viene rimosso il sovraccarico di replica associato e il nodo primario può riciclare i log delle transazioni senza alcun ostacolo da parte dell'istanza.

Tabelle temporanee

Le tabelle temporanee, create utilizzando la parola chiave `TEMPORARY`, risiedono nello schema temporaneo (ad esempio, pg_temp_xxx) e sono accessibili solo alla sessione che le ha create. Le tabelle temporanee vengono eliminate al termine della sessione. Tuttavia, queste tabelle sono invisibili al processo di autovacuum di PostgreSQL e devono sottoposte manualmente a vacuum dalla sessione che le ha create. Il tentativo di eseguire il processo di vacuum sulla tabella temporanea da un'altra sessione non ha alcun effetto.

In circostanze insolite, una tabella temporanea esiste senza che una sessione attiva ne sia proprietaria. Se la sessione proprietaria si conclude inaspettatamente a causa di un crash irreversibile, di un problema di rete o di un evento simile, è possibile che la pulizia della tabella temporanea non venga eseguita e che di conseguenza la tabella rimanga "orfana". Quando il processo di autovacuum di PostgreSQL rileva una tabella temporanea orfana, registra il seguente messaggio:

```
LOG: autovacuum: found orphan temp table \"%s\".\"%s\" in database \"%s\"
```

La funzione `postgres_get_av_diag()` visualizza un output simile a quanto segue quando identifica in una tabella temporanea un elemento bloccante. Affinché la funzione mostri correttamente l'output relativo alle tabelle temporanee, deve essere eseguita all'interno dello stesso database in cui esistono tali tabelle.

```
blocker               | Temporary table
database              | my_database
blocker_identifier    | pg_temp_14.ttemp
wait_event            | Not applicable
autovacuum_lagging_by | 1,805,802,632
suggestion            | Connect to database "my_database". Review carefully, you
 may consider dropping temporary table using command in suggested_action. For more
 information, see Working with PostgreSQL autovacuum in the Amazon RDS User Guide.
suggested_action      | {"DROP TABLE ttemp;"}
```

Operazione suggerita

Segui le istruzioni fornite nella colonna `suggestion` dell'output per identificare e rimuovere la tabella temporanea che impedisce l'esecuzione del processo di autovacuum. Utilizza il comando seguente per eliminare la tabella temporanea riportata da `postgres_get_av_diag()`. Sostituisci il nome della tabella in base all'output fornito dalla funzione `postgres_get_av_diag()`.

```
DROP TABLE my_temp_schema.my_temp_table;
```

La seguente query può essere utilizzata per identificare le tabelle temporanee:

```
SELECT
    oid,
    relname,
    relnamespace::regnamespace,
    age(relfrozenxid)
FROM
    pg_class
WHERE
relpersistence = 't'
ORDER BY
    age(relfrozenxid) DESC;
```

Risoluzione dei blocchi non identificabili per i processi di vacuum in Aurora PostgreSQL

Questa sezione esplora ulteriori motivi che possono impedire l'avanzamento del processo di autovacuum. Al momento tali problemi non sono direttamente identificabili tramite la funzione `postgres_get_av_diag()`.

Argomenti

- Incoerenza dell'indice
- Tasso di transazione eccezionalmente elevato

Incoerenza dell'indice

Un indice logicamente incoerente può impedire il progresso del processo di autovacuum. I seguenti errori o errori simili vengono registrati durante la fase di vacuum dell'indice o quando si accede all'indice tramite istruzioni SQL.

```
ERROR: right sibling's left-link doesn't match:block 5 links to 10 instead of expected
 2 in index ix_name
```

```
ERROR: failed to re-find parent key in index "XXXXXXXXXX" for deletion target page XXX
CONTEXT:  while vacuuming index index_name of relation schema.table
```

Linea guida

Crea nuovamente l'indice o salta gli indici utilizzando `INDEX_CLEANUP` sul `VACUUM FREEZE` manuale.

- Utilizzo dell'opzione CONCURRENTLY: prima della versione 12 di PostgreSQL, per creare nuovamente un indice era richiesto un blocco di tabella esclusivo, che limitava l'accesso alla tabella Con la versione 12 di PostgreSQL e le versioni successive, l'opzione CONCURRENTLY consente di eseguire il blocco a livello di riga e ciò migliora la disponibilità della tabella in modo significativo. Il comando è il seguente:

  ```
  REINDEX INDEX ix_name CONCURRENTLY;
  ```

  Sebbene CONCURRENTLY sia meno dannoso, può funzionare più lentamente sulle tabelle più utilizzate. Se possibile, valuta la possibilità di creare l'indice durante i periodi di traffico ridotto.

Per ulteriori informazioni, consulta la sezione dedicata a REINDEX nella documentazione di PostgreSQL.

- Utilizzo dell'opzione INDEX_CLEANUP FALSE: se gli indici sono di grandi dimensioni e si stima che richiedano un lungo lasso di tempo per completare l'operazione, puoi sbloccare l'autovacuum eseguendo un VACUUM FREEZE manuale escludendo gli indici. Questa funzionalità è disponibile in PostgreSQL 12 e versioni successive.

  Bypassare gli indici ti consentirà di saltare il processo di vacuum dell'indice incoerente e mitigare il problema di wraparound. Tuttavia, ciò non risolverà il problema sottostante relativo alla pagina non valida. Per risolvere completamente il problema relativo alla pagina non valida, dovrai comunque creare di nuovo l'indice.

Tasso di transazione eccezionalmente elevato

In PostgreSQL, i tassi di transazione elevati possono influire in modo significativo sulle prestazioni di autovacuum, portando a una pulizia più lenta delle tuple inattive e a un aumento del rischio di wraparound dell'ID di transazione. È possibile monitorare il tasso di transazione misurando la differenza in `max(age(datfrozenxid))` tra due periodi di tempo, in genere al secondo. Inoltre, puoi utilizzare le seguenti metriche dei contatori di Approfondimenti sulle prestazioni di RDS per misurare il tasso di transazione (la somma di xact_commit e xact_rollback) che è il numero totale delle transazioni.

Contatore	Tipo	Unità	Parametro
xact_commit	Transazioni	Commit al secondo	db.Transactions.xact_commit
xact_rollback	Transazioni	Rollback al secondo	db.Transactions.xact_rollback

Un aumento rapido indica un carico di transazioni elevato, che può sovraccaricare il processo di autovacuum, con conseguenti aumento di dimensioni, conflitti di blocco e potenziali problemi di prestazioni. Ciò può influire negativamente sul processo di autovacuum in due modi:

- Attività delle tabelle: la tabella specifica in fase di vacuum potrebbe registrare un volume elevato di transazioni, con conseguenti ritardi.

- Risorse di sistema: l'intero sistema potrebbe essere sovraccarico e ciò rende difficile per l'autovacuum accedere alle risorse necessarie per garantire un funzionamento efficiente.

Esamina le seguenti strategie per consentire all'autovacuum di funzionare in modo più efficace e tenere il passo con le sue attività:

1. Riduzione, se possibile, del tasso di transazioni. Possibilità di raggruppare le transazioni simili, ove applicabile.
2. Concentrazione delle tabelle aggiornate di frequente con operazioni `VACUUM FREEZE` manuali con frequenza notturna, settimanale o bisettimanale nelle ore non di punta.
3. Possibilità di aumentare verticalmente la classe di istanza per allocare più risorse di sistema al fine di gestire l'elevato volume di transazioni e l'autovacuum.

Risoluzione dei problemi relativi alle prestazioni di vacuum in Aurora PostgreSQL

Questa sezione illustra i fattori che spesso contribuiscono a rallentare le prestazioni di vacuum e spiega come risolvere tali problemi.

Argomenti

- Processo di vacuum su indici di grandi dimensioni
- Quantità eccessiva di tabelle o database da sottoporre a vacuum
- Esecuzione di un processo di vacuum aggressivo (per evitare il wraparound)

Processo di vacuum su indici di grandi dimensioni

Il processo VACUUM funziona in fasi sequenziali: inizializzazione, scansione degli heap, vacuum di indici e heap, pulizia degli indici, troncamento degli heap e pulizia finale. Durante la scansione degli heap, il processo elimina alcune pagine, le deframmenta e le blocca. Dopo aver completato la scansione degli heap, VACUUM pulisce gli indici, restituisce le pagine vuote al sistema operativo ed esegue le attività di pulizia finale, come il vacuum della mappa dello spazio libero e l'aggiornamento delle statistiche.

Il processo di vacuum degli indici può richiedere più passaggi quando `maintenance_work_mem` (o `autovacuum_work_mem`) è insufficiente per elaborare l'indice. In PostgreSQL 16 e versioni precedenti, poiché per l'archiviazione degli ID delle tuple morte esisteva un limite di memoria di 1 GB, spesso era obbligatorio eseguire più passaggi sugli indici di grandi dimensioni. In PostgreSQL 17 è

stato introdotto `TidStore`, che alloca la memoria in modo dinamico anziché utilizzare un array con allocazione singola. In questo modo viene eliminato il vincolo di 1 GB, la memoria viene utilizzata in modo più efficiente e viene ridotta la necessità di scansioni multiple per ogni indice.

È comunque possibile che gli indici di grandi dimensioni richiedano più passaggi in PostgreSQL 17 se la memoria disponibile non è in grado di supportare l'elaborazione simultanea degli interi indici. In genere, gli indici di dimensioni maggiori contengono un numero superiore di tuple morte che richiedono più passaggi.

Rilevamento di operazioni di vacuum a esecuzione lenta

La funzione `postgres_get_av_diag()` è in grado di rilevare le situazioni in cui le operazioni di vacuum vengono eseguite lentamente per memoria insufficiente. Per ulteriori informazioni su questa funzione, consulta Installazione di strumenti di diagnostica e monitoraggio dei processi di autovacuum in Aurora PostgreSQL.

La funzione `postgres_get_av_diag()` genera i seguenti avvisi quando la memoria disponibile non è sufficiente per completare il processo di vacuum di un indice in un singolo passaggio.

`rds_tools` 1.9

```
NOTICE: Your database is currently running aggressive vacuum to prevent wraparound and
 it might be slow.
```

```
NOTICE: The current setting of autovacuum_work_mem is XX might not be sufficient.
 Consider increasing the setting to XXX, and if necessary, scaling up the RDS instance
 class for more
       memory. The suggested value is an estimate based on the current number of dead
 tuples for the table being vacuumed, which might not fully reflect the latest state.
 Additionally, review the possibility of manual
       vacuum with exclusion of indexes using (VACUUM (INDEX_CLEANUP FALSE, VERBOSE
 TRUE) table_name;). For more information, see

       Working with PostgreSQL autovacuum in the Amazon Aurora User Guide.
```

> **Note**
>
> La funzione `postgres_get_av_diag()` si affida a `pg_stat_all_tables.n_dead_tup` per stimare la quantità di memoria richiesta per il vacuum dell'indice.

Quando identifica un'operazione di vacuum a esecuzione lenta che richiede più scansioni dell'indice per `autovacuum_work_mem` insufficiente, la funzione `postgres_get_av_diag()` genera il seguente messaggio:

```
NOTICE: Your vacuum is performing multiple index scans due to insufficient
 autovacuum_work_mem:XXX for index vacuuming.
       For more information, see Working with PostgreSQL autovacuum in the Amazon
 Amazon RDS User Guide.
```

Linee guida

È possibile avvalersi delle seguenti soluzioni alternative utilizzando il processo `VACUUM FREEZE` manuale per velocizzare il blocco della tabella.

Incremento della memoria per il processo di vacuum

Come suggerito dalla funzione `postgres_get_av_diag()`, è consigliabile incrementare il valore del parametro `autovacuum_work_mem` per superare potenziali vincoli di memoria a livello di istanza. Sebbene `autovacuum_work_mem` sia un parametro dinamico, per applicare la nuova impostazione di memoria occorre che il daemon di autovacuum riavvii i propri worker. A tale scopo, procedi come segue:

1. Verifica che la nuova impostazione sia effettiva.
2. Termina i processi che hanno in corso un'operazione di autovacuum.

Questo approccio garantisce che la nuova quantità di memoria allocata venga applicata alle successive operazioni di autovacuum.

Per risultati più immediati, valuta la possibilità di eseguire manualmente un'operazione `VACUUM FREEZE` con un'impostazione `maintenance_work_mem` incrementata nella sessione in corso:

```
SET maintenance_work_mem TO '1GB';
VACUUM FREEZE VERBOSE table_name;
```

Se utilizzi Amazon RDS e ritieni di aver bisogno di memoria aggiuntiva per supportare valori più elevati per `maintenance_work_mem` o `autovacuum_work_mem`, valuta la possibilità di passare a una classe di istanze con più memoria. Tale operazione può fornire le risorse necessarie per ottimizzare sia le operazioni di vacuum manuali che quelle automatiche, con conseguente miglioramento delle prestazioni complessive del database e del processo di vacuum.

Disabilitazione di INDEX_CLEANUP

Il processo `VACUUM` manuale in PostgreSQL versione 12 e successive consente di saltare la fase di pulizia degli indici, mentre l'autovacuum di emergenza in PostgreSQL versione 14 e successive esegue tale operazione automaticamente in base al parametro `vacuum_failsafe_age`.

> **Warning**
>
> La mancata esecuzione della pulizia degli indici può causare il bloat degli indici e influire negativamente sulle prestazioni delle query. Per risolvere questo problema, valuta la possibilità di eseguire una nuova indicizzazione o un'operazione di vacuum sugli indici interessati durante una finestra di manutenzione.

Per ulteriori indicazioni sulla gestione degli indici di grandi dimensioni, consulta la pagina Gestione di autovacuum con indici di grandi dimensioni .

Vacuum degli indici in parallelo

A partire da PostgreSQL 13, gli indici possono essere sottoposti a vacuum e pulizia in parallelo per impostazione predefinita tramite un'operazione `VACUUM` manuale, con un processo worker di vacuum assegnato a ciascun indice. Tuttavia, affinché PostgreSQL possa determinare se un'operazione di vacuum è idonea per l'esecuzione in parallelo, occorre che siano soddisfatti criteri specifici:

- Devono esistere almeno due indici.
- Il parametro `max_parallel_maintenance_workers` deve essere impostato almeno su 2.
- Le dimensioni degli indici devono superare il limite di `min_parallel_index_scan_size`, che è di 512 KB per impostazione predefinita.

L'impostazione `max_parallel_maintenance_workers` può essere modificata in base al numero di vCPU disponibili nell'istanza Amazon RDS in uso e al numero di indici nella tabella per ottimizzare i tempi di risposta del processo di vacuum.

Per ulteriori informazioni, consulta Parallel vacuuming in Amazon RDS for PostgreSQL and Amazon Aurora PostgreSQL.

Quantità eccessiva di tabelle o database da sottoporre a vacuum

Come descritto nella sezione The Autovacuum Daemon della documentazione di PostgreSQL, il daemon di autovacuum opera attraverso l'esecuzione di più processi. È incluso un programma di avvio automatico persistente responsabile dell'avvio dei processi worker di autovacuum per ogni database all'interno del sistema. Il programma di avvio pianifica l'avvio dei worker affinché si verifichi all'incirca ogni `autovacuum_naptime` secondi per database.

Con N database, un nuovo processo worker viene avviato approssimativamente ogni [`autovacuum_naptime`/N secondi]. Tuttavia, il numero totale di processi worker simultanei è limitato dall'impostazione `autovacuum_max_workers`. Se il numero di database o di tabelle che richiedono un processo di vacuum supera tale limite, l'elaborazione del database successivo o della tabella successiva avverrà non appena sarà disponibile un processo worker.

Quando è necessario eseguire il processo di vacuum su più tabelle o database di grandi dimensioni contemporaneamente, tutti i processi worker di autovacuum disponibili possono rimanere occupati per un periodo di tempo prolungato, ritardando così la manutenzione di altri database e tabelle. In ambienti con grandi volumi di transazioni, questo collo di bottiglia può peggiorare rapidamente e potenzialmente causare problemi con il processo di vacuum per wraparound nell'istanza Amazon RDS in uso.

Quando `postgres_get_av_diag()` rileva un numero elevato di tabelle o database, fornisce il seguente suggerimento:

```
NOTICE: Your database is currently running aggressive vacuum to prevent wraparound and
 it might be slow.
```

```
NOTICE: The current setting of autovacuum_max_workers:3 might not be sufficient.
 Consider increasing the setting and, if necessary, consider scaling up the Amazon RDS
 instance class for more workers.
```

Linee guida

Incremento del numero di autovacuum_max_workers

Per velocizzare il processo di vacuum, si consiglia di regolare il parametro `autovacuum_max_workers` per consentire l'utilizzo simultaneo di più processi worker di autovacuum. Se continuano a verificarsi colli di bottiglia nelle prestazioni, valuta la possibilità di aumentare verticalmente le risorse dell'istanza Amazon RDS in uso passando a una classe con più vCPU, così da migliorare ulteriormente le capacità di elaborazione in parallelo.

Esecuzione di un processo di vacuum aggressivo (per evitare il wraparound)

L'età del database (MaximumUsedTransactionIDs) in PostgreSQL diminuisce solo dopo il corretto completamento di un processo di vacuum aggressivo (per evitare il wraparound). Fino al completamento di tale processo di vacuum, l'età continua ad aumentare in funzione del volume di transazioni.

La funzione `postgres_get_av_diag()` genera il seguente messaggio `NOTICE` quando rileva un processo di vacuum aggressivo, ma attiva questo output solo dopo che il processo di vacuum è rimasto attivo per almeno due minuti.

```
NOTICE: Your database is currently running aggressive vacuum to prevent wraparound,
 monitor autovacuum performance.
```

Per ulteriori informazioni sui processi di vacuum aggressivi, consulta la sezione Scenario in cui è già in esecuzione un processo di vacuum aggressivo.

È possibile verificare se è in corso un processo di vacuum aggressivo utilizzando la seguente query:

```
SELECT
    a.xact_start AS start_time,
    v.datname "database",
    a.query,
    a.wait_event,
    v.pid,
    v.phase,
    v.relid::regclass,
    pg_size_pretty(pg_relation_size(v.relid)) AS heap_size,
    (
        SELECT
            string_agg(pg_size_pretty(pg_relation_size(i.indexrelid)) || ':' ||
 i.indexrelid::regclass || chr(10), ', ')
        FROM
            pg_index i
        WHERE
            i.indrelid = v.relid
    ) AS index_sizes,
    trunc(v.heap_blks_scanned * 100 / NULLIF(v.heap_blks_total, 0)) AS step1_scan_pct,
    v.index_vacuum_count || '/' || (
        SELECT
            count(*)
        FROM
```

```
            pg_index i
        WHERE
            i.indrelid = v.relid
    ) AS step2_vacuum_indexes,
    trunc(v.heap_blks_vacuumed * 100 / NULLIF(v.heap_blks_total, 0)) AS
 step3_vacuum_pct,
    age(CURRENT_TIMESTAMP, a.xact_start) AS total_time_spent_sofar
FROM
    pg_stat_activity a
    INNER JOIN pg_stat_progress_vacuum v ON v.pid = a.pid;
```

È possibile determinare se si tratta di un processo di vacuum aggressivo (per evitare il wraparound) controllando la colonna della query nell'output. La dicitura "to prevent wraparound" (letteralmente, per evitare il wraparound) indica che si tratta di un processo di vacuum aggressivo.

```
query                 | autovacuum: VACUUM public.t3 (to prevent wraparound)
```

Ad esempio, si supponga di avere un elemento bloccante corrispondente a un'età delle transazioni pari a 1 miliardo e una tabella che richiede un processo di vacuum aggressivo per evitare il wraparound in corrispondenza dello stesso valore di età delle transazioni. Inoltre, esiste un altro elemento bloccante corrispondente a un'età delle transazioni pari a 750 milioni. Una volta eliminato l'elemento bloccante corrispondente a un'età delle transazioni pari a 1 miliardo, l'età delle transazioni non scende immediatamente a 750 milioni ma rimane elevata fino al completamento della tabella che richiede un processo di vacuum aggressivo o di eventuali transazioni con età superiore a 750 milioni. Durante tale periodo, l'età delle transazioni del cluster PostgreSQL continua a crescere. Una volta completato il processo di vacuum, l'età delle transazioni scende a 750 milioni e quindi ricomincia ad aumentare fino al termine di un ulteriore processo di vacuum. Questo ciclo continua fintanto che persistono le suddette condizioni, ossia finché l'età delle transazioni non sarà scesa al livello configurato per l'istanza Amazon RDS in uso, specificato dal parametro `autovacuum_freeze_max_age`.

Spiegazione dei messaggi NOTICE in Aurora PostgreSQL

La funzione `postgres_get_av_diag()` fornisce i seguenti messaggi NOTICE:

Scenario in cui l'età non ha ancora raggiunto la soglia di monitoraggio

> Per impostazione predefinita, la soglia di monitoraggio per `postgres_get_av_diag()` che consente di identificare gli elementi bloccanti è di 500 milioni di transazioni. Se

`postgres_get_av_diag()` genera il seguente messaggio NOTICE, significa che l'età delle transazioni non ha ancora raggiunto tale soglia.

```
NOTICE: postgres_get_av_diag() checks for blockers that prevent aggressive vacuums
 only, it does so only after exceeding dvb_threshold which is 500,000,000 and age of
 this PostgreSQL cluster is currently at 2.
```

Mancata connessione al database con l'età dell'ID transazione meno recente

La funzione `postgres_get_av_diag()` fornisce l'output più accurato quando è connessa al database con l'età dell'ID transazione meno recente. Nel caso in questione, il database con l'età dell'ID transazione meno recente riportato da `postgres_get_av_diag()` è diverso da "my_database". In caso di mancata connessione al database corretto, viene generato il seguente messaggio NOTICE:

```
NOTICE: You are not connected to the database with the age of oldest transaction
 ID. Connect to my_database database and run postgres_get_av_diag() for accurate
 reporting.
```

La connessione al database con l'età della transazione meno recente è importante per i seguenti motivi:

- Identificazione degli elementi bloccanti costituiti da tabelle temporanee: i metadati per le tabelle temporanee sono specifici di ogni database, pertanto solitamente sono contenuti nel database in cui sono stati creati. Tuttavia, se una tabella temporanea è l'elemento bloccante principale ed è contenuta nel database con la transazione meno recente, la situazione potrebbe risultare fuorviante. La connessione al database corretto garantisce l'identificazione accurata dell'elemento bloccante costituito da una tabella temporanea.
- Diagnosi dei problemi correlati a processi di vacuum a esecuzione lenta: le informazioni sul numero di tabelle e i metadati degli indici sono specifici di ogni database e sono necessari per diagnosticare eventuali problemi correlati a processi di vacuum a esecuzione lenta.

Scenario in cui il database con la transazione meno recente per età è un database rdsadmin o template0

In alcuni casi, è possibile che un database `rdsadmin` o `template0` venga identificato come il database con l'età dell'ID transazione meno recente. In tali casi, `postgres_get_av_diag()` genera il seguente messaggio NOTICE:

```
NOTICE: The database with the age of oldest transaction ID is rdsadmin or template0,
 reach out to support if the reported blocker is in rdsadmin or template0.
```

Verifica che l'elemento bloccante indicato non provenga da nessuno di questi due database. Se viene segnalata la presenza dell'elemento bloccante nel database `rdsadmin` o `template0`, contatta il supporto poiché tali database non sono accessibili all'utente e richiedono un intervento.

È altamente improbabile che il database `rdsadmin` o `template0` contenga un elemento bloccante principale.

Scenario in cui è già in esecuzione un processo di vacuum aggressivo

La funzione `postgres_get_av_diag()` è progettata per segnalare quando è in corso un processo di vacuum aggressivo, ma attiva questo output solo dopo che il processo di vacuum è rimasto attivo per almeno 1 minuto. Questo ritardo intenzionale aiuta a ridurre le possibilità di falsi positivi. L'attesa fa sì che la funzione segnali solo i processi di vacuum efficaci e significativi, permettendo un monitoraggio più accurato e affidabile dell'attività di vacuum.

La funzione `postgres_get_av_diag()` genera il seguente messaggio NOTICE quando rileva uno o più processi di vacuum aggressivi in esecuzione.

```
NOTICE: Your database is currently running aggressive vacuum to prevent wraparound,
 monitor autovacuum performance.
```

Come indicato nel messaggio NOTICE, occorre continuare a monitorare le prestazioni dell'attività di vacuum. Per ulteriori informazioni sui processi di vacuum aggressivi, consulta Esecuzione di un processo di vacuum aggressivo (per evitare il wraparound).

Scenario in cui la funzionalità di autovacuum è disattivata

La funzione `postgres_get_av_diag()` genera il seguente messaggio NOTICE se la funzionalità di autovacuum è disabilitata in un'istanza database:

```
NOTICE: Autovacuum is OFF, we strongly recommend to enable it, no restart is
 necessary.
```

Autovacuum è una funzionalità fondamentale delle istanze database Aurora PostgreSQL in quanto garantisce il corretto funzionamento del database. Essa rimuove automaticamente le versioni precedenti delle righe, recupera spazio di archiviazione e impedisce il bloat delle tabelle, garantendo così l'efficacia di tabelle e indici per prestazioni ottimali. Inoltre, evita il wraparound

degli ID transazione, che può interrompere le transazioni sulle istanze Amazon RDS. Disabilitando la funzionalità di autovacuum, è possibile che la stabilità e le prestazioni del database subiscano una riduzione a lungo termine. Pertanto suggeriamo di lasciarla sempre attivata. Per ulteriori informazioni, consulta Informazioni sull'autovacuum in ambienti Aurora PostgreSQL.

> **Note**
>
> La disattivazione della funzionalità di autovacuum non blocca i processi di vacuum aggressivi. Tali problemi continueranno a verificarsi quando le tabelle raggiungeranno la soglia `autovacuum_freeze_max_age`.

Scenario in cui il numero di transazioni rimanenti è estremamente basso

La funzione `postgres_get_av_diag()` genera il seguente messaggio NOTICE quando è imminente un processo di vacuum per wraparound. Questo messaggio NOTICE viene generato quando, in un'istanza Amazon RDS, mancano solo 100 milioni di transazioni prima che possa verificarsi il rifiuto di nuove transazioni.

```
WARNING: Number of transactions remaining is critically low, resolve issues with
 autovacuum or perform manual VACUUM FREEZE before your instance stops accepting
 transactions.
```

È necessaria un'azione immediata per evitare un tempo di inattività del database. Occorre monitorare attentamente le operazioni di vacuum e valutare la possibilità di avviare manualmente un processo `VACUUM FREEZE` sul database interessato per evitare errori nelle transazioni.

Gestione delle controversie relative a TOAST OID in

TOAST (The Oversized-Attribute Storage Technique) è una funzionalità di PostgreSQL progettata per gestire grandi valori di dati che superano la dimensione tipica dei blocchi di database di 8 KB. PostgreSQL non consente alle righe fisiche di estendersi su più blocchi. La dimensione del blocco funge da limite superiore alla dimensione delle righe. TOAST supera questa restrizione suddividendo i valori di campo di grandi dimensioni in blocchi più piccoli. Li memorizza separatamente in una tabella TOAST dedicata collegata alla tabella principale. Per ulteriori informazioni, consulta il meccanismo di archiviazione TOAST di PostgreSQL e la documentazione di implementazione.

Argomenti

- Comprendere le operazioni TOAST
- Identificazione delle sfide prestazionali
- Raccomandazioni
- Monitoraggio

Comprendere le operazioni TOAST

TOAST esegue la compressione e memorizza valori di campo di grandi dimensioni fuori linea. TOAST assegna un OID (Object Identifier) univoco a ogni blocco di dati di grandi dimensioni archiviato nella tabella TOAST. La tabella principale memorizza l'ID del valore TOAST e l'ID della relazione sulla pagina per fare riferimento alla riga corrispondente nella tabella TOAST. Ciò consente a PostgreSQL di localizzare e gestire in modo efficiente questi blocchi TOAST. Tuttavia, man mano che la tabella TOAST cresce, il sistema rischia di esaurire le risorse disponibili OIDs, con conseguente peggioramento delle prestazioni e potenziali tempi di inattività dovuti all'esaurimento degli OID.

Identificatori di oggetti in TOAST

Un Object Identifier (OID) è un identificatore univoco a livello di sistema utilizzato da PostgreSQL per fare riferimento a oggetti del database come tabelle, indici e funzioni. Questi identificatori svolgono un ruolo fondamentale nelle operazioni interne di PostgreSQL, consentendo al database di localizzare e gestire gli oggetti in modo efficiente.

Per le tabelle con set di dati idonei per il toast, PostgreSQL assegna l'identificazione univoca di ogni blocco di dati di OIDs grandi dimensioni archiviato nella tabella TOAST associata. Il sistema associa ogni blocco a `unchunk_id`, che aiuta PostgreSQL a organizzare e localizzare questi blocchi in modo efficiente all'interno della tabella TOAST.

Identificazione delle sfide prestazionali

La gestione degli OID di PostgreSQL si basa su un contatore globale a 32 bit, che viene eseguito dopo aver generato 4 miliardi di valori univoci. Sebbene il cluster di database condivida questo contatore, l'allocazione OID prevede due passaggi durante le operazioni TOAST:

- Contatore globale per l'allocazione: il contatore globale assegna un nuovo OID al cluster.
- Ricerca locale dei conflitti: la tabella TOAST garantisce che il nuovo OID non entri in conflitto con quelli esistenti OIDs già utilizzati in quella tabella specifica.

Il peggioramento delle prestazioni può verificarsi quando:

- La tabella TOAST presenta un'elevata frammentazione o un uso intensivo dell'OID, con conseguenti ritardi nell'assegnazione dell'OID.
- Il sistema esegue spesso allocazioni e riutilizzi OIDs in ambienti con un elevato tasso di abbandono dei dati o tabelle ampie che utilizzano ampiamente TOAST.

Per ulteriori informazioni, consulta i limiti di dimensione della tabella TOAST di PostgreSQL e la documentazione sull'allocazione OID:

Un contatore globale genera OIDs e riassume ogni 4 miliardi di valori, in modo che di tanto in tanto il sistema generi nuovamente un valore già utilizzato. PostgreSQL lo rileva e riprova con l'OID successivo. Un INSERT lento potrebbe verificarsi se c'è una serie molto lunga di valori OID usati senza spazi vuoti nella tabella TOAST. Queste problematiche diventano più evidenti man mano che lo spazio OID si riempie, con conseguente rallentamento degli inserimenti e degli aggiornamenti.

Identificazione del problema

- `INSERT`Le affermazioni semplici richiedono molto più tempo del solito in modo incoerente e casuale.
- I ritardi si verificano solo per le `UPDATE` dichiarazioni che `INSERT` coinvolgono operazioni TOAST.
- Le seguenti voci di registro vengono visualizzate nei log di PostgreSQL quando il sistema fatica a trovare la disponibilità nelle tabelle TOAST: OIDs

```
LOG: still searching for an unused OID in relation "pg_toast_20815"
DETAIL: OID candidates have been checked 1000000 times, but no unused OID has been
 found yet.
```

- Performance Insights indica un numero elevato di sessioni attive medie (AAS) associate `LWLock:buffer_io` a eventi di `LWLock:OidGenLock` attesa.

  È possibile eseguire la seguente query SQL per identificare transazioni INSERT di lunga durata con eventi di attesa:

```
SELECT
    datname AS database_name,
    usename AS database_user,
    pid,
    now() - pg_stat_activity.xact_start AS transaction_duration,
```

```
    concat(wait_event_type, ':', wait_event) AS wait_event,
    substr(query, 1, 30) AS TRANSACTION,
    state
FROM
    pg_stat_activity
WHERE (now() - pg_stat_activity.xact_start) > INTERVAL '60 seconds'
    AND state IN ('active', 'idle in transaction', 'idle in transaction (aborted)',
 'fastpath function call', 'disabled')
    AND pid <> pg_backend_pid()
AND lower(query) LIKE '%insert%'
ORDER BY
    transaction_duration DESC;
```

Esempi di risultati di query che mostrano le operazioni INSERT con tempi di attesa prolungati:

```
 database_name |  database_user  |  pid  | transaction_duration |     wait_event
 |          transaction           | state
---------------+-----------------+-------+----------------------
+---------------------+--------------------------------+--------
 postgres       | db_admin_user| 70965 | 00:10:19.484061      | LWLock:buffer_io    |
 INSERT INTO "products" (......... | active
 postgres       | db_admin_user| 69878 | 00:06:14.976037      | LWLock:buffer_io    |
 INSERT INTO "products" (......... | active
 postgres       | db_admin_user| 68937 | 00:05:13.942847      | :                   |
 INSERT INTO "products" (......... | active
```

Isolamento del problema

- Test small insert: inserisce un record inferiore alla `toast_tuple_target` soglia. Ricorda che la compressione viene applicata prima dell'archiviazione TOAST. Se funziona senza problemi di prestazioni, il problema è legato alle operazioni TOAST.
- Prova una nuova tabella: crea una nuova tabella con la stessa struttura e inserisci un record più grande di`toast_tuple_target`. Se funziona senza problemi, il problema viene localizzato nell'allocazione OID della tabella originale.

Raccomandazioni

I seguenti approcci possono aiutare a risolvere i problemi di contesa di TOAST OID.

- Pulizia e archiviazione dei dati: rivedi ed elimina tutti i dati obsoleti o non necessari per liberarli per utilizzi OIDs futuri o archivia i dati. Considera i seguenti limiti:
  - Scalabilità limitata, poiché le pulizie future potrebbero non essere sempre possibili.
  - Possibile operazione VACUUM a lunga durata per rimuovere le tuple morte risultanti.
- Scrivi su una nuova tabella: crea una nuova tabella per inserti futuri e usa una `UNION ALL` vista per combinare dati vecchi e nuovi per le query. Questa visualizzazione presenta i dati combinati delle tabelle vecchie e nuove, consentendo alle query di accedervi come un'unica tabella. Considera i seguenti limiti:
  - Gli aggiornamenti sulla vecchia tabella potrebbero comunque causare l'esaurimento degli OID.
- Partizione o frammento: partiziona i dati della tabella o del frammento per migliorare la scalabilità e le prestazioni. Considera i seguenti limiti:
  - Maggiore complessità nella logica e nella manutenzione delle query, potenziale necessità di modifiche alle applicazioni per gestire correttamente i dati partizionati.

Monitoraggio

Utilizzo delle tabelle di sistema

È possibile utilizzare le tabelle di sistema di PostgreSQL per monitorare la crescita dell'utilizzo degli OID.

> **Warning**
>
> A seconda del numero di elementi OIDs presenti nella tabella TOAST, il completamento potrebbe richiedere del tempo. Si consiglia di pianificare il monitoraggio durante le ore non lavorative per ridurre al minimo l'impatto.

Il seguente blocco anonimo conta il numero di elementi distinti OIDs utilizzati in ogni tabella TOAST e visualizza le informazioni della tabella principale:

```
DO $$
DECLARE
    r record;
    o bigint;
    parent_table text;
```

```
    parent_schema text;
BEGIN
    SET LOCAL client_min_messages TO notice;
    FOR r IN
    SELECT
        c.oid,
        c.oid::regclass AS toast_table
    FROM
        pg_class c
    WHERE
        c.relkind = 't'
        AND c.relowner != 10 LOOP
            -- Fetch the number of distinct used OIDs (chunk IDs) from the TOAST table
            EXECUTE 'SELECT COUNT(DISTINCT chunk_id) FROM ' || r.toast_table INTO o;
            -- If there are used OIDs, find the associated parent table and its schema
            IF o <> 0 THEN
                SELECT
                    n.nspname,
                    c.relname INTO parent_schema,
                    parent_table
                FROM
                    pg_class c
                    JOIN pg_namespace n ON c.relnamespace = n.oid
                WHERE
                    c.reltoastrelid = r.oid;
                -- Raise a concise NOTICE message
                RAISE NOTICE 'Parent schema: % | Parent table: % | Toast table: %
 | Number of used OIDs: %', parent_schema, parent_table, r.toast_table, TO_CHAR(o,
 'FM9,999,999,999,999');
            END IF;
        END LOOP;
END
$$;
```

Esempio di output che mostra le statistiche sull'utilizzo di OID tramite la tabella TOAST:

```
NOTICE:  Parent schema: public | Parent table: my_table | Toast table:
 pg_toast.pg_toast_16559 | Number of used OIDs: 45,623,317
NOTICE:  Parent schema: public | Parent table: my_table1 | Toast table:
 pg_toast.pg_toast_45639925 | Number of used OIDs: 10,000
NOTICE:  Parent schema: public | Parent table: my_table2 | Toast table:
 pg_toast.pg_toast_45649931 | Number of used OIDs: 1,000,000
DO
```

Il seguente blocco anonimo recupera l'OID massimo assegnato per ogni tabella TOAST non vuota:

```
DO $$
DECLARE
    r record;
    o bigint;
    parent_table text;
    parent_schema text;
BEGIN
    SET LOCAL client_min_messages TO notice;
    FOR r IN
    SELECT
        c.oid,
        c.oid::regclass AS toast_table
    FROM
        pg_class c
    WHERE
        c.relkind = 't'
        AND c.relowner != 10 LOOP
            -- Fetch the max(chunk_id) from the TOAST table
            EXECUTE 'SELECT max(chunk_id) FROM ' || r.toast_table INTO o;
            -- If there's at least one TOASTed chunk, find the associated parent table
 and its schema
            IF o IS NOT NULL THEN
                SELECT
                    n.nspname,
                    c.relname INTO parent_schema,
                    parent_table
                FROM
                    pg_class c
                    JOIN pg_namespace n ON c.relnamespace = n.oid
                WHERE
                    c.reltoastrelid = r.oid;
                -- Raise a concise NOTICE message
                RAISE NOTICE 'Parent schema: % | Parent table: % | Toast table:
 % | Max chunk_id: %', parent_schema, parent_table, r.toast_table, TO_CHAR(o,
 'FM9,999,999,999,999');
            END IF;
        END LOOP;
END
$$;
```

Esempio di output che mostra il blocco massimo per le tabelle IDs TOAST:

```
NOTICE:  Parent schema: public | Parent table: my_table | Toast table:
 pg_toast.pg_toast_16559 | Max chunk_id: 45,639,907
NOTICE:  Parent schema: public | Parent table: my_table1 | Toast table:
 pg_toast.pg_toast_45639925 | Max chunk_id: 45,649,929
NOTICE:  Parent schema: public | Parent table: my_table2 | Toast table:
 pg_toast.pg_toast_45649931 | Max chunk_id: 46,649,935
DO
```

Uso di Approfondimenti sulle prestazioni

Gli eventi `LWLock:buffer_io` di attesa `LWLock:OidGenLock` vengono visualizzati in Performance Insights durante le operazioni che richiedono l'assegnazione di nuovi identificatori di oggetti ()OIDs. Le sessioni AAS (High Average Active Sessions) per questi eventi in genere indicano situazioni di conflitto durante l'assegnazione degli OID e la gestione delle risorse. Ciò è particolarmente comune in ambienti con un elevato tasso di abbandono dei dati, un utilizzo esteso di grandi quantità di dati o la creazione frequente di oggetti.

LWLockCiò è particolarmente comune in ambienti con un elevato tasso di abbandono dei dati, un ampio utilizzo di dati o la creazione frequente di oggetti. ----sep----:buffer_io

`LWLock:buffer_io`è un evento di attesa che si verifica quando una sessione PostgreSQL è in I/O attesa del completamento delle operazioni su un buffer condiviso. Ciò si verifica in genere quando il database legge i dati dal disco alla memoria o scrive pagine modificate dalla memoria al disco. L'evento di `BufferIO` attesa garantisce la coerenza impedendo a più processi di accedere o modificare lo stesso buffer mentre I/O le operazioni sono in corso. Un numero elevato di occorrenze di questo evento di attesa può indicare colli di bottiglia sul disco o un'attività eccessiva I/O nel carico di lavoro del database.

Durante le operazioni TOAST:

- PostgreSQL OIDs alloca oggetti di grandi dimensioni e ne garantisce l'unicità scansionando l'indice della tabella TOAST.
- Gli indici TOAST di grandi dimensioni possono richiedere l'accesso a più pagine per verificare l'unicità dell'OID. Ciò si traduce in un aumento dell'I/O del disco, specialmente quando il buffer pool non è in grado di memorizzare nella cache tutte le pagine richieste.

La dimensione dell'indice influisce direttamente sul numero di pagine buffer a cui è necessario accedere durante queste operazioni. Anche se l'indice non è gonfio, le sue dimensioni possono

aumentare l'I/O del buffer, in particolare in ambienti con elevata concorrenza o elevato tasso di abbandono. Per ulteriori informazioni, consulta:Guida alla risoluzione dei problemi relativi agli eventi Bufferio wait. LWLock

LWLock:OidGenLock

`OidGenLock`è un evento di attesa che si verifica quando una sessione PostgreSQL è in attesa di allocare un nuovo identificatore di oggetto (OID). Questo blocco garantisce che OIDs vengano generati in modo sequenziale e sicuro, consentendo la generazione di un solo processo alla volta. OIDs

Durante le operazioni TOAST:

- Allocazione OID per blocchi nella tabella TOAST: PostgreSQL assegna i blocchi nelle tabelle OIDs TOAST quando si gestiscono record di dati di grandi dimensioni. Ogni OID deve essere unico per evitare conflitti nel catalogo di sistema.
- Concorrenza elevata: poiché l'accesso al generatore OID è sequenziale, quando più sessioni creano contemporaneamente oggetti che lo richiedono OIDs, possono verificarsi contese per. `OidGenLock` Ciò aumenta la probabilità che le sessioni attendano il completamento dell'allocazione OID.
- Dipendenza dall'accesso al catalogo di sistema: l'allocazione OIDs richiede aggiornamenti alle tabelle del catalogo di sistema condiviso come e. `pg_class` `pg_type` Se queste tabelle presentano un'attività intensa (a causa di frequenti operazioni DDL), è possibile che si verifichi un aumento del conflitto di blocchi per. `OidGenLock`
- Elevata richiesta di allocazione OID: i carichi di lavoro pesanti di TOAST con record di dati di grandi dimensioni richiedono un'allocazione OID costante, con conseguente aumento del conflitto.

Fattori aggiuntivi che aumentano la contesa OID:

- Creazione frequente di oggetti: i carichi di lavoro che creano e rilasciano frequentemente oggetti, come le tabelle temporanee, amplificano il conflitto sul contatore OID globale.
- Controblocco globale: il contatore OID globale è accessibile in serie per garantire l'unicità, creando un unico punto di contesa in ambienti ad alta concorrenza.

Utilizzo di Babelfish per Aurora PostgreSQL

Babelfish per Aurora PostgreSQL estende il cluster database Aurora PostgreSQL con la possibilità di accettare connessioni al database dai client SQL Server. Con Babelfish, le applicazioni create originariamente per SQL Server possono utilizzare direttamente Aurora PostgreSQL con poche modifiche al codice rispetto a una migrazione tradizionale e senza modificare i driver del database. Per ulteriori informazioni sulla migrazione, consulta Migrazione di un database SQL Server a Babelfish per Aurora PostgreSQL.

Babelfish fornisce un endpoint aggiuntivo per un cluster di database Aurora PostgreSQL che consente di comprendere il protocollo wire-level di SQL Server e le istruzioni SQL Server comunemente utilizzate. Le applicazioni client che utilizzano il protocollo wire Tabular Data Stream (TDS) possono connettersi in maniera nativa alla porta del listener TDS su Aurora PostgreSQL. Per ulteriori informazioni su TDS, consulta [MS-TDS]: Tabular Data Stream Protocol sul sito Web di Microsoft.

> **Note**
>
> Babelfish per Aurora PostgreSQL supporta versioni TDS da 7.1 a 7.4.

Babelfish fornisce inoltre l'accesso ai dati utilizzando la connessione PostgreSQL. Per impostazione predefinita, entrambi i dialetti SQL supportati da Babelfish sono disponibili tramite i protocolli wire nativi presso le seguenti porte:

- Dialetto SQL Server (T-SQL), i client si connettono alla porta 1433.
- Dialetto PostgreSQL (PL/pgSQL), i client si connettono alla porta 5432.

Babelfish esegue il linguaggio Transact-SQL (T-SQL) con alcune differenze. Per ulteriori informazioni, consulta Differenze tra Babelfish per Aurora PostgreSQL e SQL Server.

Nelle seguenti sezioni sono disponibili informazioni sulla configurazione e l'utilizzo di un cluster database Babelfish per Aurora PostgreSQL.

Argomenti

- Limitazioni di Babelfish
- Comprendere l'architettura e la configurazione di Babelfish

- Creazione di un cluster database Babelfish per Aurora PostgreSQL
- Migrazione di un database SQL Server a Babelfish per Aurora PostgreSQL
- Autenticazione del database con Babelfish per Aurora PostgreSQL
- Connessione a un cluster database Babelfish
- Utilizzo di Babelfish
- Risoluzione dei problemi relativi a Babelfish
- Disattivazione di Babelfish
- Gestione degli aggiornamenti della versione di Babelfish per Aurora PostgreSQL
- Informazioni di riferimento su Babelfish per Aurora PostgreSQL

Limitazioni di Babelfish

Le seguenti limitazioni si applicano attualmente a Babelfish per Aurora PostgreSQL:

- Babelfish non supporta le seguenti funzionalità di Aurora:
  - AWS Identity and Access Management
  - Flussi di attività di database (DAS)
  - API dati RDS con Aurora PostgreSQL Serverless v2 e istanze allocate
  - Server proxy per RDS con RDS per SQL Server
  - SCRAM (Salted Challenge Response Authentication Mechanism)
  - Editor della query
- Babelfish non fornisce il seguente supporto API per i driver client:
  - Le richieste API con gli attributi di connessione correlati a Microsoft Distributed Transaction Coordinator (MSDTC) non sono supportate. Queste includono le chiamate XA da parte della SQLServer XAResource classe nel driver JDBC del server SQL.
- Babelfish attualmente non supporta le seguenti estensioni Aurora PostgreSQL:
  - `bloom`
  - `btree_gin`
  - `btree_gist`
  - `citext`
  - `cube`
  - `hstore`
  - `hypopg`
  - Replica logica con `pglogical`
  - `ltree`
  - `pgcrypto`
  - Gestione del piano di query utilizzando `apg_plan_mgmt`

  Per ulteriori informazioni sulle estensioni PostgreSQL, consulta Utilizzo di estensioni e wrapper di dati esterni.
- Il driver jTDS open source progettato come alternativa al driver Microsoft JDBC non è supportato.

Comprendere l'architettura e la configurazione di Babelfish

La gestione del cluster database Aurora PostgreSQL-Compatible Edition che esegue Babelfish è simile a quella di qualsiasi cluster database Aurora. Ovvero, è possibile sfruttare la scalabilità, l'elevata disponibilità con il supporto per il failover e la replica integrata forniti da un cluster database Aurora. Per ulteriori informazioni su queste funzionalità, consulta Gestione delle prestazioni e del dimensionamento dei cluster DB Aurora, Elevata disponibilità di Amazon Aurora e Replica con Amazon Aurora. È disponibile inoltre l'accesso a molti altri strumenti e utilità AWS, inclusi i seguenti:

- Amazon CloudWatch è un servizio di monitoraggio e osservabilità che fornisce dati e approfondimenti fruibili. Per ulteriori informazioni, consulta Monitoraggio dei parametri di Amazon Aurora con Amazon CloudWatch.
- Performance Insights è una funzionalità di ottimizzazione e monitoraggio delle prestazioni del database che consente di valutare rapidamente il carico sul database. Per ulteriori informazioni, consulta Monitoraggio del carico DB con Performance Insights su Amazon Aurora.
- I database globali Aurora si sviluppano su più Regioni AWS, consentendo letture globali a bassa latenza e offrendo un ripristino di emergenza rapido in caso di eventuali interruzioni che potrebbero interessare un'intera Regione AWS. Per ulteriori informazioni, consulta Utilizzo di Database globale Amazon Aurora.
- Le patch software automatiche mantengono aggiornato il database con le ultime patch di sicurezza e funzionalità quando diventano disponibili.
- Gli eventi Amazon RDS comunicano via e-mail o SMS eventi importanti del database, ad esempio un failover automatico. Per ulteriori informazioni, consulta Monitoraggio di eventi Amazon Aurora.

Di seguito sono fornite ulteriori informazioni sull'architettura Babelfish e su come i database SQL Server migrati vengono gestiti da Babelfish. Quando crei il cluster database Babelfish, devi prendere alcune decisioni in anticipo su database singoli o multipli, regole di confronto e altri dettagli.

Argomenti

- Architettura babelfish
- Impostazioni del gruppo di parametri del cluster database per Babelfish
- Informazioni sulle regole di confronto in Babelfish per Aurora PostgreSQL.
- Gestione degli errori di Babelfish con escape hatch

Architettura babelfish

Quando crei un cluster Aurora PostgreSQL con Babelfish attivato, Aurora fornisce il cluster un database PostgreSQL denominato `babelfish_db`. Questo database è dove risiedono tutti gli oggetti e le strutture di SQL Server migrati.

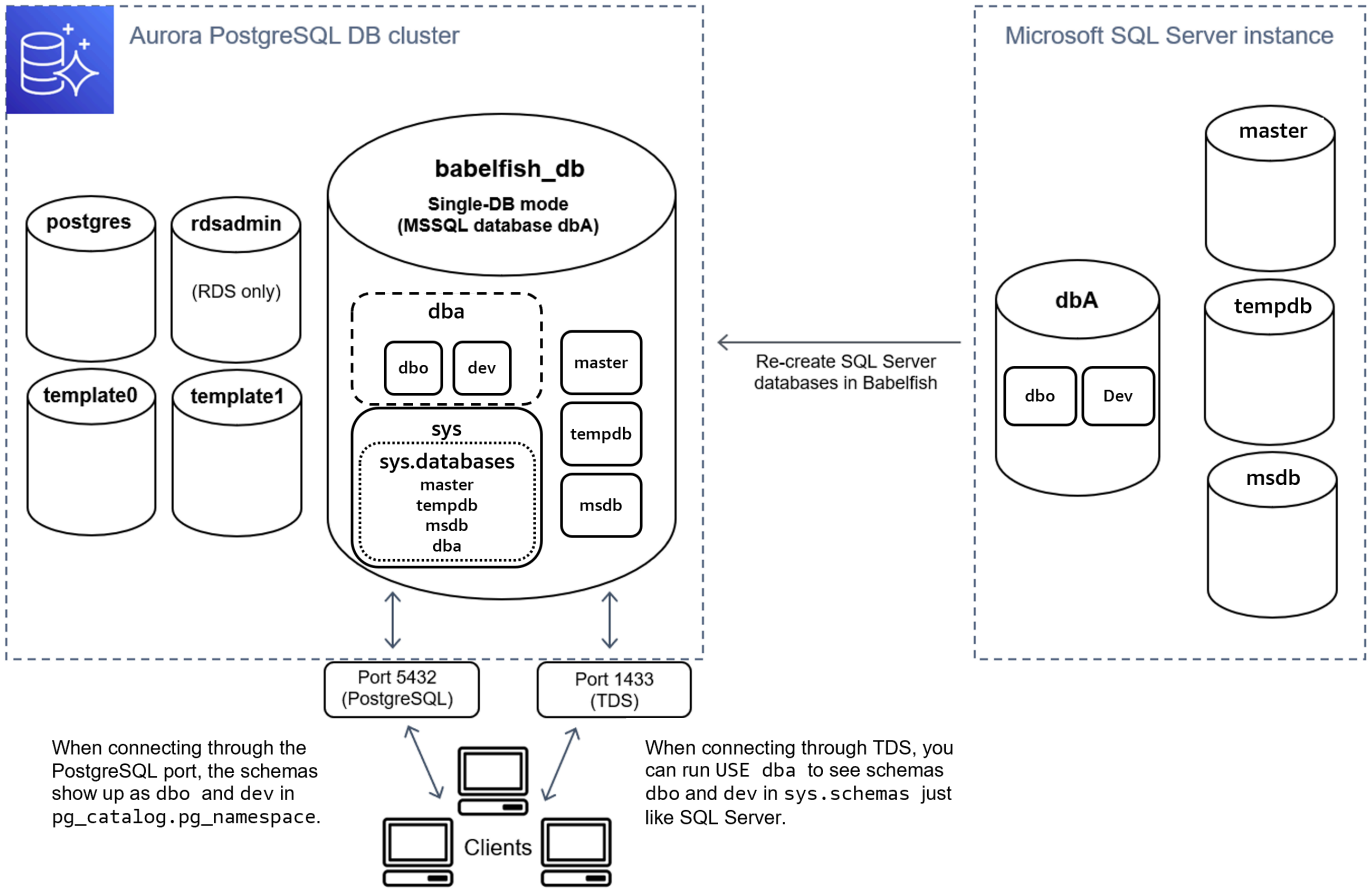
> **Note**
>
> In un cluster Aurora PostgreSQL, il nome del database `babelfish_db` è riservato a Babelfish. La creazione del proprio database "babelfish_db" su un cluster database Babelfish impedisce ad Aurora di eseguire correttamente il provisioning di Babelfish.

Quando ci si connette alla porta TDS, la sessione viene inserita nel database `babelfish_db`. Da T-SQL, la struttura sembra simile alla connessione a un'istanza di SQL Server. Puoi visualizzare i database `master`, `msdb` e `tempdb` e il catalogo `sys.databases`. È possibile creare database utente aggiuntivi e passare da un database all'altro con l'istruzione USE. Quando crei un database utente di SQL Server, viene appiattito nel Database PostgreSQL `babelfish_db`. Il database mantiene la sintassi e la semantica tra database uguali o simili a quelli forniti da SQL Server.

Utilizzo di Babelfish con un singolo database o più database

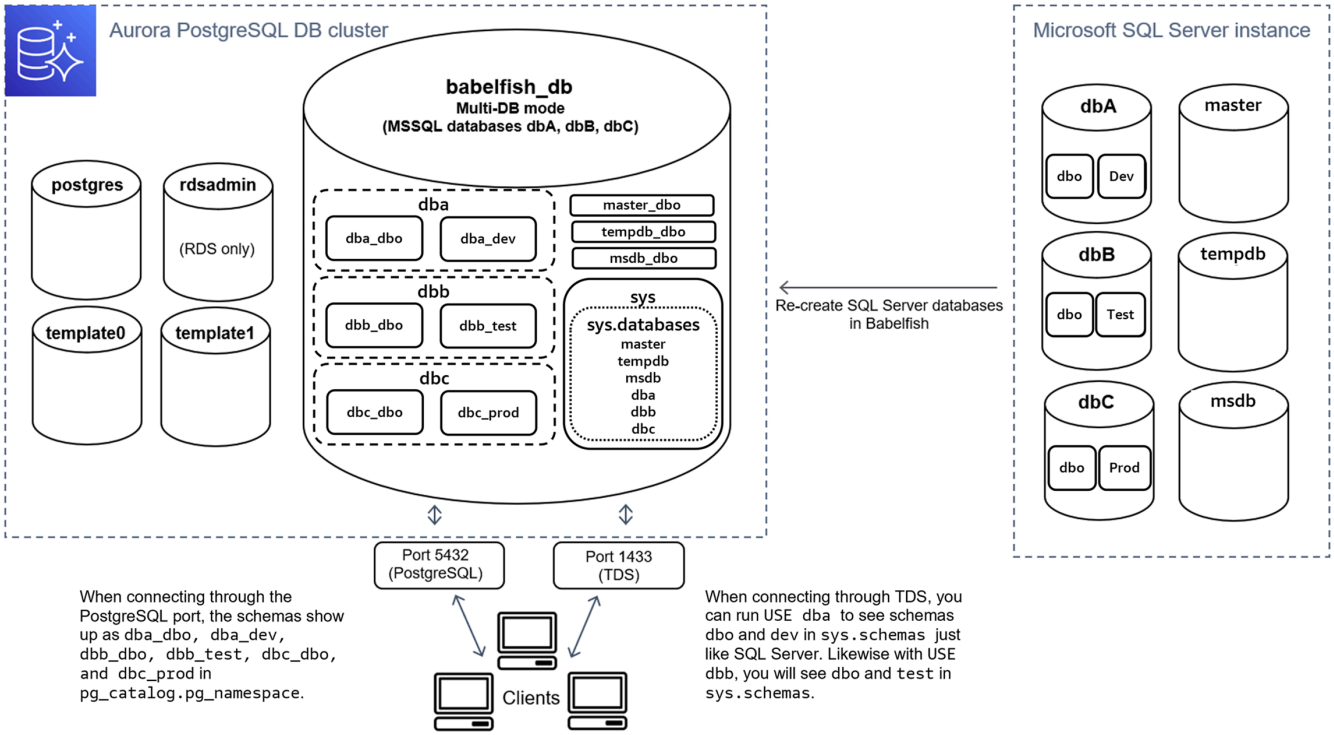
Quando si crea un cluster Aurora PostgreSQL da utilizzare con Babelfish, è possibile scegliere tra l'utilizzo di un singolo database SQL Server sul proprio database SQL Server o più database SQL Server insieme. La tua scelta influisce sul modo in cui i nomi degli schemi di SQL Server all'interno del database `babelfish_db` vengono visualizzati da Aurora PostgreSQL. La modalità di migrazione è memorizzata nel parametro `migration_mode`. Non è necessario modificare questo parametro dopo aver creato il cluster poiché si potrebbe perdere l'accesso a tutti gli oggetti SQL creati in precedenza.

In modalità database singolo, i nomi dello schema del database SQL Server rimangono uguali nel database `babelfish_db` di PostgreSQL. Se scegli di migrare un solo singolo database, i nomi dello schema del database dell'utente migrato possono essere usati come riferimento in PostgreSQL con gli stessi nomi utilizzati in SQL Server. Ad esempio, gli schemi `dbo` e `smith` risiedono all'interno del database `dbA` .



Quando ci si connette tramite TDS, è possibile eseguire `USE dba` per vedere gli schemi dbo e dev da T-SQL, come si farebbe in SQL Server. I nomi dello schema invariati sono visibili anche da PostgreSQL.

In modalità di database multipli, i nomi dello schema dei database utente diventano `dbname_schemaname` se l'accesso viene eseguito da PostgreSQL. I nomi dello schema rimangono gli stessi se l'accesso viene eseguito da T-SQL.



Come mostrato nell'immagine, la modalità database multipli e la modalità database singolo sono le stesse di SQL Server durante la connessione tramite la porta TDS e l'utilizzo di T-SQL. Ad esempio, `USE dbA` elenca schemi `dbo` e `dev` proprio come in SQL Server. I nomi dello schema mappati, come `dba_dbo` e `dba_dev`, sono visibili da PostgreSQL.

Ogni database contiene ancora i tuoi schemi. Il nome di ogni database è preceduto al nome dello schema di SQL Server, utilizzando un trattino di sottolineatura come delimitatore, ad esempio:

- `dba` contiene `dba_dbo` e `dba_dev`.
- `dbb` contiene `dbb_dbo` e `dbb_test`.
- `dbc` contiene `dbc_dbo` e `dbc_prod`.

All'interno del database`babelfish_db`, l'utente T-SQL deve ancora eseguire `USE dbname` per modificare il contesto del database, in modo che l'aspetto rimanga simile a SQL Server.

Scegliere una modalità di migrazione

Ciascuna modalità di migrazione presenta vantaggi e svantaggi. Scegli la modalità di migrazione in base al numero di database utente e ai piani di migrazione. Dopo aver creato un cluster da utilizzare con Babelfish, non devi modificare la modalità di migrazione poiché potresti perdere l'accesso a tutti

gli oggetti SQL creati in precedenza. Quando scegli una modalità di migrazione, considera i requisiti dei database utente e dei client.

Quando crei un cluster da utilizzare con Babelfish, Aurora PostgreSQL crea i database di sistema, `master` e `tempdb`. Se sono stati creati o modificati oggetti nei database di sistema (`master` o `tempdb`), assicurati di ricreare questi oggetti nel nuovo cluster. A differenza di SQL Server, Babelfish non si reinizializza `tempdb` dopo il riavvio di un cluster.

Utilizzare la modalità di migrazione di database singolo nei seguenti casi:

- Se si sta eseguendo la migrazione di un singolo database SQL Server. In modalità database singolo, se l'accesso viene eseguito da PostgreSQL i nomi degli schemi migrati sono identici ai nomi dello schema originale di SQL Server. Ciò riduce le modifiche al codice per le query SQL esistenti se desideri ottimizzarle per l'esecuzione con una connessione PostgreSQL.
- Se il tuo obiettivo finale è una migrazione completa alla nativa Aurora PostgreSQL. Prima di eseguire la migrazione, consolidare gli schemi in un unico schema (`dbo`) e quindi migrare in un singolo cluster per ridurre le modifiche richieste.

Utilizzare la modalità di migrazione di più database nei seguenti casi:

- Se desideri l'esperienza SQL Server predefinita con più database utente nella stessa istanza.
- Se è necessario migrare insieme più database dell'utente.

Impostazioni del gruppo di parametri del cluster database per Babelfish

Quando crei un cluster database Aurora PostgreSQL e scegli Turn on Babelfish (Attiva Babelfish), un gruppo di parametri del cluster database viene creato automaticamente se scegli Create new (Crea nuovo). Questo gruppo di parametri del cluster database si basa sul gruppo di parametri del cluster database Aurora PostgreSQL per la versione di Aurora PostgreSQL scelta per l'installazione, ad esempio Aurora PostgreSQL versione 14. Viene chiamato utilizzando il seguente modello generale:

```
custom-aurora-postgresql14-babelfish-compat-3
```

Durante il processo di creazione del cluster puoi modificare le seguenti impostazioni, ma alcune di queste non possono essere modificate dopo che sono state memorizzate nel gruppo di parametri personalizzati, quindi scegli attentamente:

- Database singolo o database multipli
- Impostazioni locali delle regole di confronto predefinite
- Nome della regola di confronto
- DB parameter group (Gruppo di parametri database)

Per utilizzare un gruppo di parametri esistente del cluster database Aurora PostgreSQL versione 13 o successive, modifica il gruppo e imposta il parametro `babelfish_status` su on. Specificare tutte le opzioni Babelfish prima di creare il cluster Aurora PostgreSQL. Per ulteriori informazioni, consulta Gruppi di parametri per Amazon Aurora.

I seguenti parametri controllano le preferenze di Babelfish. Salvo laddove diversamente indicato nella descrizione, i parametri sono modificabili. Il valore predefinito è incluso nella descrizione. Per visualizzare i valori consentiti per qualsiasi parametro, procedi come segue:

> **Note**
>
> Quando si associa un nuovo gruppo parametri del database a un'istanza database, i parametri statici e dinamici modificati vengono applicati solo dopo il riavvio dell'istanza database. Tuttavia, se modifichi i parametri dinamici nel gruppo di parametri database associato all'istanza database, tali modifiche vengono applicate immediatamente senza eseguire il riavvio.

1. Accedi alla Console di gestione AWS e apri la console Amazon RDS all'indirizzo https://console.aws.amazon.com/rds/.
2. Nel menu di navigazione scegli Parameter groups (Gruppi di parametri).
3. Nell'elenco, scegli il gruppo di parametri del cluster database `default.aurora-postgresql14`.
4. Inserisci il nome di un parametro nel campo di ricerca. Ad esempio, inserisci `babelfishpg_tsql.default_locale` nel campo di ricerca per visualizzare questo parametro e il suo valore predefinito e le impostazioni consentite.

> **Note**
>
> I database globali Babelfish per Aurora PostgreSQL funzionano nelle Regioni secondarie solo se in tali regioni sono attivati i seguenti parametri.

Parametro	Descrizione	Applica tipo	Is Modifiable
babelfishpg_tsql.apg_enable_correlated_scalar_transform	Consente al pianificatore di trasformare la sottoquery scalare correlata in Babelfish. (Predefinito: on) (Consentito: on, off)	dynamic	true
babelfishpg_tsql.apg_enable_subquery_cache	Abilita l'uso della cache per sottoquery scalari correlate in Babelfish. (Predefinito: on) (Consentito: on, off)	dynamic	true
babelfishpg_tds.tds_default_numeric_scale	Imposta la scala predefinita di tipo numerico da inviare nei metadati della colonna TDS se il motore non ne	dynamic	true

Parametro	Descrizione	Applica tipo	Is Modifiable
	specifica uno. (Predefinito: 8) (Consentito: 0–38)		
babelfishpg_tds.tds_default_numeric_precision	Un valore intero che imposta la precisione predefinita del tipo numerico da inviare nei metadati della colonna TDS se il motore non ne specifica uno. (Predefinito: 38) (Consentito: 1–38)	dynamic	true
babelfishpg_tds.tds_default_packet_size	Un valore intero che imposta la dimensione predefinita del pacchetto per la connessione di client SQL Server. (Predefinito: 4096) (Consentito: 512–32767)	dynamic	true

Parametro	Descrizione	Applica tipo	Is Modifiable
babelfishpg_tds.tds_default_protocol_version	Un valore intero che imposta una versione del protocollo TDS predefinita per la connessione dei client. (Predefinito: DEFAULT) (Consentito: TDSv7.0, TDSv7.1, TDSv7.1.1, TDSv7.2, TDSv7.3A, TDSv7.3B, TDSv7.4, DEFAULT)	dynamic	true
babelfishpg_tds.default_server_name	Una stringa che identifica il nome predefinito del server Babelfish. (Predefinito: Microsoft SQL Server) (Consentito: null)	dynamic	true
babelfishpg_tds.tds_debug_log_level	Un valore intero che imposta il livello di registrazione in TDS; 0 disattiva la registrazione. (Predefinito: 1) (Consentito: 0, 1, 2, 3)	dynamic	true

Parametro	Descrizione	Applica tipo	Is Modifiable
babelfishpg_tds.listen_address	Una stringa che imposta il nome host o l'indirizzo IP o gli indirizzi su cui ascoltare TDS. Questo parametro non può essere modificato dopo la creazione del cluster database Babelfish . (Predefinito: *) (Consentito: null)	–	false
babelfishpg_tds.port	Un valore intero che imposta la porta TCP utilizzata per le richieste nella sintassi di SQL Server. (Predefinito: 1433) (Consentito: 1–65535)	static	true
babelfishpg_tds.tds_ssl_encrypt	Un valore booleano che attiva o disattiva la crittografia (0) (1) per i dati che attraversano la porta del listener TDS. Per informazioni dettagliate sull'utilizzo di SSL per le connessioni client, consulta Impostazioni SSL Babelfish e connessioni client. (Predefinito: 0) (Consentito: 0, 1)	dynamic	true

Parametro	Descrizione	Applica tipo	Is Modifiable
babelfishpg_tds.tds_ssl_max_protocol_version	Una stringa che specifica la versione del protocollo SSL/TLS più alta da utilizzare per la sessione TDS. (Predefinito: 'TLSv1.2') (Consentito: 'TLSv1', 'TLSv1.1', 'TLSv1.2')	dynamic	true
babelfishpg_tds.tds_ssl_min_protocol_version	Una stringa che specifica la versione del protocollo SSL/TLS minima da utilizzare per la sessione TDS. (Predefinito: ’TLSv1.2’ da Aurora PostgreSQL versione 16, ’TLSv1’ per le versioni precedenti ad Aurora PostgreSQL versione 16) (Consentito: ’TLSv1’, ’TLSv1.1’, ’TLSv1.2’)	dynamic	true

Parametro	Descrizione	Applica tipo	Is Modifiable
babelfishpg_tds.unix_socket_directories	Una stringa che identifica la directory socket Unix server TDS. Questo parametro non può essere modificato dopo la creazione del cluster database Babelfish. (Predefinito: /tmp) (Consentito: null)	–	false
babelfishpg_tds.unix_socket_group	Una stringa che identifica il gruppo socket Unix server TDS. Questo parametro non può essere modificato dopo la creazione del cluster database Babelfish. (Predefinito: rdsdb) (Consentito: null)	–	false

Parametro	Descrizione	Applica tipo	Is Modifiable
babelfishpg_tsql.default_locale	Una stringa che specifica le impostazioni locali predefinite utilizzate per le regole di confronto Babelfish. Le impostazioni locali predefinite sono le uniche e non includono eventuali qualificatori. Impostare questo parametro quando si esegue il provisioning di un cluster Babelfish DB. Dopo aver eseguito il provisioning del cluster database, le modifiche apportate a questo parametro vengono ignorate. (Predefinito: en_US) (Consentito: consulta tabelle)	static	true

Parametro	Descrizione	Applica tipo	Is Modifiable
babelfishpg_tsql.migration_mode	Un elenco non modificabile che specifica il supporto per database a un solo utente o più utenti. Impostare questo parametro quando si esegue il provisioning di un cluster Babelfish DB. Dopo aver eseguito il provisioning del cluster database, non è possibile modificare il valore di questo parametro. (Predefinito: multi-db da Aurora PostgreSQL versione 16, single db per versioni precedenti ad Aurora PostgreSQL versione 16) (Consentito: single db, multi-db, null)	static	true

Parametro	Descrizione	Applica tipo	Is Modifiable
babelfishpg_tsql.server_collation_name	Una stringa che specifica il nome della regola di confronto utilizzata per le azioni a livello di server. Impostare questo parametro quando si esegue il provisioning di un cluster Babelfish DB. Dopo aver eseguito il provisioning del cluster di database, non modificare il valore di questo parametro. (Predefinito: bbf_unicode_general_ci_as) (Consentito: consulta tabelle)	static	true
babelfishpg_tsql.versione	Una stringa che imposta l'output della variabile @@VERSION. Non modificare questo valore per i cluster Aurora PostgreSQL DB. (Predefinito: null) (Consentito: default)	dynamic	true

Parametro	Descrizione	Applica tipo	Is Modifiable
rds.babelfish_status	Una stringa che imposta lo stato della funzionalità Babelfish. Quando questo parametro è impostato su `datatypesonly` , Babelfish è disattivato ma i tipi di dati di SQL Server sono ancora disponibili. (Predefinito: off) (Consentito: on, off, datatypesonly)	static	true
unix_socket_permissions	Un valore intero che imposta le autorizzazioni socket Unix server TDS. Questo parametro non può essere modificato dopo la creazione del cluster database Babelfish. (Predefinito: 0700) (Consentito: 0–511)	–	false

Impostazioni SSL Babelfish e connessioni client

Per richiedere le connessioni SSL/TLS al cluster di database Babelfish per Aurora PostgreSQL, utilizzare il parametro `rds.force_ssl`.

- Per richiedere le connessioni SSL/TLS, impostare il valore del parametro `rds.force_ssl` su 1 (on).
- Per disattivare le connessioni SSL/TLS richieste, impostare il valore del parametro `rds.force_ssl` su 0 (off).

Il valore predefinito di questo parametro dipende dalla versione di Aurora PostgreSQL:

- Per Aurora PostgreSQL versioni 17 e successive: il valore predefinito è 1 (on).
- Per Aurora PostgreSQL versioni 16 e precedenti: il valore predefinito è 0 (off).

> **Note**
>
> Quando si esegue un aggiornamento di versione principale da Aurora PostgreSQL versione 16 o precedente alla versione 17 o successiva, il valore predefinito del parametro cambia da 0 (off) a 1 (on). Questa modifica può causare errori di connettività per le applicazioni non configurate per SSL. È possibile ripristinare il comportamento predefinito precedente impostando questo parametro su 0 (off).

Per informazioni specifiche del driver, consulta Connessione a un cluster database Babelfish.

Quando un client si connette alla porta TDS (default `1433`), Babelfish confronta l'impostazione Secure Sockets Layer (SSL) inviata durante l'handshake del client con l'impostazione del parametro SSL Babelfish (`tds_ssl_encrypt`). Babelfish determina quindi se è consentita una connessione. Se è consentita una connessione, il comportamento di crittografia viene applicato o meno, a seconda delle impostazioni dei parametri e del supporto per la crittografia offerto dal client.

La tabella seguente mostra come si comporta Babelfish per ogni combinazione.

Impostazione del client dell'app	Impostazione SSL Babelfish	rds.force_ssl	Connessione consentita?	Valore restituito al client
ENCRYPT_ON	Qualsiasi	Qualsiasi	Consentito, l'intera connessione è crittografata	ENCRYPT_ON
ENCRYPT_OFF	tds_ssl_encrypt=1	Qualsiasi	Consentito, l'intera connessione è crittografata	ENCRYPT_REQ
ENCRYPT_OFF	tds_ssl_encrypt=0	rds.force_ssl=0	Consentito, il pacchetto di accesso è crittografato	ENCRYPT_OFF
ENCRYPT_OFF	tds_ssl_encrypt=0	rds.force_ssl=1	No, connessione chiusa	ENCRYPT_OFF
ENCRYPT_NOT_SUP	tds_ssl_encrypt=0	rds.force_ssl=0	Sì	ENCRYPT_NOT_SUP
ENCRYPT_NOT_SUP	tds_ssl_encrypt=1	Qualsiasi	No, connessione chiusa	ENCRYPT_REQ
ENCRYPT_NOT_SUP	tds_ssl_encrypt=0	rds.force_ssl=1	No, connessione chiusa	ENCRYPT_NOT_SUP
ENCRYPT_CLIENT_CERT	Qualsiasi	Qualsiasi	No, connessione chiusa	Non supportato.

Informazioni sulle regole di confronto in Babelfish per Aurora PostgreSQL.

Quando crei un cluster database Aurora PostgreSQL, scegli una regola di confronto per i dati. Una regola di confronto specifica i modelli di ordinamento e bit che producono il testo o i caratteri in una determinata lingua scritta. Una regola di confronto include regole che confrontano i dati per un determinato set di modelli di bit. La regola di confronto è correlata alla localizzazione. Le diverse impostazioni locali influenzano la mappatura dei caratteri, l'ordinamento e simili. Gli attributi della regola di confronto si riflettono nei nomi di varie regole di confronto. Per ulteriori informazioni sugli attributi, consulta Babelfish collation attributes table.

Babelfish mappa le raccolte di SQL Server a raccolte comparabili fornite da Babelfish. Babelfish predefinisce le raccolte Unicode con raccolte di stringhe culturalmente sensibili e ordini di ordinamento. Babelfish fornisce anche un modo per tradurre le raccolte nel database di SQL Server nella raccolta Babelfish più vicina. Vengono fornite raccolte specifice per le diverse lingue e regioni.

Alcune raccolte specificano una pagina codice corrispondente a una codifica lato client. Babelfish traduce automaticamente dalla codifica del server alla codifica client a seconda della raccolta di ciascuna colonna di output.

Babelfish supporta le regole di confronto elencate nella Babelfish supported collations table. Babelfish mappa le raccolte di SQL Server a raccolte comparabili fornite da Babelfish.

Babelfish utilizza la versione 153.80 della libreria delle regole di confronto International Components for Unicode (ICU). Per ulteriori informazioni sulle regole di confronto ICU, consulta Collation (Regola di confronto) nella documentazione di ICU. Per ulteriori informazioni su PostgreSQL e regola di confronto, consulta Collation Support (Supporto per regole di confronto) nella documentazione di PostgreSQL.

Argomenti

- Parametri del cluster database che controllano regole di confronto e impostazioni locali
- Regole di confronto deterministiche e non deterministiche in Babelfish
- Regole di confronto supportate a livello di database in Babelfish
- Regole di confronto di server e oggetti in Babelfish
- Comportamento della regola di confronto predefinita in Babelfish
- Gestione delle raccolte
- Limitazioni di regole di confronto e differenze di comportamento

Parametri del cluster database che controllano regole di confronto e impostazioni locali

I seguenti parametri influiscono sul comportamento della regola di confronto.

babelfishpg_tsql.default_locale

Questo parametro specifica le impostazioni locali predefinite utilizzate dalla regola di confronto. Questo parametro viene utilizzato in combinazione con gli attributi elencati nella Babelfish collation attributes table per personalizzare le regole di confronto per una lingua e una regione specifici. Il valore predefinito per questo parametro è `en-US`.

Le impostazioni locali predefinite si applicano a tutti i nomi delle regole di confronto Babelfish che iniziano con "BBF" e a tutte le regole di confronto SQL Server mappate alle regole di confronto Babelfish. La modifica dell'impostazione di questo parametro su un cluster database Babelfish esistente non influenza le impostazioni locali delle regole di confronto esistenti. Per l'elenco delle regole di confronto, consulta Babelfish supported collations table.

babelfishpg_tsql.server_collation_name

Questo parametro specifica la regola di confronto predefinita per il server (istanza del cluster database Aurora PostgreSQL) e il database. Il valore predefinito è `sql_latin1_general_cp1_ci_as`. Il `server_collation_name` deve essere una raccolta `CI_AS` perché in T-SQL, la raccolta determina il modo in cui vengono confrontati gli identificatori.

Quando crei il tuo cluster database Babelfish, scegli il Collation name (Nome regola di confronto) dall'elenco selezionabile. Questi includono le regole di confronto elencate in Babelfish supported collations table. Non modificare il `server_collation_name` dopo la creazione del database Babelfish.

Le impostazioni scelte durante la creazione del cluster database Babelfish per Aurora PostgreSQL vengono memorizzate nel gruppo di parametri del cluster database associato al cluster per questi parametri e impostano il comportamento della regola di confronto.

Regole di confronto deterministiche e non deterministiche in Babelfish

Babelfish supporta le raccolte deterministiche e non deterministiche:

- Una regola di confronto deterministica valuta come uguali i caratteri con sequenze di byte identiche. Ciò significa che `x` e `X` non sono uguali in una regola di confronto deterministica. Le regole di confronto deterministiche possono essere con distinzione maiuscole/minuscole (CS) e con distinzione caratteri accentati/non accentati (AS).

- Una regola di confronto non deterministica non richiede una corrispondenza identica. Una raccolta non deterministica valuta x e X come uguale. Le regole di confronto non deterministiche sono senza distinzione tra maiuscole e minuscole (CI) e senza distinzione tra caratteri accentati e non accentati (AI).

Nella tabella seguente sono riportate alcune differenze di comportamento tra Babelfish e PostgreSQL quando si utilizzano le regole di confronto non deterministiche.

Babelfish	PostgreSQL
Supporta la clausola LIKE per le regole di confronto CI_AS.	Non supporta la clausola LIKE sulle regole di confronto non deterministiche.
Supporta la clausola LIKE solo per le seguenti regole di confronto AI di Babelfish versione 4.2.0: • bbf_unicode_cp1250_ci_ai • bbf_unicode_cp1250_cs_ai • bbf_unicode_cp1257_ci_ai • bbf_unicode_cp1257_cs_ai • bbf_unicode_cp1_ci_ai • bbf_unicode_cp1_cs_ai • estonian_ci_ai • finnish_swedish_ci_ai • french_ci_ai • latin1_general_ci_ai • latin1_general_cs_ai • modern_spanish_ci_ai • polish_ci_ai • sql_latin1_general_cp1_ci_ai • sql_latin1_general_cp1_cs_ai • traditional_spanish_ci_ai	Non supporta la clausola LIKE sulle regole di confronto non deterministiche.

Per un elenco di altre limitazioni e differenze di comportamento per Babelfish rispetto a SQL Server e PostgreSQL, consulta Limitazioni di regole di confronto e differenze di comportamento.

Babelfish e SQL Server seguono una convenzione di denominazione per le raccolte che descrivono gli attributi di raccolta, come illustrato nella tabella seguente.

Attributo	Descrizione
Intelligenza artificiale	Insensibile all'accento.
AS	sensibile agli accenti.
BIN2	BIN2 richiede che i dati siano ordinati in ordine di codice. L'ordine dei punti di codice Unicode è lo stesso ordine di caratteri per le codifiche UTF-8, UTF-16 e UCS-2. L'ordine dei punti di codice è una raccolta deterministica rapida.
CI	Senza distinzione tra maiuscole e minuscole
CS	Distinzione tra lettere maiuscole e minuscole
PREF	Per ordinare le lettere maiuscole prima delle lettere minuscole, utilizzar e una raccolta PREF. Se la raccolta è priva di distinzione tra maiuscole e minuscole, la versione maiuscola di una lettera viene ordinata prima della versione minuscola, se non esiste altra distinzione. La libreria ICU supporta le preferenze maiuscole con `colCaseFirst=upper` , ma non per le raccolte CI_AS. La PREF può essere applicata solo a raccolte deterministiche `CS_AS`.

Regole di confronto supportate a livello di database in Babelfish

Le seguenti regole di confronto sono supportate a livello di database in Babelfish:

- bbf_unicode_bin2
- bbf_unicode_cp1_ci_ai
- bbf_unicode_cp1_ci_as
- bbf_unicode_cp1250_ci_ai

- bbf_unicode_cp1250_ci_as
- bbf_unicode_cp1257_ci_ai
- bbf_unicode_cp1257_ci_as
- estonian_ci_ai
- estonian_ci_as
- finnish_swedish_ci_ai
- finnish_swedish_ci_as
- french_ci_ai
- french_ci_as
- latin1_general_bin2
- latin1_general_ci_ai
- latin1_general_ci_as
- latin1_general_90_bin2
- latin1_general_100_bin2
- latin1_general_140_bin2
- modern_spanish_ci_ai
- modern_spanish_ci_as
- polish_ci_ai
- polish_ci_as
- sql_latin1_general_cp1_ci_ai
- SQL_Latin1_General_CP1_CI_AS (default)
- sql_latin1_general_cp1250_ci_as
- sql_latin1_general_cp1251_ci_as
- sql_latin1_general_cp1257_ci_as
- traditional_spanish_ci_ai
- traditional_spanish_ci_as

> **Note**
>
> Per utilizzare regole di confronto diverse a livello di database, è necessario che corrispondano a quelle a livello di server. Per ulteriori informazioni, consulta Regole di confronto di server e oggetti in Babelfish

Regole di confronto di server e oggetti in Babelfish

Utilizzare le raccolte seguenti come raccolta del server o come raccolta di oggetti.

ID raccolta	Note
bbf_unicode_general_ci_as	Supporta il confronto tra maiuscole e minuscole e l'operatore LIKE.
bbf_unicode_cp1_ci_as	Raccolta non deterministica nota anche come CP1252.
bbf_unicode_CP1250_ci_as	Raccolta non deterministica utilizzata per rappresentare testi nelle lingue dell'Europa centrale e dell'Europa orientale che utilizzano lo script latino.
bbf_unicode_CP1251_ci_as	Raccolta non deterministica per le lingue che utilizzano lo script cirillico.
bbf_unicode_cp1253_ci_as	Raccolta non deterministica usata per rappresentare il greco moderno.
bbf_unicode_cp1254_ci_as	Raccolta non deterministica che supporta il turco.
bbf_unicode_cp1255_ci_as	Raccolta non deterministica che supporta l'ebraico.
bbf_unicode_cp1256_ci_as	Raccolta non deterministica utilizzata per scrivere lingue che utilizzano lo script arabo.

ID raccolta	Note
bbf_unicode_cp1257_ci_as	Raccolta non deterministica utilizzata per supportare le lingue estone, lettone e lituano.
bbf_unicode_cp1258_ci_as	Raccolta non deterministica utilizzata per scrivere caratteri vietnamiti.
bbf_unicode_cp874_ci_as	Raccolta non deterministica utilizzata per scrivere caratteri thailandesi.
sql_latin1_general_cp1250_ci_as	Codifica di caratteri a byte singolo non deterministico usata per rappresentare caratteri latini.
sql_latin1_general_cp1251_ci_as	Raccolta non deterministica che supporta i caratteri latini.
SQL_Latin1_General_CP1_CI_AS (default)	Raccolta non deterministica che supporta i caratteri latini.
sql_latin1_general_cp1253_ci_as	Raccolta non deterministica che supporta i caratteri latini.
sql_latin1_general_cp1254_ci_as	Raccolta non deterministica che supporta i caratteri latini.
sql_latin1_general_cp1255_ci_as	Raccolta non deterministica che supporta i caratteri latini.
sql_latin1_general_cp1256_ci_as	Raccolta non deterministica che supporta i caratteri latini.

ID raccolta	Note
sql_latin1_general_cp1257_ci_as	Raccolta non deterministica che supporta i caratteri latini.
sql_latin1_general_cp1258_ci_as	Raccolta non deterministica che supporta i caratteri latini.
chinese_PRC_CI_AS	Raccolta non deterministica che supporta il cinese (RPC).
cyrillic_general_ci_as	Raccolta non deterministica che supporta il cirillico.
finnish_swedish_ci_as	Raccolta non deterministica che supporta che supporta il finlandese.
french_ci_as	Raccolta non deterministica che supporta il francese.
japanese_ci_as	Regola di confronto non deterministica che supporta il giapponese. Supportata in Babelfish 2.1.0 e versioni successive.
korean_wansung_ci_as	Raccolta non deterministica che supporta il coreano (con ordinamento del dizionario).
latin1_general_ci_as	Raccolta non deterministica che supporta i caratteri latini.
modern_spanish_ci_as	Raccolta non deterministica che supporta lo spagnolo moderno.
polish_ci_as	Raccolta non deterministica che supporta il polacco.
thai_ci_as	Raccolta non deterministica che supporta il tailandese.

ID raccolta	Note
traditional_spanish_ci_as	Raccolta non deterministica che supporta lo spagnolo (classificazione tradizionale).
turkish_ci_as	Raccolta non deterministica che supporta il turco.
ukrainian_ci_as	Raccolta non deterministica che supporta l'ucraino.
vietnamese_ci_as	Raccolta non deterministica che supporta il vietnamita.

È possibile utilizzare le seguenti raccolte come raccolte tra oggetti.

Dialetto	Opzioni deterministiche	Opzioni non deterministiche
Arabo	Arabic_cs_as	Arabic_CI_AS Arabic_CI_AI
Alfabeto arabo	BBF_Unicode_CP1256_CS_AS BBF_Unicode_Pref_CP1256_CS_AS	BBF_Unicode_CP1256_CI_AI BBF_Unicode_CP1256_CS_AI
Binario	latin1_general_bin2 BBF_Unicode_BIN2	–
Lingue dell’Europa centrale e dell’Europa orientale che utilizzano l’alfabeto latino	BBF_Unicode_CP1250_CS_AS BBF_Unicode_Pref_CP1250_CS_AS	BBF_Unicode_CP1250_CI_AI BBF_Unicode_CP1250_CS_AI

Dialetto	Opzioni deterministiche	Opzioni non deterministiche
Cinese	Chinese_PRC_CS_AS	Chinese_PRC_CI_AS Chinese_PRC_CI_AI
Cyrillic_General	Cyrillic_general_cs_as	Cyrillic_General_CI_AS Cyrillic_General_CI_AI
Alfabeto cirillico	BBF_Unicode_CP1251_CS_AS BBF_Unicode_Pref_CP1251_CS_AS	BBF_Unicode_CP1251_CI_AI BBF_Unicode_CP1251_CS_AI
Estone	Estonian_cs_as	Estonian_CI_AS Estonian_CI_AI
Estone, lettone e lituano	BBF_Unicode_CP1257_CS_AS BBF_Unicode_Pref_CP1257_CS_AS	BBF_Unicode_CP1257_CI_AI BBF_Unicode_CP1257_CS_AI
Finnish_Swedish	Finnish_Swedish_CS_AS	Finnish_Swedish_CI_AS Finnish_Swedish_CI_AI
Francese	French_CS_AS	French_CI_AS French_CI_AI
Greco	Greek_CS_AS	Greek_CI_AS Greek_CI_AI

Dialetto	Opzioni deterministiche	Opzioni non deterministiche
Ebraico	BBF_Unicode_CP1255_CS_AS BBF_Unicode_Pref_CP1255_CS_AS Hebrew_CS_AS	BBF_Unicode_CP1255_CI_AI BBF_Unicode_CP1255_CS_AI Hebrew_CI_AS Hebrew_CI_AI
Giapponese (Babelfish 2.1.0 e versioni successive)	Japanese_CS_AS	Japanese_CI_AI Japanese_CI_AS
Korean_Wamsung	Korean_Wamsung_CS_AS	Korean_Wamsung_CI_AS Korean_Wamsung_CI_AI
Caratteri latini per la tabella codici CP1252	latin1_general_cs_as BBF_Unicode_General_CS_AS BBF_Unicode_General_Pref_CS_AS BBF_Unicode_Pref_CP1_CS_AS BBF_Unicode_CP1_CS_AS	latin1_general_ci_as latin1_general_ci_ai latin1_general_cs_ai BBF_Unicode_General_CI_AI BBF_Unicode_General_CS_AI BBF_Unicode_CP1_CI_AI BBF_Unicode_CP1_CS_AI
Greco moderno	BBF_Unicode_CP1253_CS_AS BBF_Unicode_Pref_CP1253_CS_AS	BBF_Unicode_CP1253_CI_AI BBF_Unicode_CP1253_CS_AI

Dialetto	Opzioni deterministiche	Opzioni non deterministiche
Modern_Spanish	Modern_Spanish_CS_AS	Modern_Spanish_CI_AS Modern_Spanish_CI_AI
Mongolo	Mongolian_CS_AS	Mongolian_CI_AS Mongolian_CI_AI
Polacco	Polish_CS_AS	Polish_CI_AS Polish_CI_AI
Thai	BBF_Unicode_CP874_CS_AS BBF_Unicode_Pref_CP874_CS_AS Thai_CS_AS	BBF_Unicode_CP874_CI_AI BBF_Unicode_CP874_CS_AI Thai_CI_AS, Thai_CI_AI
Tradition al_Spanish	Traditional_Spanish_CS_AS	Traditional_Spanish_CI_AS Traditional_Spanish_CI_AI
Turco	BBF_Unicode_CP1254_CS_AS BBF_Unicode_Pref_CP1254_CS_AS Turkish_CS_AS	BBF_Unicode_CP1254_CI_AI BBF_Unicode_CP1254_CS_AI Turkish_CI_AS, Turkish_CI_AI
Ucraino	Ukranian_CS_AS	Ukranian_CI_AS Ukranian_CI_AI

Dialetto	Opzioni deterministiche	Opzioni non deterministiche
Vietnamita	BBF_Unicode_CP1258_CS_AS	BBF_Unicode_CP1258_CI_AI
	BBF_Unicode_Pref_CP1258_CS_AS	BBF_Unicode_CP1258_CS_AI
	Vietnamese_CS_AS	Vietnamese_CI_AS
		Vietnamese_CI_AI

Comportamento della regola di confronto predefinita in Babelfish

In precedenza, la regola di confronto predefinita dei tipi di dati di regola di confronto era `pg_catalog.default`. I tipi di dati e gli oggetti che dipendono da questi tipi di dati seguono la regola di confronto con distinzione tra maiuscole e minuscole. Questa condizione ha un potenziale impatto sugli oggetti T-SQL del set di dati con la regola di confronto senza distinzione tra maiuscole e minuscole. A partire da Babelfish 2.3.0, la regola di confronto predefinita per i tipi di dati di regola di confronto (eccetto TEXT e NTEXT) è uguale alla regola di confronto del parametro `babelfishpg_tsql.server_collation_name`. Quando esegui l'aggiornamento a Babelfish 2.3.0, la regola di confronto predefinita viene selezionata automaticamente al momento della creazione del cluster database, senza alcun impatto visibile.

Gestione delle raccolte

La libreria ICU fornisce il tracciamento delle raccolte per garantire che gli indici che dipendono dalle raccolte possano essere reindicizzati quando diventa disponibile una nuova versione di ICU. Per verificare se il database corrente contiene regole di confronto che richiedono l'aggiornamento, puoi utilizzare la seguente query dopo aver effettuato la connessione utilizzando `psql` o `pgAdmin`:

```
SELECT pg_describe_object(refclassid, refobjid,
    refobjsubid) AS "Collation",
    pg_describe_object(classid, objid, objsubid) AS "Object"
    FROM pg_depend d JOIN pg_collation c ON refclassid = 'pg_collation'::regclass
    AND refobjid = c.oid WHERE c.collversion <> pg_collation_actual_version(c.oid)
    ORDER BY 1, 2;
```

Questa query restituisce un output del tipo seguente:

```
 Collation | Object
```

```
-----------+--------
(0 rows)
```

In questo esempio, non è necessario aggiornare regole di confronto.

Per ottenere un elenco delle regole di confronto predefinite nel database Babelfish, puoi utilizzare `psql` o `pgAdmin` con la seguente query:

```
SELECT * FROM pg_collation;
```

Le raccolte predefinite sono memorizzate nella tabella `sys.fn_helpcollations`. È possibile utilizzare il seguente comando per visualizzare informazioni su una raccolta (ad esempio i flag lcid, style e collate). Per ottenere un elenco di tutte le regole di confronto utilizzando `sqlcmd`, esegui la connessione alla porta T-SQL (1433, per impostazione predefinita) ed esegui la seguente query:

```
1> :setvar SQLCMDMAXVARTYPEWIDTH 40
2> :setvar SQLCMDMAXFIXEDTYPEWIDTH 40
3> SELECT * FROM fn_helpcollations()
4> GO
name                                    description
--------------------------------------- ----------------------------------------
arabic_cs_as                            Arabic, case-sensitive, accent-sensitive
arabic_ci_ai                            Arabic, case-insensitive, accent-insensi
arabic_ci_as                            Arabic, case-insensitive, accent-sensiti
bbf_unicode_bin2                        Unicode-General, case-sensitive, accent-
bbf_unicode_cp1250_ci_ai                Default locale, code page 1250, case-ins
bbf_unicode_cp1250_ci_as                Default locale, code page 1250, case-ins
bbf_unicode_cp1250_cs_ai                Default locale, code page 1250, case-sen
bbf_unicode_cp1250_cs_as                Default locale, code page 1250, case-sen
bbf_unicode_pref_cp1250_cs_as           Default locale, code page 1250, case-sen
bbf_unicode_cp1251_ci_ai                Default locale, code page 1251, case-ins
bbf_unicode_cp1251_ci_as                Default locale, code page 1251, case-ins
bbf_unicode_cp1254_ci_ai                Default locale, code page 1254, case-ins
...
(124 rows affected)
```

Le righe 1 e 2 mostrate nell'esempio restringono l'output solo ai fini della leggibilità della documentazione.

```
1> SELECT SERVERPROPERTY('COLLATION')
2> GO
```

```
serverproperty
--------------------------------------------------------------------------
sql_latin1_general_cp1_ci_as

(1 rows affected)
1>
```

Limitazioni di regole di confronto e differenze di comportamento

Babelfish utilizza la libreria ICU per il supporto delle raccolte. PostgreSQL è costruito con una versione specifica di ICU e può corrispondere al massimo a una versione di un confronto. Le variazioni tra le versioni sono inevitabili, così come le piccole variazioni nel tempo man mano che le lingue evolvono. Nella sezione seguente sono disponibili le limitazioni note e le variazioni di comportamento delle regole di confronto Babelfish:

- Indici e dipendenza tipo regola di confronto: un indice su un tipo definito dall'utente che dipende dalla libreria di regole di confronto ICU (International Components for Unicode) (la libreria utilizzata da Babelfish) non viene invalidato quando la versione della libreria viene modificata.
- Funzione COLLATIONPROPERTY: le proprietà delle regole di confronto sono implementate solo per le regole di confronto Babelfish BBF supportate. Per ulteriori informazioni, consulta Babelfish supported collations table.
- Differenze regole di ordinamento Unicode: le regole di confronto SQL per SQL Server ordinano i dati con codifica Unicode (`nchar` e `nvarchar`) in modo diverso rispetto ai dati senza codifica Unicode (`char` e `varchar`). I database Babelfish sono sempre codificati UTF-8 e applicano sempre regole di ordinamento Unicode in modo coerente, a prescindere dal tipo di dati, pertanto l'ordinamento per `char` o `varchar` è uguale a quello per `nchar` o `nvarchar`.
- Regole di confronto secondarie uguali e comportamento di ordinamento: la regola di confronto predefinita Unicode secondaria uguale (`CI_AS`) ordina i segni di punteggiatura e altri caratteri non alfanumerici prima dei caratteri numerici e i caratteri numerici prima dei caratteri alfabetici. Tuttavia, l'ordine di punteggiatura e altri caratteri speciali sono diversi.
- Regole di confronto terziarie, soluzione alternativa per ORDER BY: le regole di confronto SQL, ad esempio `SQL_Latin1_General_Pref_CP1_CI_AS`, supportano la funzione `TERTIARY_WEIGHTS` e la capacità di ordinare stringhe che si confrontano allo stesso modo in una regola di confronto `CI_AS` da ordinare prima in maiuscolo: `ABC`, `ABc`, `AbC`, `Abc`, `aBC`, `aBc`, `abC` e infine `abc`. Pertanto la funzione analitica `DENSE_RANK OVER (ORDER BY column)` valuta queste stringhe come aventi lo stesso rango, ma le ordina prima in maiuscolo all'interno di una partizione.

Puoi ottenere un risultato simile con Babelfish aggiungendo una clausola `COLLATE`alla clausola `ORDER BY` che specifica una raccolta terziaria `CS_AS` che specifica `@colCaseFirst=upper`. Tuttavia, il modificatore `colCaseFirst` si applica esclusivamente alle stringhe terziarie uguali (piuttosto che secondarie-uguali come una regola di confronto `CI_AS`). Pertanto, non è possibile emulare raccolte SQL terziarie utilizzando una singola relazione ICU.

Come soluzione alternativa, consigliamo di modificare le applicazioni che utilizzano la raccolta `SQL_Latin1_General_Pref_CP1_CI_AS` per utilizzare la raccolta `BBF_SQL_Latin1_General_CP1_CI_AS` prima di iniziare la raccolta. Quindi aggiungere `COLLATE BBF_SQL_Latin1_General_Pref_CP1_CS_AS`a qualsiasi clausola `ORDER BY` per questa colonna.

- Espansione di caratteri: un'espansione di caratteri tratta un singolo carattere come uguale a una sequenza di caratteri a livello primario. La regola di confronto `CI_AS` predefinita di SQL Server supporta l'espansione di caratteri. Le regole di confronto ICU supportano l'espansione di caratteri solo per regole di confronto senza distinzione caratteri accentati/non accentati.

  Quando è richiesta l'espansione dei caratteri, utilizzare una raccolta `AI` per confronti. Tuttavia, tali raccolte non sono attualmente supportate dall'operatore LIKE.

- Codifica char e varchar: quando le regole di confronto SQL vengono utilizzate per i tipi di dati `char` o `varchar`, l'ordinamento per i caratteri precedenti a ASCII 127 è determinato dalla pagina codice specifica per tale regola di confronto SQL. Per le regole di confronto SQL, le stringhe dichiarate come `char` o `varchar` potrebbero essere ordinate in modo diverso rispetto alle stringhe dichiarate come `nchar` o `nvarchar`.

  PostgreSQL codifica tutte le stringhe con la codifica del database, pertanto tutti i caratteri vengono convertiti in UTF-8 e ordinati utilizzando regole Unicode.

  Poiché le regole di confronto SQL ordinano i tipi di dati nchar e nvarchar utilizzando regole Unicode, Babelfish codifica tutte le stringhe sul server utilizzando UTF-8. Babelfish ordina le stringhe nchar e nvarchar nello stesso modo in cui ordina le stringhe char e varchar, usando regole Unicode.

- Caratteri supplementari: le funzioni SQL Server `NCHAR`, `UNICODE`, e `LEN` supportano i caratteri per i punti di codice al di fuori di Unicode Basic Multilingual Plane (BMP). Al contrario, le raccolte non SC utilizzano caratteri di coppia surrogata per gestire caratteri supplementari. Per i tipi di dati Unicode, SQL Server può rappresentare fino a 65.535 caratteri utilizzando UCS-2 o l'intervallo Unicode completo (1,114,114 caratteri) se si utilizzano caratteri supplementari.

- Regole di confronto con distinzione Kana (KS): una regola di confronto con distinzione Kana (KS) gestisce i caratteri Kana giapponesi `Hiragana` e `Katakana` in modo diverso. ICU supporta lo standard di raccolta giapponese`JIS X 4061`. Il modificatore di localizzazione `colhiraganaQ [on | off]` oramai obsoleto potrebbe fornire la stessa funzionalità delle raccolte KS. Tuttavia, le raccolte KS con lo stesso nome di SQL Server non sono attualmente supportate da Babelfish.
- Regole di confronto con distinzione della larghezza (WS): quando un carattere a byte singolo (mezza larghezza) e lo stesso carattere rappresentato come carattere a doppio byte (larghezza intera) vengono trattati in modo diverso, la regola di confronto viene chiamata con distinzione della larghezza (WS). Le raccolte WS con lo stesso nome di SQL Server non sono attualmente supportate da Babelfish.
- Regole di confronto con distinzione selettore di variazione (VSS): le regole di confronto con distinzione selettore di variazione (VSS) distinguono tra selettori variazione ideografica nelle regole di confronto giapponesi `Japanese_Bushu_Kakusu_140` e `Japanese_XJIS_140`. Una sequenza di varianti è costituita da un carattere base più un selettore di varianti aggiuntivo. Se non viene selezionata l'opzione `_VSS`, il selettore delle varianti non è considerato nel confronto.

  Le relazioni VSS attualmente non sono supportate da Babelfish.
- Regole di confronto BIN e BIN2: una regola di confronto BIN2 ordina i caratteri in base all'ordine dei punti di codice. L'ordine binario byte per byte di UTF-8 conserva l'ordine dei punti di codice Unicode, quindi è probabile che questo sia anche il confronto con le migliori raccolte. Se l'ordine dei punti di codice Unicode funziona per un'applicazione, prendere in considerazione l'utilizzo di una raccolta BIN2. Tuttavia, l'utilizzo di una raccolta BIN2 può comportare la visualizzazione dei dati sul client in un ordine culturalmente inaspettato. I nuovi mapping con caratteri minuscoli vengono aggiunti a Unicode man mano che il tempo progredisce, quindi la funzione `LOWER` potrebbe funzionare in modo diverso su diverse versioni di ICU. Questo è un caso speciale del problema di controllo delle versioni di raccolta più generale piuttosto che come qualcosa di specifico per la raccolta BIN2.

  Babelfish fornisce la raccolta `BBF_Latin1_General_BIN2` con la distribuzione Babelfish da collare nell'ordine dei punti di codice Unicode. In una raccolta BIN solo il primo carattere viene ordinato come wchar. I caratteri rimanenti vengono ordinati byte per byte, in modo efficace in ordine di codice in base alla sua codifica. Questo approccio non segue le regole di confronto Unicode e non è supportato da Babelfish.
- Regole di confronto non deterministiche e limitazione CHARINDEX: per versioni di Babelfish precedenti alla versione 2.1.0, non è possibile utilizzare CHARINDEX con regole di confronto non deterministiche. Per impostazione predefinita, Babelfish utilizza una regola di confronto senza

distinzione maiuscole/minuscole (non deterministica). L'utilizzo di CHARINDEX per le versioni precedenti di Babelfish genera il seguente errore di runtime:

```
nondeterministic collations are not supported for substring searches
```

> **Note**
>
> Questa limitazione e la soluzione alternativa si applicano solo a Babelfish versione 1.x (Aurora PostgreSQL versioni 13.x). Babelfish 2.1.0 e versioni successive non presentano questo problema.

Puoi risolvere il problema in uno dei seguenti modi:

- Converti esplicitamente l'espressione in un confronto con distinzione tra maiuscole e minuscole e trasforma entrambi gli argomenti applicando LOWER o UPPER. Ad esempio, `SELECT charindex('x', a) FROM t1` diventa ciò che è indicato qui di seguito:

  ```
  SELECT charindex(LOWER('x'), LOWER(a COLLATE sql_latin1_general_cp1_cs_as)) FROM t1
  ```

- Crea una funzione SQL f_charindex e sostituisci le chiamate CHARINDEX con le chiamate alla seguente funzione:

  ```
  CREATE function f_charindex(@s1 varchar(max), @s2 varchar(max)) RETURNS int
  AS
  BEGIN
  declare @i int = 1
  WHILE len(@s2) >= len(@s1)
  BEGIN
    if LOWER(@s1) = LOWER(substring(@s2,1,len(@s1))) return @i
    set @i += 1
    set @s2 = substring(@s2,2,999999999)
  END
  return 0
  END
  go
  ```

Gestione degli errori di Babelfish con escape hatch

Babelfish simula il comportamento SQL per flusso di controllo e stato della transazione quando possibile. Se Babelfish rileva un errore, restituisce un codice di errore simile al codice di errore di SQL Server. Se Babelfish non riesce a mappare l'errore a un codice SQL Server, restituisce un codice di errore fisso (`33557097`) ed esegue azioni specifiche in base al tipo di errore, come segue:

- Per errori in fase di compilazione, Babelfish esegue il rollback della transazione.
- Per errori di runtime, Babelfish termina il batch ed esegue il rollback della transazione.
- Per errore di protocollo tra client e server, non viene eseguito il rollback della transazione.

Se un codice di errore non può essere mappato a un codice equivalente e il codice per un errore simile è disponibile, il codice di errore viene mappato al codice alternativo. Ad esempio, i comportamenti che causano i codici SQL Server `8143` e `8144` sono entrambi mappati su `8143`.

Gli errori che non possono essere mappati non rispettano un costrutto `TRY... CATCH`.

È possibile utilizzare `@@ERROR` per restituire un codice di errore di SQL Server oppure la funzione `@@PGERROR` per restituire un codice di errore PostgreSQL. È possibile utilizzare anche la funzione `fn_mapped_system_error_list` per restituire un elenco di codici di errore mappati. Per informazioni sui codici di errore PostgreSQL, consulta il sito web di https://www.postgresql.org/docs/current/errcodes-appendix.html.

Modifica delle impostazioni dell'escape hatch Babelfish

Per gestire meglio le istruzioni che potrebbero non andare a buon fine, Babelfish definisce alcune opzioni chiamate escape hatch. Una via di fuga è un'opzione che specifica il comportamento di Babelfish quando incontra una caratteristica o una sintassi non supportata.

Puoi utilizzare la procedura memorizzata `sp_babelfish_configure` per controllare le impostazioni di una via di fuga. Usa lo script per impostare la via di fuga su `ignore` o `strict`. Se è impostata su `strict`, Babelfish restituisce un errore che è necessario correggere prima di continuare.

Includere la parola chiave `server` per applicare le modifiche alla sessione corrente e a livello di cluster.

L'utilizzo è il seguente:

- Per elencare tutte le vie di fuga e il relativo stato, oltre alle informazioni sull'utilizzo, eseguire `sp_babelfish_configure`.
- Per elencare i tratteggi escape denominati e i relativi valori, per la sessione corrente o in tutto il cluster, eseguire il comando `sp_babelfish_configure 'hatch_name'` dove *`hatch_name`* è l'identificatore di una o più vie di fuga. *`hatch_name`* può utilizzare caratteri jolly SQL, ad esempio '%'.
- Per impostare una o più vie di fuga sul valore specificato, eseguire `sp_babelfish_configure ['hatch_name' [, 'strict'|'ignore' [, 'server']]]`. Per rendere le impostazioni permanenti a livello di cluster, includi la parola chiave `server`, come nell'esempio seguente:

  ```
  EXECUTE sp_babelfish_configure 'escape_hatch_unique_constraint', 'ignore', 'server'
  ```

  Per impostarle solo per la sessione corrente, non utilizzare `server`.
- Per ripristinare tutti gli escape hatch ai valori predefiniti, esegui `sp_babelfish_configure 'default'` (Babelfish 1.2.0 e versioni successive).

La stringa che identifica il tratteggio (o i tratteggi) può includere caratteri jolly SQL. Ad esempio, il seguente imposta tutte le trattezze di escape sintassi su `ignore` per il cluster Aurora PostgreSQL.

```
EXECUTE sp_babelfish_configure '%', 'ignore', 'server'
```

Nella tabella seguente è possibile trovare descrizioni e valori predefiniti per gli escape hatch predefiniti di Babelfish.

Via di fuga	Descrizione	Default
`escape_hatch_checkpoint`	Consente l'uso dell'istruzione CHECKPOINT nel codice procedurale, ma l'istruzione CHECKPOINT non è attualmente implementata.	ignora
`escape_hatch_constraint_name_for_default`	Controlla il comportamento di Babelfish correlato ai nomi dei vincoli predefiniti.	ignora

Via di fuga	Descrizione	Default
`escape_hatch_database_misc_options`	Controlla il comportamento di Babelfish correlato alle seguenti opzioni su CREATE DATABASE: CONTAINMENT, DB_CHAINING, TRUSTWORTHY, PERSISTENT_LOG_BUFFER.	ignora
`escape_hatch_for_replication`	Controlla il comportamento di Babelfish correlato alla clausola [NOT] FOR REPLICATION durante la creazione o la modifica di una tabella.	strict
`escape_hatch_fulltext`	Controlla il comportamento di Babelfish correlato alle funzionalità FULLTEXT, ad esempio DEFAULT_FULLTEXT_LANGUAGE in CREATE/ALTER DATABASE, CREATE FULLTEXT INDEX o sp_fulltext_database.	ignora
`escape_hatch_ignore_dup_key`	Controlla il comportamento di Babelfish relativo a CREATE/ALTER TABLE e CREATE INDEX. Quando IGNORE_DUP_KEY=ON, genera un errore quando impostato su `strict` (impostazione predefinita) o ignora l'errore quando impostato su `ignore` (Babelfish 1.2.0 e versioni successive).	strict

Via di fuga	Descrizione	Default
`escape_hatch_index_clustering`	Controlla il comportamento di Babelfish correlato alle parole chiave CLUSTERED o NONCLUSTERED per indici e vincoli PRIMARY KEY o UNIQUE. Quando CLUSTERED viene ignorato, l'indice o il vincolo viene ancora creato come se fosse stato specificato NON CLUSTERED.	ignora
`escape_hatch_index_columnstore`	Controlla il comportamento di Babelfish relativo alla clausola COLUMNSTORE. Se si specifica `ignore`, Babelfish crea un normale indice B-tree.	strict
`escape_hatch_join_hints`	Controlla il comportamento delle parole chiave in un operatore JOIN: LOOP, HASH, MERGE, REMOTE, REDUCE, REDISTRIB UITE, REPLICATE.	ignora

Via di fuga	Descrizione	Default
`escape_hatch_language_non_english`	Controlla il comportamento di Babelfish relativo a lingue diverse dall'inglese per i messaggi sullo schermo. Al momento, Babelfish supporta solo `us_english` per messaggi sullo schermo. SET LANGUAGE potrebbe utilizzare una variabile contenente il nome della lingua, quindi la lingua effettiva impostata può essere rilevata solo in fase di esecuzione.	strict
`escape_hatch_login_hashed_password`	Se ignorato, elimina l'errore per la parola chiave `HASHED` per `CREATE LOGIN` e `ALTER LOGIN`.	strict
`escape_hatch_login_misc_options`	Se ignorato, elimina l'errore per altre parole chiave oltre `HASHED`, `MUST_CHANGE` , `OLD_PASSWORD` , e `UNLOCK` per `CREATE LOGIN` and `ALTER LOGIN`.	strict
`escape_hatch_login_old_password`	Se ignorato, elimina l'errore per la parola chiave `OLD_PASSWORD` per `CREATE LOGIN` e `ALTER LOGIN`.	strict
`escape_hatch_login_password_must_change`	Se ignorato, elimina l'errore per la parola chiave `MUST_CHANGE` per `CREATE LOGIN` e `ALTER LOGIN`.	strict

Via di fuga	Descrizione	Default
`escape_hatch_login_password_unlock`	Se ignorato, elimina l'errore per la parola chiave `UNLOCK` per `CREATE LOGIN` e `ALTER LOGIN`.	strict
`escape_hatch_nocheck_add_constraint`	Controlla il comportamento di Babelfish relativo alla clausola WITH CHECK o NOCHECK per i vincoli.	strict
`escape_hatch_nocheck_existing_constraint`	Controlla il comportamento di Babelfish correlato ai vincoli FOREIGN KEY o CHECK.	strict
`escape_hatch_query_hints`	Controlla il comportamento di Babelfish correlato ai suggerimenti sulle query. Quando questa opzione è impostata per ignorare, il server ignora i suggerimenti che utilizzano la clausola OPTION (...) per specificare gli aspetti di elaborazione delle query. Esempi includono SELECT FROM... OPZIONE (MERGE JOIN HASH, MAXRECURSION 10)).	ignora
`escape_hatch_rowversion`	Controlla il comportamento dei tipi di dati ROWVERSION e TIMESTAMP. Per informazioni sull'utilizzo, consulta Utilizzo di funzionalità Babelfish con implementazione limitata.	strict

Via di fuga	Descrizione	Default
`escape_hatch_schemabinding_function`	Controlla il comportamento di Babelfish relativo alla clausola WITH SCHEMABINDING. Per impostazione predefinita, la clausola WITH SCHEMABINDING viene ignorata quando specificata con il comando CREATE o ALTER FUNCTION.	ignora
`escape_hatch_schemabinding_procedure`	Controlla il comportamento di Babelfish relativo alla clausola WITH SCHEMABINDING. Per impostazione predefinita, la clausola WITH SCHEMABINDING viene ignorata quando specificata con il comando CREATE o ALTER PROCEDURE.	ignora
`escape_hatch_rowguidcol_column`	Controlla il comportamento di Babelfish correlato alla clausola ROWGUIDCOL durante la creazione o la modifica di una tabella.	strict
`escape_hatch_schemabinding_trigger`	Controlla il comportamento di Babelfish relativo alla clausola WITH SCHEMABINDING. Per impostazione predefinita, la clausola WITH SCHEMABINDING viene ignorata quando specificata con il comando CREATE o ALTER TRIGGER.	ignora

Via di fuga	Descrizione	Default
`escape_hatch_schemabinding_view`	Controlla il comportamento di Babelfish relativo alla clausola WITH SCHEMABINDING. Per impostazione predefinita, la clausola WITH SCHEMABINDING viene ignorata quando specificata con il comando CREATE o ALTER VIEW.	ignora
`escape_hatch_session_settings`	Controlla il comportamento di Babelfish verso affermazioni SET a livello di sessione non supportate.	ignora
`escape_hatch_showplan_all`	Controlla il comportamento di Babelfish relativo a SET SHOWPLAN_ALL e SET STATISTICS PROFILE. Se impostati su "ignore", si comportano come SET BABELFISH_SHOWPLAN_ALL e SET BABELFISH_STATISTICS PROFILE, se impostati su "strict", vengono ignorati silenziosamente.	strict
`escape_hatch_storage_on_partition`	Controlla il comportamento di Babelfish correlato alla clausola `ON partition_scheme column` durante la definizione del partizionamento. Babelfish attualmente non implementa il partizionamento.	strict

Via di fuga	Descrizione	Default
`escape_hatch_storage_options`	Via di fuga su qualsiasi opzione di archiviazione utilizzata in CREATE, ALTER DATABASE, TABLE, INDEX. Sono incluse le clausole (LOG) ON, TEXTIMAGE_ON, FILESTREAM_ON che definiscono le posizioni di archiviazione (partizioni, gruppi di file) per tabelle, indici e vincoli e anche per un database. Questa impostazione di escape hatch si applica a tutte queste clausole (inclusi ON [PRIMARY] e ON “DEFAULT”). L'eccezione è quando viene specificata una partizione per una tabella o un indice con ON partition_scheme (colonna).	ignora
`escape_hatch_table_hints`	Controlla il comportamento dei suggerimenti di tabella specificati utilizzando la clausola WITH (...).	ignora

Via di fuga	Descrizione	Default
`escape_hatch_unique_constraint`	Se impostato su strict, un'oscura differenza semantica tra SQL Server e PostgreSQL nella gestione dei valori NULL nelle colonne indicizzate può generare errori. La differenza semantica emerge solo in casi d'uso non realistici, quindi puoi impostare questo hatch di escape su "ignore" per evitare di vedere l'errore. Deprecato dalle seguenti versioni: 3.6.0 e versioni successive, 4.2.0 e versioni successive	strict

Creazione di un cluster database Babelfish per Aurora PostgreSQL

Babelfish per Aurora PostgreSQL è supportato su Aurora PostgreSQL versione 13.4 e successive.

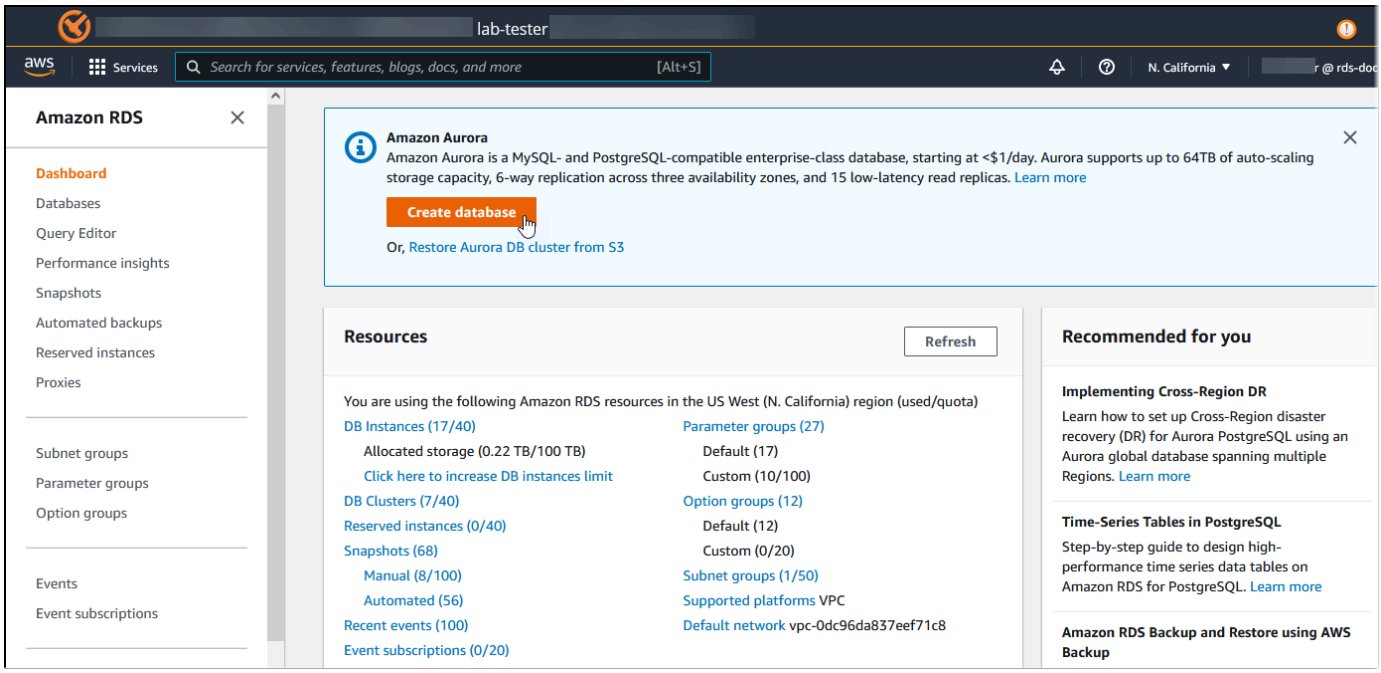
Puoi utilizzare Console di gestione AWS oppure AWS CLI per creare un cluster database Aurora PostgreSQL con Babelfish.

> **Note**
>
> In un cluster Aurora PostgreSQL, il nome del database `babelfish_db` è riservato a Babelfish. Creare il proprio database “babelfish_db” su Babelfish per Aurora PostgreSQL impedisce ad Aurora di eseguire correttamente il provisioning di Babelfish.

Console

Per creare un cluster con Babelfish in esecuzione con Console di gestione AWS

1. Apri la console Amazon RDS all'indirizzo https://console.aws.amazon.com/rds/ e scegli Create database (Creazione di database).

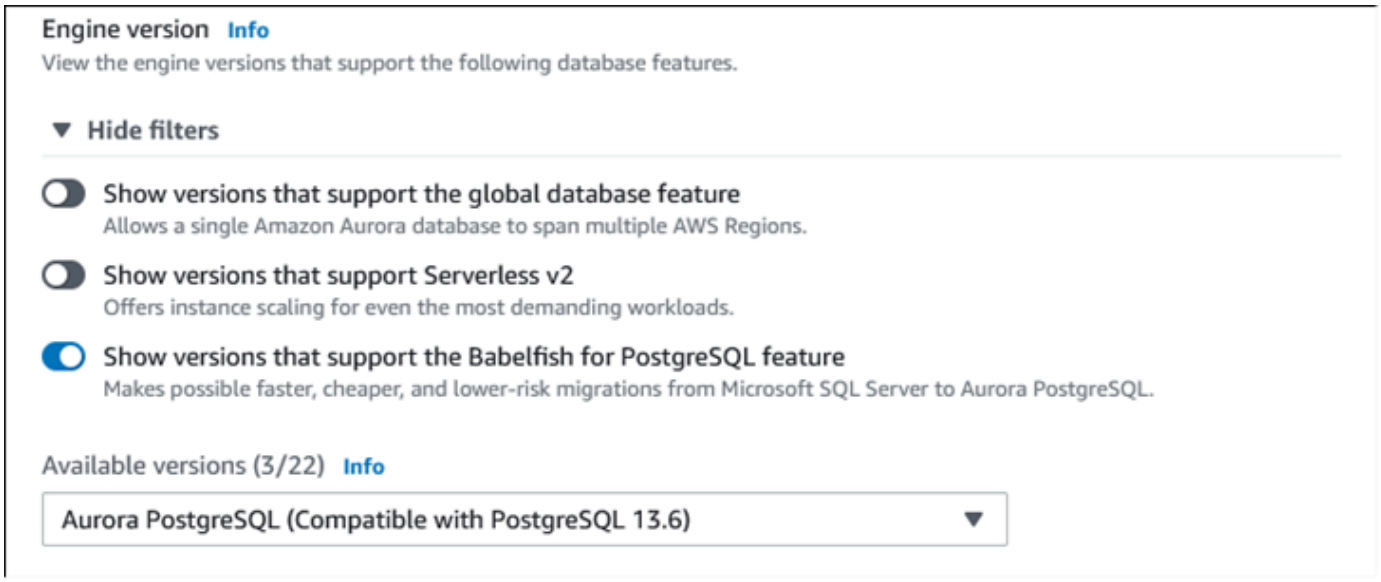


2. Per Scegli un metodo di creazione del database, procedi in uno dei seguenti modi:

   - Per specificare opzioni dettagliate del motore, scegli Standard create (Creazione standard).

- Per utilizzare opzioni preconfigurate che supportano le best practice per un cluster Aurora, scegli Creazione semplice.

3. Per Tipo di motore scegli Aurora (compatibile con PostgreSQL).
4. Scegli Mostra filtrie quindi scegli Mostra versioni che supportano la funzionalità Babelfish per PostgreSQL per elencare i tipi di motore che supportano Babelfish. Babelfish è attualmente supportato su Aurora PostgreSQL 13.4 e versioni successive.
5. Per Versioni disponibili, scegli una versione Aurora PostgreSQL. Per ottenere le funzionalità di Babelfish più recenti, scegli la versione principale di Aurora PostgreSQL più alta.

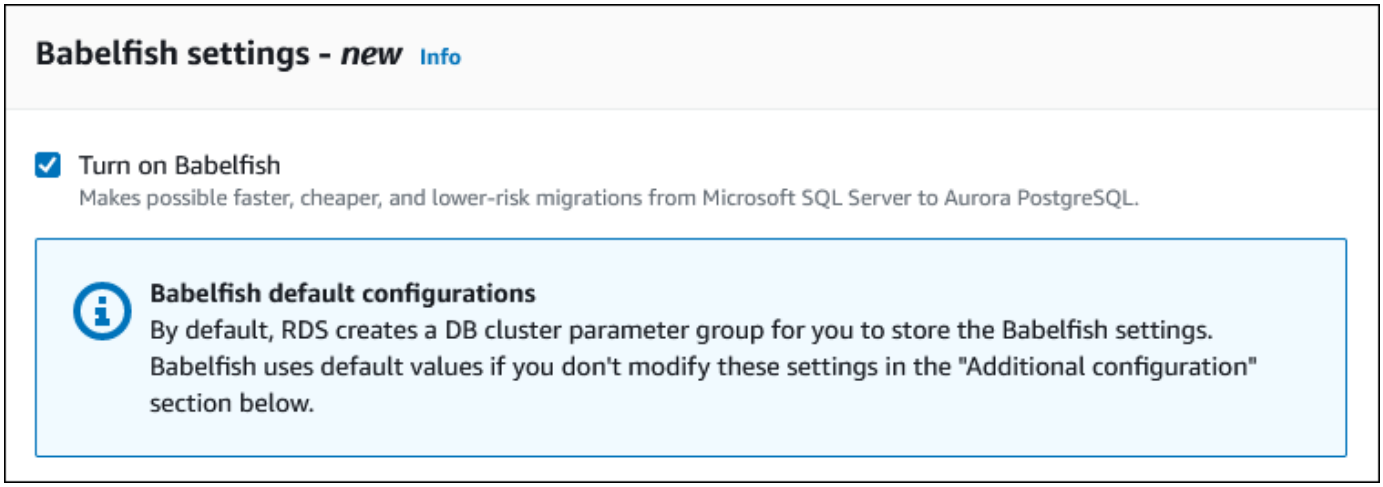


6. In Templates (Modelli), seleziona il modello che corrisponde al proprio caso d'uso.
7. Per Identificatore cluster di database, inserisci un nome che puoi facilmente trovare in seguito nell'elenco dei cluster di database.
8. Per Master username (Nome utente master), immetti un nome utente amministratore. Il valore predefinito per Aurora PostgreSQL è `postgres`. Puoi accettare il valore predefinito o scegliere un nome diverso. Ad esempio, per seguire la convenzione di denominazione utilizzata nei database SQL Server, puoi immettere `sa` (amministratore di sistema) per Master username (Nome utente Master).

   Se non crei un utente chiamato `sa` in questo momento, puoi crearne uno in un secondo momento con la tua scelta del cliente. Dopo aver creato l'utente, utilizza il comando `ALTER SERVER ROLE` per aggiungerlo al gruppo (ruolo) `sysadmin` per il cluster.

> **Warning**
>
> Il nome utente master deve sempre utilizzare caratteri minuscoli, altrimenti il cluster di database non può connettersi a Babelfish tramite la porta TDS.

9. Per Master password (Password master), crea una password complessa e confermala.
10. Per le opzioni che seguono, fino alla sezione Impostazioni di Babelfish,specifica le impostazioni del cluster di database. Per informazioni su ciascuna impostazione, consulta Impostazioni per cluster di database Aurora.
11. Per rendere disponibile la funzionalità Babelfish, seleziona la casella Turn on Babelfish (Attiva Babelfish).



12. Per Gruppo di parametri del cluster database, procedere in uno dei seguenti modi:

    - Scegliere Crea nuovo per creare un nuovo gruppo di parametri con Babelfish attivato.
    - Scegliere Scegli esistente per utilizzare un gruppo di parametri esistente. Se si utilizza un gruppo esistente, assicurarsi di modificare il gruppo prima di creare il cluster e aggiungere valori per i parametri Babelfish. Per informazioni sull'estensione dei parametri consulta Impostazioni del gruppo di parametri del cluster database per Babelfish.

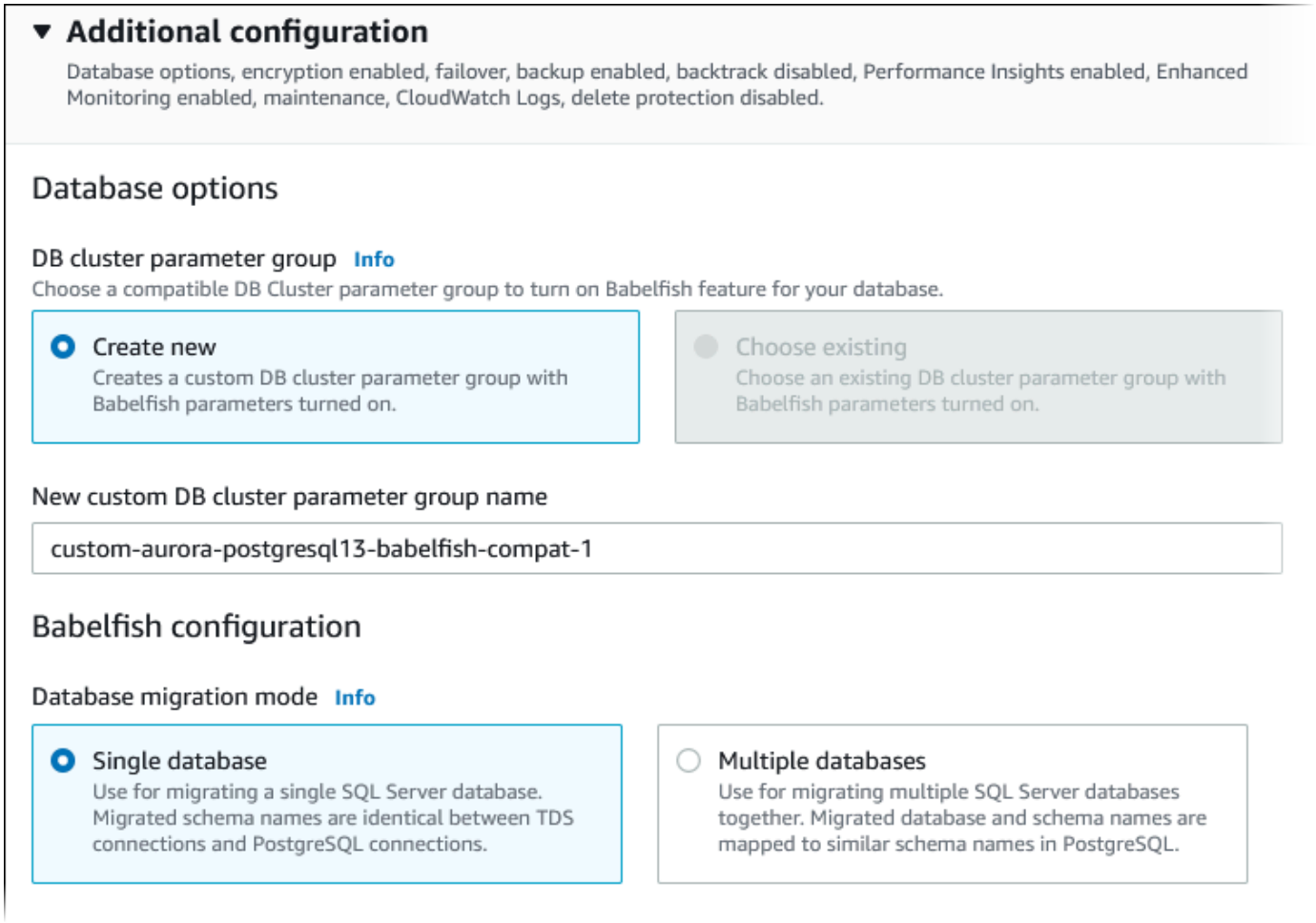
      Se si utilizza un gruppo esistente, specificare il nome del gruppo nella casella che segue.
13. Per Database migration mode (Modalità di migrazione del database), scegliere una delle seguenti opzioni:

    - Database singolo per eseguire la migrazione di un singolo database di SQL Server.

In alcuni casi, potresti migrare più database utente insieme, con il tuo obiettivo finale una migrazione completa a Aurora PostgreSQL nativa senza Babelfish. Se le applicazioni finali richiedono schemi consolidati (un singolo schema dbo), assicurarsi di consolidare innanzitutto i database di SQL Server in un singolo database SQL Server. Quindi effettuare la migrazione a Babelfish usando la modalità Database singolo.

- Database multipli per eseguire la migrazione di più database di SQL Server (provenienti da una singola installazione di SQL Server). La modalità di database multipli non consolida più database che non provengono da una singola installazione di SQL Server. Per ulteriori informazioni sulla migrazione di più database, consulta Utilizzo di Babelfish con un singolo database o più database.

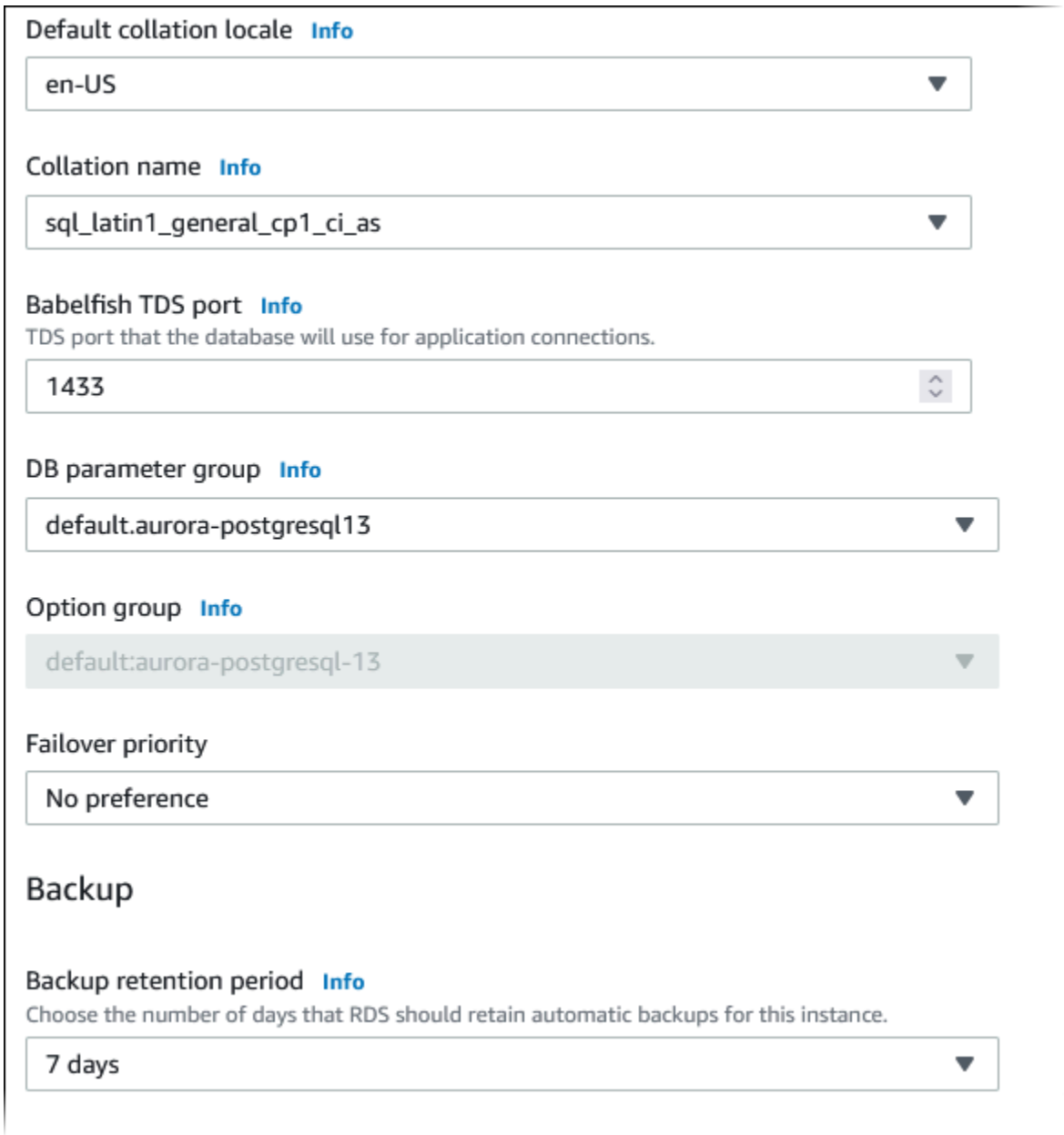
> **Note**
>
> A partire da Aurora PostgreSQL 16, per impostazione predefinita viene scelta la modalità Database multipli come modalità di migrazione del database.



14. Per Localizzazione raccolta predefinita, immetti le impostazioni internazionali del server. Il valore predefinito è `en-US`. Per informazioni dettagliate sulle raccolte consulta Informazioni sulle regole di confronto in Babelfish per Aurora PostgreSQL..
15. Per il campo Collation name (Nome raccolta), inserisci le regole di confronto predefinite. Il valore predefinito è `sql_latin1_general_cp1_ci_as`. Per informazioni dettagliate, consulta Informazioni sulle regole di confronto in Babelfish per Aurora PostgreSQL..
16. Per Porta TDS Babelfish inserisci la porta predefinita `1433`. Attualmente, Babelfish supporta solo la porta `1433` per il cluster di database.
17. Per DB parameter group (Gruppo di parametri database), scegli un gruppo di parametri o fai in modo che Aurora crei un nuovo gruppo per te con impostazioni predefinite.
18. Per Failover priority (Priorità failover), seleziona una priorità di failover per l'istanza. Se non specifichi alcun valore, l'impostazione predefinita è `tier-1`. Questa priorità determina l'ordine

di promozione delle repliche durante il ripristino da un errore dell'istanza primaria. Per ulteriori informazioni, consulta Tolleranza ai guasti di un cluster DB Aurora.

19. Per Backup retention period (Tempo di conservazione del backup), scegli l'arco di tempo (da 1 a 35 giorni), nel quale Aurora conserverà le copie di backup del database. Puoi utilizzare le copie di backup per operazioni di ripristino point-in-time (PITR) del database al secondo. Il periodo di conservazione predefinito è 60 giorni.



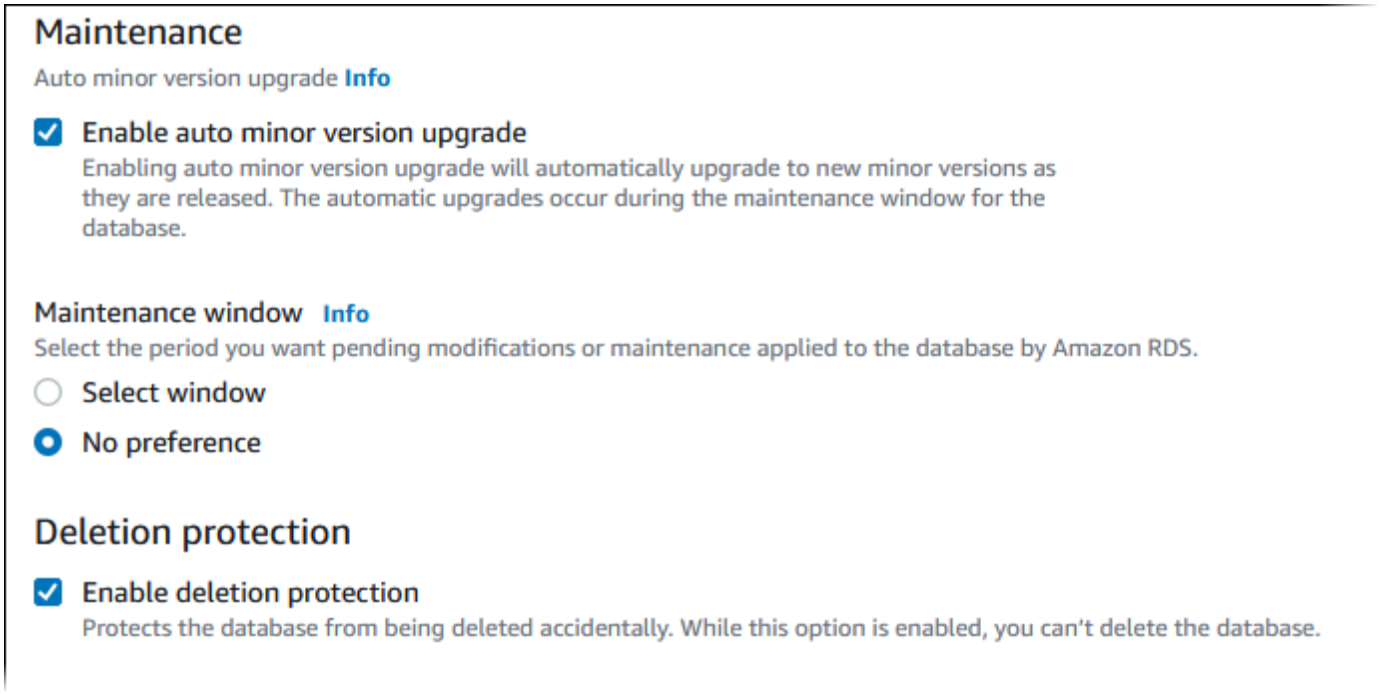
20. Selezionare Copy tags to snapshots (Copy tag a snapshot) per copiare i tag dell'istanza database in uno snapshot DB quando si crea uno snapshot.

> **Note**
>
> Quando si ripristina un cluster di database da uno snapshot, non viene ripristinato come cluster di database Babelfish per Aurora PostgreSQL. È necessario attivare i parametri che controllano le preferenze di Babelfish nel gruppo di parametri del cluster di database per abilitare nuovamente Babelfish. Per ulteriori informazioni sui parametri di Babelfish, consulta Impostazioni del gruppo di parametri del cluster database per Babelfish.

21. Scegliere Enable encryption (Abilita crittografia) per attivare la crittografia a riposo (crittografia dello storage Aurora) per questo cluster database.
22. Scegliere Enable Performance Insights (Abilita Performance Insights)per attivare Amazon RDS Performance Insights.
23. Scegliere Enable Enhanced monitoring (Abilita monitoraggio avanzato) per avviare la raccolta di parametri in tempo reale per il sistema operativo su cui viene eseguito il cluster di database.
24. Scegliere Postgre SQL (Log di PostgreSQL) per pubblicare i file di log su Amazon CloudWatch Logs.
25. Scegliere Enable auto minor version upgrade (Abilita aggiornamento automatico versione secondaria) per aggiornare automaticamente il cluster Aurora DB quando è disponibile un aggiornamento di versione secondaria.
26. Per Maintenance window (Finestra di manutenzione), eseguire le seguenti operazioni:

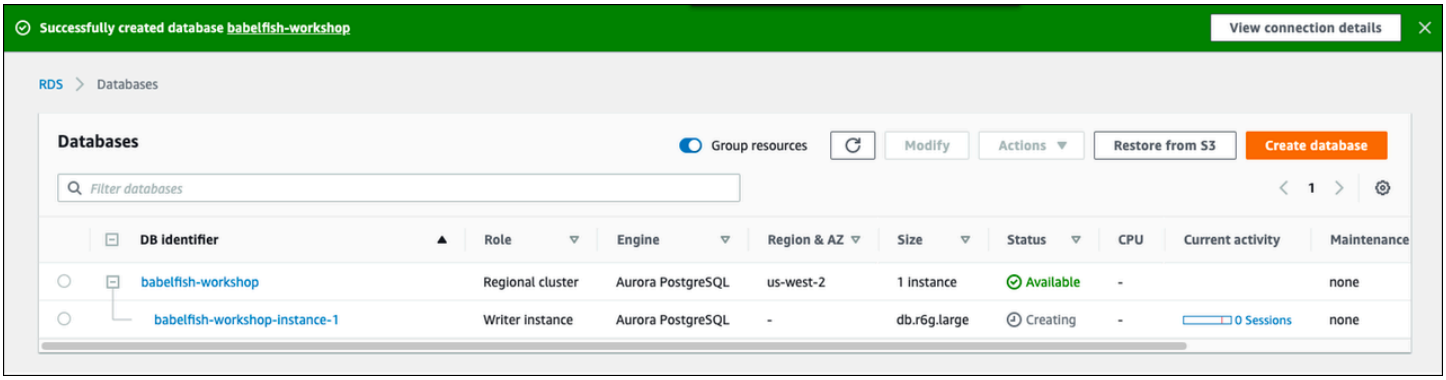
    - Per scegliere un orario per fare apportare modifiche o far eseguire la manutenzione da Amazon RDS, scegliere Select window (Seleziona finestra).
    - Per eseguire la manutenzione di Amazon RDS in un momento non programmato, scegliere No preference (Nessuna preferenza).
27. Selezionare la casella Enable deletion protection (Abilita protezione da eliminazione) per proteggere il database dall'eliminazione accidentale.

    Se si attiva questa funzione, non è possibile eliminare direttamente il database. Al contrario, è necessario modificare il cluster di database e disattivare questa funzione prima di eliminare il database.



28. Scegliere Crea database.

Puoi trovare il tuo nuovo database configurato per Babelfish nell'elenco Database. La colonna Status (Stato) visualizza Available (Disponibile) quando la distribuzione è completata.



AWS CLI

Quando si crea un Babelfish per Aurora PostgreSQL; utilizzando AWS CLI, è necessario passare al comando il nome del gruppo di parametri del cluster database da utilizzare per il cluster. Per ulteriori informazioni, consulta Prerequisiti per i cluster di database.

Prima di poter utilizzare AWS CLI per creare un cluster Aurora PostgreSQL con Babelfish, procedi nel seguente modo:

- Seleziona l'URL dell'endpoint dall'elenco dei servizi all'indirizzo Endpoint e quote di Amazon Aurora.
- Creare gruppo di parametri del cluster di database Per ulteriori informazioni sui gruppi di parametri, consultare Gruppi di parametri per Amazon Aurora.
- Modificare il gruppo di parametri, aggiungendo il parametro che attiva Babelfish.

Come creare un cluster database Aurora PostgreSQL tramite AWS CLI

Gli esempi che seguono utilizzano il nome utente master predefinito, `postgres`. Sostituisci in base alle esigenze con il nome utente creato per il cluster database, ad esempio `sa` o qualsiasi nome utente scelto se non è stato accettato il valore predefinito.

1. Per creare un gruppo di parametri.

   Per Linux, macOS o Unix:

   ```
   aws rds create-db-cluster-parameter-group \
   --endpoint-url endpoint-url \
   --db-cluster-parameter-group-name parameter-group \
   --db-parameter-group-family aurora-postgresql14 \
   --description "description"
   ```

   Per Windows:

   ```
   aws rds create-db-cluster-parameter-group ^
   --endpoint-url endpoint-URL ^
   --db-cluster-parameter-group-name parameter-group ^
   --db-parameter-group-family aurora-postgresql14 ^
   --description "description"
   ```

2. Modifica il gruppo di parametri per attivare Babelfish.

   Per Linux, macOS o Unix:

   ```
   aws rds modify-db-cluster-parameter-group \
   --endpoint-url endpoint-url \
   --db-cluster-parameter-group-name parameter-group \
   --parameters
    "ParameterName=rds.babelfish_status,ParameterValue=on,ApplyMethod=pending-reboot"
   ```

Per Windows:

```
aws rds modify-db-cluster-parameter-group ^
--endpoint-url endpoint-url ^
--db-cluster-parameter-group-name paramater-group ^
--parameters
 "ParameterName=rds.babelfish_status,ParameterValue=on,ApplyMethod=pending-reboot"
```

3. Identificare il gruppo di sottoreti del database e l'ID del gruppo di sicurezza VPC per il nuovo cluster di database, quindi richiamare il comando create-db-cluster.

   Per Linux, macOS o Unix:

```
aws rds create-db-cluster \
--db-cluster-identifier  cluster-name\
--master-username postgres \
--manage-master-user-password \
--engine aurora-postgresql \
--engine-version 14.3            \
--vpc-security-group-ids security-group \
--db-subnet-group-name subnet-group-name \
--db-cluster-parameter-group-name parameter-group
```

   Per Windows:

```
aws rds create-db-cluster ^
--db-cluster-identifier cluster-name ^
--master-username postgres ^
--manage-master-user-password ^
--engine aurora-postgresql ^
--engine-version 14.3 ^
--vpc-security-group-ids security-group ^
--db-subnet-group-name subnet-group ^
--db-cluster-parameter-group-name parameter-group
```

   In questo esempio è specificata l'opzione `--manage-master-user-password` per generare la password dell'utente master e gestirla in Secrets Manager. Per ulteriori informazioni, consulta Gestione delle password con Amazon Aurora e Gestione dei segreti AWS. In alternativa, puoi utilizzare l'opzione `--master-password` per specificare e gestire personalmente la password.

4. Crea in maniera esplicita l'istanza primaria del cluster database. Utilizza il nome del cluster creato nella fase 3 per l'argomento `--db-cluster-identifier` quando chiami il comando create-db-instance, come illustrato di seguito.

   Per Linux, macOS o Unix:

   ```
   aws rds create-db-instance \
   --db-instance-identifier instance-name \
   --db-instance-class db.r6g \
   --db-subnet-group-name subnet-group \
   --db-cluster-identifier cluster-name \
   --engine aurora-postgresql
   ```

   Per Windows:

   ```
   aws rds create-db-instance ^
   --db-instance-identifier instance-name ^
   --db-instance-class db.r6g ^
   --db-subnet-group-name subnet-group ^
   --db-cluster-identifier cluster-name ^
   --engine aurora-postgresql
   ```

Migrazione di un database SQL Server a Babelfish per Aurora PostgreSQL

Puoi utilizzare Babelfish per Aurora PostgreSQL per eseguire la migrazione di un database SQL Server a un cluster database Amazon Aurora PostgreSQL. Prima di eseguire la migrazione, consulta Utilizzo di Babelfish con un singolo database o più database.

Argomenti

- Panoramica del processo di migrazione
- Valutazione e gestione delle differenze tra SQL Server e Babelfish
- Strumenti di importazione/esportazione per la migrazione da SQL Server a Babelfish

Panoramica del processo di migrazione

Il seguente riepilogo elenca i passaggi necessari migrare l'applicazione SQL Server e farla funzionare con Babelfish. Per informazioni sugli strumenti che puoi utilizzare per i processi di esportazione e importazione e per ulteriori dettagli, consulta Strumenti di importazione/esportazione per la migrazione da SQL Server a Babelfish. Per caricare i dati, è consigliabile utilizzare AWS DMS con un cluster di database Aurora PostgreSQL come endpoint di destinazione.

1. Crea un nuovo cluster database Aurora PostgreSQL con Babelfish attivato. Per scoprire come, consulta Creazione di un cluster database Babelfish per Aurora PostgreSQL.

   Per importare i vari artefatti SQL esportati dal database SQL Server, esegui la connessione al cluster Babelfish utilizzando uno strumento SQL Server, ad esempio sqlcmd. Per ulteriori informazioni, consulta Utilizzo di un client SQL Server per la connessione al cluster database.
2. Nel database SQL Server che desideri migrare, esporta Data Definition Language (DDL). Il DDL è un codice SQL che descrive gli oggetti di database che contengono dati utente (ad esempio tabelle, indici e viste) e il codice di database scritto dall'utente (come procedure memorizzate, funzioni definite dall'utente e trigger).

   Per ulteriori informazioni, consulta Utilizzo di SQL Server Management Studio (SSMS) per eseguire la migrazione a Babelfish.
3. Esegui uno strumento di valutazione per valutare l'ambito di eventuali modifiche che potrebbe essere necessario apportare per consentire a Babelfish di supportare efficacemente l'applicazione in esecuzione su SQL Server. Per ulteriori informazioni, consulta Valutazione e gestione delle differenze tra SQL Server e Babelfish.

4. Esamina le limitazioni dell'endpoint di destinazione AWS DMS e aggiorna lo script DDL se necessario. Per ulteriori informazioni, consulta Limitazioni all'utilizzo di un endpoint di destinazione PostgreSQL con le tabelle Babelfish in Utilizzo di Babelfish per Aurora PostgreSQL come destinazione.
5. Sul nuovo cluster database Babelfish, esegui il DDL nel database T-SQL specificato per creare solo gli schemi, i tipi di dati definiti dall'utente e le tabelle con i vincoli della chiave primaria.
6. Utilizza AWS DMS per migrare i tuoi dati da SQL Server alle tabelle Babelfish. Per la replica continua con SQL Server Change Data Capture o Replica di SQL, usa Aurora PostgreSQL anziché Babelfish come endpoint. Per farlo, vedi Utilizzo di Babelfish per Aurora PostgreSQL come obiettivo per AWS Database Migration Service.
7. Al termine del caricamento dei dati, crea tutti gli oggetti T-SQL rimanenti che supportano l'applicazione sul cluster Babelfish.
8. Riconfigura l'applicazione client per connettersi all'endpoint Babelfish anziché al database SQL Server. Per ulteriori informazioni, consulta Connessione a un cluster database Babelfish.
9. Modifica l'applicazione come richiesto e ripeti il test. Per ulteriori informazioni, consulta Differenze tra Babelfish per Aurora PostgreSQL e SQL Server.

Devi comunque valutare le query SQL sul lato client. Gli schemi generati dall'istanza di SQL Server convertono solo il codice SQL sul lato server. Si consiglia di effettuare la procedura seguente:

- Acquisisci le query sul lato client utilizzando SQL Server Profiler con il modello predefinito TSQL_Replay. Questo modello acquisisce le informazioni sull'istruzione T-SQL che possono essere riprodotte per l'ottimizzazione e il test iterativo. Puoi avviare il profiler dal menu Tools (Strumenti) di SQL Server Management Studio. Scegli SQL Server Profiler per aprire il profiler e selezionare il modello TSQL_Replay.

  Da utilizzare per la migrazione di Babelfish, avvia una traccia e quindi esegui l'applicazione utilizzando i test funzionali. Il profiler acquisisce le istruzioni T-SQL. Al termine del test, interrompi la traccia. Salva il risultato in un file XML con le query sul lato client (File > Save as > Trace XML File for Replay) (File > Salva con nome > Traccia file XML per la ripetizione).

  Per ulteriori informazioni, consulta SQL Server Profiler nella documentazione di Microsoft. Per ulteriori informazioni sul modello TSQL_Replay, consulta SQL Server Profiler Templates (Modelli di SQL Server Profiler).
- Per applicazioni con query SQL complesse sul lato client, si consiglia di utilizzare Babelfish Compass per analizzarle per la compatibilità delle query con Babelfish. Se l'analisi indica che le

istruzioni SQL lato client contengono funzionalità SQL non supportate, esamina gli aspetti SQL nell'applicazione client e modifica in base alle esigenze.

- È inoltre possibile acquisire le query SQL come eventi estesi (formato .xel). A tale scopo, utilizza il profiler XEvent di SSMS. Dopo aver generato il file .xel, estrai le istruzioni SQL nei file .xml che Compass può elaborare. Per ulteriori informazioni, consulta Use the SSMS XEvent Profler (Uso del profiler XEvent di SQL Server Management Studio) nella documentazione di Microsoft.

Se sei soddisfatto di tutti i test, dell'analisi e delle eventuali modifiche necessarie all'applicazione migrata, puoi iniziare a utilizzare il database Babelfish in produzione. A questo scopo, interrompi il database originale e reindirizza le applicazioni client live per utilizzare la porta Babelfish TDS.

> **Note**
>
> AWS DMS ora supporta la replica dei dati da Babelfish. Per ulteriori informazioni, consulta AWS DMS now supports Babelfish for Aurora PostgreSQL as a source.

Valutazione e gestione delle differenze tra SQL Server e Babelfish

Per risultati ottimali, si consiglia di valutare il DDL/DML generato e il codice delle query client prima di migrare effettivamente l'applicazione di database SQL Server a Babelfish. A seconda della versione di Babelfish e delle funzionalità specifiche di SQL Server implementate dall'applicazione, potrebbe essere necessario rifattorizzare l'applicazione o utilizzare alternative per funzionalità che non sono ancora completamente supportate in Babelfish.

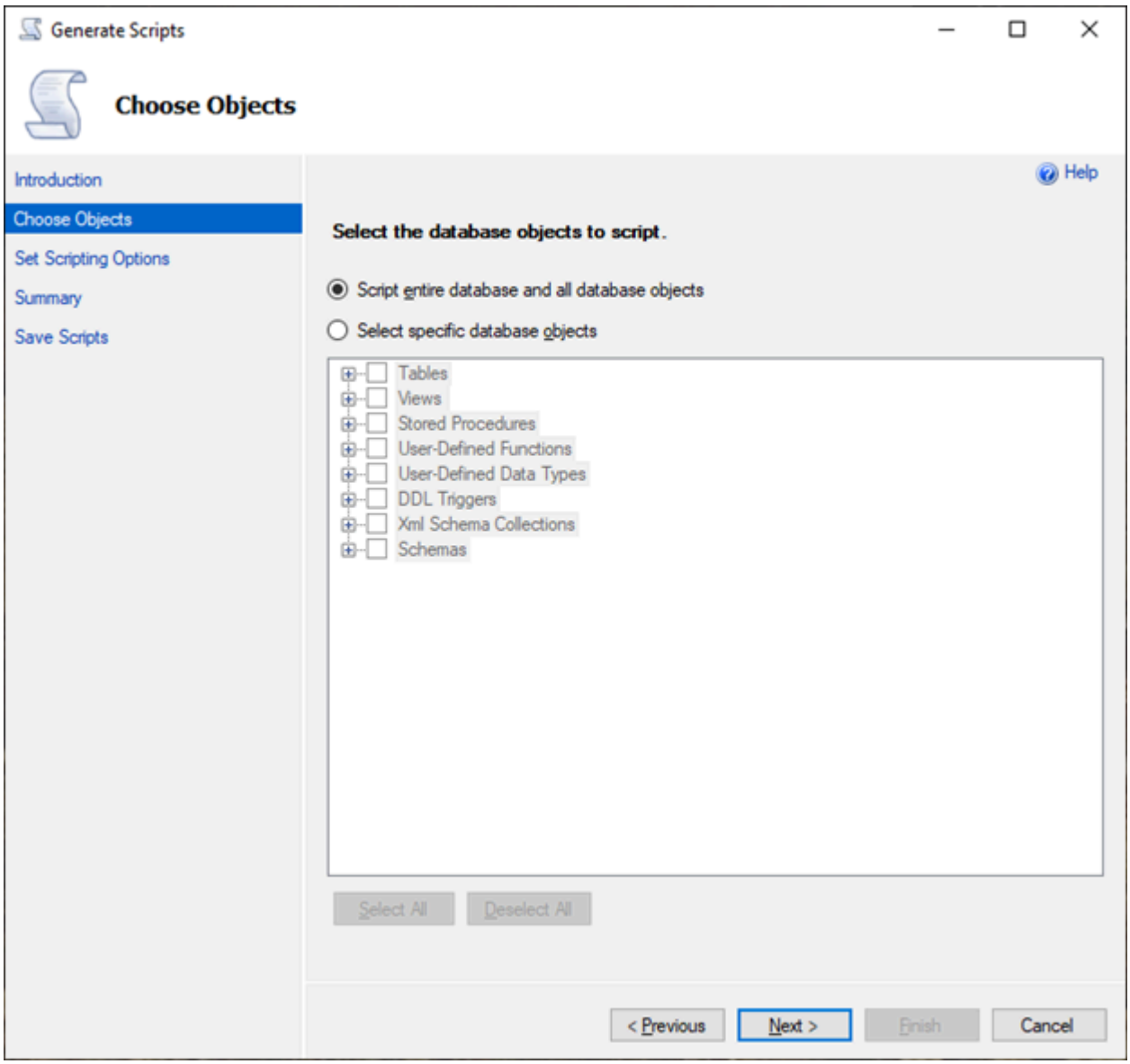
- Per valutare il codice dell'applicazione SQL Server, utilizza Babelfish Compass sul DDL generato per determinare fino a che punto il codice T-SQL è supportato da Babelfish. Identifica il codice T-SQL che potrebbe richiedere modifiche prima di essere eseguito su Babelfish. Per ulteriori informazioni su questo strumento, consulta Babelfish Compass tool su GitHub.

  > **Note**
  >
  > Babelfish Compass è uno strumento open source. Segnala eventuali problemi con Babelfish Compass tramite GitHub piuttosto che con il Supporto AWS.

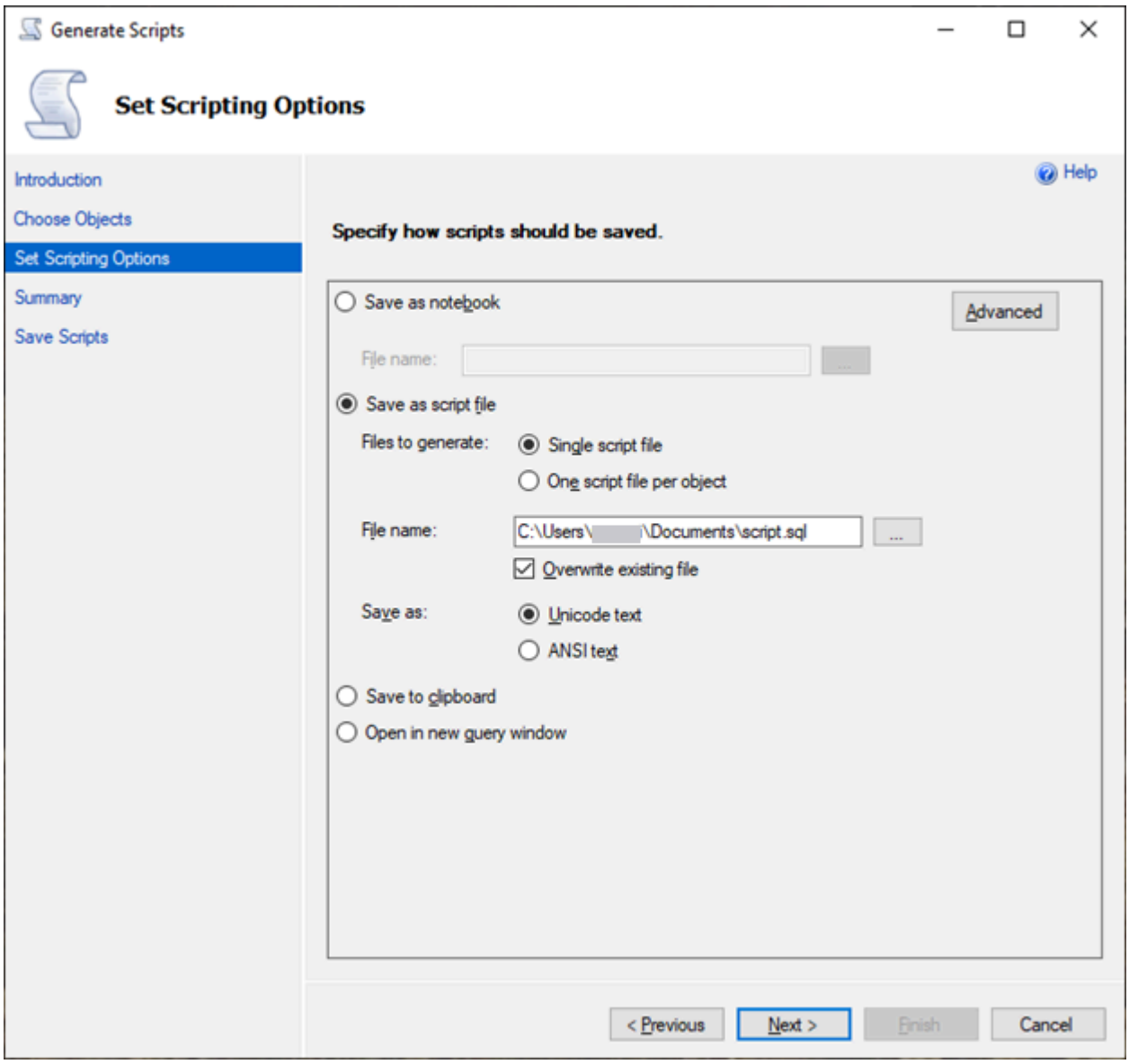
È possibile utilizzare la procedura guidata Generate Scripts (Genera script) con SQL Server Management Studio (SSMS) per generare il file SQL valutato da Babelfish Compass o dall'interfaccia

della linea di comando AWS Schema Conversion Tool. Consigliamo di effettuare i seguenti passaggi per semplificare la valutazione.

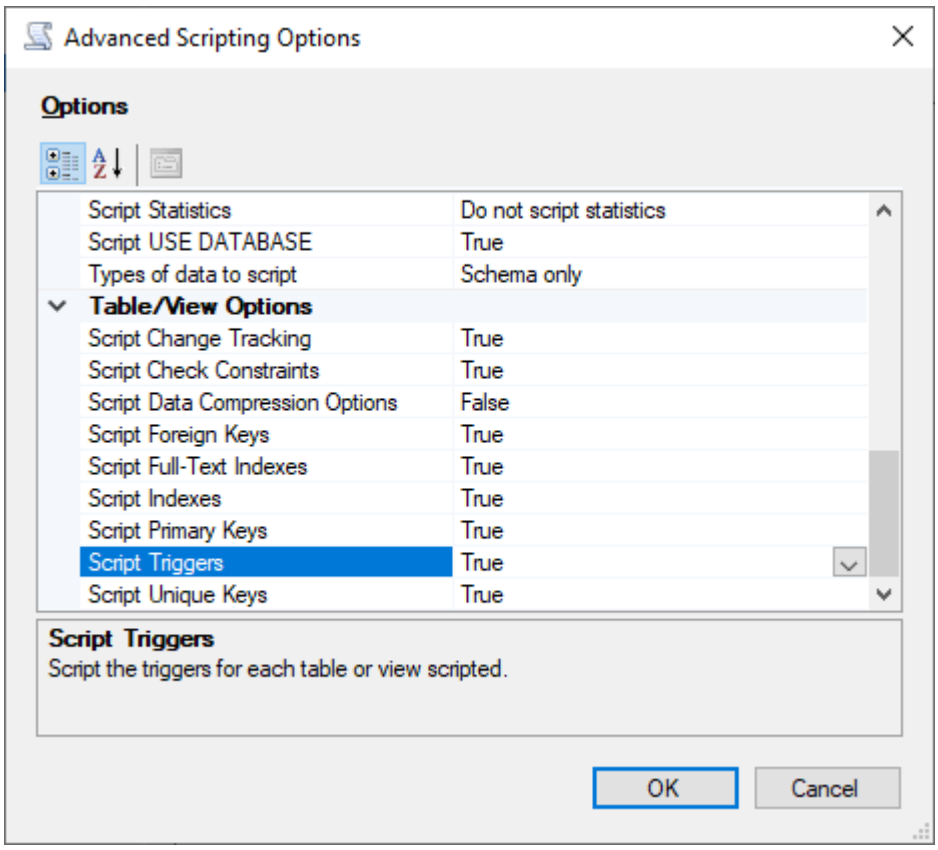
1. Nella pagina Choose Objects (Scegli oggetti), scegli Script entire database and all database objects (Script per l'intero database e tutti gli oggetti del database).



2. Per Set Scripting Options (Imposta le opzioni di script), scegli Save as script file (Salva come file script) come Single script file (File script singolo).



3. Scegli Advanced (Avanzato) per modificare le opzioni di script predefinite e identificare le funzionalità che normalmente sono impostate su false per una valutazione completa:
   - Script Change Tracking (Monitoraggio delle modifiche allo script) su True
   - Script Full-Text Indexes (Indici di testo completo dello script) su True
   - Script Triggers (Trigger script) su True
   - Script Logins (Accessi di script) su True
   - Script Owner (Proprietario script) su True
   - Script Object-Level Permissions (Autorizzazioni a livello di oggetto script) su True
   - Script Collations (Regole di confronto script) su True



4. Segui i restanti passaggi della procedura guidata per generare il file.

Strumenti di importazione/esportazione per la migrazione da SQL Server a Babelfish

Si consiglia di utilizzare AWS DMS come strumento principale per la migrazione da SQL Server a Babelfish. Tuttavia, Babelfish supporta diversi altri modi per la migrazione dei dati con gli strumenti SQL Server, tra cui i seguenti.

- SQL Server Integration Services (SSIS) per tutte le versioni di Babelfish. Per ulteriori informazioni, consulta Migrate from SQL Server to Aurora PostgreSQL using SSIS and Babelfish (Migrazione da SQL Server ad Aurora PostgreSQL utilizzando SSIS e Babelfish).
- Utilizza la procedura guidata di importazione/esportazione di SSMS per Babelfish 2.1.0 e versioni successive. Questo strumento è disponibile tramite SSMS, ma anche come strumento autonomo. Per ulteriori informazioni, consulta Welcome to SQL Server Import and Export Wizard nella documentazione di Microsoft.
- Questa utilità Bulk Data Copy Program (bcp) di Microsoft consente di copiare i dati da un'istanza Microsoft SQL Server in un file di dati nel formato specificato. Per ulteriori informazioni, consulta bpc Utility nella documentazione di Microsoft. Babelfish ora supporta la migrazione dei dati

utilizzando il client BCP e l'utilità bcp ora supporta il flag `-E` (per le colonne di identità) e il flag -b (per gli inserimenti in batch). Alcune opzioni bcp non sono supportate, incluse `-C`, `-T`, `-G`, `-K`, `-R`, `-V` e `-h`.

Utilizzo di SQL Server Management Studio (SSMS) per eseguire la migrazione a Babelfish

Si consiglia di generare file separati per ciascun tipo di oggetto specifico. È possibile utilizzare prima la procedura guidata Generate Scripts (Genera script) di SSMS per ogni set di istruzioni DDL e quindi modificare gli oggetti come gruppo per risolvere eventuali problemi riscontrati durante la valutazione.

Esegui questi passaggi per migrare i dati utilizzando AWS DMS o altri metodi di migrazione dei dati. Esegui prima questi tipi di script di creazione per un approccio migliore e più veloce per caricare i dati nelle tabelle Babelfish in Aurora PostgreSQL.

1. Esegui le istruzioni `CREATE SCHEMA`.
2. Esegui le istruzioni `CREATE TYPE` per creare i tipi di dati definiti dall'utente.
3. Esegui le istruzioni `CREATE TABLE` di base con le chiavi primarie o i vincoli univoci.

Esegui il caricamento dei dati utilizzando lo strumento di importazione/esportazione consigliato. Esegui gli script modificati per i passaggi seguenti in modo da aggiungere gli oggetti del database rimanenti. Sono necessarie le istruzioni create table per eseguire questi script per i vincoli, i trigger e gli indici. Dopo la generazione degli script, elimina le istruzioni create table.

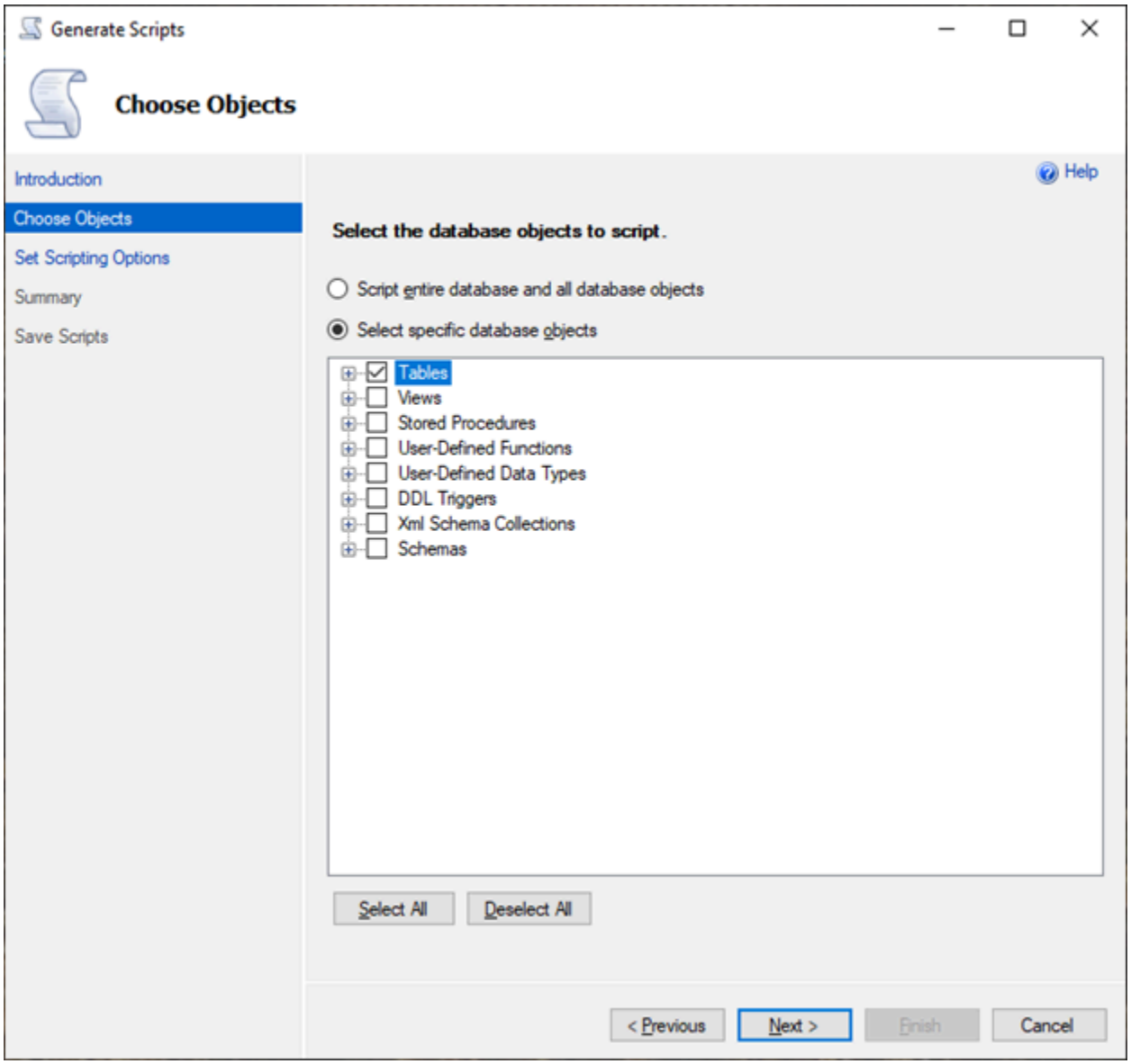
1. Esegui le istruzioni `ALTER TABLE` per i vincoli di controllo, i vincoli di chiave esterna, i vincoli predefiniti.
2. Esegui le istruzioni `CREATE TRIGGER`.
3. Esegui le istruzioni `CREATE INDEX`.
4. Esegui le istruzioni `CREATE VIEW`.
5. Esegui le istruzioni `CREATE STORED PROCEDURE`.

Per generare gli script per ogni tipo di oggetto

Utilizza i seguenti passaggi per creare le istruzioni di base per la creazione di tabelle utilizzando la procedura guidata Generate Scripts (Genera script) di SSMS. Segui la stessa procedura per generare gli script per i diversi tipi di oggetto.

1. Eseguire la connessione all'istanza SQL Server esistente.

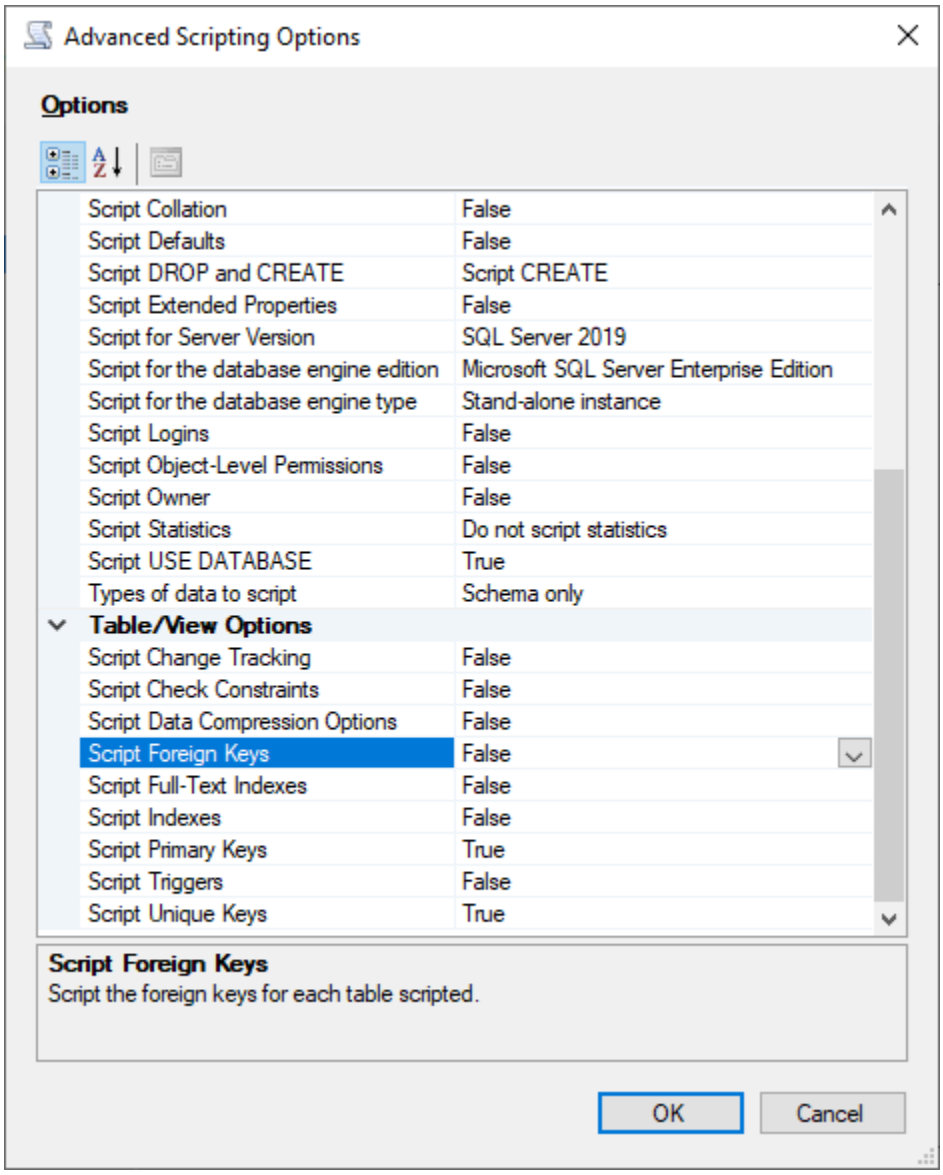
2. Apri il menu contestuale (tasto destro del mouse) per il nome di un database.
3. Scegli Tasks (Attività), quindi seleziona Generate Scripts... (Genera script...).
4. Nel riquadro Choose Objects (Scegli oggetti), seleziona Select specific database objects (Seleziona oggetti di database specifici). Scegli Tables (Tabelle), seleziona tutte le tabelle. Seleziona Successivo per continuare.



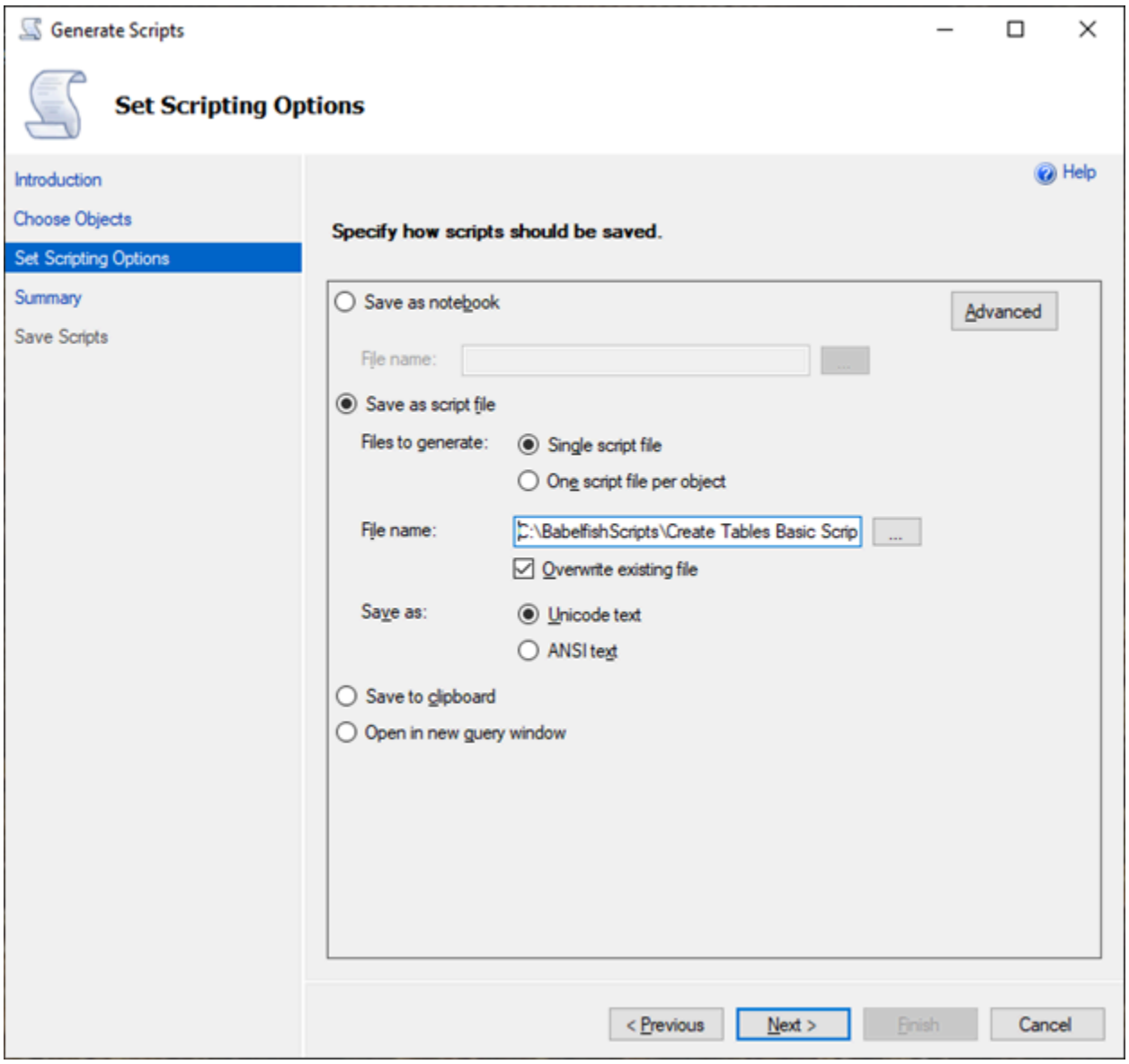
5. Nella pagina Set Scripting Options (Imposta opzioni di scripting), scegli Advanced (Avanzato) per aprire le impostazioni Options (Opzioni). Per generare le istruzioni di base create table, modifica i seguenti valori predefiniti:

   - Script Defaults (Valori predefiniti script) su False.

- Script Extended Properties (Proprietà estese script) su False. Babelfish non supporta le proprietà estese.
- Script Check Constraints (Vincoli di controllo script) su False. Script Foreign Keys (Chiavi esterne script) su False.



6. Scegli OK.
7. Nella pagina Set Scripting Options (Imposta le opzioni di script), scegli Save as script file (Salva come file script), quindi seleziona l'opzione Single script file (File script singolo). Immetti un valore nel campo File name (Nome file).



8. Scegli Next (Successivo) per visualizzare la pagina Summary wizard (Procedura guidata di riepilogo).
9. Scegli Next (Successivo) per avviare la generazione dello script.

   Nella procedura guidata è possibile continuare a generare script per altri tipi di oggetti. Anziché scegliere Finish (Fine) dopo il salvataggio del file, seleziona il pulsante Previous (Precedente) per tre volte per tornare alla pagina Choose Objects (Scegli oggetti). Quindi ripeti i passaggi della procedura guidata per generare gli script per altri tipi di oggetti.

Autenticazione del database con Babelfish per Aurora PostgreSQL

Babelfish per Aurora PostgreSQL supporta due metodi di autenticazione degli utenti del database. L'autenticazione con password è disponibile per impostazione predefinita per tutti i cluster database Babelfish. È possibile aggiungere l'autenticazione Kerberos per lo stesso cluster database.

Argomenti

- Autenticazione password con Babelfish
- Autenticazione Kerberos con Babelfish
- Impostazione dell'autenticazione Kerberos utilizzando i gruppi di sicurezza Active Directory per Babelfish

Autenticazione password con Babelfish

Babelfish per Aurora PostgreSQL supporta l'autenticazione con password. Le password vengono memorizzate in forma crittografata su disco. Per ulteriori informazioni sull'autenticazione su un cluster Aurora PostgreSQL, consulta Sicurezza con Amazon Aurora PostgreSQL.

Potrebbero essere richieste credenziali ogni volta che ti connetti a Babelfish. Qualsiasi utente migrato o creato su Aurora PostgreSQL può utilizzare le stesse credenziali sia sulla porta di SQL Server che sulla porta PostgreSQL. Babelfish non applica le policy sulle password, ma è consigliabile effettuare le seguenti operazioni:

- Richiedi una password complessa contenente almeno otto (8) caratteri.
- Applica un criterio di scadenza della password.

Per esaminare un elenco completo degli utenti del database, utilizza il comando `SELECT * FROM pg_user;`.

Autenticazione Kerberos con Babelfish

La versione Babelfish per Aurora PostgreSQL 15.2 supporta l'autenticazione al cluster di database tramite Kerberos. Questo metodo ti consente di usare l'autenticazione di Microsoft Windows per gli utenti che si connettono al database Babelfish, ma per utilizzarlo devi prima configurare il cluster di database in modo da impiegare AWS Directory Service for Microsoft Active Directory per l'autenticazione Kerberos. Per ulteriori informazioni, consulta Cos'è Directory Service? nella Guida per l'amministrazione di AWS Directory Service.

Configurazione dell'autenticazione Kerberos

Il cluster di database Babelfish per Aurora PostgreSQL può connettersi utilizzando due porte diverse, ma la configurazione dell'autenticazione Kerberos è un processo unico. Pertanto, per prima cosa è necessario configurare l'autenticazione Kerberos per il cluster database. Per ulteriori informazioni, consulta Configurazione dell'autenticazione Kerberos. Dopo aver completato la configurazione, verifica di poterti connettere a un client PostgreSQL usando Kerberos. Per ulteriori informazioni, consulta Connessione con Autenticazione Kerberos.

Accesso e allocazione degli utenti in Babelfish

Gli accessi Windows creati dalla porta TDS (Tabular Data Stream) possono essere utilizzati con la porta TDS o con la porta PostgreSQL. Innanzitutto, l'accesso che può utilizzare Kerberos per l'autenticazione deve essere sottoposto a provisioning dalla porta TDS prima che questa venga usata dagli utenti e dalle applicazioni T-SQL per connettersi a un database Babelfish. Quando si creano gli accessi Windows, gli amministratori possono fornire l'accesso utilizzando il nome di dominio DNS o il nome di dominio NetBIOS. In genere, il dominio NetBIOS è il sottodominio del nome di dominio DNS. Ad esempio, se il nome di dominio DNS è `CORP.EXAMPLE.COM`, il dominio NetBIOS può essere `CORP`. Se per l'accesso viene usato il formato del nome di dominio NetBIOS, è necessario impostare una mappatura al nome di dominio DNS.

Gestione della mappatura del nome di dominio NetBIOS al nome di dominio DNS

Per gestire le mappature tra il nome di dominio NetBIOS e il nome di dominio DNS, Babelfish include le stored procedure di sistema per aggiungere, rimuovere e troncare le mappature. Solo un utente con il ruolo `sysadmin` può eseguire queste procedure.

Per creare una mappatura tra i nomi di dominio NetBIOS e DNS, usare la stored procedure di sistema `babelfish_add_domain_mapping_entry` fornita da Babelfish. Entrambi gli argomenti devono avere un valore valido e non essere NULL.

Example

```
EXEC babelfish_add_domain_mapping_entry 'netbios_domain_name',
 'fully_qualified_domain_name'
```

L'esempio seguente mostra come creare la mappatura tra il nome NetBIOS CORP e il nome di dominio DNS CORP.EXAMPLE.COM.

Example

```
EXEC babelfish_add_domain_mapping_entry 'corp', 'corp.example.com'
```

Per eliminare una voce di mappatura esistente, usaRE la stored procedure di sistema babelfish_remove_domain_mapping_entry:

Example

```
EXEC babelfish_remove_domain_mapping_entry 'netbios_domain_name'
```

L'esempio seguente mostra come rimuovere la mappatura per il nome NetBIOS CORP.

Example

```
EXEC babelfish_remove_domain_mapping_entry 'corp'
```

Per rimuovere tutte le voci di mappatura esistenti, usa la stored procedure di sistema babelfish_truncate_domain_mapping_table:

Example

```
EXEC babelfish_truncate_domain_mapping_table
```

Per visualizzare tutte le mappature tra i nomi di dominio NetBIOS e DNS, utilizzare la seguente query.

Example

```
SELECT netbios_domain_name, fq_domain_name FROM babelfish_domain_mapping;
```

Gestione degli accessi

Creazione degli accessi

Esegui la connessione al database tramite l'endpoint TDS utilizzando un accesso con le autorizzazioni corrette. Se non è stato creato un utente del database, l'accesso viene mappato all'utente guest. Se l'utente guest non è abilitato, il tentativo di accesso ha esito negativo.

Crea un accesso Windows utilizzando la seguente query. L'opzione `FROM WINDOWS` consente l'autenticazione tramite Active Directory.

```
CREATE LOGIN login_name FROM WINDOWS [WITH DEFAULT_DATABASE=database]
```

Example

L'esempio seguente mostra come creare un accesso per l'utente di Active Directory [corp\test1] con il database predefinito db1.

```
CREATE LOGIN [corp\test1] FROM WINDOWS WITH DEFAULT_DATABASE=db1
```

Questo esempio presuppone che esista una mappatura tra il dominio NetBIOS CORP e il nome di dominio DNS CORP.EXAMPLE.COM. Se non è presente una mappatura, è necessario fornire il nome di dominio DNS [CORP.EXAMPLE.COM\test1].

> **Note**
>
> Gli accessi basati sugli utenti di Active Directory sono limitati a nomi contenenti meno di 21 caratteri.

Eliminazione di un accesso

Per eliminare un accesso, usa la stessa sintassi utilizzata per qualsiasi accesso, come mostrato nell'esempio seguente:

```
DROP LOGIN [DNS domain name\login]
```

Modifica di un accesso

Per modificare un accesso, usa la stessa sintassi utilizzata per qualsiasi accesso, come mostrato nell'esempio seguente:

```
ALTER LOGIN [DNS domain name\login] { ENABLE|DISABLE|WITH DEFAULT_DATABASE=[master] }
```

Il comando ALTER LOGIN supporta un numero limitato di opzioni per gli accessi Windows, tra cui le seguenti:

- DISABLE: per disabilitare un accesso. Non puoi usare un accesso disabilitato per l'autenticazione.
- ENABLE: per abilitare un accesso disabilitato.
- DEFAULT_DATABASE: per modificare il database predefinito di un accesso.

> **Note**
>
> La gestione delle password viene completamente eseguita tramite Directory Service, quindi il comando ALTER LOGIN non consente agli amministratori del database di modificare o impostare le password per gli accessi Windows.

Connessione a Babelfish per Aurora PostgreSQL con l'autenticazione Kerberos

In genere, gli utenti del database eseguono l'autenticazione tramite Kerberos usando computer client che sono membri del dominio Active Directory. Utilizzano l'autenticazione di Windows dalle loro applicazioni client per accedere al server Babelfish per Aurora PostgreSQL sulla porta TDS.

Connessione a Babelfish per Aurora PostgreSQL sulla porta PostgreSQL con l'autenticazione Kerberos

Puoi utilizzare gli accessi creati dalla porta TDS con la porta TDS o la porta PostgreSQL. Tuttavia, PostgreSQL utilizza per impostazione predefinita il confronto con distinzione tra maiuscole e minuscole per i nomi utente. Affinché Aurora PostgreSQL interpreti i nomi utente Kerberos senza distinzione tra maiuscole e minuscole, è necessario impostare il parametro `krb_caseins_users` su `true` nel gruppo di parametri del cluster Babelfish personalizzato. Questo parametro è impostato su `false` per impostazione predefinita. Per ulteriori informazioni, consulta Configurazione per nomi utente senza distinzione tra maiuscole e minuscole. Inoltre, è necessario specificare il nome utente di accesso nel formato <accesso@nome dominio DNS> dalle applicazioni client PostgreSQL. Non è possibile usare il formato <nome dominio DNS\accesso>.

Errori frequenti

Puoi configurare una relazione di trust tra foreste tra Microsoft Active Directory on-premise e AWS Managed Microsoft AD. Per maggiori informazioni, consulta Creazione di una relazione di trust. Quindi, dovrai connetterti utilizzando un endpoint specifico del dominio specializzato anziché utilizzare il dominio Amazon `rds.amazonaws.com` nell'endpoint host. Se non utilizzi l'endpoint specifico del dominio corretto, potresti riscontrare il seguente errore:

```
Error: “Authentication method "NTLMSSP" not supported (Microsoft SQL Server, Error:
 514)"
```

Questo errore si verifica quando il client TDS non riesce a memorizzare nella cache il ticket di servizio per l'URL dell'endpoint fornito. Per ulteriori informazioni, consulta Connessione con Kerberos.

Impostazione dell’autenticazione Kerberos utilizzando i gruppi di sicurezza Active Directory per Babelfish

A partire da Babelfish versione 4.2.0, è possibile configurare l’autenticazione Kerberos per Babelfish con i gruppi di sicurezza Active Directory. Di seguito sono riportati i prerequisiti da soddisfare per configurare l’autenticazione Kerberos tramite Active Directory:

- È necessario seguire tutti i passaggi indicati in Autenticazione Kerberos con Babelfish.
- Assicurati che l’istanza database sia associata ad Active Directory. Puoi visualizzare lo stato dell’appartenenza al dominio nella console o eseguendo il comando AWS CLI describe-db-instances.

  Lo stato dell’istanza database deve essere Kerberos abilitato. Per ulteriori informazioni sull’appartenenza al dominio, consulta Appartenenza al dominio.
- Verifica le mappature tra il nome di dominio NetBIOS e il nome di dominio DNS utilizzando la seguente query:

  ```
  SELECT netbios_domain_name, fq_domain_name FROM babelfish_domain_mapping;
  ```
- Prima di procedere, verifica che l’autenticazione Kerberos tramite singolo accesso funzioni come previsto. La connessione che utilizza l’autenticazione Kerberos come utente di Active Directory deve avere esito positivo. In caso di problemi, consulta Errori frequenti.

Configurazione dell’estensione pg_ad_mapping

È necessario seguire tutti i passaggi indicati in Configurazione dell’estensione pg_ad_mapping. Per verificare che l’estensione sia installata, esegui la seguente query dall’endpoint TDS:

```
1> SELECT extname, extversion FROM pg_extension where extname like 'pg_ad_mapping';
2> GO
extname       extversion
------------- ----------
pg_ad_mapping 0.1

(1 rows affected)
```

Gestione degli accessi di gruppo

Crea gli accessi di gruppo seguendo i passaggi indicati in Gestione degli accessi. Sebbene non sia obbligatorio, è consigliabile che il nome di accesso sia uguale al nome del gruppo di sicurezza Active Directory (AD) per semplificare la manutenzione. Per esempio:

```
CREATE LOGIN [corp\accounts-group] FROM WINDOWS [WITH DEFAULT_DATABASE=database]
```

Mappatura degli accessi del gruppo T-SQL con il gruppo di sicurezza AD

È necessario allocare in modo esplicito un accesso del gruppo Windows T-SQL per ogni gruppo di sicurezza AD che richiede l’accesso al server di database. Un utente AD, che fa parte di almeno un gruppo di sicurezza AD allocato, avrà accesso al server del database.

> **Note**
>
> Questo accesso T-SQL non può più autenticarsi utilizzando un’autenticazione basata su password.

Ad esempio, accounts-group è un gruppo di sicurezza in AD e per fornire questo gruppo di sicurezza in Babelfish, è necessario utilizzare il formato [corp\accounts-group].

- Gruppo di sicurezza AD: accounts-group
- Accesso TSQL: [corp\accounts-group]
- Ruolo PG equivalente per un determinato accesso TSQL: accounts-group@CORP.EXAMPLE.COM

L'amministratore può ora procedere alla creazione della mappatura tra il gruppo di sicurezza AD e l'accesso T-SQL dall'endpoint PostgreSQL utilizzando il seguente comando psql. Per ulteriori informazioni sull'utilizzo della funzione, consulta Utilizzo delle funzioni dall'estensione `pg_ad_mapping`.

> **Note**
>
> L'accesso T-SQL deve essere specificato nel formato login_name@FQDN durante l'aggiunta della mappatura. I pesi vengono ignorati durante la connessione tramite l'endpoint TDS. Per ulteriori informazioni sull'utilizzo dei pesi, consulta Connessione a Babelfish tramite l'endpoint PostgreSQL sulla porta PostgreSQL.

```
postgres=>select pgadmap_set_mapping('accounts-group', 'accounts-
group@CORP.EXAMPLE.COM', <SID>, <Weight>);
```

Per informazioni sul recupero del SID del gruppo di sicurezza AD, consulta Recupero del SID del gruppo Active Directory in PowerShell.

La tabella seguente mostra un esempio di mappatura dai gruppi di sicurezza AD agli accessi T-SQL:

Gruppi di sicurezza AD	Accessi TSQL	Ruolo PG equivalente per un determinato accesso TSQL	Weight
accounts-group	[corp\accounts-group]	accounts-group@CORP.EXAMPLE.COM	7
sales-group	[corp\sales-group]	sales-group@CORP.EXAMPLE.COM	10
dev-group	[corp\dev-group]	dev-group@CORP.EXAMPLE.COM	7

```
postgres=> select admap.ad_sid, admap.ad_grp, lgn.orig_loginname, lgn.rolname,
 admap.weight from pgadmap_read_mapping() as admap, sys.babelfish_authid_login_ext as
 lgn where admap.pg_role = lgn.rolname;
    ad_sid    |     ad_grp     |    orig_loginname   |             rolname
 | weight
--------------+---------------+---------------------+-------------------------------
+--------
 S-1-5-67-890 | accounts-group | corp\accounts-group  | accounts-group@CORP.EXAMPLE.COM
 |     7
 S-1-2-34-560 | sales-group    | corp\sales-group     | sales-group@CORP.EXAMPLE.COM
 |     10
 S-1-8-43-612 | dev-group      | corp\dev-group       | dev-group@CORP.EXAMPLE.COM
 |     7
 (7 rows)
```

Connessione a Babelfish tramite l’endpoint TDS

Nell’esempio seguente, user1 è membro di accounts-group e sales-group e user2 è membro di accounts-group e dev-group.

Username	Appartenenza ai gruppi di sicurezza AD
user1	accounts-group, sales-group
user2	accounts-group, dev-group

La connessione al server di database Babelfish viene eseguita utilizzando l’utilità sqlcmd. È possibile verificare se un utente (user1 in questo esempio) è stato autenticato utilizzando Kerberos con l’esempio seguente:

```
1> select principal, gss_authenticated from pg_stat_gssapi where pid =
 pg_backend_pid();
2>  GO
principal               gss_authenticated
----------------------  -----------------
user1@CORP.EXAMPLE.COM  1

((1 rows affected))
```

```
1> select suser_name();
2>  GO
suser_name
----------
corp\user1

(1 rows affected)
```

In questo esempio, user1 erediterà i privilegi di accounts-group e sales-group. È possibile verificare l'appartenenza al gruppo utilizzando la visualizzazione di sistema `sys.login_token`.

```
1> SELECT name, type FROM sys.login_token;
2>  GO
name                type
------------------- ----
corp\accounts-group WINDOWS GROUP
corp\sales-group    WINDOWS GROUP

(2 rows affected)
```

Audit e registrazione dei log

Per determinare l'identità principale di sicurezza AD, utilizza i seguenti comandi:

```
1> select suser_name();
2> GO
suser_name
----------
corp\user1

(1 rows affected)
```

Attualmente, l'identità dell'utente AD non è visibile nei log. È possibile attivare il parametro `log_connections` per registrare nei log l'istituzione della sessione di database. Per ulteriori informazioni, consulta log_connections. L'output include l'identità dell'utente AD come principale, come mostrato nell'esempio seguente. Il PID di backend associato a questo output può quindi aiutare ad attribuire le azioni all'utente AD effettivo.

```
bbf_group_ad_login@babelfish_db:[615]:LOG: connection authorized:
 user=bbf_group_ad_login database=babelfish_db application_name=sqlcmd GSS
 (authenticated=yes, encrypted=yes, principal=user1@CORP.EXAMPLE.COM)
```

Utilizzo dei privilegi di appartenenza al gruppo di sicurezza AD

Eredità dei privilegi a livello di server

Gli utenti AD che sono membri di un determinato gruppo di sicurezza AD ereditano i privilegi a livello di server concessi all'accesso del gruppo Windows mappato. Ad esempio, considera il gruppo di sicurezza AD `accounts-group`, a cui viene concessa l'appartenenza al ruolo del server `sysadmin` su Babelfish. È possibile ereditare i privilegi a livello di server utilizzando il comando seguente:

```
1> ALTER SERVER ROLE sysadmin ADD MEMBER [corp\accounts-group];
```

Di conseguenza, qualsiasi utente Active Directory che è membro del gruppo di sicurezza AD `accounts-group` erediterà i privilegi a livello di server associati al ruolo `sysadmin`. Ciò significa che un utente come `corp\user1`, in qualità di membro di `accounts-group`, ora avrà la possibilità di eseguire operazioni a livello di server all'interno di Babelfish.

> **Note**
>
> Per eseguire le DDL a livello di server, è necessario l'accesso Windows per il singolo utente AD. Per ulteriori informazioni, consulta Limitazioni.

Eredità dei privilegi a livello di database

Per concedere i privilegi a livello di database, è necessario creare e mappare un utente del database con l'accesso del gruppo Windows. Gli utenti AD che sono membri di un determinato gruppo di sicurezza AD ereditano i privilegi a livello di database concessi a tale utente del database. Nell'esempio seguente, è possibile vedere come vengono assegnati i privilegi a livello di database per il gruppo Windows [corp\accounts-group].

```
1> CREATE DATABASE db1;
2> GO
```

```
1> USE db1;
2> GO
Changed database context to 'db1'.
1> CREATE TABLE dbo.t1(a int);
2> GO
```

Crea un utente del database [corp\sales-group] per l'accesso al gruppo Windows [corp\accounts-group]. Per eseguire questo passaggio, connettiti tramite l'endpoint TDS utilizzando l'accesso come membro di sysadmin.

```
1> CREATE USER [corp\accounts-group] FOR LOGIN [corp\accounts-group];
2> GO
```

Ora, connettiti come l'utente AD user1 per verificare l'accesso alla tabella t1. Poiché non abbiamo ancora concesso i privilegi a livello di database, si verificherà un errore di autorizzazione negata.

```
1> SELECT * FROM dbo.t1;
2> GO
Msg 33557097, Level 16, State 1, Server db-inst, Line 1
permission denied for table t1
```

Esegui GRANT SELECT sulla tabella t1 per l'utente del database [corp\accounts-group]. Per eseguire questo passaggio, connettiti tramite l'endpoint TDS utilizzando l'accesso come membro di sysadmin.

```
1> GRANT SELECT ON dbo.t1 TO [corp\accounts-group];
2> GO
```

Connettiti come l'utente AD user1 per convalidare l'accesso.

```
1> SELECT * FROM dbo.t1;
2> GO
```

```
a
-----------

(0 rows affected)
```

Gestione del comportamento dell'istruzione DDL in base allo schema predefinito o esplicito

Quando si utilizza una sessione autenticata da AD, lo schema predefinito per la sessione corrente è determinato dalle seguenti condizioni:

- Se esiste un singolo utente del database, lo schema predefinito dell'utente viene considerato lo schema predefinito della sessione corrente.
- Se esiste lo schema predefinito per un utente del database di gruppo, lo schema predefinito dell'utente del database di gruppo viene considerato lo schema predefinito della sessione corrente con l'ID principale più piccolo.

Informazioni sul comportamento dell'istruzione CREATE DDL

Se non è specificato uno schema esplicito nell'istruzione CREATE DDL, la creazione dell'oggetto avviene nello schema predefinito della sessione corrente. Se non è possibile determinare se lo schema è predefinito o esplicito, l'istruzione DDL genererà il seguente errore:

```
"Babelfish Unsupported Command : Schema required for CREATE DDLs when connecting with
 Active Directory Group authentication. Assign default schema to group user or specify
 schema in command."
```

Example : lo schema predefinito non esiste per gli utenti del gruppo Windows

L'utente del gruppo Windows [corp\accounts-group] ha uno schema predefinito NULL e l'utente AD user1 sta tentando di eseguire l'istruzione DDL senza specificare lo schema in modo esplicito. Poiché l'accesso e l'utente Windows individuali non esistono per user1, otterrà i privilegi a livello di database solo dell'utente del gruppo Windows [corp\accounts-group].

```
1> create TABLE t2(a int);
```

```
2> GO

Msg 33557097, Level 16, State 1, Server db-inst, Line 1
Babelfish Unsupported Command : Schema required for CREATE DDLs when connecting with
 Active Directory Group authentication. Assign default schema to group user or specify
 schema in command.
```

> **Note**
>
> L’accesso e l’utente Windows individuali non esistono per l’utente AD user1

Example : esiste uno schema predefinito per gli utenti del gruppo Windows

Creare un utente del gruppo Windows per l’accesso [corp\accounts-group] con lo schema predefinito utilizzando sysadmin.

```
1> CREATE USER [corp\accounts-group] FOR LOGIN [corp\accounts-group] WITH
 DEFAULT_SCHEMA = sch_acc;
2> GO
1> CREATE SCHEMA sch_acc AUTHORIZATION [gad\accounts-group];
2> GO
1> SELECT name, principal_id, default_schema_name FROM sys.database_principals WHERE
 name = 'corp\accounts-group';
2> GO

name               principal_id default_schema_name
------------------ ------------ ------------------
corp\accounts-group 24162        sch_acc

(1 rows affected)
```

Provare a creare un oggetto senza specificare lo schema in modo esplicito utilizzando l’utente AD user1. La tabella t2 verrà creata nello schema predefinito dell’utente del gruppo Windows [corp \accounts-group]. Il proprietario di questo oggetto sarà lo stesso del proprietario dello schema sch_acc.

```
1> CREATE TABLE t_group(a int);
2> GO
1> SELECT name, schema_name(schema_id) FROM sys.objects WHERE name like 't_group';
2> GO

name    schema_name
------- -----------
t_group sch_acc

(1 rows affected)
```

> **Note**
>
> L’accesso e l’utente Windows individuali non esistono per l’utente AD user1

Example : esiste anche un singolo utente del database per un utente AD

Se esiste anche un singolo utente del database per un utente AD, gli oggetti verranno sempre creati nello schema associato al singolo utente del database. Se lo schema non esiste per l’utente del database, verrà utilizzato lo schema dbo. Creare un singolo accesso Windows e utente del database per l’utente AD user1. Eseguire la connessione tramite l’endpoint TDS utilizzando un accesso sysadmin.

```
1> CREATE LOGIN [corp\user1] FROM WINDOWS;
2> GO
1> CREATE USER [corp\user1] FOR LOGIN [corp\user1] WITH DEFAULT_SCHEMA = sch1;
2> GO
1> CREATE SCHEMA sch1 AUTHORIZATION [corp\user1];
2> GO
1> SELECT name, default_schema_name FROM sys.database_principals WHERE name = 'corp
\user1';
2> GO

name      default_schema_name
--------- -------------------
corp\user1 sch1
```

```
(1 rows affected)
```

Eseguire la connessione utilizzando l’utente AD user1 e prova a creare un oggetto senza specificare lo schema in modo esplicito. La tabella t2 verrà creata nello schema sch1. Il proprietario di questo oggetto sarà lo stesso del proprietario dello schema sch1.

```
1> CREATE TABLE t2(a int);
2> GO
1> SELECT name, schema_name(schema_id) FROM sys.objects WHERE name like 't2';
2> GO

name schema_name
---- -----------
t2   sch1

(1 rows affected)
```

Limitazioni

- L’utilità Dump/Restore non supporta il dump delle mappature dell’estensione pg_ad_mapping. È necessario ricreare queste mappature dopo il ripristino.
- L’implementazione blu/verde non è supportata per Babelfish e le istanze Aurora PostgreSQL con `pg_ad_mapping`.
- La creazione implicita dello schema non è supportata. Le istruzioni DDL che richiedono la creazione implicita dello schema non sono supportate.
- Le DDL a livello di server ALTER AUTHORIZATION ON DATABASE, CREATE DATABASE, CREATE LOGIN, ALTER LOGIN, ALTER SERVER ROLE, ALTER DATABASE non sono supportate in una sessione di gruppo autenticata da AD quando non esiste un singolo accesso Windows, ma esiste solo l’accesso Windows di gruppo. Per ovviare a questa limitazione, si consiglia di eseguire queste operazioni in una sessione autenticata tramite password o di creare un singolo accesso Windows.
- La creazione implicita dell’utente non è supportata. Comportamento T-SQL ideale [non ancora supportato in Babelfish]: in alcuni casi, come le istruzioni DDL e di controllo degli accessi come

GRANT/REVOKE, in cui il nome dell'utente AD è specificato nel comando ma non esiste nel database, l'utente del database denominato come utente AD viene creato implicitamente.

- Per le DDL nelle procedure o nelle funzioni PL/pgSQL create dall'endpoint PSQL ed eseguite dall'endpoint TDS nella sessione autenticata di gruppo AD:
  - Le istruzioni ALTER/DROP sono supportate.
  - CREATE TABLE, CREATE VIEW, CREATE INDEX, CREATE FUNCTION/PROC, CREATE TYPE, CREATE SEQUENCE, CREATE TRIGGER, SELECT INTO, CREATE FULLTEXT INDEX, CREATE UNIQUE INDEX genereranno un errore se lo schema non viene fornito in modo esplicito e lo schema predefinito è nullo per la sessione corrente.
  - CREATE DATABASE, CREATE EXTENSION e tutte le altre istruzioni CREATE per oggetti specifici PG (non in T-SQL) CREATE subscription, CREATE tablespace, CREATE policy, CREATE conversion non saranno supportate.
- Le DDL dell'endpoint PostgreSQL non sono supportate nella sessione di gruppo autenticata da AD. Come soluzione alternativa, è sempre possibile connettersi utilizzando l'utente principale o qualsiasi altro utente tramite un meccanismo di autenticazione basato su password.
- Gli oggetti di sistema come SUSER_SID(), IS_SRVROLEMEMBER(), IS_MEMBER(), sys.dm_exec_sessions presentano le seguenti limitazioni.
  - SUSER_SID() non restituirà il SID quando viene fornito un utente AD o un gruppo di sicurezza AD.
  - IS_SRVROLEMEMBER() non prenderà in considerazione l'appartenenza al ruolo se l'utente AD corrente eredita l'appartenenza al ruolo del server dall'appartenenza al ruolo del server di un accesso del gruppo Windows.
  - IS_MEMBER() restituirà false per qualsiasi query relativa al gruppo Windows.
  - sys.dm_exec_sessions non mostrerà i valori previsti nelle colonne login_name, nt_user_name.

Connessione a Babelfish tramite l'endpoint PostgreSQL sulla porta PostgreSQL

È possibile utilizzare gli accessi di gruppo creati dalla porta TDS per connettersi anche tramite la porta PostgreSQL. Per connettersi tramite la porta PostgreSQL, è necessario specificare il nome utente AD nel formato `<ad_username@FQDN>` dalle applicazioni client PostgreSQL. Non è possibile utilizzare il formato `<DNS domain name\ad_username>`.

PostgreSQL utilizza per impostazione predefinita il confronto con distinzione tra maiuscole e minuscole per i nomi utente. Affinché Aurora PostgreSQL interpreti i nomi utente Kerberos senza distinzione tra maiuscole e minuscole, è necessario impostare il parametro krb_caseins_users su true

nel gruppo di parametri del cluster Babelfish personalizzato. Questo parametro è impostato su false per impostazione predefinita. Per ulteriori informazioni, consulta Configurazione del cluster database Aurora PostgreSQL per nomi utente senza distinzione tra maiuscole e minuscole.

Differenze di comportamento tra gli endpoint T-SQL e PostgreSQL quando un utente AD fa parte di più gruppi

L'utente AD user1 fa parte di due gruppi di sicurezza AD [corp\accounts-group] e [corp\sales-group] e l'amministratore di database ha impostato la mappatura dell'utente nel modo seguente.

```
postgres=> select * from pgadmap_read_mapping();

ad_sid       | pg_role                          | weight | ad_grp
-------------+----------------------------------+--------+---------------
S-1-5-67-980 | accounts-group@CORP.EXAMPLE.COM | 7      | accounts-group
S-1-2-34-560 | sales-group@CORP.EXAMPLE.COM    | 10     | sales-group
(2 rows)
```

Se l'utente si connette dall'endpoint T-SQL, durante l'autorizzazione erediterà i privilegi da tutti gli accessi T-SQL associati. In questo esempio user1 erediterà l'unione dei privilegi da entrambi gli accessi di gruppo T-SQL e i pesi verranno ignorati. Questo approccio è in linea con il comportamento T-SQL standard.

Tuttavia, se lo stesso utente si connette dall'endpoint PostgreSQL, può ereditare i privilegi da un solo accesso T-SQL associato con il peso più elevato. Se ai due accessi di gruppo T-SQL è stato assegnato lo stesso peso, l'utente AD erediterà i privilegi dell'accesso T-SQL corrispondente alla mappatura aggiunta più di recente. Per PostgreSQL, è consigliabile specificare pesi che riflettano le autorizzazioni/privilegi relativi dei singoli ruoli di database per evitare l'ambiguità. Nell'esempio seguente, user1 si è connesso tramite l'endpoint PSQL e ha ereditato solo i privilegi dei sales-groups.

```
babelfish_db=> select session_user, current_user;

   session_user               |   current_user
------------------------------+--------------------------
 sales-group@CORP.EXAMPLE.COM | sales-group@CORP.EXAMPLE.COM
(1 row)
```

```
babelfish_db=> select principal, gss_authenticated from pg_stat_gssapi where pid =
 pg_backend_pid();

     principal          | gss_authenticated
------------------------+-------------------
 user1@CORP.EXAMPLE.COM | t
(1 row)
```

Connessione a un cluster database Babelfish

Per connetterti a Babelfish, esegui la connessione all'endpoint del cluster Aurora PostgreSQL che esegue Babelfish. Il client può utilizzare uno dei seguenti driver client conformi a TDS versione 7.1 fino a 7.4:

- Open Database Connectivity (ODBC)
- Driver database OLE/msoledBSQL
- Java Database Connectivity (JDBC) versione 8.2.2 (mssql-jdbc-8.2.2) e versioni successive
- Microsoft SQLClient Data Provider per SQL Server
- Provider di dati .NET per SQL Server
- SQL Server Native Client 11.0 (obsoleto)
- Provider OLE DB/SQLOLEDB (obsoleto)

Con Babelfish esegui le seguenti operazioni:

- Strumenti, applicazioni e sintassi di SQL Server sulla porta TDS, per impostazione predefinita della porta 1433.
- Strumenti, applicazioni e sintassi PostgreSQL sulla porta PostgreSQL, per impostazione predefinita 5432.

Per ulteriori informazioni sulla connessione ad Aurora PostgreSQL, consulta Connessione a un cluster di database Amazon Aurora PostgreSQL.

> **Note**
>
> Gli strumenti di sviluppo di terze parti che utilizzano il provider SQL Server OLEDB per accedere ai metadati non sono supportati. È consigliabile utilizzare connessioni client SQL Server JDBC, ODBC o SQL Native per questi strumenti.

A partire da Babelfish versione 5.1.0, la crittografia delle connessioni end-to-end viene applicata per impostazione predefinita. Per garantire una connettività continua:

- Configura la crittografia SSL/TLS per le connessioni. Per ulteriori informazioni, consulta Utilizzo SSL/TLS per crittografare una connessione a un'.

- Importa i certificati richiesti nei computer client. Per ulteriori informazioni, consulta Utilizzo di SSL con un'istanza database Microsoft SQL Server.

Se desideri continuare a utilizzare le impostazioni di crittografia di una versione precedente di Babelfish (precedente alla versione 5.1.0), puoi impostare il parametro `rds.force_ssl` su `0` nel gruppo di parametri del cluster di database.

Argomenti

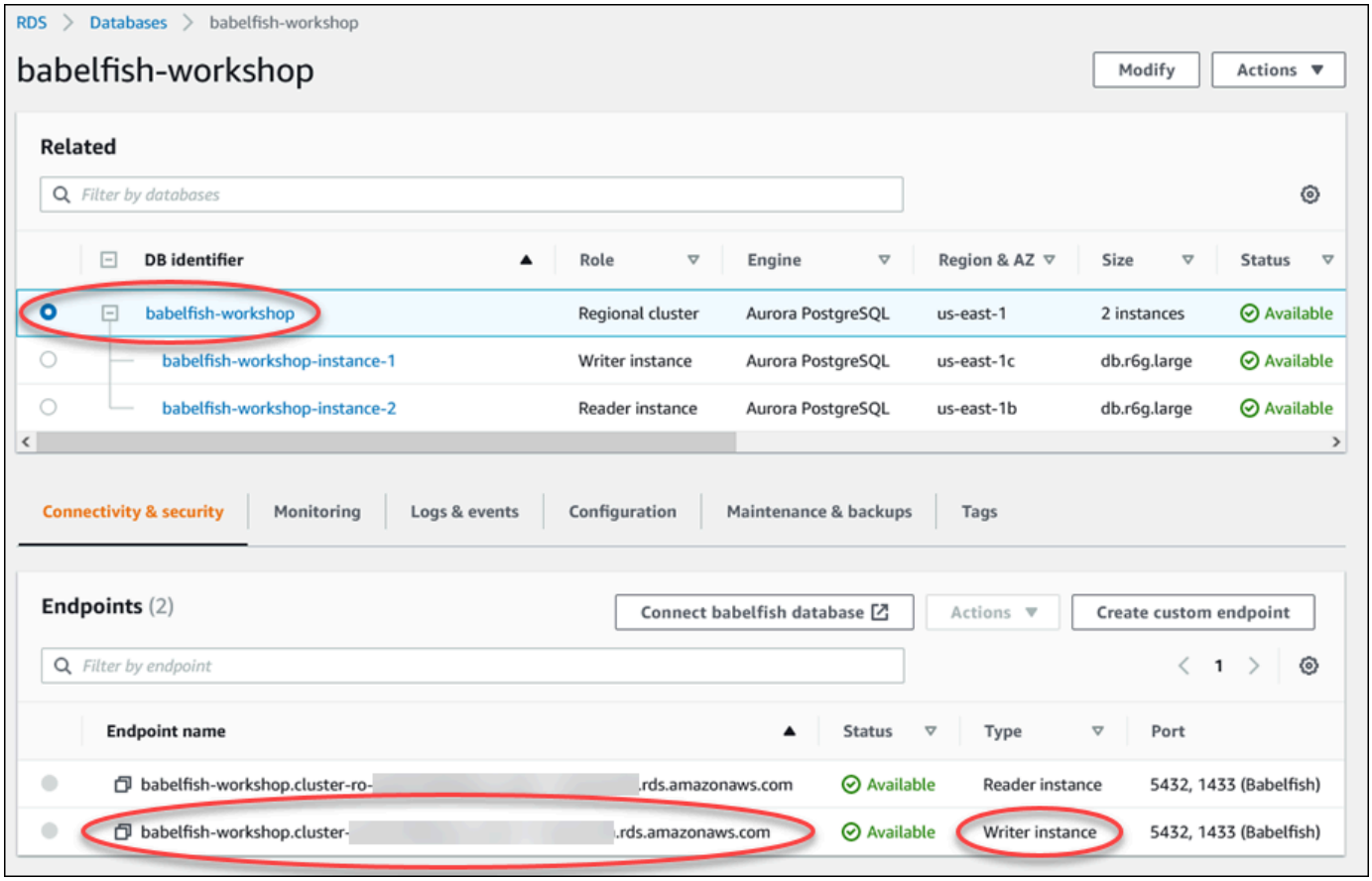
- Ricerca dell'endpoint del writer e del numero di porta
- Creazione di connessioni client C # o JDBC Babelfish
- Utilizzo di un client SQL Server per la connessione al cluster database
- Utilizzo di un client PostgreSQL per connetterti al cluster di database

Ricerca dell'endpoint del writer e del numero di porta

Per connetterti al cluster database Babelfish, utilizza l'endpoint associato all'istanza di scrittura (primaria) del cluster database. Lo stato dell'istanza deve essere Available (Disponibile). Dopo la creazione del cluster database Babelfish per Aurora PostgreSQL, possono essere necessari fino a 20 minuti affinché le istanze siano disponibili.

Per trovare l'endpoint del database

1. Apri la console per Babelfish.
2. Nel pannello di navigazione seleziona Database.
3. Scegli il cluster database Babelfish per Aurora PostgreSQL tra quelli elencati per visualizzarne i dettagli.
4. Sulla scheda Connettività e sicurezza, prendi nota dei valori Endpoint del cluster disponibile. Utilizza l'endpoint del cluster per l'istanza del writer nelle tue stringhe di connessione per tutte le applicazioni che eseguono operazioni di scrittura o lettura del database.



Per ulteriori informazioni sui dettagli del cluster database Aurora, consulta Creazione di un cluster database Amazon Aurora.

Creazione di connessioni client C # o JDBC Babelfish

Di seguito sono disponibili alcuni esempi di utilizzo delle classi C # e JDBC per connettersi a un Babelfish per Aurora PostgreSQL.

Example : Utilizzo del codice C # per connettersi a un cluster database

```
string dataSource = 'babelfishServer_11_24';

//Create connection
connectionString = @"Data Source=" + dataSource
    +";Initial Catalog=your-DB-name"
    +";User ID=user-id;Password=password";

// [optional] To validate server certificate during TLS/SSL connection
connectionString = connectionString + ";ServerCertificate=/path/to/certificate.pem";


SqlConnection cnn = new SqlConnection(connectionString);
cnn.Open();
```

Example : Utilizzo di classi e interfacce API JDBC generiche per connettersi a un cluster database

```
String dbServer =
   "database-babelfish.cluster-123abc456def.us-east-1-rds.amazonaws.com";
String connectionUrl = "jdbc:sqlserver://" + dbServer + ":1433;" +
    "databaseName=your-DB-name;user=user-id;password=password";

// [optional] To validate server certificate during TLS/SSL connection
connectionUrl = connectionUrl + ";serverCertificate=/path/to/certificate.pem";

// Load the SQL Server JDBC driver and establish the connection.
System.out.print("Connecting Babelfish Server ... ");
Connection cnn = DriverManager.getConnection(connectionUrl);
```

Example : Utilizzo di classi e interfacce JDBC specifiche di SQL Server per connettersi a un cluster database

```
// Create datasource.
SQLServerDataSource ds = new SQLServerDataSource();
ds.setUser("user-id");
ds.setPassword("password");
```

```
String babelfishServer =
   "database-babelfish.cluster-123abc456def.us-east-1-rds.amazonaws.com";

ds.setServerName(babelfishServer);
ds.setPortNumber(1433);
ds.setDatabaseName("your-DB-name");

// [optional] To validate server certificate during TLS/SSL connection
ds.setServerCertificate("/path/to/certificate.pem");

Connection con = ds.getConnection();
```

Important

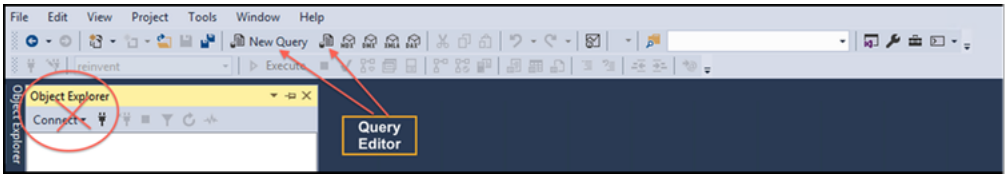
Assicurati che il certificato corrisponda all'Autorità di certificazione mostrata nella configurazione del cluster di database sulla Console di gestione AWS.

Utilizzo di un client SQL Server per la connessione al cluster database

È possibile utilizzare un client SQL Server per connettersi a Babelfish sulla porta TDS. A partire da Babelfish 2.1.0 e versioni successive, puoi utilizzare SSMS Object Explorer o l'editor di query SSMS per connetterti al cluster Babelfish.

Limitazioni

- In Babelfish 2.1.0 e versioni precedenti, l'utilizzo di `PARSE` per controllare la sintassi SQL non funziona come previsto. Invece di controllare la sintassi senza eseguire la query, il comando `PARSE` esegue la query ma non visualizza alcun risultato. L'utilizzo della combinazione di tasti <Ctrl><F5> SMSS per verificare la sintassi ha lo stesso comportamento anomalo, ovvero, Babelfish esegue inaspettatamente la query senza fornire alcun output.
- Babelfish non supporta MARS (Multiple Active Result Set). Assicurati che tutte le applicazioni client utilizzate per connetterti a Babelfish non siano impostate per utilizzare MARS.
- Per Babelfish 1.3.0 e versioni precedenti, solo l'editor di query è supportato per SSMS. Per utilizzare SSMS con Babelfish, assicurati di aprire la finestra di dialogo di connessione dell'editor di query in SSMS e non Object Explorer. Se si apre la finestra di dialogo Object Explorer, annulla la finestra di dialogo e apri nuovamente l'editor di query. Nell'immagine seguente, sono disponibili le opzioni di menu per scegliere quando connetterti a Babelfish 1.3.0 o versioni precedenti.



Per ulteriori informazioni sull'interoperabilità e le differenze di comportamento tra SQL Server e Babelfish, consulta Differenze tra Babelfish per Aurora PostgreSQL e SQL Server.

Utilizzo di sqlcmd per connettersi al cluster di database

È possibile connettersi e interagire con un cluster di database Aurora PostgreSQL che supporta Babelfish solo utilizzando il client di riga di comando SQL Server `sqlcmd` versione 19.1 e precedenti. La versione 19.2 di SSMS non è supportata per la connessione a un cluster Babelfish. Utilizzare il seguente comando.

```
sqlcmd -S endpoint,port -U login-id -P password -d your-DB-name
```

Le opzioni sono le seguenti:

- `-S` è l'endpoint e la porta TDS (opzionale) del cluster database.
- `-U` è il nome di accesso dell'utente.
- `-P` è la password associata all'utente
- `-d` è il nome del database Babelfish.

Dopo la connessione, è possibile utilizzare molti degli stessi comandi utilizzati con SQL Server. Per alcuni esempi, consulta Recupero di informazioni dal catalogo di sistema Babelfish.

Utilizzo di SSMS per connettersi al cluster database

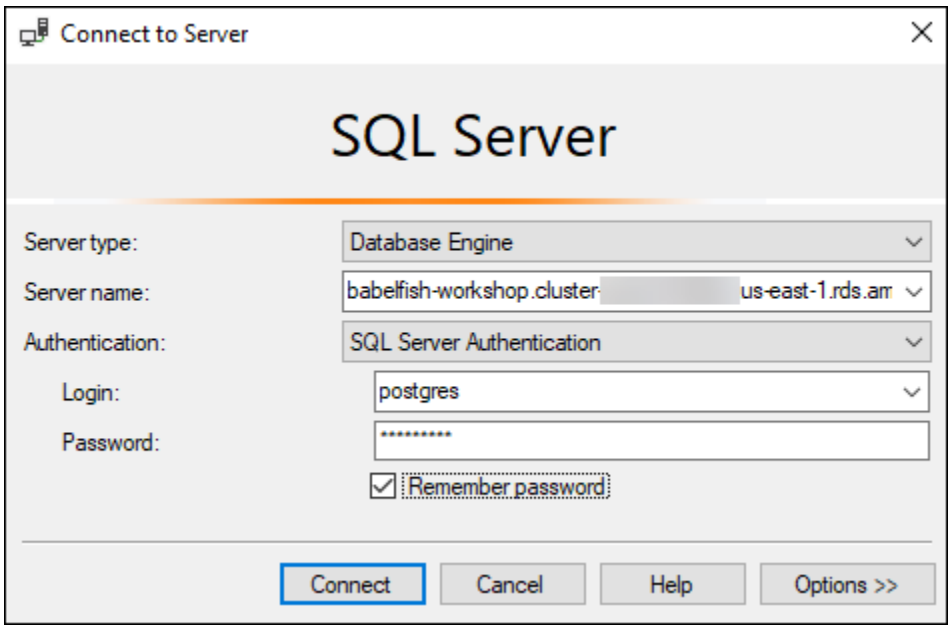
Puoi connetterti a un cluster database Aurora PostgreSQL che esegue Babelfish utilizzando Microsoft SQL Server Management Studio (SSMS). SSMS include una varietà di strumenti, incluso Importazione/Esportazione guidata di SQL Server discussa in Migrazione di un database SQL Server a Babelfish per Aurora PostgreSQL. Per ulteriori informazioni su SSMS, consulta Download SQL Server Management Studio (SSMS) nella documentazione di Microsoft. Per configurare SSL/TLS, consulta Utilizzo di SSL con un'istanza database Microsoft SQL Server.

Connessione al tuo database Babelfish con SSMS

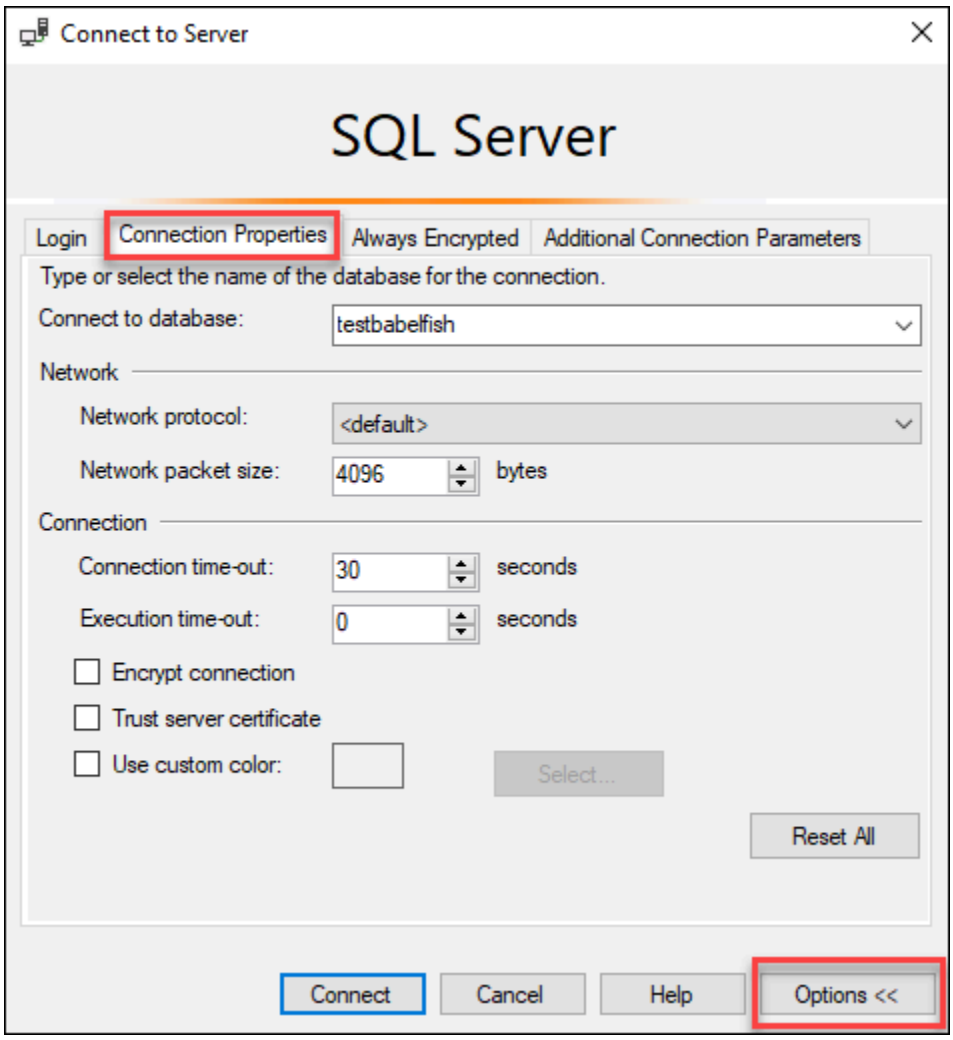
1. Avvia SSMS.
2. Apri la finestra di dialogo Connect to Server (Connettiti al server). Per continuare con la connessione, esegui una delle seguenti operazioni:
   - Scegliere New Query (Nuova query).
   - Se l'editor di query è aperto, scegliere Query,Connection (Connessione),Connect (Connetti).
3. Fornisci le seguenti informazioni per il tuo database:
   a. In Server type (Tipo di server) scegliere Database Engine (Motore di database).
   b. Per Server name (Server name), immettere il nome DNS. Ad esempio, il nome server dovrebbe essere simile a quanto segue.

      ```
      cluster-name.cluster-555555555555.aws-region.rds.amazonaws.com,1433
      ```

   c. In Authentication (Autenticazione) selezionare SQL Server Authentication (Autenticazione SQL Server).
   d. Per Login, immettere il nome utente scelto al momento della creazione del database.
   e. Per Password, immettere la password scelta al momento della creazione del database.



4. (Facoltativo) Scegli Opzioni e quindi scegli la scheda Connection Properties (Proprietà della connessione).



5. (Opzionale) Per Connect to database (Connetti al database), specificare il nome del database SQL Server migrato a cui connettersi e scegliere Connect (Collegarsi).

   Se viene visualizzato un messaggio che indica che SSMS non può applicare stringhe di connessione, scegliere OK.

   Se hai problemi a connetterti a Babelfish, consulta Errore della connessione.

   Per ulteriori informazioni sui problemi di connessione, consulta Risoluzione dei problemi di connessione all'istanza database di SQL Server nella Guida per l'utente di Amazon RDS.

Utilizzo di un client PostgreSQL per connetterti al cluster di database

È possibile utilizzare un client PostgreSQL per connettersi a Babelfish sulla porta PostgreSQL. A partire dalla versione 5.1.0, il server Babelfish applica la crittografia della connessione end-to-end per impostazione predefinita. Aggiorna le applicazioni per utilizzare i certificati SSL/TLS. Per ulteriori informazioni sulla configurazione dei certificati SSL/TLS, consulta the section called “Sicurezza dei dati Aurora PostgreSQL con SSL/TLS”.

Usare psql per connettersi al cluster di database

Puoi scaricare il client PostgreSQL dal sito Web di PostgreSQL. Per installare psql, segui le istruzioni relative al tuo sistema operativo.

Puoi eseguire query su un cluster di database Aurora PostgreSQL che supporta Babelfish con il client a riga di comando `psql`. Durante la connessione, utilizza la porta PostgreSQL (per impostazione predefinita, porta 5432). In genere, non è necessario specificare il numero di porta, a meno che non sia stato modificato rispetto a quello predefinito. Utilizza il seguente comando per connetterti a Babelfish dal client `psql`:

```
psql -h bfish-db.cluster-123456789012.aws-region.rds.amazonaws.com
-p 5432 -U postgres -d babelfish_db
```

I parametri sono i seguenti:

- `-h` – Nome host dell'istanza del cluster di database (endpoint del cluster) cui desideri accedere.
- `-p` – Numero di porta PostgreSQL usato per connettersi alla istanza database.
- `-d`: database a cui si desidera connettersi. Il valore predefinito è `babelfish_db`.
- `-U` – L'account utente del database cui desideri accedere. (L'esempio mostra il nome utente master predefinito.)

Quando si esegue un comando SQL sul client psql, si termina il comando con un punto e virgola. Ad esempio, il comando SQL seguente esegue la query della vista sistemi pg_tables per restituire informazioni su ogni tabella del database.

```
SELECT * FROM pg_tables;
```

Il client psql ha anche un set di metacomandi integrati. Un metacomando è un collegamento che regola la formattazione o fornisce una scorciatoia che restituisce i metadati in un formato facile da

usare. Ad esempio, il seguente metacomando restituisce informazioni simili al precedente comando SQL:

```
\d
```

I metacomandi non devono essere terminati con un punto e virgola (;).

Per uscire dal client psql, immettere \q.

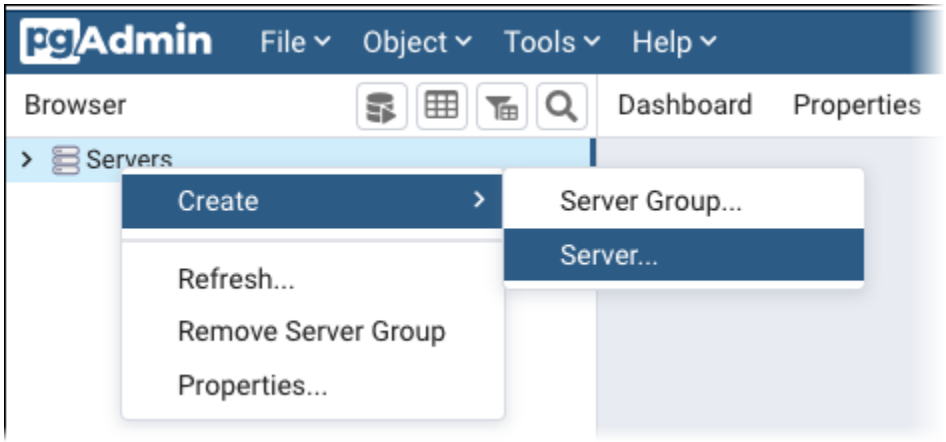
Per ulteriori informazioni sull'utilizzo del client psql per eseguire query su un cluster Aurora PostgreSQL, consulta la documentazione di PostgreSQL.

Uso di pgAdmin per connettersi al cluster di database

È possibile utilizzare il client pgAdmin per accedere ai dati nel dialetto PostgreSQL nativo.

Per connettersi al cluster con il client pgAdmin

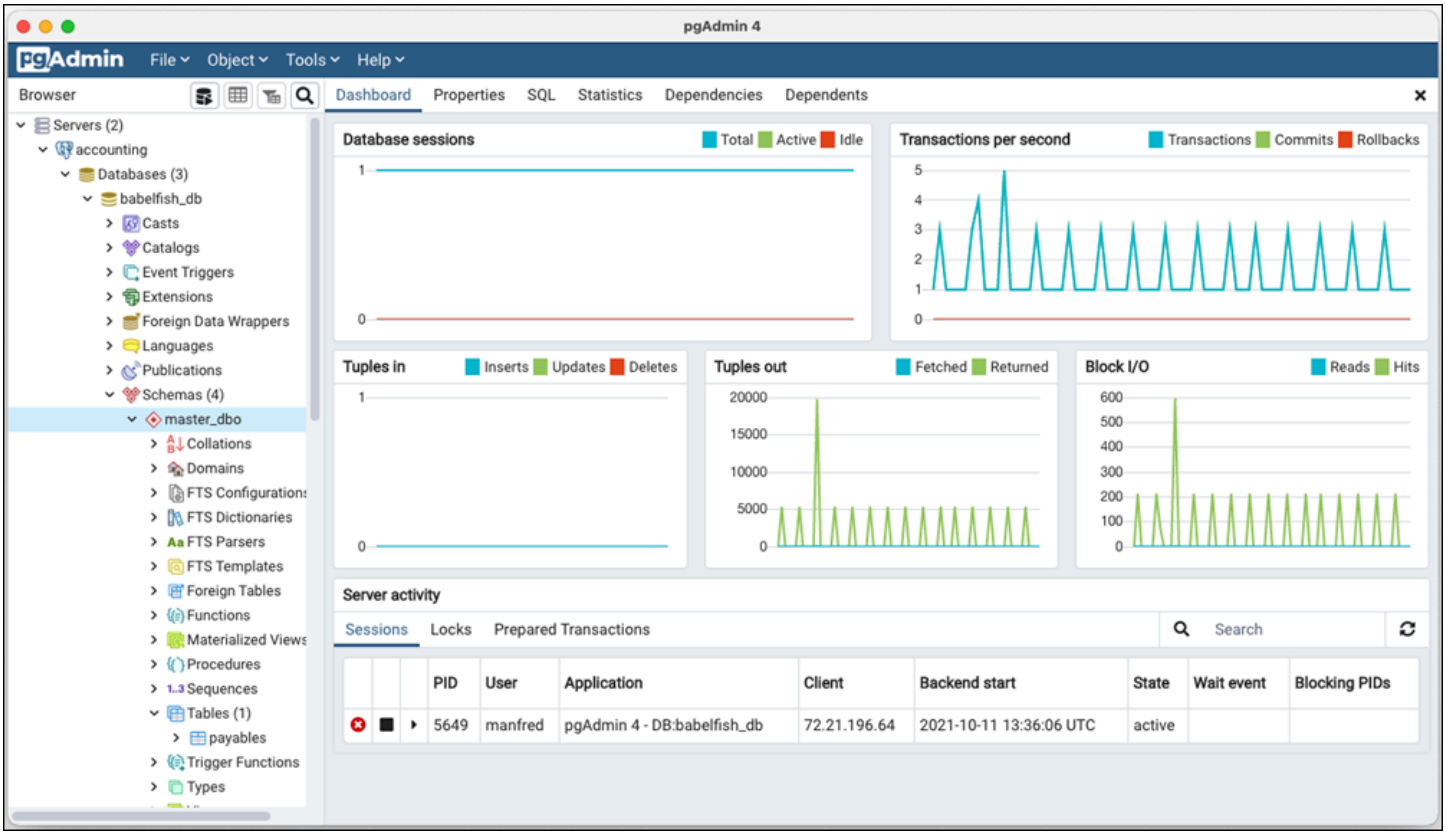
1. Scarica e installa il client pgAdmin dal sito web pgAdmin.
2. Apri il client e esegui l'autenticazione con pgAdmin.
3. Aprire il menu contestuale (pulsante destro del mouse) per Server e quindi scegliere Create (Crea), Server.



4. Inserimento delle informazioni nella finestra di dialogo Create Server (Crea Server).

   Sulla scheda Connection (Connessione), aggiungi l'indirizzo del cluster Aurora PostgreSQL per Host e il numero di porta PostgreSQL (per impostazione predefinita, 5432) per Port (Porta). Fornisci i dettagli di autenticazione e scegli Save (Salva).

Dopo la connessione, è possibile utilizzare la funzionalità PGAmin per monitorare e gestire il cluster Aurora PostgreSQL sulla porta PostgreSQL.



Per ulteriori informazioni, consulta la pagina Web pgAdmin.

Utilizzo di Babelfish

Di seguito sono disponibili informazioni di utilizzo per Babelfish, incluse alcune delle differenze tra l'utilizzo di Babelfish e SQL Server e tra i database Babelfish e PostgreSQL.

Argomenti

- Recupero di informazioni dal catalogo di sistema Babelfish
- Gestione delle autorizzazioni e del controllo degli accessi in Babelfish per Aurora PostgreSQL
- Differenze tra Babelfish per Aurora PostgreSQL e SQL Server
- Utilizzo di funzionalità Babelfish con implementazione limitata
- Miglioramento delle prestazioni della query Babelfish
- Utilizzo delle estensioni Aurora PostgreSQL con Babelfish
- Babelfish supporta i server collegati
- Utilizzo della ricerca full-text in Babelfish
- Babelfish supporta i tipi di dati geospaziali
- Informazioni sul partizionamento in Babelfish

Recupero di informazioni dal catalogo di sistema Babelfish

Puoi ottenere informazioni sugli oggetti di database archiviati nel cluster Babelfish eseguendo query su molte delle stesse viste di sistema utilizzate in SQL Server. Ogni nuova versione di Babelfish aggiunge supporto per ulteriori viste di sistema. Per un elenco delle viste attualmente disponibili, consulta la tabella SQL Server system catalog views.

Queste viste di sistema forniscono informazioni dal catalogo di sistema (`sys.schemas`). Nel caso di Babelfish, queste viste contengono entrambi gli schemi di sistema SQL Server e PostgreSQL. Per eseguire query su Babelfish per informazioni sul catalogo di sistema, puoi utilizzare la porta TDS o la porta PostgreSQL, come illustrato negli esempi seguenti.

- Esegui query sulla porta T-SQL utilizzando **`sqlcmd`** o un altro client SQL Server.

```
1> SELECT * FROM sys.schemas
2> GO
```

  Questa query restituisce gli schemi di sistema SQL Server e Aurora PostgreSQL, come mostrato di seguito.

```
name
------------------------------------------------------------
demographic_dbo
public
sys
master_dbo
tempdb_dbo
...
```

- Esegui query sulla porta PostgreSQL utilizzando **`psql`** o **`pgAdmin`**. In questo esempio viene utilizzato il meta-comando degli schemi di elenco `psql` (`\dn`):

  ```
  babelfish_db=> \dn
  ```

  La query restituisce lo stesso set di risultati restituito da `sqlcmd` sulla porta T-SQL.

  ```
         List of schemas
              Name
  ------------------------------

   demographic_dbo

   public
   sys
   master_dbo
   tempdb_dbo
  ...
  ```

Cataloghi di sistema SQL Server disponibili in Babelfish

Nella tabella seguente sono disponibili le viste SQL Server attualmente implementate in Babelfish. Per ulteriori informazioni sui cataloghi di sistema in SQL Server, consulta System Catalog Views (Transact-SQL) nella documentazione di Microsoft.

Nome della vista	Descrizione o limitazione Babelfish (se presente)
`sys.all_columns`	Tutte le colonne di tutte le tabelle e tutte le viste

Nome della vista	Descrizione o limitazione Babelfish (se presente)
`sys.all_objects`	Tutti gli oggetti in tutti gli schemi
`sys.all_sql_modules`	L'unione di `sys.sql_modules` e `sys.system_sql_modules`
`sys.all_views`	Tutte le visualizzazioni in tutti gli schemi
`sys.columns`	Tutte le colonne in tabelle e viste definite dall'utente
`sys.configurations`	Supporto Babelfish limitato a una singola configurazione di sola lettura.
`sys.data_spaces`	Contiene una riga per ogni spazio dati. Può essere un filegroup, uno schema di partizione o un filegroup di dati FILESTREAM.
`sys.database_files`	Una vista per database contenente una riga per ogni file di un database come archiviato nel database stesso.
`sys.database_mirroring`	Per informazioni, consulta sys.database_mirroring nella documentazione di Microsoft Transact-SQL.
`sys.database_principals`	Per informazioni, consulta sys.database_principals nella documentazione di Microsoft Transact-SQL.
`sys.database_role_members`	Per informazioni, consulta sys.database_role_members nella documentazione di Microsoft Transact-SQL.
`sys.databases`	Tutti i database in tutti gli schemi

Nome della vista	Descrizione o limitazione Babelfish (se presente)
`sys.dm_exec_connections`	Per informazioni, consulta sys.dm_exec_connections nella documentazione di Microsoft Transact-SQL.
`sys.dm_exec_sessions`	Per informazioni, consulta sys.dm_exec_sessions nella documentazione di Microsoft Transact-SQL.
`sys.dm_hadr_database_replica_states`	Per informazioni, consulta sys.dm_hadr_database_replica_states nella documentazione di Microsoft Transact-SQL.
`sys.dm_os_host_info`	Per informazioni, consulta sys.dm_os_host_info nella documentazione di Microsoft Transact-SQL.
`sys.endpoints`	Per informazioni, consulta sys.endpoints nella documentazione di Microsoft Transact-SQL.
`sys.indexes`	Per informazioni, consulta sys.indexes nella documentazione di Microsoft Transact-SQL.
`sys.languages`	Per informazioni, consulta sys.languages nella documentazione di Microsoft Transact-SQL.
`sys.schemas`	Tutti gli schemi
`sys.server_principals`	Tutti gli accessi e i ruoli
`sys.sql_modules`	Per informazioni, consulta sys.sql_modules nella documentazione di Microsoft Transact-SQL.
`sys.sysconfigures`	Supporto Babelfish limitato a una singola configurazione di sola lettura.

Nome della vista	Descrizione o limitazione Babelfish (se presente)
`sys.syscurconfigs`	Supporto Babelfish limitato a una singola configurazione di sola lettura.
`sys.sysprocesses`	Per informazioni, consulta sys.sysprocesses nella documentazione di Microsoft Transact-SQL.
`sys.system_sql_modules`	Per informazioni, consulta sys.system_sql_modules nella documentazione di Microsoft Transact-SQL.
`sys.table_types`	Per informazioni, consulta sys.table_types nella documentazione di Microsoft Transact-SQL.
`sys.tables`	Tutte le tabelle in uno schema
`sys.xml_schema_collections`	Per informazioni, consulta sys.xml_schema_collections nella documentazione di Microsoft Transact-SQL.

PostgreSQL implementa cataloghi di sistema simili alle viste del catalogo oggetti di SQL Server. Per l'elenco completo dei cataloghi di sistema, consulta System Catalogs (Cataloghi di sistema) nella documentazione di PostgreSQL.

Esportazioni DDL supportate da Babelfish

A partire dalle versioni 2.4.0 e 3.1.0, Babelfish supporta le esportazioni DDL utilizzando vari strumenti. Ad esempio, puoi utilizzare questa funzionalità di SQL Server Management Studio (SSMS) per generare gli script di definizione dei dati per vari oggetti in un database Babelfish per Aurora PostgreSQL. Quindi puoi utilizzare i comandi DDL generati in questo script per creare gli stessi oggetti in un altro database Babelfish per Aurora PostgreSQL o SQL Server.

Babelfish supporta le esportazioni DDL per i seguenti oggetti nelle versioni specificate.

Elenco di oggetti	2.4.0	3.1.0
Tabelle dell'utente	Sì	Sì
Chiavi primarie	Sì	Sì
Chiavi esterne	Sì	Sì
Vincoli univoci	Sì	Sì
Indici	Sì	Sì
Vincoli check	Sì	Sì
Visualizzazioni	Sì	Sì
Stored procedure	Sì	Sì
Funzioni definite dall'utente	Sì	Sì
Funzioni con valori tabellari	Sì	Sì
Trigger	Sì	Sì
Tipi di dati definiti dall'utente	No	No
Tipi di tabella definiti dall'utente	No	No
Utenti	No	No
Accessi	No	No
Sequenze	No	No
Roles	No	No

Limitazioni con le DDL esportate

- Usa gli escape hatch prima di ricreare gli oggetti con le DDL esportate: Babelfish non supporta tutti i comandi nello script DDL esportato. Usa gli escape hatch per evitare errori causati durante

la ricreazione degli oggetti dai comandi DDL in Babelfish. Per ulteriori informazioni sugli escape hatch, consulta Gestione degli errori di Babelfish con escape hatch.

- Oggetti contenenti vincoli CHECK con clausole COLLATE esplicite: gli script con questi oggetti generati da un database SQL Server hanno regole di confronto diverse ma equivalenti a quelle del database Babelfish. Ad esempio, alcune raccolte, come sql_latin1_general_cp1_cs_as, sql_latin1_general_cp1251_cs_as e latin1_general_cs_as vengono generate come latin1_general_cs_as, che è la regola di confronto di Windows più vicina.

Gestione delle autorizzazioni e del controllo degli accessi in Babelfish per Aurora PostgreSQL

In Babelfish per Aurora PostgreSQL, è possibile gestire le autorizzazioni e il controllo degli accessi per database, schemi e oggetti. Le tabelle seguenti illustrano i comandi SQL specifici per la concessione delle autorizzazioni in Babelfish per realizzare vari scenari di controllo degli accessi. Sono inclusi i casi d'uso supportati che possono essere implementati, nonché le soluzioni alternative per i casi attualmente non supportati. Ciò consente di configurare le autorizzazioni appropriate per soddisfare i requisiti di sicurezza e conformità quando lavori con i database Babelfish.

Casi di utilizzo supportati

Nella seguente tabella sono descritti i casi d'uso supportati in Babelfish. Per ogni caso d'uso, la tabella mostra l'azione necessaria per ottenerlo e alcuni esempi di comandi SQL.

Caso d'uso	Azione	Comandi SQL	Commenti	Compatibilità della versione di Babelfish
Consentire l'accesso per eseguire SELECT/DML/DDL in qualsiasi database	Aggiungere l'accesso al ruolo del server sysadmin	ALTER SERVER ROLE sysadmin ADD MEMBER `login`	Nessuno	Tutte le versioni
Consentire l'accesso per eseguire	Rendere l'accesso	ALTER AUTHORIZATION ON	Un database può avere un solo proprietario.	Versione 3.4 e successive

Caso d'uso	Azione	Comandi SQL	Commenti	Compatibilità della versione di Babelfish
SELECT/DML/DDL in un database	proprietario del database	DATABASE::`database` TO `login`		
Consentire all'utente del database di eseguire SELECT/DML su uno schema	Concedere all'utente del database l'autorizzazione per lo schema	GRANT SELECT/EXECUTE/INSERT/UPDATE/DELETE ON SCHEMA::`schema` TO `user`	Nessuno	Versione 3.6 e successive, 4.2 e successive
Consentire all'utente del database di eseguire SELECT/DML su uno schema	Rendere l'utente del database proprietario dello schema al momento della creazione dello schema	CREATE SCHEMA `schema` AUTHORIZATION `user`	La modifica della proprietà dello schema dopo la creazione non è attualmente supportata.	Versione 1.2 e successive
Consentire all'utente del database di eseguire SELECT/DML su un oggetto	Concedere all'utente del database l'autorizzazione per l'oggetto	GRANT SELECT/EXECUTE/INSERT/UPDATE/DELETE ON OBJECT::`object` TO `user`	Nessuno	Tutte le versioni

Caso d'uso	Azione	Comandi SQL	Commenti	Compatibilità della versione di Babelfish
Consentire all'utente del database di eseguire SELECT/DML/DDL in un database, inclusa l'eliminazione del database	Aggiungere l'utente al ruolo fisso del database db_owner	ALTER ROLE db_owner ADD MEMBER `user`	Solo gli utenti possono essere aggiunti al ruolo fisso del database db_owner. L'aggiunta di ruoli al ruolo db_owner non è ancora supportata.	Versione 4.5 e successive, 5.1 e successive
Consentire all'utente o ai membri di un ruolo di database personalizzato di eseguire solo SELECT in un database	Aggiungere un utente o ruolo al ruolo fisso del database db_datareader	ALTER ROLE db_datareader ADD MEMBER `user` / `role`	Nessuno	Versione 4.5 e successive, 5.1 e successive
Consentire all'utente o ai membri di un ruolo di database personalizzato di eseguire solo DML in un database	Aggiungere un utente o ruolo al ruolo fisso del database db_datawriter	ALTER ROLE db_datawriter ADD MEMBER `user` / `role`	Nessuno	Versione 4.5 e successive, 5.1 e successive

Caso d'uso	Azione	Comandi SQL	Commenti	Compatibilità della versione di Babelfish
Consentire all'utente o ai membri di un ruolo di database personalizzato di eseguire solo DDL in un database	Aggiungere un utente o ruolo al ruolo fisso del database db_accessadmin	ALTER ROLE db_accessadmin ADD MEMBER `user` / `role`	Nessuno	Versione 4.5 e successive, 5.1 e successive
Consentire all'utente o ai membri di un ruolo di database personalizzato di eseguire solo CREATE/ALTER/DROP per i ruoli personalizzati, GRANT/REVOKE per le autorizzazioni degli oggetti in un database e/o CREATE SCHEMA in un database	Aggiungere un utente o ruolo al ruolo fisso del database db_securityadmin	ALTER ROLE db_securityadmin ADD MEMBER `user` / `role`	Nessuno	Versione 4.5 e successive, 5.1 e successive

Caso d'uso	Azione	Comandi SQL	Commenti	Compatibilità della versione di Babelfish
Consentire all'utente o ai membri di un ruolo di database personalizzato di eseguire solo CREATE/ALTER/DROP per qualsiasi utente, concedere e revocare l'accesso al database e mappare gli account utente agli accessi e/o CREATE SCHEMA in un database	Aggiungere un utente o ruolo al ruolo fisso del database db_accessadmin	ALTER ROLE db_accessadmin ADD MEMBER `user` / `role`	Nessuno	Versione 4.5 e successive, 5.1 e successive
Consentire l'accesso solo per eseguire CREATE/DROP/ALTER per qualsiasi database	Aggiungere l'accesso al ruolo fisso del server dbcreator	ALTER SERVER ROLE dbcreator ADD MEMBER `login`	Possono essere modificati solo i database a cui dbcreator ha accesso.	Versione 4.5 e successive, 5.1 e successive

Caso d'uso	Azione	Comandi SQL	Commenti	Compatibilità della versione di Babelfish
Consentire l’accesso solo per eseguire CREATE/ALTER/DROP per qualsiasi accesso	Aggiungere l’accesso al ruolo fisso del server securityadmin	ALTER SERVER ROLE securityadmin ADD MEMBER `login`	Nessuno	Versione 4.5 e successive, 5.1 e successive

Casi d’uso non supportati con soluzioni alternative

La tabella seguente spiega i casi d’uso che non sono supportati in Babelfish, ma che possono essere ottenuti utilizzando una soluzione alternativa.

Caso d'uso	Azione	Comandi SQL	Commenti	Compatibilità della versione Babelfish per soluzioni alternative
Consentire all’utente di eseguire SELECT/DML su un oggetto/schema insieme all’opzione GRANT per concedere queste autorizzazioni ad altri utenti	Eseguire GRANT per concedere le autorizzazioni direttamente a tutti gli altri utenti	GRANT SELECT/EXECUTE/INSERT/UPDATE/DELETE ON OBJECT/SCHEMA::`object/sc` TO `user`	GRANT ... WITH GRANT OPTION non è attualmente supportato.	Versione 3.6 e successive, 4.2 e successive

Caso d'uso	Azione	Comandi SQL	Commenti	Compatibilità della versione Babelfish per soluzioni alternative
Consentire all'utente del database di eseguire SELECT/DML/DDL in un database, inclusa l'eliminazione del database	Aggiungere membri al ruolo fisso del database db_owner	ALTER ROLE db_owner ADD MEMBER `user`	L'aggiunta di ruoli al ruolo db_owner non è attualmente supportata.	Versione 4.5 e successive, 5.1 e successive

Casi di utilizzo non supportati

Nella seguente tabella sono descritti i casi d'uso non supportati in Babelfish.

Caso d'uso	Commenti
Impedire all'utente o ai membri di un ruolo di database personalizzato di eseguire SELECT in un database	Il ruolo fisso del database db_denydatareader non è ancora supportato
Impedire all'utente o ai membri di un ruolo di database personalizzato di eseguire DML in un database	Il ruolo fisso del database db_denydatawriter non è attualmente supportato.
Consentire l'accesso solo per eseguire KILL su una connessione al database	Il ruolo fisso del server processadmin non è attualmente supportato.
Consentire l'accesso solo per aggiungere o rimuovere server collegati	Il ruolo fisso del server setupadmin non è attualmente supportato.

Risoluzione delle differenze di proprietà degli oggetti dopo l'aggiornamento

Babelfish versione 4.6 e successive e versione 5.2 e successive includono una modifica alla gestione della proprietà degli oggetti tramite l'endpoint TDS. Quando si creano nuovi oggetti tramite l'endpoint TDS, questi oggetti sono ora di proprietà del proprietario dello schema anziché dell'utente corrente. Questo cambio di proprietà potrebbe influire sul comportamento delle autorizzazioni per i nuovi oggetti rispetto agli oggetti esistenti quando si esegue l'aggiornamento da versioni precedenti alla 4.6 o 5.2.

Per risolvere queste differenze di proprietà, Babelfish fornisce la funzione `sys.generate_alter_ownership_statements()`. Questa funzione genera istruzioni SQL che allineano la proprietà degli oggetti alla proprietà dello schema.

Tieni presente le seguenti limitazioni relative alla proprietà degli oggetti:

- Gli utenti con autorizzazioni CREATE fornite tramite l'endpoint PostgreSQL non possono creare oggetti tramite l'endpoint TDS in tali schemi.
- La modifica delle autorizzazioni degli oggetti T-SQL tramite l'endpoint PostgreSQL non è consigliata e potrebbe causare un comportamento T-SQL non corretto.
- Le autorizzazioni di accesso potrebbero differire tra oggetti vecchi e nuovi a causa della mancata corrispondenza tra la proprietà degli oggetti. Ad esempio, considera uno schema di proprietà di `sch_own` che include oggetti di proprietà di `dbo`. In questo caso, gli oggetti di proprietà di `dbo` che sono stati creati prima dell'aggiornamento potrebbero avere autorizzazioni di accesso diverse rispetto agli oggetti di proprietà di `sch_own` creati dopo l'aggiornamento. Ciò può influire sulle operazioni come SELECT e INSERT.

Se il cluster di database include oggetti creati nelle versioni di Babelfish precedenti alla 4.6 o 5.2, valuta la possibilità di allinearne la proprietà.

Per risolvere le differenze di proprietà degli oggetti

1. Connettiti al database `babelfish_db` nel cluster di database utilizzando l'endpoint PostgreSQL.
2. Esegui il comando seguente:

   ```
   SELECT * from sys.generate_alter_ownership_statements();
   ```

   Questo comando genera un elenco di istruzioni SQL destinate a standardizzare la proprietà tra gli oggetti.

3. Esegui le istruzioni generate in un ambiente di test per convalidarne l'effetto prima di applicarle all'ambiente di produzione.

È consigliabile eseguire queste istruzioni per ottenere un modello di proprietà degli oggetti coerente in tutto il database.

Differenze tra Babelfish per Aurora PostgreSQL e SQL Server

Babelfish è una funzionalità di Aurora PostgreSQL in evoluzione, con nuove funzionalità aggiunte in ogni versione a partire dall'offerta iniziale in Aurora PostgreSQL 13.4. È progettata per fornire semantica T-SQL nella parte superiore di PostgreSQL, attraverso il dialetto T-SQL utilizzando la porta TDS. Ogni nuova versione di Babelfish aggiunge funzionalità e funzioni che si allineano meglio alle funzionalità e al comportamento di T-SQL, come mostrato nella tabella Funzionalità supportate in Babelfish per versione. Per ottenere risultati ottimali durante l'utilizzo di Babelfish, si consiglia di comprendere le differenze attualmente esistenti tra il T-SQL supportato da SQL Server e Babelfish per la versione più recente. Per ulteriori informazioni, consulta Differenze T-SQL in Babelfish.

Oltre alle differenze tra T-SQL supportato da Babelfish e SQL Server, potrebbe anche essere necessario considerare i problemi di interoperabilità tra Babelfish e PostgreSQL nel contesto del cluster database Aurora PostgreSQL. Come citato in precedenza, Babelfish supporta la semantica T-SQL nella parte superiore di PostgreSQL, attraverso il dialetto T-SQL utilizzando la porta TDS. Allo stesso tempo, è anche possibile accedere al database Babelfish tramite la porta PostgreSQL standard con istruzioni SQL PostgreSQL. Se si intende utilizzare entrambe le funzionalità di PostgreSQL e Babelfish in un'implementazione della produzione, occorre tenere presente i potenziali problemi di interoperabilità tra i nomi degli schemi, gli identificatori, le autorizzazioni, la semantica transazionale, i set di risultati multipli, le regole di fascicolazione predefinite e così via. In poche parole, quando le istruzioni PostgreSQL o l'accesso PostgreSQL si verificano nel contesto di Babelfish, si può verificare un'interferenza tra PostgreSQL e Babelfish che può potenzialmente influire su sintassi, semantica e compatibilità quando vengono rilasciate nuove versioni di Babelfish. Per informazioni complete e le linee guida su tutte le considerazioni, consulta Guidance on Babelfish Interoperability nella documentazione di Babelfish per PostgreSQL.

> **Note**
>
> Prima di utilizzare la funzionalità nativa di PostgreSQL e la funzionalità di Babelfish nello stesso contesto di applicazione, è opportuno considerare i problemi discussi in Guidance on Babelfish Interoperability nella documentazione di Babelfish per PostgreSQL. Questi problemi

di interoperabilità (Aurora PostgreSQL e Babelfish) sono pertinenti solo se si prevede di utilizzare l'istanza database PostgreSQL nello stesso contesto di applicazione di Babelfish.

Argomenti

- Dump e ripristino di Babelfish
- Differenze T-SQL in Babelfish
- Livelli di isolamento della transazione in Babelfish

Dump e ripristino di Babelfish

A partire dalle versioni 4.0.0 e 3.4.0, gli utenti Babelfish possono ora utilizzare le utilità di dump e ripristino per eseguire il backup e il ripristino dei propri database. Per ulteriori informazioni, consulta Dump e ripristino di Babelfish. Questa funzionalità si basa sulle utilità di dump e ripristino PostgreSQL. Per ulteriori informazioni, consulta pg_dump e pg_restore. Per utilizzare efficacemente questa funzionalità in Babelfish, è necessario utilizzare strumenti basati su PostgreSQL specificamente adattati per Babelfish. La funzionalità di backup e ripristino per Babelfish differisce notevolmente da quella di SQL Server. Per ulteriori informazioni su queste differenze, consulta Dump and restore functionality differences : Babelfish and SQL Server. Babelfish per Aurora PostgreSQL offre funzionalità aggiuntive per il backup e il ripristino dei cluster di database Amazon Aurora PostgreSQL. Per ulteriori informazioni, consulta Backup e ripristino di un cluster DB Amazon Aurora.

Differenze T-SQL in Babelfish

Di seguito, è possibile trovare una tabella delle funzionalità T-SQL supportate nella versione corrente di Babelfish con alcune note sulle differenze nel comportamento rispetto a quello di SQL Server.

Per ulteriori informazioni sul supporto delle diverse versioni, consulta Funzionalità supportate in Babelfish per versione. Per informazioni sulle funzionalità attualmente non supportate, consulta Funzionalità non supportate in Babelfish.

Babelfish è disponibile con Aurora PostgreSQL-Compatible Edition. Per ulteriori informazioni sulle versioni di Babelfish, consulta Release notes for Amazon Aurora PostgreSQL-Compatible Edition.

Funzionalità o sintassi	Descrizione del comportamento o della differenza
\ (carattere di continuazione della riga)	Il carattere di continuazione della riga (una barra rovesciata prima di una nuova riga) per stringhe di caratteri ed esadecima

Funzionalità o sintassi	Descrizione del comportamento o della differenza
	li non è attualmente supportato. Per le stringhe di caratteri, la barra rovesciata-nuova riga viene interpretata come caratteri nella stringa. Per le stringhe esadecimali, la barra rovesciata-nuova riga genera un errore di sintassi.
@version	Il formato del valore restituito da `@@version` è leggermente diverso dal valore restituito da SQL Server. Il codice potrebbe non funzionare correttamente se dipende dalla formattazione di `@@version` .
Funzioni di aggregazione	Le funzioni di aggregazione sono parzialmente supportate (sono supportate AVG, COUNT, COUNT_BIG, GROUPING, MAX, MIN, STRING_AGG e SUM). Per gli elenchi di funzionalità non supportate, vedere Funzioni non supportate.
ALTER TABLE	Supporta l'aggiunta o l'eliminazione di una sola colonna o di un solo vincolo.
ALTER TABLE..ALTER COLUMN	Al momento non è possibile specificare NULL e NOT NULL. Per modificare la nullabilità di una colonna, usa l'istruzione PostgreSQL ALTER TABLE..{SET|DROP} NOT NULL.

AT TIME ZONE	Durante il passaggio dall'ora legale (DST) all'ora solare, il periodo di sovrapposizione viene visualizzato utilizzando l'offset dell'ora solare. Per chiarire, considerare il seguente esempio:
	`SELECT CONVERT(DATETIME2(0), '2022-10-30T02:00:00', 126) AT TIME ZONE 'Central European Standard Time';`
	`GO;`
	`Result: 2022-10-30 02:00:00 +01:00`

Funzionalità o sintassi	Descrizione del comportamento o della differenza
Nomi di colonna vuoti senza alias di colonna	Le utility `sqlcmd` e `psql` gestiscono le colonne con nomi vuoti in modo diverso: • SQL Server `sqlcmd` restituisce un nome di colonna vuoto. • PostgreSQL `psql` restituisce un nome di colonna generato.
Funzione CHECKSUM	Babelfish e SQL Server utilizzano algoritmi di hashing diversi per la funzione CHECKSUM. Di conseguenza, i valori hash generati dalla funzione CHECKSUM in Babelfish potrebbero essere diversi da quelli generati dalla funzione CHECKSUM in SQL Server.
Impostazione predefinita colonna	Quando si crea una colonna predefinita, il nome del vincolo viene ignorato. Per eliminare una colonna predefinita, utilizza la sintassi seguente: `ALTER TABLE...ALTER COLUMN..DROP DEFAULT...`
Constraint_name	In SQL Server, i nomi dei vincoli devono essere univoci all'interno dello schema a cui appartiene la tabella. Tuttavia, in Babelfish, ciò si applica solo ai vincoli PRIMARY KEY e UNIQUE. Altri tipi di vincoli non sono soggetti a questa restrizione.
Vincoli	PostgreSQL non supporta l'attivazione e la disattivazione di singoli vincoli. L'istruzione DDL viene ignorata e viene emesso un avviso.
Vincoli con IGNORE_DUP_KEY	I vincoli vengono creati senza questa proprietà.

Funzionalità o sintassi	Descrizione del comportamento o della differenza
CREA, ALTERA, RILASCIA IL RUOLO DEL SERVER	RILASCIA IL RUOLO DEL SERVER è supportato solo per `sysadmin`. Tutta l'altra sintassi non è supportata. L'utente T-SQL in Babelfish ha un'esperienza simile a SQL Server per i concetti di accesso (principale del server), di un database e di un utente del database (principale del database).
Le clausole CREATE, ALTER LOGIN sono supportate con una sintassi limitata	CREATE LOGIN... Sono supportate la clausola PASSWORD, DEFAULT_DATABASE e DEFAULT_LANGUAGE. L'ALTER LOGIN... La clausola PASSWORD è supportata, ma ALTER LOGIN... La clausola OLD_PASSWORD non è supportata. Solo un login che è un membro di sysadmin può modificare una password.
Raccolta distinzione tra lettere maiuscole e minuscole CREATE DATABASE	Le raccolte con distinzione tra maiuscole e minuscole non sono supportate con l'affermazione CREATE DATABASE
Parole chiave e clausole CREATE DATABASE	Le opzioni tranne COLLATE e CONTAINMENT=NONE non sono supportate. La clausola COLLATE è accettata ed è sempre impostata sul valore di `babelfishpg_tsql.server_collation_name` .
Clausole di supporto CREATE SCHEMA…	È possibile utilizzare il comando CREATE SCHEMA per creare uno schema vuoto. Utilizzare comandi aggiuntivi per creare oggetti dello schema.
I valori ID del database sono diversi su Babelfish	Ai database master e tempdb non corrispondono gli ID database 1 e 2.

Funzionalità o sintassi	Descrizione del comportamento o della differenza
La funzione FORMAT del tipo di data è supportata con le seguenti limitazioni	Il meridiano a singolo carattere non è supportato. Il formato "yyy" in SQL Server restituisce 4 cifre per l'anno superiore a 1000, ma solo 3 cifre per gli altri. I formati "g" e "R" non sono supportati La traduzione locale "vi-VN" è leggermente diversa.
Identificatori superiori a 63 caratteri	PostgreSQL supporta un massimo di 63 caratteri per gli identific atori. Babelfish converte gli identificatori più lunghi di 63 caratteri in un nome che include un hash del nome originale. Ad esempio, una tabella creata come "AB(ABC12345678901234567890 12345678901234567890123456789012345678901234567890" potrebbe essere convertita in "ABC12345678901234567890123 45678901234567890123456789012345678901234567890".
Supporto delle colonne IDENTITY	Le colonne `IDENTITY` sono supportate per i tipi di dati `tinyint`, `smallint`, `int`, `bigint`. `numeric` e `decimal`. SQL Server supporta la precisione fino a 38 posizioni per i tipi di dati `numeric` e `decimal` nelle colonne IDENTITY. PostgreSQL supporta la precisione fino a 19 posizioni per i tipi di dati `numeric` e `decimal` nelle colonne IDENTITY.
Indici con IGNORE_DUP_KEY	La sintassi che crea un indice che include IGNORE_DUP_KEY crea un indice come se questa proprietà fosse omessa.
Indici con più di 32 colonne	Un indice non può includere più di 32 colonne. Le colonne di indice incluse contano fino al massimo in PostgreSQL ma non in SQL Server.
Indici (cluster)	Gli indici cluster vengono creati come se NON CLUSTERED fosse stato specificato.

Funzionalità o sintassi	Descrizione del comportamento o della differenza
Clausole Indice	Le seguenti clausole vengono ignorate: FILLFACTOR, ALLOW_PAGE_LOCKS, ALLOW_ROW_LOCKS, PAD_INDEX , STATISTICS_NORECOMPUTE, OPTIMIZE_FOR_SEQUE NTIAL_KEY, SORT_IN_TEMPDB, DROP_EXISTING, ONLINE, COMPRESSION_DELAY, MAXDOP e DATA_COMPRESSION
Supporto JSON	L'ordine delle coppie nome-valore non è garantito. Ma il tipo di array rimane inalterato.
Oggetti LOGIN	Non sono supportate tutte le opzioni per gli oggetti LOGIN, ad eccezione di PASSWORD, DEFAULT_DATABASE, DEFAULT_LANGUAGE, ENABLE, DISABLE.
La funzione NEWSEQUEN TIALID	Implementato come NEWID; il comportamento sequenziale non è garantito. Quando si richiama `NEWSEQUENTIALID` , PostgreSQL genera un nuovo valore GUID.
La clausola OUTPUT è supportata con le seguenti limitazioni	OUTPUT e OUTPUT INTO non sono supportati nella stessa query DML. I riferimenti alla tabella non di destinazione delle operazioni UPDATE o DELETE in una clausola OUTPUT non sono supportati. OUTPUT... DELETED *, INSERTED * non sono supportati nella stessa query.
Limite dei parametri di procedura o funzione	Babelfish supporta un massimo di 100 parametri per una procedura o una funzione.
ROWGUIDCOL	Al momento, questa clausola è ignorata. Le query che fanno riferimento a `$GUIDGOL` causano un errore di sintassi.
Supporto degli oggetti SEQUENCE	Gli oggetti SEQUENCE sono supportati per i tipi di dati tinyint, smallint, int, bigint, numerico e decimale. Aurora PostgreSQL supporta la precisione di 19 posizioni per i tipi di dati numerici e decimali in una SEQUENCE.

Funzionalità o sintassi	Descrizione del comportamento o della differenza
Ruoli a livello di server	Il ruolo a livello di server `sysadmin` è supportato. Altri ruoli a livello server (diversi da `sysadmin`) non sono supportati.
Ruoli a livello di database diversi da `db_owner`	Sono supportati i ruoli a livello di database `db_owner` e i ruoli a livello di database definiti dall'utente. Altri ruoli a livello server (diversi da db_owner) non sono supportati.
Parola chiave SQL `SPARSE`	La parola chiave SPARSE viene accettata e ignorata.
Clausola parola chiave SQL `ON filegroup`	Al momento, questa clausola è ignorata.
Parole chiave SQL `CLUSTERED` e `NONCLUSTERED` per indici e vincoli	Babelfish accetta e ignora le parole chiave `CLUSTERED` e `NONCLUSTERED`
`sysdatabases.cmptlevel`	`sysdatabases.cmptlevel` è sempre impostato su 120.
tempdb non viene reinizializzato al riavvio	Gli oggetti permanenti (come tabelle e procedure) creati in tempdb non vengono rimossi al riavvio del database.
Gruppo di file TEXTIMAGE_ON	Babelfish ignora la clausola `TEXTIMAGE_ON` *`filegroup`* .
Precisione temporale	Babelfish supporta la precisione a 6 cifre per secondi frazionari. Non sono previsti effetti avversi con questo comportamento.
Livelli di isolamento della transazione	READUNCOMMITTED è trattato allo stesso modo di READCOMMITTED.
Colonne computate virtuali (non persistenti)	Le colonne computate virtuali vengono create come persistenti.
Senza clausola SCHEMABINDING	Questa clausola non è supportata in funzioni, procedure, trigger o viste. L'oggetto viene creato, ma come se WITH SCHEMABINDING fosse stato specificato.

Livelli di isolamento della transazione in Babelfish

Babelfish supporta i livelli di isolamento delle transazioni `READ UNCOMMITTED`, `READ COMMITTED` e `SNAPSHOT`. A partire da Babelfish versione 3.4 sono supportati i livelli di isolamento aggiuntivi `REPEATABLE READ` e `SERIALIZABLE`. Tutti i livelli di isolamento in Babelfish sono supportati con il comportamento dei livelli di isolamento corrispondenti in PostgreSQL. SQL Server e Babelfish utilizzano diversi meccanismi sottostanti per implementare i livelli di isolamento delle transazioni (blocco per l'accesso simultaneo, blocchi eseguiti dalle transazioni, gestione degli errori e così via). Inoltre, ci sono alcune lievi differenze nel modo in cui l'accesso simultaneo può funzionare per carichi di lavoro diversi. Per ulteriori informazioni su questo comportamento di PostgreSQL, consulta Transaction Isolation.

Argomenti

- Panoramica dei livelli di isolamento delle transazioni
- Impostazione dei livelli di isolamento delle transazioni
- Abilitazione o disabilitazione dei livelli di isolamento delle transazioni
- Confronto tra i livelli di isolamento di Babelfish e SQL Server

Panoramica dei livelli di isolamento delle transazioni

I livelli originali di isolamento delle transazioni di SQL Server si basano su un blocco pessimistico in cui esiste solo una copia dei dati e le query devono bloccare le risorse, ad esempio le righe, prima di accedervi. Successivamente, è stata introdotta una variazione del livello di isolamento `READ COMMITTED`. Questa variazione consente l'uso di versioni di riga per fornire una migliore simultaneità tra letture e scritture utilizzando un accesso senza blocco. Inoltre, è disponibile un nuovo livello di isolamento chiamato `SNAPSHOT`. Utilizza inoltre le versioni di riga per fornire una simultaneità migliore rispetto al livello di isolamento `REPEATABLE READ`, evitando blocchi condivisi sui dati di lettura che vengono conservati fino alla fine della transazione.

A differenza di SQL Server, tutti i livelli di isolamento delle transazioni in Babelfish si basano sul blocco ottimistico (MVCC). Ogni transazione vede uno snapshot dei dati all'inizio dell'istruzione (`READ COMMITTED`) o all'inizio della transazione (`REPEATABLE READ`, `SERIALIZABLE`), indipendentemente dallo stato corrente dei dati sottostanti. Pertanto, il comportamento di esecuzione delle transazioni simultanee in Babelfish potrebbe differire da quello di SQL Server.

Ad esempio, si consideri una transazione con livello di isolamento `SERIALIZABLE` inizialmente bloccata in SQL Server ma che ha esito positivo in seguito. Potrebbe non riuscire in Babelfish a

causa di un conflitto di serializzazione con una transazione simultanea che legge o aggiorna le stesse righe. Potrebbero esserci anche casi in cui l'esecuzione di più transazioni simultanee produce un risultato finale diverso in Babelfish rispetto a SQL Server. Le applicazioni che utilizzano i livelli di isolamento devono essere accuratamente testate negli scenari di concorrenza.

Livelli di isolamento in SQL Server	Livello di isolamento di Babelfish	Livello di isolamento di PostgreSQL	Commenti
`READ UNCOMMITTED`	`READ UNCOMMITTED`	`READ UNCOMMITTED`	`READ UNCOMMITTED` è uguale a `READ COMMITTED` in Babelfish o PostgreSQL
`READ COMMITTED`	`READ COMMITTED`	`READ COMMITTED`	`READ COMMITTED` di SQL Server è basato sul blocco pessimistico, `READ COMMITTED` di Babelfish è basato su snapshot (MVCC).
`READ COMMITTED SNAPSHOT`	`READ COMMITTED`	`READ COMMITTED`	Entrambi sono basati su snapshot (MVCC) ma non sono esattamente uguali.
`SNAPSHOT`	`SNAPSHOT`	`REPEATABLE READ`	Esattamente uguali.
`REPEATABLE READ`	`REPEATABLE READ`	`REPEATABLE READ`	`REPEATABLE READ` di SQL Server è basato sul blocco pessimistico, `REPEATABLE READ` di Babelfish è basato su snapshot (MVCC).

Livelli di isolamento in SQL Server	Livello di isolamento di Babelfish	Livello di isolamento di PostgreSQL	Commenti
`SERIALIZABLE`	`SERIALIZABLE`	`SERIALIZABLE`	`SERIALIZABLE` di SQL Server è basato sull'isolamento pessimistico, `SERIALIZABLE` di Babelfish è basato su snapshot (MVCC).

> **Note**
>
> I suggerimenti della tabella non sono attualmente supportati e il comportamento è controllato utilizzando l'escape hatch predefinito di Babelfish `escape_hatch_table_hints`.

Impostazione dei livelli di isolamento delle transazioni

Il seguente comando consente di impostare il livello di isolamento delle transazioni:

Example

```
SET TRANSACTION ISOLATION LEVEL { READ UNCOMMITTED | READ COMMITTED | REPEATABLE READ |
 SNAPSHOT | SERIALIZABLE }
```

Abilitazione o disabilitazione dei livelli di isolamento delle transazioni

I livelli di isolamento delle transazioni `REPEATABLE READ` e `SERIALIZABLE` sono disabilitati per impostazione predefinita in Babelfish ed è necessario abilitarli esplicitamente impostando l'escape hatch `babelfishpg_tsql.isolation_level_serializable` o `babelfishpg_tsql.isolation_level_repeatable_read` su `pg_isolation` utilizzando `sp_babelfish_configure`. Per ulteriori informazioni, consulta Gestione degli errori di Babelfish con escape hatch.

Di seguito sono riportati alcuni esempi per abilitare o disabilitare l'uso di `REPEATABLE READ` e `SERIALIZABLE` nella sessione corrente impostando i rispettivi escape hatch. Facoltativamente, si include il parametro `server` per impostare l'escape hatch per la sessione corrente e per tutte le nuove sessioni successive.

Per abilitare l'uso di `SET TRANSACTION ISOLATION LEVEL REPEATABLE READ` solo nella sessione corrente.

Example

```
EXECUTE sp_babelfish_configure 'isolation_level_repeatable_read', 'pg_isolation'
```

Per abilitare l'uso di `SET TRANSACTION ISOLATION LEVEL REPEATABLE READ` nella sessione corrente e in tutte le nuove sessioni successive.

Example

```
EXECUTE sp_babelfish_configure 'isolation_level_repeatable_read', 'pg_isolation',
 'server'
```

Per disabilitare l'uso di `SET TRANSACTION ISOLATION LEVEL REPEATABLE READ` nella sessione corrente e nelle nuove sessioni successive.

Example

```
EXECUTE sp_babelfish_configure 'isolation_level_repeatable_read', 'off', 'server'
```

Per abilitare l'uso di `SET TRANSACTION ISOLATION LEVEL SERIALIZABLE` solo nella sessione corrente.

Example

```
EXECUTE sp_babelfish_configure 'isolation_level_serializable', 'pg_isolation'
```

Per abilitare l'uso di `SET TRANSACTION ISOLATION LEVEL SERIALIZABLE` nella sessione corrente e in tutte le nuove sessioni successive.

Example

```
EXECUTE sp_babelfish_configure 'isolation_level_serializable', 'pg_isolation', 'server'
```

Per disabilitare l'uso di `SET TRANSACTION ISOLATION LEVEL SERIALIZABLE` nella sessione corrente e nelle nuove sessioni successive.

Example

```
EXECUTE sp_babelfish_configure 'isolation_level_serializable', 'off', 'server'
```

Confronto tra i livelli di isolamento di Babelfish e SQL Server

Di seguito sono riportati alcuni esempi delle differenze nel modo in cui SQL Server e Babelfish implementano i livelli di isolamento ANSI.

> **Note**
>
> - I livelli di isolamento `REPEATABLE READ` e `SNAPSHOT` sono gli stessi in Babelfish.
> - I livelli di isolamento `READ UNCOMMITTED` e `READ COMMITTED` sono gli stessi in Babelfish.

L'esempio seguente mostra come creare la tabella di base per tutti gli esempi citati di seguito:

```
CREATE TABLE employee (
    id sys.INT NOT NULL PRIMARY KEY,
    name sys.VARCHAR(255)NOT NULL,
    age sys.INT NOT NULL
);
INSERT INTO employee (id, name, age) VALUES (1, 'A', 10);
INSERT INTO employee (id, name, age) VALUES (2, 'B', 20);
INSERT INTO employee (id, name, age) VALUES (3, 'C', 30);
```

Argomenti

- READ UNCOMMITTED di Babelfish rispetto al livello di isolamento READ UNCOMMITTED di SQL Server

- READ COMMITTED di Babelfish rispetto al livello di isolamento READ COMMITTED di SQL Server
- READ COMMITTED di Babelfish rispetto al livello di isolamento READ COMMITTED SNAPSHOT di SQL Server
- REPEATABLE READ di Babelfish rispetto al livello di isolamento REPEATABLE READ di SQL Server
- SERIALIZABLE di Babelfish rispetto al livello di isolamento SERIALIZABLE di SQL Server

`READ UNCOMMITTED` di Babelfish rispetto al livello di isolamento **`READ UNCOMMITTED`** di SQL Server

La tabella seguente fornisce dettagli sulle letture dirty quando vengono eseguite transazioni simultanee. Mostra i risultati osservati quando si utilizza il livello di isolamento `READ UNCOMMITTED` in SQL Server rispetto all'implementazione Babelfish.

Transazione 1	Transazione 2	**`READ UNCOMMITTED`** di SQL Server	**`READ UNCOMMITTED`** di Babelfish
`BEGIN TRANSACTION`	`BEGIN TRANSACTION`	Nessuno	Nessuna
`SET TRANSACTION ISOLATION LEVEL READ UNCOMMITTED;`	`SET TRANSACTION ISOLATION LEVEL READ UNCOMMITTED;`	Nessuna	Nessuno
Inattivo in transazione	`UPDATE employee SET age=0;`	Aggiornamento completato.	Aggiornamento completato.
Inattivo in transazione	`INSERT INTO employee VALUES (4, 'D', 40);`	Inserimento completato.	Inserimento completato.
`SELECT * FROM employee;`	Inattivo in transazione	La transazione 1 può visualizzare le modifiche di cui non è stato eseguito il	Come `READ COMMITTED` in Babelfish. Le modifiche di cui non

Transazione 1	Transazione 2	**READ UNCOMMITTED** di SQL Server	**READ UNCOMMITTED** di Babelfish
		commit dalla transazione 2.	è stato eseguito il commit dalla transazione 2 non sono visibili nella transazione 1.
Inattivo in transazione	`COMMIT`	Nessuno	Nessuno
`SELECT * FROM employee;`	Inattivo in transazione	Vede le modifiche di cui è stato eseguito il commit dalla transazione 2.	Vede le modifiche di cui è stato eseguito il commit dalla transazione 2.

`READ COMMITTED` di Babelfish rispetto al livello di isolamento **`READ COMMITTED`** di SQL Server

La tabella seguente fornisce dettagli sul comportamento del blocco di lettura-scrittura quando vengono eseguite transazioni simultanee. Mostra i risultati osservati quando si utilizza il livello di isolamento `READ COMMITTED` in SQL Server rispetto all'implementazione Babelfish.

Transazione 1	Transazione 2	**READ COMMITTED** di SQL Server	**READ COMMITTED** di Babelfish
`BEGIN TRANSACTION`	`BEGIN TRANSACTION`	Nessuno	Nessuna
`SET TRANSACTION ISOLATION LEVEL READ COMMITTED;`	`SET TRANSACTION ISOLATION LEVEL READ COMMITTED;`	Nessuna	Nessuno
`SELECT * FROM employee;`	Inattivo in transazione	Nessuno	Nessuno
Inattivo in transazione	`UPDATE employee SET age=100 WHERE id = 1;`	Aggiornamento completato.	Aggiornamento completato.

Transazione 1	Transazione 2	**READ COMMITTED** di SQL Server	**READ COMMITTED** di Babelfish
`UPDATE employee SET age = 0 WHERE age IN (SELECT MAX(age) FROM employee);`	Inattivo in transazione	Passaggio bloccato fino al commit della transazione 2.	Le modifiche alla transazione 2 non sono ancora visibili. Aggiorna la riga con id=3.
Inattivo in transazione	`COMMIT`	La transazione 2 viene correttamente sottoposta al commit. La transazione 1 è ora sbloccata e vede l'aggiornamento dalla transazione 2.	La transazione 2 viene correttamente sottoposta al commit.
`SELECT * FROM employee;`	Inattivo in transazione	La transazione 1 aggiorna la riga con id = 1.	La transazione 1 aggiorna la riga con id = 3.

`READ COMMITTED` di Babelfish rispetto al livello di isolamento **`READ COMMITTED SNAPSHOT`** di SQL Server

La tabella seguente fornisce dettagli sul comportamento del blocco delle righe appena inserite quando vengono eseguite transazioni simultanee. Mostra i risultati osservati quando si utilizza il livello di isolamento `READ COMMITTED SNAPSHOT` in SQL Server rispetto all'implementazione `READ COMMITTED` di Babelfish.

Transazione 1	Transazione 2	**READ COMMITTED SNAPSHOT** di SQL Server	**READ COMMITTED** di Babelfish
`BEGIN TRANSACTION`	`BEGIN TRANSACTION`	Nessuno	Nessuna

Transazione 1	Transazione 2	**READ COMMITTED SNAPSHOT** di SQL Server	**READ COMMITTED** di Babelfish
`SET TRANSACTION ISOLATION LEVEL READ COMMITTED;`	`SET TRANSACTION ISOLATION LEVEL READ COMMITTED;`	Nessuna	Nessuno
`INSERT INTO employee VALUES (4, 'D', 40);`	Inattivo in transazione	Nessuno	Nessuno
Inattivo in transazione	`UPDATE employee SET age = 99;`	Il passaggio è bloccato fino al commit della transazione 1. La riga inserita è bloccata dalla transazione 1.	Sono state aggiornate tre righe. La riga appena inserita non è ancora visibile.
`COMMIT`	Inattivo in transazione	Commit completato. La transazione 2 è ora sbloccata.	Commit completato.
Inattivo in transazione	`SELECT * FROM employee;`	Tutte e 4 le righe hanno age=99.	La riga con id = 4 ha il valore di age 40 perché non era visibile alla transazione 2 durante la query di aggiornamento. Le altre righe vengono aggiornate ad age=99.

`REPEATABLE READ` di Babelfish rispetto al livello di isolamento **`REPEATABLE READ`** di SQL Server

La tabella seguente fornisce dettagli sul comportamento del blocco di lettura-scrittura quando vengono eseguite transazioni simultanee. Mostra i risultati osservati quando si utilizza il livello di

isolamento `REPEATABLE READ` in SQL Server rispetto all'implementazione `REPEATABLE READ` di Babelfish.

Transazione 1	Transazione 2	**`REPEATABLE READ`** di SQL Server	**`REPEATABLE READ`** di Babelfish
`BEGIN TRANSACTION`	`BEGIN TRANSACTION`	Nessuno	Nessuna
`SET TRANSACTION ISOLATION LEVEL REPEATABLE READ;`	`SET TRANSACTION ISOLATION LEVEL REPEATABLE READ;`	Nessuna	Nessuno
`SELECT * FROM employee;`	Inattivo in transazione	Nessuno	Nessuno
`UPDATE employee SET name='A_TXN1' WHERE id=1;`	Inattivo in transazione	Nessuno	Nessuno
Inattivo in transazione	`SELECT * FROM employee WHERE id != 1;`	Nessuno	Nessuno
Inattivo in transazione	`SELECT * FROM employee;`	La transazione 2 è bloccata fino al commit della transazione 1.	La transazione 2 procede normalmente.
`COMMIT`	Inattivo in transazione	Nessuno	Nessuno
Inattivo in transazione	`SELECT * FROM employee;`	L'aggiornamento dalla transazione 1 è visibile.	L'aggiornamento dalla transazione 1 non è visibile.
`COMMIT`	Inattivo in transazione	Nessuno	Nessuno

Transazione 1	Transazione 2	**REPEATABLE READ** di SQL Server	**REPEATABLE READ** di Babelfish
Inattivo in transazione	`SELECT * FROM employee;`	Vede l'aggiornamento dalla transazione 1.	Vede l'aggiornamento dalla transazione 1.

La tabella seguente fornisce dettagli sul comportamento del blocco di scrittura-scrittura quando vengono eseguite transazioni simultanee. Mostra i risultati osservati quando si utilizza il livello di isolamento `REPEATABLE READ` in SQL Server rispetto all'implementazione `REPEATABLE READ` di Babelfish.

Transazione 1	Transazione 2	**REPEATABLE READ** di SQL Server	**REPEATABLE READ** di Babelfish
`BEGIN TRANSACTION`	`BEGIN TRANSACTION`	Nessuno	Nessuna
`SET TRANSACTION ISOLATION LEVEL REPEATABLE READ;`	`SET TRANSACTION ISOLATION LEVEL REPEATABLE READ;`	Nessuna	Nessuno
`UPDATE employee SET name='A_TXN1' WHERE id=1;`	Inattivo in transazione	Nessuno	Nessuno
Inattivo in transazione	`UPDATE employee SET name='A_TXN2' WHERE id=1;`	La transazione 2 è bloccata.	La transazione 2 è bloccata.
`COMMIT`	Inattivo in transazione	Il commit è stato completato e la transazione 2 è stata sbloccata.	Il commit è stato completato e la transazione 2 non riesce con l'errore indicante la mancata

Transazione 1	Transazione 2	**REPEATABLE READ** di SQL Server	**REPEATABLE READ** di Babelfish
			serializzazione dell'accesso a causa di un aggiornamento simultaneo.
Inattivo in transazione	`COMMIT`	Commit completato.	La transazione 2 è già stata interrotta.
Inattivo in transazione	`SELECT * FROM employee;`	La riga con id=1 ha name='A_TX2'.	La riga con id=1 ha name='A_TX1'.

La tabella seguente fornisce dettagli sul comportamento delle letture fantasma quando vengono eseguite transazioni simultanee. Mostra i risultati osservati quando si utilizza il livello di isolamento `REPEATABLE READ` in SQL Server rispetto all'implementazione `REPEATABLE READ` di Babelfish.

Transazione 1	Transazione 2	**REPEATABLE READ** di SQL Server	**REPEATABLE READ** di Babelfish
`BEGIN TRANSACTION`	`BEGIN TRANSACTION`	Nessuno	Nessuna
`SET TRANSACTION ISOLATION LEVEL REPEATABLE READ;`	`SET TRANSACTION ISOLATION LEVEL REPEATABLE READ;`	Nessuna	Nessuno
`SELECT * FROM employee;`	Inattivo in transazione	Nessuno	Nessuno
Inattivo in transazione	`INSERT INTO employee VALUES (4, 'NewRowName', 20);`	La transazione 2 procede senza blocchi.	La transazione 2 procede senza blocchi.

Transazione 1	Transazione 2	**REPEATABLE READ** di SQL Server	**REPEATABLE READ** di Babelfish
Inattivo in transazione	`SELECT * FROM employee;`	La riga appena inserita è visibile.	La riga appena inserita è visibile.
Inattivo in transazione	`COMMIT`	Nessuno	Nessuno
`SELECT * FROM employee;`	Inattivo in transazione	La nuova riga inserita dalla transazione 2 è visibile.	La nuova riga inserita dalla transazione 2 non è visibile.
`COMMIT`	Inattivo in transazione	Nessuno	Nessuno
`SELECT * FROM employee;`	Inattivo in transazione	La riga appena inserita è visibile.	La riga appena inserita è visibile.

La tabella seguente fornisce dettagli sull'esecuzione di transazioni simultanee e sui diversi risultati finali quando si utilizza il livello di isolamento `REPEATABLE READ` in SQL Server rispetto all'implementazione `REPEATABLE READ` di Babelfish.

Transazione 1	Transazione 2	**REPEATABLE READ** di SQL Server	**REPEATABLE READ** di Babelfish
`BEGIN TRANSACTION`	`BEGIN TRANSACTION`	Nessuno	Nessuna
`SET TRANSACTION ISOLATION LEVEL REPEATABLE READ;`	`SET TRANSACTION ISOLATION LEVEL REPEATABLE READ;`	Nessuna	Nessuno
`UPDATE employee SET age = 100 WHERE age IN (SELECT MIN(age) FROM employee);`	Inattivo in transazione	La transazione 1 aggiorna la riga con id 1.	La transazione 1 aggiorna la riga con id 1.

Transazione 1	Transazione 2	**REPEATABLE READ** di SQL Server	**REPEATABLE READ** di Babelfish
Inattivo in transazione	`UPDATE employee SET age = 0 WHERE age IN (SELECT MAX(age) FROM employee);`	La transazione 2 è bloccata perché l'istruzione SELECT tenta di leggere le righe bloccate dalla query UPDATE nella transazione 1.	La transazione 2 procede senza blocchi perché la lettura non viene mai bloccata, l'istruzione SELECT viene eseguita e la riga con id = 3 viene aggiornata perché le modifiche alla transazione 1 non sono ancora visibili.
Inattivo in transazione	`SELECT * FROM employee;`	Questo passaggio viene eseguito dopo il commit della transazione 1. La riga con id = 1 viene aggiornata dalla transazione 2 nel passaggio precedente ed è visibile.	La riga con id = 3 viene aggiornata dalla transazione 2.
`COMMIT`	Inattivo in transazione	La transazione 2 è ora sbloccata.	Commit completato.
Inattivo in transazione	`COMMIT`	Nessuno	Nessuno
`SELECT * FROM employee;`	Inattivo in transazione	Entrambe le transazioni eseguono l'aggiornamento sulla riga con id = 1.	Le diverse righe vengono aggiornate dalle transazioni 1 e 2.

`SERIALIZABLE` di Babelfish rispetto al livello di isolamento **`SERIALIZABLE`** di SQL Server

La tabella seguente fornisce dettagli sui blocchi di intervallo quando vengono eseguite transazioni simultanee. Mostra i risultati osservati quando si utilizza il livello di isolamento `SERIALIZABLE` in SQL Server rispetto all'implementazione `SERIALIZABLE` di Babelfish.

Transazione 1	Transazione 2	**`SERIALIZABLE`** di SQL Server	**`SERIALIZABLE`** di Babelfish
`BEGIN TRANSACTION`	`BEGIN TRANSACTION`	Nessuno	Nessuna
`SET TRANSACTION ISOLATION LEVEL SERILAIZABLE;`	`SET TRANSACTION ISOLATION LEVEL SERILAIZABLE;`	Nessuna	Nessuno
`SELECT * FROM employee;`	Inattivo in transazione	Nessuno	Nessuno
Inattivo in transazione	`INSERT INTO employee VALUES (4, 'D', 35);`	La transazione 2 è bloccata fino al commit della transazione 1.	La transazione 2 procede senza blocchi.
Inattivo in transazione	`SELECT * FROM employee;`	Nessuno	Nessuno
`COMMIT`	Inattivo in transazione	La transazione 1 viene correttamente sottoposta al commit. La transazione 2 è ora sbloccata.	La transazione 1 viene correttamente sottoposta al commit.
Inattivo in transazione	`COMMIT`	Nessuno	Nessuno
`SELECT * FROM employee;`	Inattivo in transazione	La riga appena inserita è visibile.	La riga appena inserita è visibile.

La tabella seguente fornisce dettagli sull'esecuzione di transazioni simultanee e sui diversi risultati finali quando si utilizza il livello di isolamento `SERIALIZABLE` in SQL Server rispetto all'implementazione `SERIALIZABLE` di Babelfish.

Transazione 1	Transazione 2	**`SERIALIZABLE`** di SQL Server	**`SERIALIZABLE`** di Babelfish
`BEGIN TRANSACTION`	`BEGIN TRANSACTION`	Nessuno	Nessuna
`SET TRANSACTION ISOLATION LEVEL SERILAIZABLE;`	`SET TRANSACTION ISOLATION LEVEL SERILAIZABLE;`	Nessuna	Nessuno
Inattivo in transazione	`INSERT INTO employee VALUES (4, 'D', 40);`	Nessuno	Nessuno
`UPDATE employee SET age =99 WHERE id = 4;`	Inattivo in transazione	La transazione 1 è bloccata fino al commit della transazione 2.	La transazione 1 procede senza blocchi.
Inattivo in transazione	`COMMIT`	La transazione 2 viene correttamente sottoposta al commit. La transazione 1 è ora sbloccata.	La transazione 2 viene correttamente sottoposta al commit.
`COMMIT`	Inattivo in transazione	Nessuno	Nessuno
`SELECT * FROM employee;`	Inattivo in transazione	La riga appena inserita è visibile con valore age = 99.	La riga appena inserita è visibile con valore age = 40.

La tabella seguente fornisce i dettagli relativi all'azione `INSERT` in una tabella con vincolo univoco. Mostra i risultati osservati quando si utilizza il livello di isolamento `SERIALIZABLE` in SQL Server rispetto all'implementazione `SERIALIZABLE` di Babelfish.

Transazione 1	Transazione 2	**SERIALIZABLE** di SQL Server	**SERIALIZABLE** di Babelfish
`BEGIN TRANSACTION`	`BEGIN TRANSACTION`	Nessuno	Nessuna
`SET TRANSACTION ISOLATION LEVEL SERILAIZABLE;`	`SET TRANSACTION ISOLATION LEVEL SERILAIZABLE;`	Nessuna	Nessuno
Inattivo in transazione	`INSERT INTO employee VALUES (4, 'D', 40);`	Nessuno	Nessuno
`INSERT INTO employee VALUES ((SELECT MAX(id)+1 FROM employee), 'E', 50);`	Inattivo in transazione	La transazione 1 è bloccata fino al commit della transazione 2.	La transazione 1 è bloccata fino al commit della transazione 2.
Inattivo in transazione	`COMMIT`	La transazione 2 viene correttamente sottoposta al commit. La transazione 1 è ora sbloccata.	La transazione 2 viene correttamente sottoposta al commit. La transazione 1 viene interrotta con errore di valore di chiave duplicato che viola un vincolo univoco.
`COMMIT`	Inattivo in transazione	La transazione 1 viene correttamente sottoposta al commit.	Il commit della transazione 1 ha esito negativo e non è stato possibile serializzare l'accesso a causa di dipendenze di

Transazione 1	Transazione 2	**SERIALIZABLE** di SQL Server	**SERIALIZABLE** di Babelfish
			lettura o scrittura tra le transazioni.
`SELECT * FROM employee;`	Inattivo in transazione	Viene inserita la riga (5, 'E', 50).	Esistono solo 4 righe.

In Babelfish, le transazioni simultanee eseguite con livello di isolamento serializzabile restituiscono un errore di anomalia di serializzazione se l'esecuzione di queste transazioni non è coerente con tutte le possibili esecuzioni seriali (una alla volta) di tali transazioni.

Le tabelle seguenti forniscono dettagli sull'anomalia di serializzazione quando vengono eseguite transazioni simultanee. Mostra i risultati osservati quando si utilizza il livello di isolamento `SERIALIZABLE` in SQL Server rispetto all'implementazione `SERIALIZABLE` di Babelfish.

Transazione 1	Transazione 2	**SERIALIZABLE** di SQL Server	**SERIALIZABLE** di Babelfish
`BEGIN TRANSACTION`	`BEGIN TRANSACTION`	Nessuno	Nessuna
`SET TRANSACTION ISOLATION LEVEL SERILAIZABLE;`	`SET TRANSACTION ISOLATION LEVEL SERILAIZABLE;`	Nessuna	Nessuno
`SELECT * FROM employee;`	Inattivo in transazione	Nessuno	Nessuno
`UPDATE employee SET age=5 WHERE age=10;`	Inattivo in transazione	Nessuno	Nessuno
Inattivo in transazione	`SELECT * FROM employee;`	La transazione 2 è bloccata fino al commit della transazione 1.	La transazione 2 procede senza blocchi.

Transazione 1	Transazione 2	**SERIALIZABLE** di SQL Server	**SERIALIZABLE** di Babelfish
Inattivo in transazione	`UPDATE employee SET age=35 WHERE age=30;`	Nessuno	Nessuno
`COMMIT`	Inattivo in transazione	La transazione 1 viene correttamente sottoposta al commit.	La transazione 1 viene sottoposta al commit per prima ed è in grado di completar lo con successo.
Inattivo in transazione	`COMMIT`	La transazione 2 viene correttamente sottoposta al commit.	Il commit della transazione 2 non riesce a causa di un errore di serializz azione, l'intera transazione è stata sottoposta a rollback. Riprova la transazione 2.
`SELECT * FROM employee;`	Inattivo in transazione	Le modifiche di entrambe le transazio ni sono visibili.	La transazione 2 è stata sottoposta a rollback. Vengono visualizzate solo le modifiche alla transazione 1.

In Babelfish, l'anomalia di serializzazione si verifica solo se tutte le transazioni simultanee vengono eseguite a livello di isolamento `SERIALIZABLE`. Nella tabella seguente è riportato l'esempio precedente con la transazione 2 impostata sul livello di isolamento `REPEATABLE READ`.

Transazione 1	Transazione 2	Livelli di isolamento di SQL Server	Livelli di isolamento di Babelfish
`BEGIN TRANSACTION`	`BEGIN TRANSACTION`	Nessuno	Nessuna
`SET TRANSACTION ISOLATION LEVEL SERILAIZABLE;`	`SET TRANSACTION ISOLATION LEVEL REPEATABLE READ;`	Nessuna	Nessuno
`SELECT * FROM employee;`	Inattivo in transazione	Nessuno	Nessuno
`UPDATE employee SET age=5 WHERE age=10;`	Inattivo in transazione	Nessuno	Nessuno
Inattivo in transazione	`SELECT * FROM employee;`	La transazione 2 è bloccata fino al commit della transazione 1.	La transazione 2 procede senza blocchi.
Inattivo in transazione	`UPDATE employee SET age=35 WHERE age=30;`	Nessuno	Nessuno
`COMMIT`	Inattivo in transazione	La transazione 1 viene correttamente sottoposta al commit.	La transazione 1 viene correttamente sottoposta al commit.
Inattivo in transazione	`COMMIT`	La transazione 2 viene correttamente sottoposta al commit.	La transazione 2 viene correttamente sottoposta al commit.
`SELECT * FROM employee;`	Inattivo in transazione	Le modifiche di entrambe le transazioni sono visibili.	Le modifiche di entrambe le transazioni sono visibili.

Utilizzo di funzionalità Babelfish con implementazione limitata

Ogni nuova versione di Babelfish aggiunge il supporto per ulteriori funzionalità che si allineano meglio alle funzionalità e al comportamento di T-SQL. Tuttavia, ci sono alcune caratteristiche e differenze non supportate nell'implementazione corrente. Di seguito, puoi trovare informazioni sulle differenze funzionali tra Babelfish e T-SQL, con alcune soluzioni alternative o note di utilizzo.

A partire dalla versione 1.2.0 di Babelfish, le seguenti funzionalità attualmente presentano implementazioni limitate:

- Cataloghi SQL Server (visualizzazioni di sistema) - I cataloghi `sys.sysconfigures`,`sys.syscurconfigs` e `sys.configurations`supportano solo una singola configurazione di sola lettura. `sp_configure` non è attualmente supportato. Per ulteriori informazioni sulle altre viste di SQL Server implementate da Babelfish, consulta Recupero di informazioni dal catalogo di sistema Babelfish.
- Autorizzazioni GRANT - GRANT...TO PUBLIC è supportato, ma GRANT..TO PUBLIC WITH GRANT OPTION non è attualmente supportato.
- Limitazioni sulla catena di proprietà e sul meccanismo di autorizzazione di SQL Server - In Babelfish, la catena di proprietà di SQL Server funziona per le viste ma non per le stored procedure. Ciò significa che alle procedure deve essere concesso un accesso esplicito ad altri oggetti di proprietà dello stesso proprietario delle procedure chiamanti. In SQL Server, concedere al chiamante le autorizzazioni EXECUTE sulla procedura è sufficiente per invocare altri oggetti di proprietà dello stesso proprietario. In Babelfish, al chiamante devono inoltre essere concesse le autorizzazioni sugli oggetti a cui si accede dalla procedura.
- Risoluzione di riferimenti agli oggetti non qualificati (senza nome dello schema) - Quando un oggetto SQL (procedura, vista, funzione o trigger) fa riferimento a un oggetto senza qualificarlo con un nome di schema, SQL Server risolve il nome dello schema dell'oggetto utilizzando il nome dello schema dell'oggetto SQL in cui si verifica il riferimento. Attualmente, Babelfish risolve questo problema in modo diverso, utilizzando lo schema predefinito dell'utente del database che esegue la procedura.
- Modifiche di schema, sessioni e connessioni predefinite - Se gli utenti modificano lo schema predefinito con`ALTER USER...WITH DEFAULT SCHEMA` , la modifica diventa immediatamente effettiva in quella sessione. Tuttavia, per altre sessioni attualmente connesse appartenenti allo stesso utente, la tempistica è diversa, come segue:
  - Per SQL Server: la modifica ha effetto su tutte le altre connessioni per questo utente immediatamente.

  - Per Babelfish: la modifica ha effetto solo per questo utente per le nuove connessioni.
- Implementazione dei tipi di dati ROWVERSION e TIMESTAMP e impostazione di escape hatch - I tipi di dati ROWVERSION e TIMESTAMP sono ora supportati in Babelfish. Per utilizzare ROWVERSION o TIMESTAMP in Babelfish, è necessario modificare l'impostazione dell'escape hatch `babelfishpg_tsql.escape_hatch_rowversion` dal suo valore predefinito (strict) a `ignore`. L'implementazione Babelfish dei tipi di dati ROWVERSION e TIMESTAMP è per lo più semanticamente identica a quella di SQL Server, con le seguenti eccezioni:
  - La funzione @@DBTS integrata si comporta in modo simile a SQL Server, ma con piccole differenze. Invece di restituire l'ultimo valore utilizzato per `SELECT @@DBTS`, Babelfish genera un nuovo timestamp, a causa del motore di database PostgreSQL sottostante e della sua implementazione del controllo della concorrenza multiversione (MVCC).
  - In SQL Server, ogni riga inserita o aggiornata ottiene un ROWVERSION/TIMESTAMP value. In Babelfish, every inserted row updated by the same statement is assigned the same ROWVERSION/TIMESTAMP valore univoco.

    Ad esempio, quando un'istruzione UPDATE o un'istruzione INSERT-SELECT influisce su più righe, in SQL Server, le righe interessate hanno tutte valori diversi nella colonna. ROWVERSION/TIMESTAMP In Babelfish (PostgreSQL), le righe hanno lo stesso valore.
  - In SQL Server, quando si crea una nuova tabella con SELECT-INTO, è possibile assegnare un valore esplicito (come NULL) a un to-be-created ROWVERSION/TIMESTAMP column. When you do the same thing in Babelfish, an actual ROWVERSION/TIMESTAMP valore assegnato da Babelfish a ogni riga della nuova tabella.

  Queste piccole differenze nei ROWVERSION/TIMESTAMP tipi di dati non dovrebbero avere un impatto negativo sulle applicazioni in esecuzione su Babelfish.
- Clausola TOP N PERCENT — Babelfish fornisce supporto per la clausola TOP N PERCENT con alcune limitazioni. Le operazioni SELECT sono supportate mentre le operazioni UPDATE, DELETE e INSERT con TOP N PERCENT non sono supportate. Inoltre, l'opzione WITH TIES e le sottoquery all'interno della clausola TOP non sono supportate. Quando il valore dell'espressione supera 100, il comportamento è diverso:
  - Per SQL Server: genera un errore.
  - Per Babelfish: considera il valore come valido e restituisce risultati.
- Creazione, proprietà e autorizzazioni dello schema - Le autorizzazioni per creare e accedere a oggetti in uno schema di proprietà di un utente non DBO (utilizzando `CREATE SCHEMA` *`schema`*

name AUTHORIZATION *user name*) differiscono per gli utenti SQL Server e Babelfish non DBO, come illustrato nella seguente tabella:

L'utente del database (non DBO) proprietario dello schema può eseguire le seguenti operazioni:	SQL Server	Babelfish
Creare oggetti nello schema senza ulteriori autorizzazioni da parte del DBO?	No	Sì
Accedere a oggetti creati dal DBO nello schema senza ulteriori autorizzazioni?	Sì	No

- Sintassi CREATE OR ALTER VIEW/ALTER VIEW: il supporto di questa sintassi in Babelfish presenta le seguenti limitazioni:
  - Queste istruzioni non possono essere utilizzate su viste a cui è collegato un trigger INSTEAD-OF.
  - Queste istruzioni non possono essere utilizzate su viste che hanno un'altra vista basata su questa vista.

Miglioramento delle prestazioni della query Babelfish

Puoi velocizzare l'elaborazione delle query in Babelfish utilizzando i suggerimenti per le query e il sistema di ottimizzazione PostgreSQL.

Argomenti

- Utilizzo del piano explain per migliorare le prestazioni delle query Babelfish
- Utilizzo di suggerimenti per le query T-SQL per migliorare le prestazioni delle query Babelfish

Puoi anche migliorare le prestazioni delle query utilizzando la procedura `sp_babelfish_volatility`. Per ulteriori informazioni, consulta sp_babelfish_volatility.

Puoi migliorare le prestazioni delle query anche utilizzando la trasformazione delle sottoquery e la cache delle sottoquery. Per ulteriori informazioni, consulta Ottimizzazione delle sottoquery correlate in Aurora PostgreSQL.

Utilizzo del piano explain per migliorare le prestazioni delle query Babelfish

A partire dalla versione 2.1.0, Babelfish include due funzioni che utilizzano in modo trasparente l'ottimizzatore PostgreSQL per generare piani di query effettivi per query T-SQL sulla porta TDS. Queste funzioni sono simili all'utilizzo di SET STATISTICS PROFILE o SET SHOWPLAN_ALL con i database SQL Server per identificare e migliorare le query a esecuzione lenta.

> **Note**
>
> Il recupero di piani di query da funzioni, flussi di controllo e cursori non è attualmente supportato.

Nella tabella è disponibile un confronto tra le funzioni explain del piano di query in SQL Server, Babelfish e PostgreSQL.

SQL Server	Babelfish	PostgreSQL
SHOWPLAN_ALL	BABELFISH_SHOWPLAN_ALL	EXPLAIN
STATISTICS PROFILE	BABELFISH_STATISTICS PROFILE	EXPLAIN ANALYZE
Utilizza l'ottimizzatore SQL Server	Utilizza l'ottimizzatore PostgreSQL	Utilizza l'ottimizzatore PostgreSQL
Formato di input e output di SQL Server	Formato di input e output di SQL Server	Formato di input e output di PostgreSQL
Impostato per la sessione	Impostato per la sessione	Si applica a un'istruzione specifica
Supporta quanto segue: • SELECT • INSERT • UPDATE	Supporta quanto segue: • SELECT • INSERT • UPDATE	Supporta quanto segue: • SELECT • INSERT • UPDATE

SQL Server	Babelfish	PostgreSQL
• DELETE • CURSOR • CREATE • EXECUTE • EXEC e funzioni, incluso il flusso di controllo (CASE, WHILE-BREAK-CONTINUE, WAITFOR, BEGIN-END, IF-ELSE e così via)	• DELETE • CREATE • EXECUTE • EXEC • RAISEERROR • THROW • PRINT • USE	• DELETE • CURSOR • CREATE • EXECUTE

Utilizza le funzioni Babelfish come segue:

- SET BABELFISH_SHOWPLAN_ALL [ON|OFF]: imposta su ON per generare un piano di esecuzione delle query stimato. Questa funzione implementa il comportamento del comando `EXPLAIN` PostgreSQL. Utilizza questo comando per ottenere il piano explain per la query specificata.
- SET BABELFISH_STATISTICS PROFILE [ON|OFF]: imposta su ON per i piani di esecuzione delle query effettivi. Questa funzione implementa il comportamento del comando `EXPLAIN ANALYZE` PostgreSQL.

Per ulteriori informazioni su `EXPLAIN` e `EXPLAIN ANALYZE` PostgreSQL, consulta EXPLAIN nella documentazione di PostgreSQL.

> **Note**
>
> A partire dalla versione 2.2.0, è possibile impostare il parametro `escape_hatch_showplan_all` su ignore per evitare l'uso del prefisso BABELFISH_ nella sintassi di SQL Server per i comandi SET `SHOWPLAN_ALL` e `STATISTICS PROFILE`.

Ad esempio, la sequenza di comandi seguente attiva la pianificazione delle query e quindi restituisce un piano di esecuzione delle query stimato per l'istruzione SELECT senza eseguire la query. In

questo esempio, viene utilizzato il database SQL Server `northwind` di esempio che utilizza lo strumento a riga di comando `sqlcmd` per eseguire query sulla porta TDS:

```
1> SET BABELFISH_SHOWPLAN_ALL ON
2> GO
1> SELECT t.territoryid, e.employeeid FROM
2> dbo.employeeterritories e, dbo.territories t
3> WHERE e.territoryid=e.territoryid ORDER BY t.territoryid;
4> GO

QUERY PLAN


------------------------------------------------------------------------------


Query Text: SELECT t.territoryid, e.employeeid FROM
dbo.employeeterritories e, dbo.territories t
WHERE e.territoryid=e.territoryid ORDER BY t.territoryid
Sort  (cost=6231.74..6399.22 rows=66992 width=10)
  Sort Key: t.territoryid NULLS FIRST
  ->  Nested Loop  (cost=0.00..861.76 rows=66992 width=10)
        ->  Seq Scan on employeeterritories e  (cost=0.00..22.70 rows=1264 width=4)
              Filter: ((territoryid)::"varchar" IS NOT NULL)
        ->  Materialize  (cost=0.00..1.79 rows=53 width=6)
              ->  Seq Scan on territories t  (cost=0.00..1.53 rows=53 width=6)
```

Al termine della revisione e della regolazione della query, disattiva la funzione come illustrato di seguito:

```
1> SET BABELFISH_SHOWPLAN_ALL OFF
```

Con BABELFISH_STATISTICS PROFILE impostato su ON, ogni query eseguita restituisce il set di risultati regolare seguito da un set di risultati aggiuntivo che mostra i piani di esecuzione effettivi delle query. Babelfish genera il piano di query che fornisce il set di risultati più rapido quando richiama l'istruzione SELECT.

```
1> SET BABELFISH_STATISTICS PROFILE ON
1>
2> GO
1> SELECT e.employeeid, t.territoryid FROM
```

```
2> dbo.employeeterritories e, dbo.territories t
3> WHERE t.territoryid=e.territoryid ORDER BY t.territoryid;
4> GO
```

Vengono restituiti il set di risultati e il piano di query (questo esempio mostra solo il piano di query).

```
QUERY PLAN


------------------------------------------------------------------------
Query Text: SELECT e.employeeid, t.territoryid FROM
dbo.employeeterritories e, dbo.territories t
WHERE t.territoryid=e.territoryid ORDER BY t.territoryid
Sort  (cost=42.44..43.28 rows=337 width=10)
  Sort Key: t.territoryid NULLS FIRST


  ->  Hash Join  (cost=2.19..28.29 rows=337 width=10)
       Hash Cond: ((e.territoryid)::"varchar" = (t.territoryid)::"varchar")
        ->  Seq Scan on employeeterritories e  (cost=0.00..22.70 rows=1270 width=36)
        ->  Hash  (cost=1.53..1.53 rows=53 width=6)
             ->  Seq Scan on territories t  (cost=0.00..1.53 rows=53 width=6)
```

Per ulteriori informazioni su come analizzare le query e i risultati restituiti dall'ottimizzatore PostgreSQL, consulta explain.depesz.com. Per ulteriori informazioni su PostgreSQL EXPLAIN e EXPLAIN ANALYZE, consulta EXPLAIN nella documentazione di PostgreSQL.

Parametri che controllano le opzioni explain di Babelfish

Puoi utilizzare i parametri mostrati nella tabella seguente per controllare il tipo di informazioni visualizzate dal piano di query.

Parametro	Descrizione
babelfishpg_tsql.explain_buffers	Un valore booleano che attiva (e disattiva) le informazioni sull'utilizzo del buffer per l'ottimiz zatore. (Predefinito: off) (Consentito: off, on)
babelfishpg_tsql.explain_costs	Un valore booleano che attiva (e disattiva) le informazioni di avvio e del totale dei costi

Parametro	Descrizione
	stimate per l'ottimizzatore. (Predefinito: on) (Consentito: off, on)
babelfishpg_tsql.explain_format	Specifica il formato di output per il piano `EXPLAIN`. (Predefinito: text) (Consentito: text, xml, json, yaml)
babelfishpg_tsql.explain_settings	Un valore booleano che attiva (o disattiva) l'inclusione di informazioni sui parametri di configurazione nell'output del piano EXPLAIN. (Predefinito: off) (Consentito: off, on)
babelfishpg_tsql.explain_summary	Un valore booleano che attiva (o disattiva) informazioni di riepilogo come il tempo totale dopo il piano di query. (Predefinito: on) (Consentito: off, on)
babelfishpg_tsql.explain_timing	Un valore booleano che attiva (o disattiva) il tempo di avvio effettivo e il tempo trascorso in ciascun nodo dell'output. (Predefinito: on) (Consentito: off, on)
babelfishpg_tsql.explain_verbose	Un valore booleano che attiva (o disattiva) la versione più dettagliata di un piano explain. (Predefinito: off) (Consentito: off, on)
babelfishpg_tsql.explain_wal	Un valore booleano che attiva (o disattiva) la generazione di informazioni sui record WAL come parte di un piano explain. (Predefinito: off) (Consentito: off, on)

Puoi verificare i valori di qualsiasi parametro relativo a Babelfish sul sistema utilizzando il client PostgreSQL o il client SQL Server. Esegui il comando seguente per ottenere i valori dei parametri correnti:

```
1> execute sp_babelfish_configure '%explain%';
```

```
2> GO
```

Nell'output seguente, puoi vedere che tutte le impostazioni di questo particolare cluster database Babelfish sono sui loro valori predefiniti. Non tutto l'output è mostrato in questo esempio.

```
             name                    setting                     short_desc
--------------------------------- --------
 ---------------------------------------------------------
babelfishpg_tsql.explain_buffers   off      Include information on buffer usage
babelfishpg_tsql.explain_costs     on       Include information on estimated startup
 and total cost
babelfishpg_tsql.explain_format    text     Specify the output format, which can be
 TEXT, XML, JSON, or YAML
babelfishpg_tsql.explain_settings  off      Include information on configuration
 parameters
babelfishpg_tsql.explain_summary   on       Include summary information (e.g.,totaled
 timing information) after the query plan
babelfishpg_tsql.explain_timing    on       Include actual startup time and time spent
 in each node in the output
babelfishpg_tsql.explain_verbose   off      Display additional information regarding
 the plan
babelfishpg_tsql.explain_wal       off      Include information on WAL record
 generation

(8 rows affected)
```

Puoi modificare l'impostazione di questi parametri utilizzando `sp_babelfish_configure`, come mostrato nell'esempio seguente.

```
1> execute sp_babelfish_configure 'explain_verbose', 'on';
2> GO
```

Per rendere le impostazioni permanenti a livello di cluster, includi la parola chiave server, come nell'esempio seguente.

```
1> execute sp_babelfish_configure 'explain_verbose', 'on', 'server';
2> GO
```

Utilizzo di suggerimenti per le query T-SQL per migliorare le prestazioni delle query Babelfish

A partire dalla versione 2.3.0, Babelfish supporta l'uso dei suggerimenti per le query utilizzando `pg_hint_plan`. In Aurora PostgreSQL, `pg_hint_plan` è installato per impostazione predefinita. Per ulteriori informazioni sull'estensione PostgreSQL `pg_hint_plan`, consulta https://github.com/ossc-db/pg_hint_plan. Per informazioni dettagliate sulla versione di questa estensione supportata da Aurora PostgreSQL, consulta Extension versions for Amazon Aurora PostgreSQL (Versioni delle estensioni di Amazon Aurora PostgreSQL) nelle Note di rilascio per Aurora PostgreSQL.

L'ottimizzatore di query è progettato per trovare il piano di esecuzione ottimale per un'istruzione SQL. Quando si seleziona un piano, l'ottimizzatore di query considera sia il modello di costo del motore sia le statistiche di colonne e tabelle. Tuttavia, il piano suggerito potrebbe non soddisfare le esigenze dei tuoi set di dati. Pertanto, i suggerimenti di query risolvono i problemi di prestazioni per migliorare i piani di esecuzione. Un `query hint` è una sintassi aggiunta allo standard SQL che indica al motore di database come eseguire la query. Ad esempio, un suggerimento può indicare al motore di seguire una scansione sequenziale e sostituire qualsiasi piano selezionato dall'ottimizzatore di query.

Attivazione dei suggerimenti di query T-SQL in Babelfish

Attualmente, Babelfish ignora tutti i suggerimenti T-SQL per impostazione predefinita. Per applicare i suggerimenti T-SQL, esegui il comando `sp_babelfish_configure` con il valore enable_pg_hint impostato su ON (Attivato).

```
EXECUTE sp_babelfish_configure 'enable_pg_hint', 'on' [, 'server']
```

Puoi rendere le impostazioni permanenti a livello di cluster includendo la parola chiave server. Per configurare l'impostazione solo per la sessione corrente, non utilizzare la parola chiave server.

Dopo aver impostato `enable_pg_hint` su ON (Attivato), Babelfish applica i seguenti suggerimenti T-SQL.

- Suggerimenti INDEX
- Suggerimenti JOIN
- Suggerimento FORCE ORDER
- Suggerimento MAXDOP

Ad esempio, la seguente sequenza di comandi attiva `pg_hint_plan`

```
1> CREATE TABLE t1 (a1 INT PRIMARY KEY, b1 INT);
```

```
2> CREATE TABLE t2 (a2 INT PRIMARY KEY, b2 INT);
3> GO
1> EXECUTE sp_babelfish_configure 'enable_pg_hint', 'on';
2> GO
1> SET BABELFISH_SHOWPLAN_ALL ON;
2> GO
1> SELECT * FROM t1 JOIN t2 ON t1.a1 = t2.a2; --NO HINTS (HASH JOIN)
2> GO
```

Nessun suggerimento viene applicato all'istruzione SELECT. Viene restituito il piano di query senza alcun suggerimento.

```
QUERY PLAN


------------------------------------------------------------------------------
Query Text: SELECT * FROM t1 JOIN t2 ON t1.a1 = t2.a2
Hash Join (cost=60.85..99.39 rows=2260 width=16)
 Hash Cond: (t1.a1 = t2.a2)
 -> Seq Scan on t1 (cost=0.00..32.60 rows=2260 width=8)
 -> Hash (cost=32.60..32.60 rows=2260 width=8)
 -> Seq Scan on t2 (cost=0.00..32.60 rows=2260 width=8)
```

```
1> SELECT * FROM t1 INNER MERGE JOIN t2 ON t1.a1 = t2.a2;
2> GO
```

Il suggerimento di query viene applicato all'istruzione SELECT. L'output seguente mostra che viene restituito il piano di query con Merge Join.

```
QUERY PLAN


------------------------------------------------------------------------------
Query Text: SELECT/*+ MergeJoin(t1 t2) Leading(t1 t2)*/ * FROM t1 INNER JOIN t2 ON
 t1.a1 = t2.a2
Merge Join (cost=0.31..190.01 rows=2260 width=16)
 Merge Cond: (t1.a1 = t2.a2)
 -> Index Scan using t1_pkey on t1 (cost=0.15..78.06 rows=2260 width=8)
```

```
 -> Index Scan using t2_pkey on t2 (cost=0.15..78.06 rows=2260 width=8)
```

```
1> SET BABELFISH_SHOWPLAN_ALL OFF;
2> GO
```

Limitazioni

Per utilizzare i suggerimenti di query, considera le seguenti restrizioni:

- Se un piano di query viene memorizzato nella cache prima dell'attivazione di `enable_pg_hint`, i suggerimenti non vengono applicati nella stessa sessione, ma nella nuova sessione.
- Se i nomi di schema vengono forniti in modo esplicito, i suggerimenti non possono essere applicati. È possibile utilizzare gli alias delle tabelle come soluzione alternativa.
- Un suggerimento di query non può essere applicato a viste e query secondarie.
- I suggerimenti non funzionano per le istruzioni UPDATE/DELETE con JOIN.
- Un suggerimento di indice per una tabella o un indice inesistente viene ignorato.
- Il suggerimento FORCE ORDER non funziona per gli HASH JOIN e i JOIN non ANSI.

Utilizzo delle estensioni Aurora PostgreSQL con Babelfish

Aurora PostgreSQL fornisce estensioni per lavorare con altri servizi. AWS Si tratta di estensioni opzionali che supportano diversi casi d'uso, ad esempio l'utilizzo di Amazon S3 con il cluster database per l'importazione o l'esportazione di dati.

- Per importare i dati da un bucket Amazon S3 nel cluster database Babelfish, è necessario configurare l'estensione Aurora PostgreSQL `aws_s3`. Questa estensione consente anche di esportare i dati dal cluster database di Aurora PostgreSQL in un bucket Simple Storage Service (Amazon S3).
- AWS Lambda è un servizio di elaborazione che consente di eseguire codice senza fornire o gestire server. È possibile utilizzare le funzioni Lambda per elaborare le notifiche di eventi da un'istanza database. Per ulteriori informazioni su Lambda, consulta Che cos'è AWS Lambda? nella Guida per gli sviluppatori di AWS Lambda . Per richiamare le funzioni Lambda dal cluster database Babelfish, è necessario configurare l'estensione Aurora PostgreSQL `aws_lambda`.

Per configurare queste estensioni per il cluster Babelfish, devi prima concedere l'autorizzazione all'utente Babelfish interno per caricare le estensioni. Dopo aver concesso l'autorizzazione, è possibile caricare le estensioni Aurora PostgreSQL.

Abilitazione delle estensioni Aurora PostgreSQL nel cluster database Babelfish

Per poter caricare le estensioni `aws_s3` o `aws_lambda`, concedere i privilegi necessari al cluster database Babelfish.

Nella procedura che segue viene utilizzato lo strumento a riga di comando `psql` di PostgreSQL per connettersi al cluster database. Per ulteriori informazioni, consulta Usare psql per connettersi al cluster di database. È possibile utilizzare anche pgAdmin. Per informazioni dettagliate, vedi Uso di pgAdmin per connettersi al cluster di database.

Questa procedura carica `aws_s3` e `aws_lambda`, uno dopo l'altro. Se si desidera utilizzare solo una di queste estensioni, non è necessario caricarle entrambe. L'estensione `aws_commons` è richiesta da ciascuno e viene caricata per impostazione predefinita come mostrato nell'output.

Per configurare il cluster database Babelfish con privilegi per le estensioni Aurora PostgreSQL

1. Connettiti al cluster database Babelfish. Utilizza il nome per l'utente "master" (-U) specificato al momento della creazione del cluster database Babelfish. Il valore predefinito (`postgres`) viene visualizzato negli esempi.

   Per Linux, macOS o Unix:

   ```
   psql -h your-Babelfish.cluster.444455556666-us-east-1.rds.amazonaws.com \
   -U postgres \
   -d babelfish_db \
   -p 5432
   ```

   Per Windows:

   ```
   psql -h your-Babelfish.cluster.444455556666-us-east-1.rds.amazonaws.com ^
   -U postgres ^
   -d babelfish_db ^
   -p 5432
   ```

   Il comando risponde con un prompt per inserire la password per il nome utente (-U).

```
Password:
```

Inserisci la password del nome utente (-U) per il cluster database. Se il comando viene eseguito correttamente, verrà visualizzato un output simile al seguente.

```
psql (13.4)
SSL connection (protocol: TLSv1.3, cipher: TLS_AES_256_GCM_SHA384, bits: 256,
 compression: off)
Type "help" for help.

postgres=>
```

2. Concedi i privilegi all'utente Babelfish interno per creare e caricare le estensioni.

```
babelfish_db=> GRANT rds_superuser TO master_dbo;
GRANT ROLE
```

3. Crea e carica l'estensione `aws_s3`. L'estensione `aws_commons` è obbligatoria e viene installata automaticamente quando viene installato `aws_s3`.

```
babelfish_db=> create extension aws_s3 cascade;
NOTICE:  installing required extension "aws_commons"
CREATE EXTENSION
```

4. Crea e carica l'estensione `aws_lambda`.

```
babelfish_db=> create extension aws_lambda cascade;
CREATE EXTENSION
babelfish_db=>
```

Utilizzo di Babelfish con Simple Storage Service (Amazon S3)

Se non hai già un bucket Simple Storage Service (Amazon S3) da utilizzare con il cluster database Babelfish, puoi creane uno. Fornisci l'accesso per qualsiasi bucket Simple Storage Service (Amazon S3) che desideri utilizzare.

Prima di provare a importare o esportare dati utilizzando un bucket Simple Storage Service (Amazon S3), completa i seguenti passaggi una tantum.

Per configurare l'accesso per la tua istanza database Babelfish al bucket Simple Storage Service (Amazon S3)

1. Se necessario, crea un bucket Simple Storage Service (Amazon S3) per la tua istanza Babelfish. Per farlo, segui le istruzioni riportate in Create a bucket nella Guida per l'utente di Amazon Simple Storage Service.
2. Carica i file nel tuo bucket Simple Storage Service (Amazon S3). A tale scopo, attieniti alla procedura descritta in Add an object to a bucket nella Guida per l'utente di Amazon Simple Storage Service.
3. Imposta le autorizzazioni come necessario:

   - Per importare dati da Amazon S3, il cluster database Babelfish richiede l'autorizzazione per accedere al bucket. Ti consigliamo di utilizzare un ruolo AWS Identity and Access Management (IAM) e di associare una policy IAM a quel ruolo per il tuo cluster. A tale scopo, segui la procedura in Utilizzo di un ruolo IAM per accedere a un bucket Amazon S3..
   - Per esportare i dati dal cluster database Babelfish, è necessario concedere al cluster l'accesso al bucket Amazon S3. Come per l'importazione, si consiglia di utilizzare un ruolo e una policy IAM. A tale scopo, segui la procedura in Configurazione dell'accesso a un bucket Amazon S3.

Ora puoi utilizzare Amazon S3 con l'estensione `aws_s3` con il cluster database Babelfish.

Per importare dati da Simple Storage Service (Amazon S3) in Babelfish ed esportare dati Babelfish in Simple Storage Service (Amazon S3)

1. Utilizzo dell'estensione `aws_s3` con il cluster database Babelfish.

   In questa situazione, assicurati di fare riferimento alle tabelle così come sono presenti nel contesto di PostgreSQL. Vale a dire, se vuoi importare in una tabella Babelfish denominata `[database].[schema].[tableA]`, fai riferimento a quella tabella come `database_schema_tableA` nella funzione `aws_s3`:

   - Per un esempio di utilizzo di una funzione `aws_s3` per importare dati, vedi Importazione di dati da Amazon S3 nel cluster database Aurora PostgreSQL.
   - Per esempi di utilizzo di funzioni `aws_s3` per esportare dati, vedi Esportazione dei dati della query utilizzando la funzione aws_s3.query_export_to_s3.

2. Assicurati di fare riferimento alle tabelle Babelfish usando la denominazione PostgreSQL quando utilizzi l'estensione `aws_s3` e Simple Storage Service (Amazon S3), come illustrato nella tabella seguente.

Tabella Babelfish	Tabella Aurora PostgreSQL
database.*schema*.*table*	*database*_*schema*_*table*

Per ulteriori informazioni sull'uso di Amazon S3 con Aurora PostgreSQL, vedi Importazione di dati da Amazon S3 in un cluster database Aurora PostgreSQL e Esportazione di dati da del cluster di database Aurora PostgreSQLRDS per PostgreSQL a Amazon S3.

Usare Babelfish con AWS Lambda

Dopo aver caricato l'estensione `aws_lambda` nel cluster database Babelfish ma prima di richiamare le funzioni Lambda, puoi fornire l'accesso a Lambda al cluster database seguendo questa procedura.

Per configurare l'accesso per il cluster database Babelfish per il funzionamento con Lambda

Questa procedura utilizza AWS CLI per creare la politica e il ruolo IAM e associarli al cluster Babelfish DB.

1. Crea una policy IAM che consente l'accesso a Lambda dal cluster database Babelfish.

```
aws iam create-policy  --policy-name rds-lambda-policy --policy-document '{
 "Version": "2012-10-17",
    "Statement": [
        {
        "Sid": "AllowAccessToExampleFunction",
        "Effect": "Allow",
        "Action": "lambda:InvokeFunction",
        "Resource": "arn:aws:lambda:aws-region:444455556666:function:my-function"
        }
    ]
}'
```

2. Crea un ruolo IAM che la policy può assumere in fase di esecuzione.

```
aws iam create-role  --role-name rds-lambda-role --assume-role-policy-document '{
    "Version": "2012-10-17",
```

```
    "Statement": [
        {
        "Effect": "Allow",
        "Principal": {
            "Service": "rds.amazonaws.com"
        },
        "Action": "sts:AssumeRole"
        }
    ]
}'
```

3. Collegare la policy al ruolo.

```
aws iam attach-role-policy \
    --policy-arn arn:aws:iam::444455556666:policy/rds-lambda-policy \
    --role-name rds-lambda-role --region aws-region
```

4. Associa il ruolo al tuo cluster database Babelfish

```
aws rds add-role-to-db-cluster \
       --db-cluster-identifier my-cluster-name \
       --feature-name Lambda \
       --role-arn  arn:aws:iam::444455556666:role/rds-lambda-role   \
       --region aws-region
```

Dopo aver completato queste attività, è possibile richiamare le funzioni Lambda. Per ulteriori informazioni ed esempi di configurazione AWS Lambda per il cluster Aurora PostgreSQL DB con, consulta. AWS LambdaAWS Lambda

Per richiamare una funzione Lambda dal cluster database Babelfish

AWS Lambda supporta funzioni scritte in Java, Node.js, Python, Ruby e altri linguaggi. Se la funzione restituisce un testo quando viene richiamata, è possibile richiamarla dal cluster database Babelfish. L'esempio seguente è una funzione python segnaposto che restituisce un saluto.

```
lambda_function.py
import json
def lambda_handler(event, context):
    #TODO implement
    return {
        'statusCode': 200,
```

```
        'body': json.dumps('Hello from Lambda!')
```

Attualmente, Babelfish non supporta JSON. Se la funzione restituisce JSON, si utilizza un wrapper per gestire il JSON. Ad esempio, diciamo che la funzione `lambda_function.py` mostrata in precedenza è memorizzata in Lambda come `my-function`.

1. Collegati al cluster database Babelfish utilizzando il client `psql` (o il client pgAdmin). Per ulteriori informazioni, consulta Usare psql per connettersi al cluster di database.
2. Crea il wrapper. Questo esempio utilizza il linguaggio procedurale di PostgreSQL per SQL, PL/pgSQL. Per ulteriori informazioni, consulta PL/pgSQL–SQL Procedural Language (PL/pgSQL - Linguaggio procedurale SQL).

   ```
   create or replace function master_dbo.lambda_wrapper()
   returns text
   language plpgsql
   as
   $$
   declare
      r_status_code integer;
      r_payload text;
   begin
      SELECT payload INTO r_payload
        FROM aws_lambda.invoke(  aws_commons.create_lambda_function_arn('my-function',
    'us-east-1')
                               ,'{"body": "Hello from Postgres!"}'::json );
      return r_payload ;
   end;
   $$;
   ```

   La funzione ora può essere eseguita dalla porta TDS Babelfish (1433) o dalla porta PostgreSQL (5433).

   a. Per richiamare (chiamare) questa funzione dalla porta PostgreSQL:

      ```
      SELECT * from aws_lambda.invoke(aws_commons.create_lambda_function_arn('my-
      function', 'us-east-1'), '{"body": "Hello from Postgres!"}'::json );
      ```

      L'output è simile a quello riportato di seguito:

```
status_code |                        payload                         |
 executed_version | log_result
-------------+--------------------------------------------------------
+-----------------+------------
         200 | {"statusCode": 200, "body": "\"Hello from Lambda!\""} | $LATEST
         |
(1 row)
```

b. Per richiamare (chiamare) questa funzione dalla porta TDS, connettiti alla porta utilizzando il client della riga di comando `sqlcmd` di SQL Server. Per informazioni dettagliate, vedi Utilizzo di un client SQL Server per la connessione al cluster database. Una volta connesso, esegui quanto segue:

```
1> select lambda_wrapper();
2> go
```

Il comando restituisce un output simile al seguente:

```
{"statusCode": 200, "body": "\"Hello from Lambda!\""}
```

Per ulteriori informazioni sull'uso di Lambda con Aurora PostgreSQL, consulta . Per ulteriori informazioni sull'utilizzo delle funzioni Lambda, consulta Nozioni di base su Lambda nella Guida per gli sviluppatori di AWS Lambda .

Utilizzo di pg_stat_statements in Babelfish

Babelfish per Aurora PostgreSQL supporta l'estensione `pg_stat_statements` dalla versione 3.3.0. Per maggiori informazioni, consulta pg_stat_statements.

Per informazioni sulla versione di questa estensione supportata da Aurora PostgreSQL, consulta Extension versions.

Creazione dell'estensione pg_stat_statements

Per attivare `pg_stat_statements`, è necessario attivare il calcolo dell'identificatore di query. Questa operazione viene eseguita automaticamente se `compute_query_id` è impostato su on o `auto` nel gruppo di parametri. Il valore predefinito del parametro `compute_query_id` è `auto`. È inoltre necessario creare questa estensione per attivare questa funzionalità. Utilizza il comando seguente per installare l'estensione dall'endpoint T-SQL:

```
1>EXEC sp_execute_postgresql 'CREATE EXTENSION pg_stat_statements WITH SCHEMA sys';
```

È possibile accedere alle statistiche delle query utilizzando la seguente query:

```
postgres=>select * from pg_stat_statements;
```

> **Note**
>
> Se durante l'installazione non fornisci il nome dello schema per l'estensione, per impostazione predefinita l'estensione verrà creata in uno schema pubblico. Per accedervi, è necessario scrivere il qualificatore dello schema tra parentesi quadre, come mostrato di seguito:

```
postgres=>select * from [public].pg_stat_statements;
```

Puoi anche creare l'estensione dall'endpoint PSQL.

Autorizzazione dell'estensione

Per impostazione predefinita, è possibile visualizzare le statistiche per le query eseguite all'interno del database T-SQL senza bisogno di autorizzazioni.

Per accedere alle statistiche delle query create da altri, è necessario disporre del ruolo PostgreSQL `pg_read_all_stats`. Segui i passaggi indicati di seguito per costruire il comando GRANT pg_read_all_stats.

1. In T-SQL, usa la seguente query che restituisce il nome del ruolo PG interno.

   ```
   SELECT rolname FROM pg_roles WHERE oid = USER_ID();
   ```

2. Connettiti al database PostgreSQL di Babelfish per Aurora con il privilegio rds_superuser e usa il seguente comando:

   ```
   GRANT pg_read_all_stats TO <rolname_from_above_query>
   ```

Esempio

Dall'endpoint T-SQL:

```
1>SELECT rolname FROM pg_roles WHERE oid = USER_ID();
2>go
```

```
rolname
-------
master_dbo
(1 rows affected)
```

Dall'endpoint PSQL:

```
babelfish_db=# grant pg_read_all_stats to master_dbo;
```

```
GRANT ROLE
```

Puoi accedere alle statistiche delle query utilizzando la vista pg_stat_statements:

```
1>create table t1(cola int);
2>go
1>insert into t1 values (1),(2),(3);
2>go
```

```
(3 rows affected)
```

```
1>select userid, dbid, queryid, query from pg_stat_statements;
2>go
```

```
userid dbid queryid             query
------ ---- -------             -----
37503 34582 6487973085327558478 select * from t1
37503 34582 6284378402749466286 SET QUOTED_IDENTIFIER OFF
37503 34582 2864302298511657420 insert into t1 values ($1),($2),($3)
10    34582 NULL                <insufficient privilege>
```

```
37503 34582 5615368793313871642 SET TEXTSIZE 4096
37503 34582 639400815330803392  create table t1(cola int)
(6 rows affected)
```

Reimpostazione delle statistiche delle query

Puoi usare `pg_stat_statements_reset()` per eseguire il ripristino delle statistiche raccolte finora da pg_stat_statements. Per maggiori informazioni, consulta pg_stat_statements. Attualmente è supportato solo tramite endpoint PSQL. Per connetterti a Babelfish per Aurora PostgreSQL con il privilegio `rds_superuser`, usa il comando seguente:

```
SELECT pg_stat_statements_reset();
```

Limitazioni

- Attualmente, `pg_stat_statements()` non è supportato tramite l'endpoint T-SQL. La vista `pg_stat_statements` è il metodo consigliato per raccogliere le statistiche.
- Alcune query potrebbero essere riscritte dal parser T-SQL implementato dal motore di Aurora PostgreSQL, in tal caso la vista `pg_stat_statements` mostrerà la query riscritta e non la query originale.

  Esempio

  ```
  select next value for [dbo].[newCounter];
  ```

  La query precedente viene riscritta come segue nella vista pg_stat_statements.

  ```
  select nextval($1);
  ```
- In base al flusso di esecuzione delle istruzioni, alcune query potrebbero non essere tracciate da pg_stat_statements e non essere visibili nella vista. Ciò include le seguenti istruzioni: `use dbname`, `goto`, `print`, `raise error`, `set`, `throw`, `declare cursor`.
- Per le istruzioni CREATE LOGIN e ALTER LOGIN, query e queryid non verranno visualizzate. Verrà visualizzato un avviso di privilegi insufficienti.
- La vista `pg_stat_statements` contiene sempre le due voci seguenti, poiché vengono eseguite internamente dal client `sqlcmd`.

- SET QUOTED_IDENTIFIER OFF
- SET TEXTSIZE 4096

Utilizzo di pgvector in Babelfish

L'estensione open source pgvector consente di cercare dati simili direttamente all'interno del database Postgres. Babelfish ora supporta questa estensione a partire dalle versioni 15.6 e 16.2. Per ulteriori informazioni, consultare la documentazione open source per pgvector.

Prerequisiti

Per abilitare la funzionalità pgvector, installare l'estensione nello schema sys utilizzando uno dei seguenti metodi:

- Eseguire il comando seguente nel client sqlcmd:

```
exec sys.sp_execute_postgresql 'CREATE EXTENSION vector WITH SCHEMA sys';
```

- Connettersi a `babelfish_db` ed eseguire il comando seguente nel client psql:

```
CREATE EXTENSION vector WITH SCHEMA sys;
```

> **Note**
>
> Dopo aver installato l'estensione pgvector, il tipo di dati vettoriali sarà disponibile solo nelle nuove connessioni al database. Le connessioni esistenti non riconoscono il nuovo tipo di dati.

Funzionalità supportata

Babelfish estende la funzionalità T-SQL per supportare:

- Archiviazione

  Babelfish ora supporta la sintassi compatibile con i tipi di dati vettoriali, migliorando la compatibilità T-SQL. Per saperne di più sull'archiviazione dei dati con pgvector, consulta Storing.

- Esecuzione di query

Babelfish espande il supporto delle espressioni T-SQL per includere operatori di somiglianza vettoriale. Tuttavia, per tutte le altre query, è ancora richiesta la sintassi T-SQL standard.

> **Note**
>
> T-SQL non supporta il tipo Matrice e i driver del database non dispongono di un'interfaccia per gestirlo. Come soluzione alternativa, Babelfish utilizza stringhe di testo (varchar/nvarchar) per archiviare i dati vettoriali. Ad esempio, quando si richiede un valore vettoriale [1,2,3], Babelfish restituisce una stringa '[1,2,3]' come risposta. È possibile analizzare e dividere questa stringa a livello di applicazione in base alle esigenze.

Per ulteriori informazioni sull'esecuzione di query sui dati con pgvector, consulta Esecuzione di query.

- Indicizzazione

  T-SQL `Create Index` ora supporta la sintassi `USING INDEX_METHOD`. È ora possibile definire l'operatore di ricerca per similarità da utilizzare su una colonna specifica durante la creazione di un indice.

  La grammatica viene inoltre estesa per supportare le operazioni di similarità vettoriale sulla colonna richiesta (controlla column_name_list_with_order_for_vector grammar).

```
CREATE [UNIQUE] [clustered] [COLUMNSTORE] INDEX <index_name> ON <table_name> [USING
 vector_index_method] (<column_name_list_with_order_for_vector>)
Where column_name_list_with_order_for_vector is:
    <column_name> [ASC | DESC] [VECTOR_COSINE_OPS | VECTOR_IP_OPS | VECTOR_L2_OPS]
 (COMMA simple_column_name [ASC | DESC] [VECTOR_COSINE_OPS | VECTOR_IP_OPS |
 VECTOR_L2_OPS])
```

  Per ulteriori informazioni sull'indicizzazione dei dati con pgvector, consulta Indicizzazione.

- Prestazioni
  - Utilizzare `SET BABELFISH_STATISTICS PROFILE ON` per eseguire il debug dei piani di query dall'endpoint T-SQL.
  - Aumentare `max_parallel_workers_get_gather` utilizzando la funzione `set_config` supportata in T-SQL.
  - Utilizzare `IVFFlat` per ricerche approssimative. Per ulteriori informazioni, consulta IVFFlat.

Per migliorare le prestazioni con pgvector, consulta Prestazioni.

Limitazioni

- Babelfish non supporta la ricerca full-text per la ricerca ibrida. Per ulteriori informazioni, consulta Hybrid Search.
- Babelfish attualmente non supporta la funzionalità di reindicizzazione. Tuttavia, è possibile utilizzare l'endpoint PostgreSQL per reindicizzare. Per ulteriori informazioni, consulta Vacuuming.

Utilizzo del machine learning Amazon Aurora con Babelfish

È possibile estendere le funzionalità del cluster di database Babelfish per Aurora PostgreSQL integrandolo con il machine learning Amazon Aurora. Questa perfetta integrazione ti consente di accedere a una gamma di potenti servizi come Amazon Comprehend o Amazon AI o SageMaker Amazon Bedrock, ciascuno personalizzato per soddisfare esigenze di machine learning distinte.

Come utente Babelfish, è possibile utilizzare le conoscenze esistenti della sintassi e della semantica T-SQL quando si lavora con il machine learning Aurora. Segui le istruzioni fornite nella AWS documentazione di Aurora PostgreSQL. Per ulteriori informazioni, consulta Utilizzo del machine learning di Amazon Aurora con Aurora PostgreSQL.

Prerequisiti

- Prima di provare a configurare il cluster di database Babelfish per Aurora PostgreSQL per utilizzare il machine learning Aurora, è necessario comprendere i requisiti e i prerequisiti correlati. Per ulteriori informazioni, consulta Requisiti per l'utilizzo del machine learning di Aurora con Aurora PostgreSQL.
- Assicurarsi di installare l'estensione `aws_ml` utilizzando l'endpoint Postgres o la stored procedure `sp_execute_postgresql`.

```
exec sys.sp_execute_postgresql 'Create Extension aws_ml'
```

> **Note**
>
> Attualmente Babelfish non supporta le operazioni a cascata con `sp_execute_postgresql`. Poiché `aws_ml` si basa su `aws_commons`, è necessario installarlo separatamente utilizzando l'endpoint Postgres.

```
create extension aws_common;
```

Gestione della sintassi e della semantica T-SQL con le funzioni **`aws_ml`**

Gli esempi seguenti spiegano come la sintassi e la semantica T-SQL vengono applicate ai servizi Amazon ML:

Example : aws_bedrock.invoke_model: una query semplice che utilizza le funzioni di Amazon Bedrock

```
aws_bedrock.invoke_model(
   model_id      varchar,
   content_type  text,
   accept_type   text,
   model_input   text)
Returns Varchar(MAX)
```

L'esempio seguente mostra come invocare un modello Anthropic Claude 2 per Bedrock utilizzando invoke_model.

```
SELECT aws_bedrock.invoke_model (
    'anthropic.claude-v2', -- model_id
    'application/json', -- content_type
    'application/json', -- accept_type
    '{"prompt": "\n\nHuman:
    You are a helpful assistant that answers questions directly
    and only using the information provided in the context below.
    \nDescribe the answerin detail.\n\nContext: %s \n\nQuestion:
    %s \n\nAssistant:","max_tokens_to_sample":4096,"temperature"
    :0.5,"top_k":250,"top_p":0.5,"stop_sequences":[]}' -- model_input
);
```

Example : aws_comprehend.detect_sentiment: una query semplice che utilizza le funzioni di Amazon Comprehend

```
aws_comprehend.detect_sentiment(
   input_text varchar,
   language_code varchar,
   max_rows_per_batch int)
Returns table (sentiment varchar, confidence real)
```

L'esempio seguente mostra come invocare il servizio Amazon Comprehend.

```
select sentiment from aws_comprehend.detect_sentiment('This is great', 'en');
```

Example : aws_sagemaker.invoke_endpoint — Una semplice query che utilizza le funzioni di Amazon SageMaker

```
aws_sagemaker.invoke_endpoint(
  endpoint_name varchar,
  max_rows_per_batch int,
  VARIADIC model_input "any") -- Babelfish inherits PG's variadic parameter type
Rerurns Varchar(MAX)
```

Poiché model_input è contrassegnato come VARIADIC e ha il tipo "any", gli utenti possono passare un elenco di qualsiasi lunghezza e qualsiasi tipo di dati alla funzione che fungerà da input per il modello. L'esempio seguente mostra come richiamare il SageMaker servizio Amazon.

```
SELECT CAST (aws_sagemaker.invoke_endpoint(
    'sagemaker_model_endpoint_name',
    NULL,
    arg1, arg2 -- model inputs are separate arguments )
AS INT) -- cast the output to INT
```

Per ulteriori informazioni sull'utilizzo del machine learning Aurora con Aurora PostgreSQL, consulta Utilizzo del machine learning di Amazon Aurora con Aurora PostgreSQL.

Limitazioni

- Sebbene Babelfish non consenta la creazione di matrici, può comunque gestire i dati che rappresentano matrici. Quando si utilizzano funzioni come `aws_bedrock.invoke_model_get_embeddings` che restituiscono matrici, i risultati vengono forniti come una stringa contenente gli elementi della matrice.

Babelfish supporta i server collegati

Babelfish per Aurora PostgreSQL supporta i server collegati utilizzando l'estensione PostgreSQL `tds_fdw` nella versione 3.1.0. Per utilizzare i server collegati, devi installare l'estensione `tds_fdw`. Per ulteriori informazioni sull'estensione `tds_fdw`, consulta Utilizzo dei wrapper di dati esterni supportati per Amazon Aurora PostgreSQL.

Installazione dell'estensione **`tds_fdw`**

È possibile installare un'estensione `tds_fdw` utilizzando i seguenti metodi.

Utilizzo di CREATE EXTENSION dall'endpoint PostgreSQL

1. Effettua la connessione alla tua istanza database PostgreSQL sul database Babelfish nella porta PostgreSQL. Usa un account con il ruolo rds_superuser.

   ```
   psql --host=your-DB-instance.aws-region.rds.amazonaws.com --port=5432 --
   username=test --dbname=babelfish_db --password
   ```

2. Installa l'estensione `tds_fdw`. Questo è un processo di installazione da compiere una sola volta. Non è necessario ripetere l'installazione al riavvio del cluster di database.

   ```
   babelfish_db=> CREATE EXTENSION tds_fdw;
   CREATE EXTENSION
   ```

Chiamata della stored procedure **`sp_execute_postgresql`** dall'endpoint TDS

Babelfish supporta l'installazione dell'estensione `tds_fdw` richiamando la procedure `sp_execute_postgresql` a partire dalla versione 3.3.0. È possibile eseguire istruzioni PostgreSQL

dall'endpoint T-SQL senza uscire dalla porta T-SQL. Per ulteriori informazioni, consulta Utilizzo delle procedure di Babelfish per Aurora PostgreSQL

1. Effettua la connessione alla tua istanza database PostgreSQL sul database Babelfish nella porta T-SQL.

```
sqlcmd -S your-DB-instance.aws-region.rds.amazonaws.com -U test -P password
```

2. Installa l'estensione `tds_fdw`.

```
1>EXEC sp_execute_postgresql N'CREATE EXTENSION tds_fdw';
2>go
```

Funzionalità supportate

Babelfish supporta l'aggiunta di endpoint remoti RDS per SQL Server o Babelfish per Aurora PostgreSQL come server collegato. È inoltre possibile aggiungere altre istanze remote SQL Server come server collegati. Quindi, usa `OPENQUERY()` per recuperare i dati da questi server collegati. A partire dalla versione 3.2.0 di Babelfish, sono supportati anche i nomi in quattro parti.

Le seguenti stored procedure e visualizzazioni del catalogo sono supportate per utilizzare i server collegati.

Stored procedure

- sp_addlinkedserver: Babelfish non supporta il parametro `@provstr`.
- sp_addlinkedsrvlogin
  - È necessario fornire un nome utente e una password remoti espliciti per connetterti all'origine dati remota. Non puoi connetterti mediante le credenziali automatiche dell'utente. Babelfish supporta solo `@useself = false`.
  - Babelfish non supporta il parametro `@locallogin` poiché non è supportata la configurazione dell'accesso del server remoto specifico per l'accesso locale.
- sp_linkedservers
- sp_helplinkedsrvlogin
- sp_dropserver
- sp_droplinkedsrvlogin: Babelfish non supporta il parametro `@locallogin` poiché non è supportata la configurazione dell'accesso del server remoto specifico per l'accesso locale.

- sp_serveroption: Babelfish supporta le seguenti opzioni del server:
  - timeout delle query (a partire dalla versione 3.2.0)
  - timeout di connessione (a partire dalla versione 3.3.0)
- sp_testlinkedserver (a partire dalla versione 3.3.0 di Babelfish)
- sp_enum_oledb_providers (a partire dalla versione 3.3.0 di Babelfish)

Visualizzazioni del catalogo

- sys.servers
- sys.linked_logins

Utilizzo della crittografia in transito per la connessione

La connessione dal server di origine Babelfish per Aurora PostgreSQL al server remoto di destinazione utilizza la crittografia in transito (TLS/SSL) in base alla configurazione del database del server remoto. Se il server remoto non è configurato per la crittografia, il server Babelfish che effettua la richiesta al database remoto esegue il fallback su non crittografato.

Per applicare la crittografia della connessione

- Se il server collegato di destinazione è un'istanza RDS per SQL Server, imposta `rds.force_ssl = on` per l'istanza SQL Server di destinazione. Per ulteriori informazioni sulla configurazione di SSL/TLS per RDS for SQL Server, consultare Utilizzo di SSL con un'istanza database Microsoft SQL Server.
- Se il server collegato di destinazione è un cluster Babelfish per Aurora PostgreSQL, imposta `babelfishpg_tds.tds_ssl_encrypt = on` e `ssl = on` per il server di destinazione. Per ulteriori informazioni su SSL/TLS, consulta Impostazioni SSL Babelfish e connessioni client.

Aggiunta di Babelfish come server collegato da SQL Server

Babelfish per Aurora PostgreSQL può essere aggiunto come server collegato da un'istanza di SQL Server. In un database SQL Server, puoi aggiungere Babelfish come server collegato utilizzando il provider Microsoft OLE DB per ODBC: MSDASQL.

Esistono due modi per configurare Babelfish come server collegato da SQL Server utilizzando il provider MSDASQL:

- Immissione di una stringa di connessione ODBC come stringa del provider.
- Immissione del DSN di sistema dell'origine dati ODBC durante l'aggiunta del server collegato.

Limitazioni

- OPENQUERY() funziona solo per SELECT e non funziona per DML.
- I nomi degli oggetti in quattro parti funzionano solo per la lettura e non per la modifica della tabella remota. UPDATE può fare riferimento a una tabella remota nella clausola FROM senza modificarla.
- L'esecuzione di stored procedure sui server collegati a Babelfish non è supportata.
- L'aggiornamento della versione principale di Babelfish potrebbe non funzionare se sono presenti oggetti dipendenti da `OPENQUERY()` o oggetti a cui si fa riferimento tramite nomi in quattro parti. Devi verificare che tutti gli oggetti che fanno riferimento a `OPENQUERY()` o a nomi in quattro parti vengano eliminati prima di un aggiornamento della versione principale.
- I seguenti tipi di dati non funzionano come previsto sul server Babelfish remoto: `nvarchar(max)`, `varchar(max)`, `varbinary(max)`, `binary(max)` e `time`. Consigliamo di utilizzare la funzione CAST per convertirli nei tipi di dati supportati.

Esempio

Nell'esempio seguente, un'istanza Babelfish per Aurora PostgreSQL si connette a un'istanza di RDS per SQL Server nel cloud.

```
EXEC master.dbo.sp_addlinkedserver @server=N'rds_sqlserver', @srvproduct=N'',
 @provider=N'SQLNCLI', @datasrc=N'myserver.CB2XKFSFFMY7.US-WEST-2.RDS.AMAZONAWS.COM';
EXEC master.dbo.sp_addlinkedsrvlogin
 @rmtsrvname=N'rds_sqlserver',@useself=N'False',@locallogin=NULL,@rmtuser=N'username',@rmtpassw
```

Quando il server collegato è attivo, puoi quindi utilizzare T-SQL OPENQUERY () o la denominazione standard in quattro parti per fare riferimento a una tabella, una vista o altri oggetti supportati sul server remoto:

```
SELECT * FROM OPENQUERY(rds_sqlserver, 'SELECT * FROM TestDB.dbo.t1');
SELECT * FROM rds_sqlserver.TestDB.dbo.t1;
```

Per eliminare il server collegato e tutti gli accessi associati:

```
EXEC master.dbo.sp_dropserver @server=N'rds_sqlserver', @droplogins=N'droplogins';
```

Risoluzione dei problemi

È possibile utilizzare lo stesso gruppo di sicurezza sia per i server di origine sia per quelli remoti per consentire la comunicare tra loro. Il gruppo di sicurezza deve consentire solo il traffico in entrata sulla porta TDS (1433 per impostazione predefinita) e l'IP di origine nel gruppo di sicurezza può essere impostato come ID del gruppo di sicurezza stesso. Per ulteriori informazioni su come impostare le regole per la connessione a un'istanza da un'altra istanza con lo stesso gruppo di sicurezza, consulta Regole per la connessione alle istanze da un'istanza con lo stesso gruppo di sicurezza.

Se l'accesso non è configurato correttamente, quando si tenta di eseguire una query sul server remoto viene visualizzato un messaggio di errore simile al seguente.

```
TDS client library error: DB #: 20009, DB Msg: Unable to connect: server is unavailable
 or does not exist (mssql2019.aws-region.rds.amazonaws.com), OS #: 110, OS Msg:
 Connection timed out, Level: 9
```

Utilizzo della ricerca full-text in Babelfish

A partire dalla versione 4.0.0, Babelfish fornisce un supporto limitato per la ricerca full-text (FTS, Full Text Search). FTS è una potente funzionalità dei database relazionali che consente di eseguire in modo efficiente la ricerca e l'indicizzazione di grandi volumi di dati di testo. Consente di eseguire ricerche di testo complesse e recuperare rapidamente i risultati pertinenti. FTS è particolarmente utile per le applicazioni che gestiscono grandi volumi di dati di testo, come sistemi di gestione dei contenuti, piattaforme di e-commerce e archivi di documenti.

Informazioni sulle funzionalità di ricerca full-text supportate da Babelfish

Babelfish supporta le seguenti funzionalità di ricerca full-text:

- Clausola CONTAINS:
  - Supporto di base per la clausola CONTAINS.

    ```
    CONTAINS (
    ```

```
    {
       column_name
    }
    , '<contains_search_condition>'
)
```

> **Note**
>
> Attualmente, è supportata solo la lingua inglese.

  - Gestione e traduzione complete delle stringhe di ricerca `simple_term`.
- Clausola `FULLTEXT INDEX`:
  - Supporta solo l'istruzione `CREATE FULLTEXT INDEX ON table_name(column_name [...n]) KEY INDEX index_name`.
  - Supporta l'istruzione completa `DROP FULLTEXT INDEX`.

> **Note**
>
> Per reindicizzare l'indice full-text, è necessario eliminare l'indice full-text e crearne uno nuovo nella stessa colonna.

- Caratteri speciali nella condizione di ricerca:
  - Babelfish garantisce che le singole occorrenze dei caratteri speciali nelle stringhe di ricerca vengano gestite in modo efficace.

> **Note**
>
> Anche se Babelfish ora identifica i caratteri speciali nella stringa di ricerca, è essenziale riconoscere che i risultati ottenuti possono variare rispetto a quelli ottenuti con T-SQL.

- Alias di tabella in column_name:
  - Con il supporto degli alias di tabella, gli utenti possono creare query SQL più concise e leggibili per la ricerca full-text.

Limitazioni della ricerca full-text in Babelfish

- Attualmente, le seguenti opzioni non sono supportate in Babelfish per la clausola `CONTAINS`.

  - I caratteri speciali e le lingue diverse dall'inglese non sono supportati. Viene visualizzato un messaggio di errore generico per i caratteri e la lingua non supportati

    ```
    Full-text search conditions with special characters or languages other than English
     are not currently supported in Babelfish
    ```

  - Più colonne come `column_list`
  - Attributo PROPERTY
  - `prefix_term`, `generation_term`, `generic_proximity_term`, `custom_proximity_term`, e `weighted_term`
  - Gli operatori booleani non sono supportati e viene visualizzato il seguente messaggio di errore se utilizzati:

    ```
    boolean operators not supported
    ```

  - I nomi degli identificatori con punti non sono supportati.
- Attualmente, le seguenti opzioni non sono supportate in Babelfish per la clausola `CREATE FULLTEXT INDEX`.
  - [TYPE COLUMN type_column_name]
  - [LANGUAGE language_term]
  - [STATISTICAL_SEMANTICS]
  - opzioni dei gruppi di file di catalogo
  - con opzioni
- La creazione di un catalogo full-text non è supportata. La creazione di un indice full-text non richiede un catalogo full-text.
- `CREATE FULLTEXT INDEX` non supporta i nomi di identificatori con punti.
- Babelfish attualmente non supporta i caratteri speciali consecutivi nelle stringhe di ricerca. Se utilizzati, viene visualizzato il seguente messaggio di errore:

  ```
  Consecutive special characters in the full-text search condition are not currently
   supported in Babelfish
  ```

Babelfish supporta i tipi di dati geospaziali

A partire dalle versioni 3.5.0 e 4.1.0, Babelfish include il supporto per i seguenti due tipi di dati spaziali:

- Tipo di dati geometrici: questo tipo di dati è destinato alla memorizzazione di dati planari o euclidei (terra piatta).
- Tipo di dati geografici: questo tipo di dati è destinato alla memorizzazione di dati ellissoidali o terra rotonda, come le coordinate GPS di latitudine e longitudine.

Questi tipi di dati consentono l'archiviazione e la manipolazione di dati spaziali, ma con limitazioni.

Informazioni sui tipi di dati geospaziali in Babelfish

- I tipi di dati geospaziali sono supportati in vari oggetti di database come viste, procedure e tabelle.
- Supporta il tipo di dati puntuale per memorizzare i dati sulla posizione come punti definiti da latitudine, longitudine e un SRID (Spatial Reference System Identifier) valido. Un punto può contenere valori Z (elevazione), M (misura) e può essere vuoto.
- Supporta il tipo di dati linestring (a partire dalla versione 5.4.0) definito da una sequenza di punti e dai segmenti di linea che li collegano e da un SRID (Spatial Reference System Identifier) valido. Una stringa di linee può contenere punti con valori Z (elevazione), M (misura) e può essere vuota.
- Le applicazioni che si connettono a Babelfish tramite driver come JDBC, ODBC, DOTNET e PYTHON possono utilizzare questa funzionalità geospaziale.

Funzioni dei tipi di dati geometrici supportate in Babelfish

- STGeomFromText (***geometry_tagged_text***, SRID) — Crea un'istanza geometrica utilizzando la rappresentazione Wooden Text (WKT).
- STPointFromText (***point_tagged_text***, SRID) — Crea un'istanza puntuale utilizzando la rappresentazione WKT.
- Point (X, Y, SRID): crea un'istanza di punti dati utilizzando valori in virgola mobile delle coordinate x e y.
- <geometry_instance>. STAsText () — Estrae la rappresentazione WKT dall'istanza geometrica.
- <geometry_instance>. STAsBinary () — Estrae la rappresentazione WKB dall'istanza geometrica.
- <geometry_instance>. STArea () — Calcola la superficie totale dell'istanza geometrica.

- <geometry_instance>. STSrid () — Estrae l'identificatore di riferimento spaziale (SRID) dell'istanza geometrica.
- <geometry_instance>. STDimension () — Recupera la dimensione spaziale dell'istanza geometrica.
- <geometry_instance>. STIsEmpty () — Verifica se l'istanza geometrica è vuota.
- <geometry_instance>. STIsClosed () — Verifica se l'istanza geometrica è chiusa.
- <geometry_instance>. STIsValid () — Verifica se l'istanza geometrica è valida.
- <geometry_instance>. STDistance (other_geometry) — Calcola la distanza tra due istanze geometriche.
- <geometry_instance>. STEquals (other_geometry) — Verifica se l'istanza geometrica rappresenta lo stesso set di punti di un'altra istanza geometrica.
- <geometry_instance>. STContains (other_geometry) — Verifica se l'istanza geometrica contiene l'istanza other_geometry.
- <geometry_instance>. STDisjoint (other_geometry) — Verifica se due istanze geometriche non hanno punti in comune.
- <geometry_instance>. STIntersects (other_geometry) — Verifica se due istanze geometriche si intersecano spazialmente.
- <geometry_instance>.STX: estrae la coordinata X (longitudine) per l'istanza di dati geometrici.
- <geometry_instance>.STY: estrae la coordinata Y (latitudine) per l'istanza di dati geometrici.

A partire dalle versioni 4.7.0 e 5.3.0, Babelfish include il supporto per le seguenti funzioni relative ai dati spaziali:

- .M — Estrae la coordinata M <geometry_instance>dell'istanza geometrica.
- .Z — Estrae <geometry_instance>la coordinata Z dell'istanza geometrica.
- .hasM <geometry_instance>— Verifica se l'istanza geometrica ha almeno un valore M.
- .hasZ <geometry_instance>— Verifica se l'istanza geometrica ha almeno un valore Z.

A partire dalla versione 5.4.0, Babelfish include il supporto per la seguente funzione di dati spaziali:

- STLineFromText (***linestring_tagged_text***, SRID) — Crea un'istanza di linestring utilizzando la rappresentazione WKT.

Funzioni dei tipi di dati geografici supportate in Babelfish

- STGeomFromText (***geography_tagged_text***, SRID) — Crea un'istanza geografica utilizzando la rappresentazione WKT.
- STPointFromText (***point_tagged_text***, SRID) — Crea un'istanza puntuale utilizzando la rappresentazione WKT.
- Point (Lat, Long, SRID): crea un'istanza di punti dati utilizzando i valori in virgola mobile di latitudine e longitudine.
- <geography_instance>. STAsText () — Estrae la rappresentazione WKT dall'istanza geografica.
- <geography_instance>. STAsBinary (): estrae la rappresentazione WKB dall'istanza geografica.
- <geography_instance>. STArea () — Calcola la superficie totale dell'istanza geografica.
- <geography_instance>. STSrid () — Estrae l'identificatore di riferimento spaziale (SRID) dell'istanza geografica.
- <geography_instance>. STDimension () — Recupera la dimensione spaziale dell'istanza geografica.
- <geography_instance>. STIsEmpty (): verifica se l'istanza geografica è vuota.
- <geography_instance>. STIsChiusa (): verifica se l'istanza geografica è chiusa.
- <geography_instance>. STIsValid (): verifica se l'istanza geografica è valida.
- <geography_instance>. STDistance (other_geography) — Calcola la distanza tra due istanze geografiche.
- <geography_instance>. STEquals (other_geography) — Verifica se l'istanza geografica rappresenta lo stesso set di punti di un'altra istanza geografica.
- <geography_instance>. STContains (other_geography) — Verifica se l'istanza geography contiene l'istanza other_geography.
- <geography_instance>. STDisjoint (other_geography) — Verifica se due istanze geografiche non hanno punti in comune.
- <geography_instance>. STIntersects (other_geography) — Verifica se due istanze geografiche si intersecano spazialmente.
- <geography_instance>.Lat: estrae il valore di latitudine per l'istanza di dati geografici.
- <geography_instance>.Long: estrae il valore di longitudine per l'istanza di dati geografici.

A partire dalle versioni 4.7.0 e 5.3.0, Babelfish include il supporto per le seguenti funzioni relative ai dati spaziali:

- .M — Estrae la coordinata M <geography_instance>dell'istanza geografica.
- .Z — Estrae <geography_instance>la coordinata Z dell'istanza geografica.
- .hasM <geography_instance>— Verifica se l'istanza geografica ha almeno un valore M.
- .hasZ<geography_instance>: verifica se l'istanza geografica ha almeno un valore Z.

A partire dalla versione 5.4.0, Babelfish include il supporto per la seguente funzione di dati spaziali:

- STLineFromText (***linestring_tagged_text***, SRID) — Crea un'istanza di linestring utilizzando la rappresentazione WKT.

Limitazioni per i tipi di dati geospaziali in Babelfish

- I tipi di geometria e geografia diversi dalle istanze point e linestring non sono attualmente supportati:
  - CircularString
  - CompoundCurve
  - Poligono
  - CurvePolygon
  - MultiPoint
  - MultiLineString
  - MultiPolygon
  - GeometryCollection
- Attualmente, l'indicizzazione spaziale non è supportata per i tipi di dati geospaziali.
- Per questi tipi di dati al momento sono supportate solo le funzioni elencate. Per ulteriori informazioni, consultare Funzioni dei tipi di dati geometrici supportate in Babelfish e Funzioni dei tipi di dati geografici supportate in Babelfish.
- STDistance l'output della funzione per i dati geografici potrebbe presentare variazioni di precisione minori rispetto a T-SQL. Ciò è dovuto all'implementazione di PostGIS sottostante. Per ulteriori informazioni, consulta ST_Distance.
- STIsL'output di una funzione valido per i dati di geometria e geografia potrebbe presentare alcune deviazioni rispetto a T-SQL. Per questo motivo, in alcuni casi le funzioni - STDistance STContains,, STInstersects, STDisjoint, STDimension, STArea, STEquals possono anche differire da T-SQL

(restituiscono un output anziché generare errori). Ciò è dovuto all'implementazione di PostGIS sottostante. Per ulteriori informazioni, vedere ST_. IsValid

- Per prestazioni ottimali, si utilizzano i tipi di dati geospaziali incorporati, senza creare ulteriori livelli di astrazione in Babelfish.
- In Babelfish, i nomi delle funzioni geospaziali vengono utilizzati come parole chiave ed eseguono operazioni sui dati spaziali solo se utilizzate nel modo previsto.

Informazioni sul partizionamento in Babelfish

A partire dalla versione 4.3.0, Babelfish include il partizionamento di tabelle e indici con supporto limitato. Le sezioni seguenti forniscono i dettagli sulle funzioni di creazione delle partizioni, sulla definizione di schemi di partizione e sull'implementazione di tabelle e indici partizionati in Babelfish.

Argomenti

- Introduzione al partizionamento in Babelfish
- Limitazioni e soluzioni alternative

Introduzione al partizionamento in Babelfish

- Funzioni di partizione:
  - `CREATE PARTITION FUNCTION:` definisce il modo in cui una tabella o un indice vengono partizionati specificando il tipo di colonna di partizionamento e l'intervallo di valori per ciascuna partizione.
  - `DROP PARTITION FUNCTION:` rimuove una funzione di partizione esistente.
- Schemi di partizione:
  - `CREATE PARTITION SCHEME:` definisce la mappatura tra partizioni e gruppi di file.

    > **Note**
    >
    > In Babelfish, i gruppi di file vengono trattati come oggetti fittizi e non rappresentano posizioni di archiviazione fisiche.

  - `DROP PARTITION SCHEME:` rimuove uno schema di partizione esistente.
- Funzione di sistema:

  - `$PARTITION:` questa funzione di sistema restituisce il numero di partizione a cui appartiene un valore specificato in una colonna di partizionamento di una tabella partizionata specificata.
- Tabelle e indici partizionati:
  - `CREATE TABLE ... ON partition_scheme_name (partition_column_name):` crea una tabella partizionata basata su uno schema di partizione e una colonna di partizionamento specificati.
  - `CREATE INDEX ... ON partition_scheme_name (partition_column_name):` crea un indice partizionato basato su uno schema di partizione e una colonna di partizionamento specificati.
- Visualizzazioni di sistema per il partizionamento dei metadati:

  Le seguenti visualizzazioni di sistema sono state aggiunte per fornire i metadati relativi al partizionamento:
  - `sys.destination_data_spaces`
  - `sys.partitions`
  - `sys.partition_functions`
  - `sys.partition_parameters`
  - `sys.partition_range_values`
  - `sys.partition_schemes`

Limitazioni e soluzioni alternative

Le seguenti funzionalità di partizionamento di SQL Server non sono ancora supportate in Babelfish:

- `ALTER PARTITION FUNCTION` e `ALTER PARTITION SCHEME`.

  > **Note**
  >
  > Babelfish non supporta le operazioni di divisione e unione. Si definiscono tutte le partizioni nelle funzioni di partizione durante la creazione perché non è possibile aggiungere o rimuovere partizioni in un secondo momento.

- Colonne calcolate come colonne di partizionamento.
- Utilità `INSERT BULK` e `BCP` per tabelle partizionate.
- Opzione del limite `LEFT` per le funzioni di partizione.

- Tipo di dati `SQL_VARIANT` per le funzioni di partizione.
- `TRUNCATE TABLE ... WITH PARTITION`.
- `ALTER TABLE ... SWITCH PARTITION`.
- Indici partizionati non allineati come lo schema di partizione e la colonna di partizione che differiscono dalla tabella partizionata.
- La migrazione DMS da un'origine Babelfish è supportata solo per le attività di caricamento completo sulle tabelle partizionate.
- Babelfish non supporta queste opzioni di sintassi ma fornisce soluzioni alternative:
  - Utilizzo dello schema di partizione con vincoli o indici nell'istruzione CREATE TABLE.
  - ALTER TABLE ... ADD CONSTRAINT ... ON partition_scheme_name (partition_column_name).

> **Note**
>
> È possibile impostare l'escape hatch `babelfishpg_tsql.escape_hatch_storage_on_partition` per ignorare lo schema. Ciò consentirà al parser di ignorare l'opzione dello schema di partizione utilizzata con vincoli o indici e al backend di creare vincoli o indici individuali per ogni partizione.

Risoluzione dei problemi relativi a Babelfish

Di seguito, puoi trovare idee per la risoluzione dei problemi e soluzioni alternative del cluster database Babelfish.

Argomenti

- Errore della connessione

Errore della connessione

Di seguito, le cause più comuni dei problemi di connessione a un nuovo cluster di database Aurora:

- Il gruppo di sicurezza non consente l'accesso - Se hai problemi a connetterti a un Babelfish, assicurati di aver aggiunto il tuo indirizzo IP al gruppo di sicurezza Amazon EC2 predefinito. È possibile utilizzare https://checkip.amazonaws.com/ per stabilite il tuo indirizzo IP e quindi aggiungerlo alla regola in ingresso per la porta TDS e la porta PostgreSQL. Per ulteriori informazioni, consulta Aggiunta di regole a un gruppo di sicurezza nella Guida per l'utente di Amazon EC2.
- Configurazioni SSL non corrispondenti - Se il parametro `rds.force_ssl` è attivato (impostato su 1) su Aurora PostgreSQL, allora i client devono connettersi a Babelfish tramite SSL. Se il client non è configurato correttamente, visualizzerai un messaggio di errore del tipo seguente:

  ```
  Cannot connect to your-Babelfish-DB-cluster, 1433
  ---------------------
  ADDITIONAL INFORMATION:
  no pg_hba_conf entry for host "256.256.256.256", user "your-user-name",
  "database babelfish_db", SSL off (Microsoft SQL Server, Error: 33557097)
  ...
  ```

  Questo errore indica un possibile problema di configurazione SSL tra il client locale e il cluster di database Babelfish e che il cluster richiede che i client utilizzino SSL (il suo parametro `rds.force_ssl` è impostato su 1). Per ulteriori informazioni sulla configurazione di SSL, consulta Utilizzo di SSL con un'istanza database PostgreSQL nella Guida per l'utente di Amazon RDS.

  Se utilizzi SQL Server Management Studio (SSMS) per connetterti a Babelfish e vedi questo errore, puoi scegliere tra le opzioni di connessione Encrypt connection (Crittografa connessione) e Trust server certificate (Considera attendibile il certificato del server) nel riquadro Connection

Properties (Proprietà connessione) e riprova. Queste impostazioni gestiscono i requisiti di connessione SSL per SSMS.

Per ulteriori informazioni sulla risoluzione dei problemi di connessione al database Aurora, consulta Impossibile connettersi all'istanza database di Amazon RDS.

Disattivazione di Babelfish

Quando non hai più bisogno di Babelfish, puoi disattivare la funzionalità Babelfish.

Ricorda alcune considerazioni:

- In alcuni casi, potresti disattivare Babelfish prima di completare una migrazione ad Aurora PostgreSQL. Se lo fai e il tuo DDL dipende dai tipi di dati di SQL Server o utilizzi qualsiasi funzionalità T-SQL nel codice, il codice non va a buon fine.
- Se dopo aver eseguito il provisioning di un'istanza Babelfish si disattiva l'estensione Babelfish, non è possibile eseguire nuovamente il provisioning dello stesso database sullo stesso cluster.

Per disattivare Babelfish, modifica il gruppo di parametri, impostando `rds.babelfish_status` su `OFF`. È possibile continuare a utilizzare i tipi di dati di SQL Server con Babelfish disattivato, impostando `rds.babelfish_status` su `datatypesonly`.

Se disattivi Babelfish nel gruppo di parametri, tutti i cluster che utilizzano quel gruppo di parametri perdono la funzionalità Babelfish.

Per ulteriori informazioni sulla modifica dei parametri dei gruppi, consulta Gruppi di parametri per Amazon Aurora. Per informazioni sui parametri specifici di Babelfish, consulta Impostazioni del gruppo di parametri del cluster database per Babelfish.

Gestione degli aggiornamenti della versione di Babelfish per Aurora PostgreSQL

Babelfish è un'opzione disponibile con Aurora PostgreSQL versione 13.4 e successive. Gli aggiornamenti a Babelfish diventano disponibili con alcune nuove versioni del motore di database Aurora PostgreSQL. Per ulteriori informazioni, consulta Release notes for Amazon Aurora PostgreSQL-Compatible Edition.

> **Note**
>
> I cluster di database Babelfish in esecuzione su qualsiasi versione di Aurora PostgreSQL 13 non possono essere aggiornati ad Aurora PostgreSQL 14.3, 14.4 e 14.5. Inoltre, Babelfish non supporta l'aggiornamento diretto da 13.x a 15.x. È necessario prima aggiornare il cluster di database 13.x alla versione 14.6 e successiva e quindi alla versione 15.x.

Per un elenco di funzionalità supportate tra versioni di Babelfish differenti, consulta Funzionalità supportate in Babelfish per versione.

Per un elenco di funzionalità attualmente non supportate, consulta Funzionalità non supportate in Babelfish.

È possibile ottenere l'elenco delle versioni di Aurora PostgreSQL che supportano Babelfish eseguendo una query sulla Regione AWS utilizzando il comando AWS CLI describe-db-engine-versions, come segue.

Per Linux, macOS o Unix:

```
$ aws rds describe-db-engine-versions --region us-east-1 \
    --engine aurora-postgresql \
    --query '*[]|[?SupportsBabelfish==`true`].[EngineVersion]' \
    --output text
```

Per ulteriori informazioni, consulta la sezione describe-db-engine-versions nella Documentazione di riferimento della AWS CLI.

Nei seguenti argomenti viene illustrato come identificare la versione di Babelfish in esecuzione sul cluster di database Aurora PostgreSQL e come eseguire l'aggiornamento a una nuova versione.

Indice

- Identificazione della versione di Babelfish
- Aggiornamento del cluster Babelfish a una nuova versione
  - Aggiornamento di Babelfish a una nuova versione secondaria
  - Aggiornamento di Babelfish a una nuova versione principale
    - Operazioni preliminari all'aggiornamento di Babelfish a una nuova versione principale
    - Esecuzione dell'aggiornamento alla versione principale
    - Operazioni successive all'aggiornamento a una nuova versione principale
    - Esempio: aggiornamento del cluster di database Babelfish a una versione principale
- Utilizzo del parametro di versione del prodotto Babelfish
  - Configurazione del parametro di versione del prodotto Babelfish
  - Query e parametri interessati
  - Interfaccia con il parametro babelfishpg_tsql.version

Identificazione della versione di Babelfish

È possibile interrogare Babelfish per trovare dettagli sulla versione Babelfish, sulla versione di Aurora PostgreSQL e sulla versione compatibile di Microsoft SQL Server. Puoi utilizzare la porta TDS o la porta PostgreSQL.

- To use the TDS port to query for version information
- To use the PostgreSQL port to query for version information

Per utilizzare la porta TDS per eseguire le query e ottenere le informazioni sulla versione

1. Utilizza `sqlcmd` o `ssms` per connetterti all'endpoint per il cluster database Babelfish.

   ```
   sqlcmd -S bfish_db.cluster-123456789012.aws-region.rds.amazonaws.com,1433 -U
       login-id -P password -d db_name
   ```

2. Per identificare la versione Babelfish, esegui la query riportata di seguito:

   ```
   1> SELECT CAST(serverproperty('babelfishversion') AS VARCHAR)
   2> GO
   ```

   La query restituisce risultati simili a quanto esposto di seguito:

```
serverproperty
------------------------------
3.4.0

(1 rows affected)
```

3. Per identificare la versione del cluster database Aurora PostgreSQL, eseguire la seguente query:

```
1> SELECT aurora_version() AS aurora_version
2> GO
```

La query restituisce risultati simili a quanto esposto di seguito:

```
aurora_version


-----------------------------------------------
15.5.0

(1 rows affected)
```

4. Per identificare la versione compatibile di Microsoft SQL Server, eseguire la seguente query:

```
1> SELECT @@VERSION AS version
2> GO
```

La query restituisce risultati simili a quanto esposto di seguito:

```
Babelfish for Aurora PostgreSQL with SQL Server Compatibility - 12.0.2000.8
Dec 7 2023 09:43:06
Copyright (c) Amazon Web Services
PostgreSQL 15.5 on x86_64-pc-linux-gnu (Babelfish 3.4.0)

(1 rows affected)
```

Come un esempio che mostra una differenza minore tra Babelfish e Microsoft SQL Server, puoi eseguire la seguente query. Su Babelfish, la query restituisce 1, mentre su Microsoft SQL Server, la query restituisce NULL.

```
SELECT CAST(serverproperty('babelfish') AS VARCHAR) AS runs_on_babelfish
```

È inoltre possibile utilizzare la porta PostgreSQL per ottenere informazioni sulla versione, come mostrato nella procedura seguente.

Per utilizzare la porta PostgreSQL per eseguire le query e ottenere le informazioni sulla versione

1. Utilizza `psql` o `pgAdmin` per connetterti all'endpoint per il cluster database Babelfish.

   ```
   psql host=bfish_db.cluster-123456789012.aws-region.rds.amazonaws.com
        port=5432 dbname=babelfish_db user=sa
   ```

2. Attiva la funzionalità estesa (`\x`) di `psql` per un output più leggibile.

   ```
   babelfish_db=> \x
   babelfish_db=> SELECT
   babelfish_db=> aurora_version() AS aurora_version,
   babelfish_db=> version() AS postgresql_version,
   babelfish_db=> sys.version() AS Babelfish_compatibility,
   babelfish_db=> sys.SERVERPROPERTY('BabelfishVersion') AS Babelfish_Version;
   ```

   La query restituisce un output simile al seguente:

   ```
   -[ RECORD 1 ]-----------
   +-------------------------------------------------------------------------------------------
   aurora_version          | 15.5.0
   postgresql_version      | PostgreSQL 15.5 on x86_64-pc-linux-gnu, compiled by
    x86_64-pc-linux-gnu-gcc (GCC) 9.5.0, 64-bit
   babelfish_compatibility | Babelfish for Aurora Postgres with SQL Server
    Compatibility - 12.0.2000.8                    +
                           | Dec 7 2023 09:43:06
                                      +
                           | Copyright (c) Amazon Web Services
                                       +
                           | PostgreSQL 15.5 on x86_64-pc-linux-gnu (Babelfish 3.4.0)
   babelfish_version       | 3.4.0
   ```

Aggiornamento del cluster Babelfish a una nuova versione

Le nuove versioni di Babelfish diventano disponibili con i nuovi rilasci del motore di database Aurora PostgreSQL a partire dalla versione 13.4. Ogni nuovo rilascio di Babelfish ha un proprio numero di versione. Come Aurora PostgreSQL, anche Babelfish utilizza per le versioni lo schema di denominazione *versione principale*.*versione secondaria*.*patch*. La prima versione di Babelfish ad esempio, Babelfish 1.0.0, è diventata disponibile come parte di Aurora PostgreSQL 13.4.0.

Babelfish non richiede un processo di installazione separato e, come indicato in Creazione di un cluster database Babelfish per Aurora PostgreSQL, Turn on Babelfishu (Attiva Babelfish) è un'opzione che si sceglie quando si crea un cluster di database Aurora PostgreSQL.

Analogamente, non è possibile aggiornare Babelfish in modo indipendente dal cluster di database Aurora che lo supporta. Per aggiornare un cluster di database Babelfish per Aurora PostgreSQL esistente a una nuova versione di Babelfish, occorre aggiornare il cluster di database Aurora PostgreSQL a una nuova versione che supporti la versione di Babelfish da utilizzare. La procedura da seguire per l'aggiornamento dipende dalla versione di Aurora PostgreSQL che supporta l'implementazione di Babelfish, come indicato di seguito.

Aggiornamenti di una versione principale

È necessario aggiornare le seguenti versioni di Aurora PostgreSQL ad Aurora PostgreSQL 14.6 e versioni successive prima di eseguire l'aggiornamento alla versione Aurora PostgreSQL 15.2.

- Aurora PostgreSQL 13.8 e tutte le versioni successive
- Aurora PostgreSQL 13.7.1 e tutte le versioni secondarie successive
- Aurora PostgreSQL 13.6.4 e tutte le versioni secondarie successive

Puoi aggiornare Aurora PostgreSQL 14.6 e versioni successive ad Aurora PostgreSQL 15.2 e versioni successive.

L'aggiornamento di un cluster di database Aurora PostgreSQL a una nuova versione principale richiede diverse attività preliminari. Per ulteriori informazioni, consulta Esecuzione di un aggiornamento alla versione principale. Per aggiornare correttamente il cluster di database Babelfish per Aurora PostgreSQL, è necessario creare un gruppo di parametri cluster di database personalizzato per la nuova versione di Aurora PostgreSQL. Questo nuovo gruppo di parametri deve contenere le impostazioni dei parametri Babelfish uguali a quelle del cluster da aggiornare. Per ulteriori informazioni e per esaminare una tabella con gli aggiornamenti delle versioni

principali di origine e di destinazione, consulta Aggiornamento di Babelfish a una nuova versione principale.

Aggiornamenti della versione secondaria e patch

Per l'aggiornamento delle versioni secondarie e delle patch non è richiesta la creazione di un nuovo gruppo di parametri cluster di database e si può utilizzare il processo di aggiornamento della versione secondaria, che può essere eseguito automaticamente o manualmente. Per ulteriori informazioni e per esaminare una tabella con le versioni di origine e di destinazione, consulta Aggiornamento di Babelfish a una nuova versione secondaria.

> **Note**
>
> Prima di eseguire un aggiornamento della versione principale o secondaria, completa tutte le attività di manutenzione in sospeso per il cluster Babelfish per Aurora PostgreSQL.

Argomenti

- Aggiornamento di Babelfish a una nuova versione secondaria
- Aggiornamento di Babelfish a una nuova versione principale

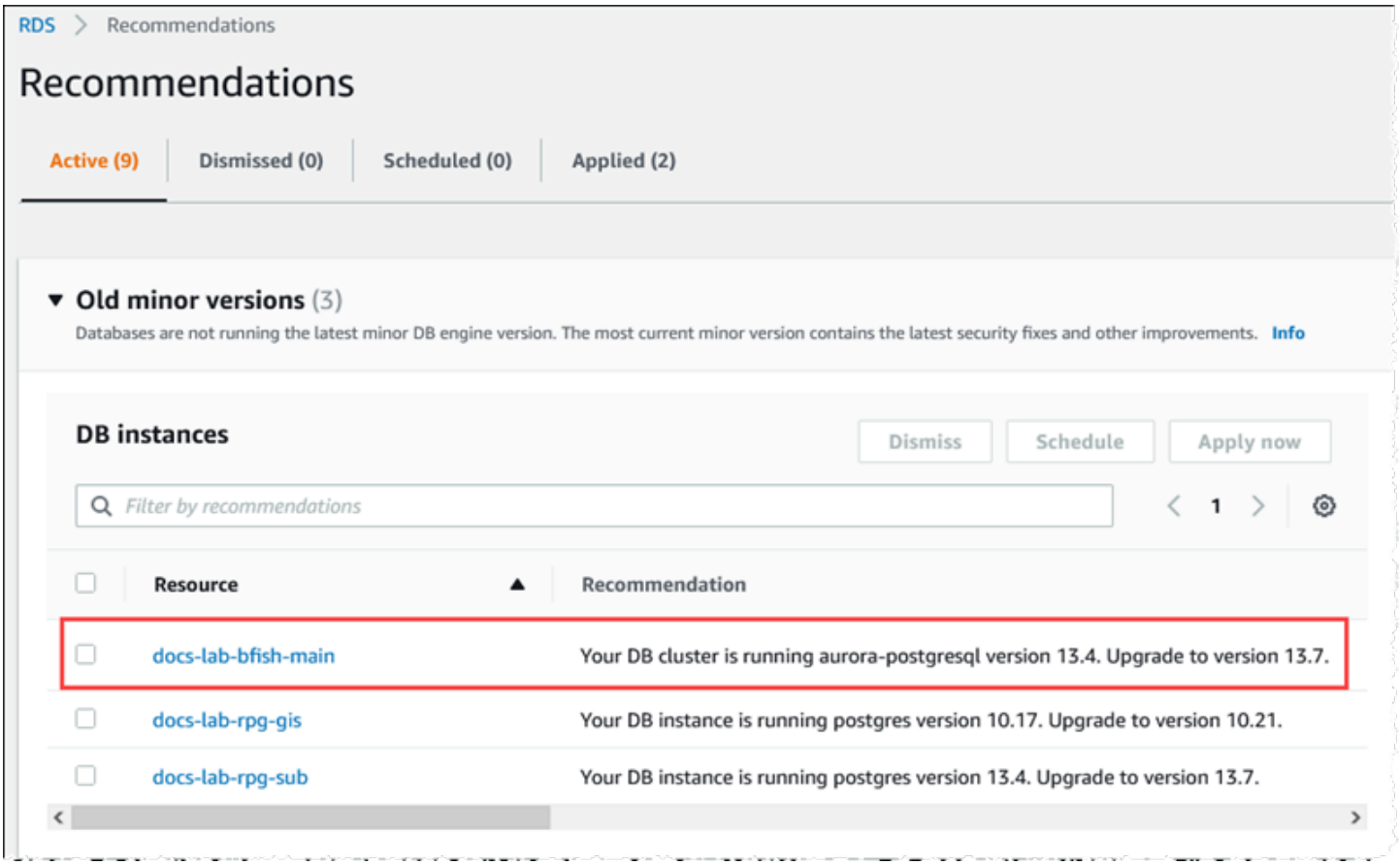
Aggiornamento di Babelfish a una nuova versione secondaria

L'aggiornamento della versione secondaria mira a mantenere la compatibilità con le versioni precedenti. Tuttavia, in alcuni casi, le correzioni di sicurezza cruciali o le correzioni di bug importanti potrebbero non essere completamente compatibili con le versioni precedenti. una patch include importanti correzioni per una versione secondaria rilasciata. Ad esempio, l'etichetta di versione del primo rilascio di Aurora PostgreSQL 13.4 era Aurora PostgreSQL 13.4.0 e successivamente sono state rilasciate diverse patch per la versione secondaria, tra cui Aurora PostgreSQL 13.4.1, 13.4.2 e 13.4.4. Le patch disponibili per ogni versione di Aurora PostgreSQL sono indicate nell'elenco Patch releases (Rilasci di patch), riportato all'inizio delle note di rilascio della versione in questione di Aurora PostgreSQL. Per un esempio, consulta PostgreSQL 14.3 nelle Note di rilascio per Aurora PostgreSQL.

Se il cluster di database Aurora PostgreSQL è configurato con l'opzione Auto minor version upgrade (Aggiornamento automatico versione secondaria), il cluster di database Babelfish per Aurora PostgreSQL viene aggiornato automaticamente durante la finestra di manutenzione del cluster.

Per ulteriori informazioni sull'aggiornamento automatico della versione secondaria (AmVU) e su come utilizzarlo, consulta Aggiornamenti automatici delle versioni secondarie per i cluster di database Aurora. Se non usi l'aggiornamento automatico della versione secondaria, puoi aggiornare manualmente il cluster database Babelfish per Aurora PostgreSQL a nuove versioni secondarie tramite le attività di manutenzione o la modifica del cluster in modo da utilizzare la nuova versione.

Quando si sceglie una versione di Aurora PostgreSQL da installare e si visualizza un cluster di database Aurora PostgreSQL esistente nella Console di gestione AWS, la versione mostra solo le cifre nello schema *`versione principale`*.*`versione secondaria`*. Ad esempio, nell'immagine seguente della console puoi vedere che per un cluster di database Babelfish per Aurora PostgreSQL con Aurora PostgreSQL 13.4 si consiglia l'aggiornamento alla versione 13.7, una nuova versione secondaria di Aurora PostgreSQL.



Per ottenere i dettagli completi della versione, incluso il livello di *`patch`*, è possibile eseguire una query sul cluster di database Aurora PostgreSQL utilizzando la funzione di Aurora PostgreSQL `aurora_version`. Per ulteriori informazioni, consulta aurora_version nella Riferimenti relativi alle funzioni Aurora PostgreSQL. Un esempio di utilizzo della funzione è disponibile nella procedura To use the PostgreSQL port to query for version information in Identificazione della versione di Babelfish.

La tabella seguente mostra le versioni di Aurora PostgreSQL e Babelfish di origine e le versioni di destinazione disponibili che supportano il processo di aggiornamento della versione secondaria.

Versioni di origine correnti	Versioni di destinazione dell'aggiornamento più recente
Aurora PostgreSQL (Babelfish)	Aurora PostgreSQL (Babelfish)
17.4 (5.1)	17.5 (5.2)
16.8 (4.5)	16.9 (4.6)
16.6 (4.4.0)	16.9 (4.6), 16.8 (4.5.0)
16.4 (4.3.0)	16.9 (4.6), 16.8 (4.5.0), 16.6 (4.4.0)
16.3 (4.2.0)	16.9 (4.6), 16.8 (4.5.0), 16.6 (4.4.0), 16.4 (4.3.0)
16.2 (4.1.0)	16.9 (4.6), 16.8 (4.5.0), 16.6 (4.4.0), 16.4 (4.3.0), 16.3 (4.2.0)
16.1 (4.0.0)	16.9 (4.6), 16.8 (4.5.0), 16.6 (4.4.0), 16.4 (4.3.0), 16.3 (4.2.0), 16.2 (4.1.0)
15.12 (3.9.0)	15.13 (3.10)
15.10 (3.8.0)	15.13 (3.10), 15.12 (3.9.0)
15.8 (3.7.0)	15.13 (3.10), 15.12 (3.9.0), 15.10 (3.8.0)
15.7 (3.6.0)	15.13 (3.10), 15.12 (3.9.0), 15.10 (3.8.0), 15.8 (3.7.0)
15.6 (3.5.0)	15.13 (3.10), 15.12 (3.9.0), 15.10 (3.8.0), 15.8 (3.7.0), 15.7 (3.6.0)
15.5 (3.4.0)	15.13 (3.10), 15.12 (3.9.0), 15.10 (3.8.0), 15.8 (3.7.0), 15.7 (3.6.0), 15.6 (3.5.0)

Versioni di origine correnti	Versioni di destinazione dell'aggiornamento più recente
15.4 (3.3.0)	15.13 (3.10), 15.12 (3.9.0), 15.10 (3.8.0), 15.8 (3.7.0), 15.7 (3.6.0), 15.6 (3.5.0), 15.5 (3.4.0)
15.12 (3.9.0), 15.3 (3.2.0)	15.13 (3.10), 15.12 (3.9.0), 15.10 (3.8.0), 15.8 (3.7.0), 15.7 (3.6.0), 15.6 (3.5.0), 15.5 (3.4.0), 15.4 (3.3.0)
15.2 (3.1.0)	15.13 (3.10), 15.12 (3.9.0), 15.10 (3.8.0), 15.8 (3.7.0), 15.7 (3.6.0), 15.6 (3.5.0), 15.5 (3.4.0), 15.4 (3.3.0), 15.3 (3.2.0)
14.17 (2.12.0)	14.18 (2.13.0)
14.15 (2.11.0)	14.18 (2.13.0), 14.17 (2.12.0)
14.13 (2.10.0)	14.18 (2.13.0), 14.17 (2.12.0), 14.15 (2.11.0)
14.12 (2.9.0)	14.18 (2.13.0), 14.17 (2.12.0), 14.15 (2.11.0), 14.13 (2.10.0)
14.11 (2.8.0)	14.18 (2.13.0), 14.17 (2.12.0), 14.15 (2.11.0), 14.13 (2.10.0), 14.12 (2.9.0)
14.10 (2.7.0)	14.18 (2.13.0), 14.17 (2.12.0), 14.15 (2.11.0), 14.13 (2.10.0), 14.12 (2.9.0), 14.11 (2.8.0)
14.9 (2.6.0)	14.18 (2.13.0), 14.17 (2.12.0), 14.15 (2.11.0), 14.13 (2.10.0), 14.12 (2.9.0), 14.11 (2.8.0), 14.10 (2.7.0)
14.8 (2.5.0)	14.18 (2.13.0), 14.17 (2.12.0), 14.15 (2.11.0), 14.13 (2.10.0), 14.12 (2.9.0), 14.11 (2.8.0), 14.10 (2.7.0), 14.9 (2.6.0)

Versioni di origine correnti	Versioni di destinazione dell'aggiornamento più recente
14.7 (2.4.0)	14.18 (2.13.0), 14.17 (2.12.0), 14.15 (2.11.0), 14.13 (2.10.0), 14.12 (2.9.0), 14.11 (2.8.0), 14.10 (2.7.0), 14.9 (2.6.0), 14.8 (2.5.0)
14.6 (2.3.0)	14.18 (2.13.0), 14.17 (2.12.0), 14.15 (2.11.0), 14.13 (2.10.0), 14.12 (2.9.0), 14.11 (2.8.0), 14.10 (2.7.0), 14.9 (2.6.0), 14.8 (2.5.0), 14.7 (2.4.0)
14.5 (2.2.0)	14.18 (2.13.0), 14.17 (2.12.0), 14.15 (2.11.0), 14.13 (2.10.0), 14.12 (2.9.0), 14.11 (2.8.0), 14.10 (2.7.0), 14.9 (2.6.0), 14.8 (2.5.0), 14.7 (2.4.0), 14.6 (2.3.0)
14.3 (2.1.0)	14.18 (2.13.0), 14.6 (2.3.0)
13.8 (1.4.0)	13.9 (1.5.0)
13,7 (1.3.0)	13.9 (1.5.0), 13.8 (1.4.0)

Aggiornamento di Babelfish a una nuova versione principale

Per eseguire l'aggiornamento della versione principale, è necessario prima aggiornare il cluster di database Babelfish per Aurora PostgreSQL a una versione che supporti l'aggiornamento della versione principale e per farlo, gli aggiornamenti di patch o della versione secondaria devono essere applicati al cluster di database. Per ulteriori informazioni, consulta Aggiornamento di Babelfish a una nuova versione secondaria.

La tabella seguente mostra le versioni di Aurora PostgreSQL e di Babelfish che possono supportare l'aggiornamento della versione principale.

Versioni di origine correnti	Versioni di destinazione dell'aggiornamento più recente
Aurora PostgreSQL (Babelfish)	Aurora PostgreSQL (Babelfish)

Versioni di origine correnti	Versioni di destinazione dell'aggiornamento più recente
16.9 (4.6.0)	17.5 (5.2.0)
16.8 (4.5.0)	17,5 (5.2.0), 17.4 (5.1.0)
16.6 (4.4.0)	17.5 (5.2.0), 17.4 (5.1.0)
16.4 (4.3.0)	17.5 (5.2.0), 17.4 (5.1.0)
16.3 (4.2.0)	17.5 (5.2.0), 17.4 (5.1.0)
16.2 (4.1.0)	17.5 (5.2.0), 17.4 (5.1.0)
16.1 (4.0.0)	17.5 (5.2.0), 17.4 (5.1.0)
15.13 (3.10)	17.5 (5.2.0) 16.9 (4.6.0)
15.12 (3.9.0)	17.5 (5.2.0), 17.4 (5.1.0) 16.9 (4.6.0), 16.8 (4.5.0)
15.10 (3.8.0)	17.5 (5.2.0), 17.4 (5.1.0) 16.9 (4.6.0), 16.8 (4.5.0), 16.6 (4.4.0)
15.8 (3.7.0)	17.5 (5.2.0), 17.4 (5.1.0) 16.9 (4.6.0), 16.8 (4.5.0), 16.6 (4.4.0), 16.4 (4.3.0)
15.7 (3.6.0)	17.5 (5.2.0), 17.4 (5.1.0) 16.9 (4.6.0), 16.8 (4.5.0), 16.6 (4.4.0), 16.4 (4.3.0), 16.3 (4.2.0)

Versioni di origine correnti	Versioni di destinazione dell'aggiornamento più recente
15.6 (3.5.0)	17.5 (5.2.0), 17.4 (5.1.0) 16.9 (4.6.0), 16.8 (4.5.0), 16.6 (4.4.0), 16.4 (4.3.0), 16.3 (4.2.0), 16.2 (4.1.0)
15.5 (3.4.0)	17.5 (5.2.0), 17.4 (5.1.0) 16.9 (4.6.0), 16.8 (4.5.0), 16.6 (4.4.0), 16.4 (4.3.0), 16.3 (4.2.0), 16.2 (4.1.0), 16.1 (4.0.0)
15.4 (3.3.0)	17.5 (5.2.0), 17.4 (5.1.0) 16.9 (4.6.0), 16.8 (4.5.0), 16.6 (4.4.0), 16.4 (4.3.0), 16.3 (4.2.0), 16.2 (4.1.0), 16.1 (4.0.0)
15.3 (3.2.0)	17.5 (5.2.0), 17.4 (5.1.0) 16.9 (4.6.0), 16.8 (4.5.0), 16.6 (4.4.0), 16.4 (4.3.0), 16.3 (4.2.0), 16.2 (4.1.0), 16.1 (4.0.0)
15.2 (3.1.0)	17.5 (5.2.0), 17.4 (5.1.0) 16.9 (4.6.0), 16.8 (4.5.0), 16.6 (4.4.0), 16.4 (4.3.0), 16.3 (4.2.0), 16.2 (4.1.0), 16.1 (4.0.0)
14.18 (2.13.0)	17.5 (5.2.0) 16.9 (4.6.0) 15.13 (3.10.0)
14.17 (2.12.0)	17.5 (5.2.0), 17.4 (5.1.0) 16.9 (4.6.0), 16.8 (4.5.0) 15.13 (3.10.0), 15.12 (3.9.0)

Versioni di origine correnti	Versioni di destinazione dell'aggiornamento più recente
14.15 (2.11.0)	17.5 (5.2.0), 17.4 (5.1.0) 16.9 (4.6.0), 16.8 (4.5.0), 16.6 (4.4.0) 15.13 (3.10.0), 15.12 (3.9.0), 15.10 (3.8.0)
14.13 (2.10.0)	17.5 (5.2.0), 17.4 (5.1.0) 16.9 (4.6.0), 16.8 (4.5.0), 16.6 (4.4.0), 16.4 (4.3.0) 15.13 (3.10.0), 15.12 (3.9.0), 15.10 (3.8.0), 15.8 (3.7.0)
14.12 (2.9.0)	17.5 (5.2.0), 17.4 (5.1.0) 16.9 (4.6.0), 16.8 (4.5.0), 16.6 (4.4.0), 16.4 (4.3.0), 16.3 (4.2.0) 15.13 (3.10.0), 15.12 (3.9.0), 15.10 (3.8.0), 15.8 (3.7.0), 15.7 (3.6.0)
14.11 (2.8.0)	17.5 (5.2.0), 17.4 (5.1.0) 16.9 (4.6.0), 16.8 (4.5.0), 16.6 (4.4.0), 16.4 (4.3.0), 16.3 (4.2.0), 16.2 (4.1.0) 15.13 (3.10.0), 15.12 (3.9.0), 15.10 (3.8.0), 15.8 (3.7.0), 15.7 (3.6.0), 15.6 (3.5.0)
14.10 (2.7.0)	17.5 (5.2.0), 17.4 (5.1.0) 16.9 (4.6.0), 16.8 (4.5.0), 16.6 (4.4.0), 16.4 (4.3.0), 16.3 (4.2.0), 16.2 (4.1.0), 16.1 (4.0.0) 15.13 (3.10.0), 15.12 (3.9.0), 15.10 (3.8.0), 15.8 (3.7.0), 15.7 (3.6.0), 15.6 (3.5.0), 15.5 (3.4.0)

Versioni di origine correnti	Versioni di destinazione dell'aggiornamento più recente
14.9 (2.6.0)	17.5 (5.2.0), 17.4 (5.1.0) 16.9 (4.6.0), 16.8 (4.5.0), 16.6 (4.4.0), 16.4 (4.3.0), 16.3 (4.2.0), 16.2 (4.1.0), 16.1 (4.0.0) 15.13 (3.10.0), 15.12 (3.9.0), 15.10 (3.8.0), 15.8 (3.7.0), 15.7 (3.6.0), 15.6 (3.5.0), 15.5 (3.4.0), 15.4 (3.3.0)
14.8 (2.5.0)	17.5 (5.2.0), 17.4 (5.1.0) 16.9 (4.6.0), 16.8 (4.5.0), 16.6 (4.4.0), 16.4 (4.3.0), 16.3 (4.2.0), 16.2 (4.1.0), 16.1 (4.0.0) 15.13 (3.10.0), 15.12 (3.9.0), 15.10 (3.8.0), 15.8 (3.7.0), 15.7 (3.6.0), 15.6 (3.5.0), 15.5 (3.4.0), 15.4 (3.3.0), 15.3 (3.2.0)
14.7 (2.4.0)	17.5 (5.2.0), 17.4 (5.1.0) 16.9 (4.6.0), 16.8 (4.5.0), 16.6 (4.4.0), 16.4 (4.3.0), 16.3 (4.2.0), 16.2 (4.1.0), 16.1 (4.0.0) 15.13 (3.10.0), 15.12 (3.9.0), 15.10 (3.8.0), 15.8 (3.7.0), 15.7 (3.6.0), 15.6 (3.5.0), 15.5 (3.4.0), 15.4 (3.3.0), 15.3 (3.2.0), 15.2 (3.1.0)
14.6 (2.3.0)	17.5 (5.2.0), 17.4 (5.1.0) 16.9 (4.6.0), 16.8 (4.5.0), 16.6 (4.4.0), 16.4 (4.3.0), 16.3 (4.2.0), 16.2 (4.1.0), 16.1 (4.0.0) 15.13 (3.10.0), 15.12 (3.9.0), 15.10 (3.8.0), 15.8 (3.7.0), 15.7 (3.6.0), 15.6 (3.5.0), 15.5 (3.4.0), 15.4 (3.3.0), 15.3 (3.2.0), 15.2 (3.1.0)
13.9 (1.5.0)	14.6 (2.3.0)

Versioni di origine correnti	Versioni di destinazione dell'aggiornamento più recente
13.8 (1.4.0)	14.6 (2.3.0)
13,7 (1.3.0)	14.6 (2.3.0)

Operazioni preliminari all'aggiornamento di Babelfish a una nuova versione principale

Un aggiornamento può comportare brevi interruzioni e per questo motivo si consiglia vivamente di eseguire o pianificare gli aggiornamenti durante la finestra di manutenzione o durante altri periodi di scarso utilizzo del sistema.

Operazioni preliminari all'aggiornamento della versione principale

1. Identifica la versione di Babelfish del cluster di database Aurora PostgreSQL esistente utilizzando i comandi descritti in Identificazione della versione di Babelfish. Le informazioni sulla versione di Aurora PostgreSQL e sulla versione di Babelfish sono gestite da PostgreSQL, quindi segui i passaggi illustrati nella procedura To use the PostgreSQL port to query for version information per i dettagli.
2. Verifica se la versione supporta l'aggiornamento della versione principale. Per l'elenco delle versioni che supportano la funzionalità di aggiornamento della versione principale, consulta Aggiornamento di Babelfish a una nuova versione secondaria ed esegui le attività necessarie prima dell'aggiornamento.

   Se, ad esempio, la versione di Babelfish è in esecuzione su un cluster di database Aurora PostgreSQL 13.5 e desideri eseguire l'aggiornamento ad Aurora PostgreSQL 15.2, devi prima applicare tutte le patch e le versioni secondarie per aggiornare il cluster ad Aurora PostgreSQL 14.6 o versione successiva. Una volta che il cluster è aggiornato alla versione 14.6 o successiva, puoi continuare il processo di aggiornamento della versione principale.
3. Crea uno snapshot manuale del cluster di database Babelfish corrente come backup, che ti consente di ripristinare il cluster alla versione di Aurora PostgreSQL e di Babelfish precedente l'aggiornamento, compresi tutti i dati. Per ulteriori informazioni, consulta Creazione di uno snapshot del cluster database. Assicurati di mantenere il gruppo di parametri cluster di database personalizzato per riutilizzarlo se decidi di ripristinare il cluster allo stato precedente l'aggiornamento. Per ulteriori informazioni, consultare Ripristino da uno snapshot cluster database e Considerazioni sui gruppi di parametri.

4. Prepara un gruppo di parametri cluster di database personalizzato per la versione del database Aurora PostgreSQL di destinazione. Duplica le impostazioni dei parametri Babelfish dal cluster di database Babelfish per Aurora PostgreSQL corrente. Per l'elenco di tutti i parametri Babelfish, consulta Impostazioni del gruppo di parametri del cluster database per Babelfish. Per l'aggiornamento della versione principale, è necessario che i seguenti parametri abbiano le stesse impostazioni del cluster di database di origine. Tutte le impostazioni devono essere uguali per poter completare l'aggiornamento.

   - rds.babelfish_status
   - babelfishpg_tds.tds_default_numeric_precision
   - babelfishpg_tds.tds_default_numeric_scale
   - babelfishpg_tsql.database_name
   - babelfishpg_tsql.default_locale
   - babelfishpg_tsql.migration_mode
   - babelfishpg_tsql.server_collation_name

   > **Warning**
   >
   > Se nel gruppo di parametri cluster di database personalizzato per la nuova versione di Aurora PostgreSQL le impostazioni dei parametri Babelfish non corrispondono ai valori dei parametri del cluster che stai aggiornando, l'operazione `ModifyDBCluster` non riesce e viene visualizzato il messaggio di errore `InvalidParameterCombination` nella Console di gestione AWS o nell'output del comando AWS CLI `modify-db-cluster`.

5. Utilizza la Console di gestione AWS o AWS CLI per creare il gruppo di parametri cluster di database personalizzato. Scegli la famiglia di Aurora PostgreSQL applicabile per la versione di Aurora PostgreSQL che desideri per l'aggiornamento.

   > **Tip**
   >
   > I gruppi di parametri sono gestiti a livello di Regione AWS. Se utilizzi AWS CLI, puoi configurare una regione predefinita invece di specificare `--region` nel comando. Per ulteriori informazioni sull'utilizzo di AWS CLI, consulta Quick setup (Configurazione rapida) nella Guida per l'utente di AWS Command Line Interface.

Esecuzione dell'aggiornamento alla versione principale

1. Aggiorna il cluster di database Aurora PostgreSQL a una nuova versione principale. Per ulteriori informazioni, consulta Aggiornamento del motore Aurora PostgreSQL a una nuova versione principale.
2. Riavvia l'istanza di scrittura del cluster per rendere effettive le impostazioni dei parametri.

Operazioni successive all'aggiornamento a una nuova versione principale

Dopo l'aggiornamento della versione principale a una nuova versione di Aurora PostgreSQL, il valore `IDENTITY` nelle tabelle con una colonna `IDENTITY` potrebbe essere maggiore (+32) del valore precedente l'aggiornamento. Di conseguenza, quando la riga successiva viene inserita nelle tabelle, il valore della colonna Identity generata diventa +32 e la sequenza inizia da quel numero. Questa condizione non influisce negativamente sulle funzioni del cluster di database Babelfish. Tuttavia, se lo desideri, puoi reimpostare l'oggetto sequenza in base al valore massimo della colonna e per farlo, ti devi connettere alla porta T-SQL nell'istanza di scrittura Babelfish utilizzando `sqlcmd` o un altro client SQL Server. Per ulteriori informazioni, consulta Utilizzo di un client SQL Server per la connessione al cluster database.

```
sqlcmd -S bfish-db.cluster-123456789012.aws-region.rds.amazonaws.com,1433 -U
     sa -P ******** -d dbname
```

Una volta stabilita la connessione, usa il seguente comando SQL per generare le istruzioni che puoi utilizzare per inizializzare l'oggetto sequenza associato. Questo comando SQL funziona per le configurazioni Babelfish con un singolo database e più database. Per ulteriori informazioni su questi due modelli di implementazione, consulta Utilizzo di Babelfish con un singolo database o più database.

```
DECLARE @schema_prefix NVARCHAR(200) = ''
IF current_setting('babelfishpg_tsql.migration_mode') = 'multi-db'
    SET @schema_prefix = db_name() + '_'
SELECT 'SELECT setval(pg_get_serial_sequence(''' + @schema_prefix +
 schema_name(tables.schema_id)
    + '.' + tables.name + ''', ''' + columns.name + '''),(select max(' + columns.name +
 ')
    FROM ' + schema_name(tables.schema_id) + '.' + tables.name + '));
    'FROM sys.tables tables JOIN sys.columns
    columns ON tables.object_id = columns.object_id
    WHERE columns.is_identity = 1
```

```
GO
```

La query genera una serie di istruzioni SELECT che è possibile eseguire per reimpostare il valore massimo di IDENTITY e risolvere eventuali problemi di sequenza. Di seguito è riportato l'output del database SQL Server di esempio Northwind in esecuzione su un cluster Babelfish.

```
------------------------------------------------------
SELECT setval(pg_get_serial_sequence('northwind_dbo.categories', 'categoryid'),(select
 max(categoryid)
    FROM dbo.categories));

SELECT setval(pg_get_serial_sequence('northwind_dbo.orders', 'orderid'),(select
 max(orderid)
    FROM dbo.orders));


SELECT setval(pg_get_serial_sequence('northwind_dbo.products', 'productid'),(select
 max(productid)
    FROM dbo.products));


SELECT setval(pg_get_serial_sequence('northwind_dbo.shippers', 'shipperid'),(select
 max(shipperid)
    FROM dbo.shippers));


SELECT setval(pg_get_serial_sequence('northwind_dbo.suppliers', 'supplierid'),(select
 max(supplierid)
    FROM dbo.suppliers));


(5 rows affected)
```

Esegui le istruzioni una alla volta per reimpostare i valori della sequenza.

Esempio: aggiornamento del cluster di database Babelfish a una versione principale

In questo esempio, puoi trovare la serie di comandi AWS CLI che spiega come aggiornare ad Aurora PostgreSQL 14.6 un cluster di database Aurora PostgreSQL 13.6.4 che esegue Babelfish versione 1.2.2. Per prima cosa crea un gruppo di parametri cluster di database personalizzato per Aurora PostgreSQL 14. Quindi, modifica i valori dei parametri in modo che corrispondano a quelli della versione Aurora PostgreSQL 13 di origine. Infine, esegui l'aggiornamento modificando il cluster di

origine. Per ulteriori informazioni, consulta Impostazioni del gruppo di parametri del cluster database per Babelfish. In questo argomento sono disponibili anche le informazioni sull'utilizzo della Console di gestione AWS per eseguire l'aggiornamento.

Usa il comando CLI create-db-cluster-parameter-group per creare il gruppo di parametri cluster di database per la nuova versione.

Per Linux, macOS o Unix:

```
aws rds create-db-cluster-parameter-group \
    --db-cluster-parameter-group-name docs-lab-babelfish-apg-14 \
    --db-parameter-group-family aurora-postgresql14 \
    --description 'New custom parameter group for upgrade to new major version' \
    --region us-west-1
```

Quando si esegue questo comando, il gruppo di parametri cluster di database personalizzato viene creato nella Regione AWS. L'output visualizzato è simile al seguente.

```
{
    "DBClusterParameterGroup": {
        "DBClusterParameterGroupName": "docs-lab-babelfish-apg-14",
        "DBParameterGroupFamily": "aurora-postgresql14",
        "Description": "New custom parameter group for upgrade to new major version",
        "DBClusterParameterGroupArn": "arn:aws:rds:us-west-1:111122223333:cluster-
pg:docs-lab-babelfish-apg-14"
    }
}
```

Per ulteriori informazioni, consulta Creazione di un gruppo di parametri del cluster di database in Amazon Aurora.

Usa il comando CLI modify-db-cluster-parameter-group per modificare le impostazioni in modo che corrispondano a quelle del cluster di origine.

Per Windows:

```
aws rds modify-db-cluster-parameter-group --db-cluster-parameter-group-name docs-lab-
babelfish-apg-14 ^
  --parameters
 "ParameterName=rds.babelfish_status,ParameterValue=on,ApplyMethod=pending-reboot" ^
```

```
 "ParameterName=babelfishpg_tds.tds_default_numeric_precision,ParameterValue=38,ApplyMethod=pen
reboot" ^
 "ParameterName=babelfishpg_tds.tds_default_numeric_scale,ParameterValue=8,ApplyMethod=pending-
reboot" ^
 "ParameterName=babelfishpg_tsql.database_name,ParameterValue=babelfish_db,ApplyMethod=pending-
reboot" ^
 "ParameterName=babelfishpg_tsql.default_locale,ParameterValue=en-
US,ApplyMethod=pending-reboot" ^
 "ParameterName=babelfishpg_tsql.migration_mode,ParameterValue=single-
db,ApplyMethod=pending-reboot" ^
 "ParameterName=babelfishpg_tsql.server_collation_name,ParameterValue=sql_latin1_general_cp1_ci
reboot"
```

La risposta è simile a quella riportata di seguito.

```
{
    "DBClusterParameterGroupName": "docs-lab-babelfish-apg-14"
}
```

Usa il comando CLI modify-db-cluster per modificare il cluster in modo da utilizzare la nuova versione e il nuovo gruppo di parametri cluster di database personalizzato. Specifica anche l'argomento `--allow-major-version-upgrade`, come indicato nell'esempio seguente.

```

aws rds modify-db-cluster \
--db-cluster-identifier docs-lab-bfish-apg-14 \
--engine-version 14.6 \
--db-cluster-parameter-group-name docs-lab-babelfish-apg-14 \
--allow-major-version-upgrade \
--region us-west-1 \
--apply-immediately
```

Usa il comando CLI reboot-db-instance per riavviare l'istanza di scrittura del cluster e rendere effettive le impostazioni dei parametri.

```

aws rds reboot-db-instance \
--db-instance-identifier docs-lab-bfish-apg-14-instance-1\
--region us-west-1
```

Utilizzo del parametro di versione del prodotto Babelfish

Un nuovo parametro Grand Unified Configuration (GUC) chiamato `babelfishpg_tds.product_version` è stato introdotto nelle versioni Babelfish 2.4.0 e 3.1.0. Questo parametro consente di impostare il numero di versione del prodotto SQL Server come output di Babelfish.

Il parametro è una stringa ID di versione in 4 parti e ogni parte deve essere separata da ".".

Sintassi

```
Major.Minor.Build.Revision
```

- Versione principale: un numero compreso tra 11 e 16.
- Versione secondaria: un numero compreso tra 0 e 255.
- Versione di build: un numero compreso tra 0 e 65535.
- Revisione: 0 e qualsiasi numero positivo.

Configurazione del parametro di versione del prodotto Babelfish

È necessario utilizzare il gruppo di parametri del cluster per impostare il parametro babelfishpg_tds.product_version nella console. Per ulteriori informazioni su come modificare il parametro del cluster di database, consulta Modifica dei parametri in un gruppo di parametri del cluster di database in Amazon Aurora.

Quando si imposta il parametro della versione del prodotto su un valore non valido, la modifica non ha effetto. Sebbene la console possa mostrare il nuovo valore, il parametro mantiene il valore precedente. Controlla il file di log del motore per i dettagli sui messaggi di errore.

Per Linux, macOS o Unix:

```
aws rds modify-db-cluster-parameter-group \
--db-cluster-parameter-group-name mydbparametergroup \
--parameters
 "ParameterName=babelfishpg_tds.product_version,ParameterValue=15.2.4000.1,ApplyMethod=immediat
```

Per Windows:

```
aws rds modify-db-cluster-parameter-group ^
--db-cluster-parameter-group-name mydbparametergroup ^
--parameters
 "ParameterName=babelfishpg_tds.product_version,ParameterValue=15.2.4000.1,ApplyMethod=immediat
```

Query e parametri interessati

Query/Parametro	Risultato	Tempo effettivo
SELECT @@VERSION	Restituisce la versione di SQL Server definita dall'utente (valore babelfishpg_tsql.version = predefinito)	Subito
SELECT SERVERPROPERTY('ProductVersion')	Restituisce la versione di SQL Server definita dall'utente	Subito
SELECT SERVERPROPERTY('ProductMajorVersion')	Restituisce la versione principale della versione di SQL Server definita dall'utente	Subito
Token VERSION nel messaggio di risposta PRELOGIN	Il server restituisce i messaggi PRELOGIN con la versione di SQL Server definita dall'utente	Ha effetto quando un utente crea una nuova sessione
SQLServerVersion in LoginAck quando si utilizza JDBC	DatabaseMetadata.getDatabaseProductVersion() restituisce la versione di SQL Server definita dall'utente	Ha effetto quando un utente crea una nuova sessione

Interfaccia con il parametro babelfishpg_tsql.version

È possibile impostare l'output di @@VERSION utilizzando i parametri babelfishpg_tsql.version e babelfishpg_tds.product_version. Gli esempi seguenti mostrano come si interfacciano questi due parametri.

- Quando il parametro babelfishpg_tsql.version è predefinito e babelfishpg_tds.product_version è 15.0.2000.8.
  - Output di @@version - 15.0.2000.8.
- Quando il parametro babelfishpg_tsql.version è impostato su 13.0.2000.8 e il parametro babelfishpg_tds.product_version è 15.0.2000.8.
  - Output di @@version - 13.0.2000.8.

Informazioni di riferimento su Babelfish per Aurora PostgreSQL

Argomenti

- Funzionalità supportate in Babelfish per versione
- Funzionalità non supportate in Babelfish
- Utilizzo delle procedure di Babelfish per Aurora PostgreSQL
- Babelfish supporta i metodi dei tipi di dati XML

Funzionalità supportate in Babelfish per versione

Nelle tabelle seguenti è possibile trovare l'elenco di tutte le funzionalità supportate dalle diverse versioni di Babelfish. Per gli elenchi di funzionalità non supportate, vedere Funzionalità non supportate in Babelfish. Per informazioni sulle varie versioni di Babelfish, consulta Release Notes for Aurora PostgreSQL.

Funzionalità di Aurora e PostgreSQL

Nella tabella seguente è possibile trovare le funzionalità di Aurora e PostgreSQL supportate dalle diverse versioni di Babelfish.

Funzionalità di Aurora e PostgreSQL	5	5	4	4	4	4	4	4	4	3	3	3	3	3	3	3	3	3	3	2	2	2	2	2	2	2	2	2	2	2	2	1.0
Servizi Aurora ML	S	S	S	S	S	S	S	S	N	S	S	S	S	S	S	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	No
Autenticazione del database con Kerberos utilizzando Directory Service	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	N	N	No
Dump e ripristino	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	No

Funzionalità di Aurora e PostgreSQL	5	5	4	4	4	4	4	4	4	3	3	3	3	3	3	3	3	3	3	2	2	2	2	2	2	2	2	2	2	2	2	2	1.0
L'estensione pg_stat_statement	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	N	N	S	S	S	S	S	S	S	N	N	N	N	N	No
pgvector	S	S	S	S	S	S	S	S	N	S	S	S	S	S	S	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	No
Applicazione di patch senza tempi di inattività (ZDP)	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	N	N	No

Funzionalità o sintassi T-SQL

Nella tabella seguente è possibile trovare le funzionalità o la sintassi T-SQL supportate dalle diverse versioni di Babelfish.

Funzionalità o sintassi T-SQL	5	5	4	4	4	4	4	4	4	3	3	3	3	3	3	3	3	3	3	2	2	2	2	2	2	2	2	2	2	2	2	2	1.0
Riferimenti ai nomi degli oggetti in 4 parti per le istruzioni SELECT	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	N	S	S	S	S	S	S	S	S	N	N	N	N	N	No
Sintassi ALTER AUTHORIZATION per modificare il proprietario del database	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	N	N	N	N	N	N	N	N	N	N	N	N	N	N	No
ALTER DATABASE *<db_name>*	S	S	S	S	S	S	S	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	No

Funzionalità o sintassi T-SQL	5	5	4	4	4	4	4	4	4	3	3	3	3	3	3	3	3	3	3	2	2	2	2	2	2	2	2	2	2	2	2	2	1.0
MODIFY NAME = *<new_db_name>*																																	
ALTER FUNCTION	S	S	S	S	S	N	N	N	N	S	S	S	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	No
ALTER PROCEDURE	S	S	S	S	S	S	S	N	N	S	S	S	S	S	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	No
ALTER ROLE	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	Sì
ALTER USER...WITH LOGIN	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	No
ALTER VIEW	S	N	S	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	No
Parola chiave AS in CREATE FUNCTION	S	S	S	S	S	S	S	S	N	S	S	S	S	S	S	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	No
Clausola AT TIME ZONE	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	No
Istanza Babelfish come server collegato	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	N	S	S	S	S	S	S	S	N	N	N	N	N	N	N	No
Operatori di confronto !< e !>	S	S	S	S	S	S	S	S	N	S	S	S	S	S	S	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	No

Funzionalità o sintassi T-SQL	5	5	4	4	4	4	4	4	4	3	3	3	3	3	3	3	3	3	3	2	2	2	2	2	2	2	2	2	2	2	2	1.0
CREATE INDEX ... ON partition _scheme_name (partition_column_ name)	S	S	S	S	S	S	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	No
CREATE trigger Instead of (DML) nelle viste SQL Server	S	S	S	S	S	S	S	N	N	S	S	S	S	S	S	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	No
CREATE PARTITION FUNCTION	S	S	S	S	S	S	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	No
CREATE PARTITION SCHEME	S	S	S	S	S	S	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	No
CREATE ROLE	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	Sì
CREATE TABLE ... ON partition _scheme_name (partition_column_ name)	S	S	S	S	S	S	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	No
CREATE TRIGGER	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	Sì
CREATE OR ALTER VIEW	S	N	S	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	No

Funzionalità o sintassi T-SQL	5	5	4	4	4	4	4	4	4	3	3	3	3	3	3	3	3	3	3	2	2	2	2	2	2	2	2	2	2	2	2	2	1.0
Creazione di indici univoci	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	Sì
Esecuzione di procedure tra database	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	No
Riferimenti tra database SELECT, INSERT, UPDATE, DELETE	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	Sì
Parametri digitati con cursore solo per i parametri di input (non di output)	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	Sì
Migrazione dei dati tramite l'utilità bcp client	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	Sì
Tipi di dati TIMESTAMP, ROWVERSION (per informazioni sull'utilizzo, consulta Funzionalità con implementazione limitata	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	Sì
Parola chiave DEFAULT nelle chiamate a stored procedure e funzioni	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	No

Funzionalità o sintassi T-SQL	5	5	4	4	4	4	4	4	4	3	3	3	3	3	3	3	3	3	3	2	2	2	2	2	2	2	2	2	2	2	2	1.0
DBCC CHECKIDENT	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	No
DROP DATABASE	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	N	No
DROP IF EXISTS (per oggetti SCHEMA, DATABASE e USER)	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	Sì
DROP INDEX index ON schema.table	S	S	S	S	S	S	S	S	N	S	S	S	S	S	S	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	No
DROP INDEX schema.table.index	S	S	S	S	S	S	S	S	N	S	S	S	S	S	S	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	No
DROP PARTITION FUNCTION	S	S	S	S	S	S	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	No
DROP PARTITION SCHEME	S	S	S	S	S	S	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	No
DROP ROLE	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	Sì
ENABLE/DISABLE TRIGGER	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	No

Funzionalità o sintassi T-SQL	5	5	4	4	4	4	4	4	4	3	3	3	3	3	3	3	3	3	3	2	2	2	2	2	2	2	2	2	2	2	2	2	1.0
FULL TEXT SEARCH	S	S	S	S	S	S	S	S	S	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	No
Ricerca full-text con clausola CONTAINS	S	S	S	S	S	S	S	S	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	No
Tipi di dati spaziali Geometry e Geography	S	S	S	S	S	S	S	S	N	S	S	S	S	S	S	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	No
GRANT	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	Sì
GRANT permission [,...n] ON SCHEMA	S	S	S	S	S	S	S	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	No
GUC babelfish pg_tds.product_version	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	N	N	N	No
Identificatori con carattere punto iniziale	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	Sì
Trigger INSTEAD OF su tabelle	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	Sì
Trigger INSTEAD-OF sulle viste	S	S	S	S	S	S	S	S	S	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	No

Funzionalità o sintassi T-SQL	5	5	4	4	4	4	4	4	4	3	3	3	3	3	3	3	3	3	2	2	2	2	2	2	2	2	2	2	2	2	1.0
KILL	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	N	N	N	N	N	N	N	N	N	N	N	N	N	N	No
$PARTITION.partition_function_name(partition_column_value)	S	S	S	S	S	S	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	No
PIVOT (utilizzare nella definizione della vista da 4.4.0 e 3.8.0, espressione di tabella comune da 3.4.0, join da 3.4.0)	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	N	N	N	N	N	N	N	N	N	N	N	N	N	N	No
REVOKE	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	Sì
REVOKE permission [,...n] ON SCHEMA	S	S	S	S	S	S	S	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	No
Clausole SELECT... OFFSET...FETCH	S	S	S	S	S	S	S	S	N	S	S	S	S	S	S	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	No
SELECT FOR JSON AUTO	S	S	S	S	S	S	S	S	N	S	S	S	S	S	S	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	No
SET BABELFISH_SHOWPLAN_ALL ON (e OFF)	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	Sì

Funzionalità o sintassi T-SQL	5	5	4	4	4	4	4	4	4	3	3	3	3	3	3	3	3	3	3	2	2	2	2	2	2	2	2	2	2	2	2	2	1.0
SET BABELFISH_STATISTICS PROFILE ON (OFF)	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	Sì
SET CONTEXT_INFO	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	N	S	S	S	S	S	S	S	N	N	N	N	N	N	No
SET LOCK_TIMEOUT	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	Sì
SET NO_BROWSETABLE	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	No
SET rowcount	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	N	S	S	S	S	S	S	S	N	N	N	N	N	N	No
SET SHOWPLAN_ALL	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	No
SET STATISTICS IO	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	No
SET TRANSACTION ISOLATION LEVEL REPEATABLE READ	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	No
SET TRANSACTION ISOLATION LEVEL SERIALIZABLE	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	No

Funzionalità o sintassi T-SQL	5	5	4	4	4	4	4	4	4	3	3	3	3	3	3	3	3	3	3	2	2	2	2	2	2	2	2	2	2	2	2	2	1.0
Sintassi SET TRANSACTION ISOLATION LEVEL	S	S	S	S	S	S	S	S	N	S	S	S	S	S	S	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	No
SSMS: connessione con la finestra di dialogo di connessione dell’esploratore di oggetti	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	Sì
SSMS: Migrazione dei dati con la procedura guidata di importazione/esportazione	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	Sì
SSMS: supporto parziale per l’esploratore di oggetti	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	Sì
STDEV	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	N	N	S	S	S	S	S	S	S	N	N	N	N	N	N	No
STDEVP	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	N	N	S	S	S	S	S	S	S	N	N	N	N	N	N	No
I trigger con più azioni DML possono fare riferimento alle tabelle di transizione	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	Sì

Funzionalità o sintassi T-SQL	5	5	4	4	4	4	4	4	4	3	3	3	3	3	3	3	3	3	3	2	2	2	2	2	2	2	2	2	2	2	2	2	1.0
Suggerimenti T-SQL (metodi di join, utilizzo degli indici, MAXDOP)	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	N	No	
Sintassi delle parentesi quadre T-SQL con il predicato LIKE	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	N	N	S	S	S	S	S	S	S	N	N	N	N	N	No	
UNPIVOT	S	N	S	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	No	
Valori stringa non quotati nei richiami di procedure memorizzate e nei valori predefiniti	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	No	
VAR	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	N	S	S	S	S	S	S	S	N	N	N	N	N	N	No	
VARP	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	N	S	S	S	S	S	S	S	N	N	N	N	N	N	No	

Funzioni integrate T-SQL

Nella tabella seguente è possibile trovare le funzioni integrate T-SQL supportate da diverse versioni di Babelfish.

Funzioni integrate T-SQL	5	5	4	4	4	4	4	4	4	3	3	3	3	3	3	3	3	3	3	2	2	2	2	2	2	2	2	2	2	2	2	2	1.0
APP_NAME	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	N	N	No	
ATN2	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	N	N	No	
CHARINDEX	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	Sì	
CHOOSE	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	Sì	
COL_LENGTH	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	No	
COL_NAME	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	No	
COLUMNS_UPDATED	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	Sì	
COLUMNPROPERTY (solo CharMaxLen, AllowsNull)	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	Sì	
CONCAT_WS	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	Sì	
CONTEXT_INFO																																	

Funzioni integrate T-SQL	5	5	4	4	4	4	4	4	4	3	3	3	3	3	3	3	3	3	3	2	2	2	2	2	2	2	2	2	2	2	2	1.0
	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	N	S	S	S	S	S	S	S	N	N	N	N	N	No
CURSOR_STATUS	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	Sì
DATABASE_ PRINCIPAL_ID	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	N	S	S	S	S	S	S	S	N	N	N	N	N	No
DATEADD	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	N	No
DATEDIFF	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	N	No
DATEDIFF_BIG	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	N	N	No
DATEFROMPARTS	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	Sì
DATENAME	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	N	No
DATEPART	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	N	No
DATETIMEF ROMPARTS	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	Sì

Funzioni integrate T-SQL	5	5	4	4	4	4	4	4	4	3	3	3	3	3	3	3	3	3	3	2	2	2	2	2	2	2	2	2	2	2	2	2	1.0
DATETIME2FROMPARTS	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	N	S	S	S	S	S	S	S	N	N	N	N	N	No	
DATETIMEOFFSETFROMPARTS	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	N	No	
DATETRUNC	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	N	N	N	S	S	S	S	S	S	N	N	N	N	N	N	No	
DATE_BUCKET	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	N	N	N	S	S	S	S	S	S	N	N	N	N	N	N	No	
EOMONTH	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	N	N	N	S	S	S	S	S	S	N	N	N	N	N	N	No	
EXECUTE AS CALLER	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	N	N	No	
fn_listextendedproperty	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	N	N	S	S	S	S	S	S	N	N	N	N	N	N	No	
FOR JSON	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	N	No	
FULLTEXTSERVICEPROPERTY	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	Sì	
HAS_DBACCESS	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	Sì	

Funzioni integrate T-SQL	5	5	4	4	4	4	4	4	4	3	3	3	3	3	3	3	3	3	3	2	2	2	2	2	2	2	2	2	2	2	2	2 1.0
HAS_PERMS_BY_NAME	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	Sì
HOST_NAME	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	No
HOST_ID	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	N	N	S	S	S	S	S	S	S	N	N	N	N	N	No
IDENTITY	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	No
IS_MEMBER	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	Sì
IS_ROLEMEMBER	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	Sì
IS_SRVROLEMEMBER	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	Sì
ISJSON	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	Sì
JSON_MODIFY	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	N	No
JSON_QUERY	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	Sì

Funzioni integrate T-SQL	5	5	4	4	4	4	4	4	4	3	3	3	3	3	3	3	3	3	3	2	2	2	2	2	2	2	2	2	2	2	2	2.0	1.0
JSON_VALUE	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	Sì	
NEXT VALUE FOR	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	N	N	No	
OBJECT_DE FINITION	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	N	N	No	
OBJECT_SC HEMA_NAME	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	N	N	No	
OPENJSON	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	Sì	
OPENQUERY	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	Sì	
ORIGINAL_LOGIN	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	Sì	
PARSENAME	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	N	N	S	S	S	S	S	S	S	N	N	N	N	N	No	
PATINDEX	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	Sì	
ROWCOUNT_BIG	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	N	S	S	S	S	S	S	S	N	N	N	N	N	No	

Funzioni integrate T-SQL	5	5	4	4	4	4	4	4	4	3	3	3	3	3	3	3	3	3	2	2	2	2	2	2	2	2	2	2	2	2	2	1.0
SCHEMA_NAME	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	Sì
SESSION_CONTEXT	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	N	N	No
SESSION_USER	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	Sì
SID_BINARY (restituisce sempre NULL)	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	No
SMALLDATETIMEFROMPARTS	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	N	N	S	S	S	S	S	S	S	N	N	N	N	No	
SQUARE	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	Sì
STR	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	N	N	No
STRING_AGG	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	N	N	N	S	S	S	N	N	N	N	N	N	N	N	No	
STRING_SPLIT	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	Sì
SUSER_SID	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	Sì

Funzioni integrate T-SQL	5	5	4	4	4	4	4	4	4	3	3	3	3	3	3	3	3	3	2	2	2	2	2	2	2	2	2	2	2	2	1.0
SUSER_SNAME	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	Sì
SWITCHOFFSET	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	N	N	N	N	N	N	N	N	N	N	N	N	N	N	No
SYSTEM_USER	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	No
TIMEFROMPARTS	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	N	S	S	S	S	S	S	N	N	N	N	N	N	No
TODATETIMEOFFSET	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	N	N	N	N	N	N	N	N	N	N	N	N	N	N	No
TO_CHAR	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	N	No
TRIGGER_NESTLEVEL supportato (solo senza argomenti)	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	Sì
TRY_CONVERT	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	N	No
TYPE_ID	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	N	N	N	N	N	N	N	N	N	N	N	N	N	N	No

Funzioni integrate T-SQL	5	5	4	4	4	4	4	4	4	3	3	3	3	3	3	3	3	3	3	2	2	2	2	2	2	2	2	2	2	2	2	1.0
TYPE_NAME	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	No
UPDATE	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	Sì

Cataloghi T-SQL INFORMATION_SCHEMA

Nella tabella seguente è possibile trovare i cataloghi T-SQL INFORMATION_SCHEMA supportati dalle diverse versioni di Babelfish.

Cataloghi T-SQL INFORMATION_SCHEMA	5	5	4	4	4	4	4	4	4	3	3	3	3	3	3	3	3	3	3	2	2	2	2	2	2	2	2	2	2	2	2	1.0
CHECK_CONSTRAINTS	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	No
COLUMN_DOMAIN_USAGE	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	No
COLUMNS	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	Sì
CONSTRAINT_COLUMN_USAGE	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	Sì
DOMAINS	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	Sì

Cataloghi T-SQL INFORMATION_SCHEMA	5	5	4	4	4	4	4	4	4	3	3	3	3	3	3	3	3	3	3	2	2	2	2	2	2	2	2	2	2	2	2	21.0
KEY_COLUMN_USAGE	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	No
ROUTINES	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	No
TABLES	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	Sì
TABLE_CONSTRAINTS	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	Sì
VIEWS	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	Sì

Variabili @@ definite dal sistema T-SQL

Nella tabella seguente è possibile trovare le variabili @@ definite dal sistema T-SQL supportate dalle diverse versioni di Babelfish.

Variabili @@ definite dal sistema T-SQL	5	5	4	4	4	4	4	4	4	3	3	3	3	3	3	3	3	3	3	2	2	2	2	2	2	2	2	2	2	2	2	21.0
@@CURSOR_ROWS	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	Sì
@@DATEFIRST	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	Sì
@@DBTS																																

Variabili @@ definite dal sistema T-SQL	5	5	4	4	4	4	4	4	4	3	3	3	3	3	3	3	3	3	3	2	2	2	2	2	2	2	2	2	2	2	2	201.0
	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	Sì
@@ERROR	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	Sì
@@ERROR=213	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	Sì
@@FETCH_S TATUS	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	Sì
@@IDENTITY	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	Sì
@@LANGUAGE	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	Sì
@@LOCK_TI MEOUT	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	Sì
@@MAX_CON NECTIONS	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	Sì
@@MAX_PRE CISION	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	Sì
@@MICROSO FTVERSION	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	Sì

Variabili @@ definite dal sistema T-SQL	5	5	4	4	4	4	4	4	4	3	3	3	3	3	3	3	3	3	3	2	2	2	2	2	2	2	2	2	2	2	2	20 1.0
@@NESTLEVEL	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	Sì
@@PROCID	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	Sì
@@ROWCOUNT	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	Sì
@@SERVERNAME	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	Sì
@@SERVICENAME	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	Sì
@@SPID	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	Sì
@@TRANCOUNT	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	Sì
@@VERSION (notare la differenza di formato come descritto in Differenze T-SQL in Babelfish.	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	Sì

Stored procedure di sistema T-SQL

Nella tabella seguente è possibile trovare le stored procedure di sistema T-SQL supportate dalle diverse versioni di Babelfish.

Stored procedure di sistema T-SQL	5	5	4	4	4	4	4	4	4	3	3	3	3	3	3	3	3	3	3	2	2	2	2	2	2	2	2	2	2	2	2	2	1.0
sp_addextendedproperty	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	N	N	S	S	S	S	S	S	S	N	N	N	N	N	No	
sp_addlinkedserver	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	N	N	No	
sp_addlinkedsrvlogin	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	N	N	No	
sp_addrole	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	N	No	
sp_addrolemember	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	N	No	
sp_babelfish_volatility	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	N	N	No	
sp_column_privileges	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	Sì	
sp_columns	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	Sì	
sp_columns_100																																	

Stored procedure di sistema T-SQL	5	5	4	4	4	4	4	4	4	3	3	3	3	3	3	3	3	3	3	2	2	2	2	2	2	2	2	2	2	2	2	2	1.0
	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	Sì
sp_column s_managed	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	Sì
sp_cursor	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	Sì
sp_cursor_list	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	Sì
sp_cursorclose	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	Sì
sp_cursorexecute	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	Sì
sp_cursorfetch	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	Sì
sp_cursoropen	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	Sì
sp_cursoroption	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	Sì
sp_cursorprepare	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	Sì

Stored procedure di sistema T-SQL	5	5	4	4	4	4	4	4	4	3	3	3	3	3	3	3	3	3	3	2	2	2	2	2	2	2	2	2	2	2	2	2.0	1.0
sp_cursorprepexec	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	Sì	
sp_cursorunprepare	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	Sì	
sp_databases	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	Sì	
sp_datatype_info	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	Sì	
sp_datatype_info_100	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	Sì	
sp_describe_cursor	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	Sì	
sp_describe_first_result_set	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	Sì	
sp_describe_undeclared_parameters	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	Sì	
sp_dropextendedproperty	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	N	N	S	S	S	S	S	S	S	N	N	N	N	N	No	
sp_droplinkedsrvlogin	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	N	N	No	

Stored procedure di sistema T-SQL	5	5	4	4	4	4	4	4	4	3	3	3	3	3	3	3	3	3	3	2	2	2	2	2	2	2	2	2	2	2	2	2	1.0
sp_droprole	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	N	No
sp_droprolemember	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	N	No
sp_dropserver	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	N	N	No
sp_enum_oledb_providers	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	N	N	S	S	S	S	S	S	S	N	N	N	N	N	No
sp_execute	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	Sì
sp_execute_postgresql(CREATE, ALTER, DROP)	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	N	N	S	S	S	S	S	S	S	N	N	N	N	N	No
sp_executesql	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	Sì
sp_fkeys	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	Sì
sp_getapplock	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	Sì
sp_helpdb	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	Sì

Stored procedure di sistema T-SQL	5	5	4	4	4	4	4	4	4	3	3	3	3	3	3	3	3	3	3	2	2	2	2	2	2	2	2	2	2	2	2	1.0
sp_helpdbfixedrole	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	N	No
sp_helplinkedsrvlogin	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	N	No
sp_helprole	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	Sì
sp_helprolemember	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	Sì
sp_helpsrvrolemember	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	No
sp_helpuser	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	Sì
sp_linkedservers	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	N	N	No
sp_oledb_ro_usrname	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	Sì
sp_pkeys	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	Sì
sp_prefix	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	No

Stored procedure di sistema T-SQL	5	5	4	4	4	4	4	4	4	3	3	3	3	3	3	3	3	3	3	2	2	2	2	2	2	2	2	2	2	2	2	2	1.0
sp_prepare	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	Sì
sp_procedure_params_100_managed	S	S	S	S	S	S	S	S	N	S	S	S	S	S	S	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	No
sp_releaseapplock	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	Sì
sp_rename	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	N	N	N	No
sp_renamedb	S	S	S	S	S	S	S	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	No
sp_rename (per object_type 'DATABASE')	S	S	S	S	S	S	S	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	No
sp_reset_connection	S	S	S	S	S	N	N	N	N	S	S	S	N	N	N	N	N	N	N	S	S	S	N	N	N	N	N	N	N	N	N	N	No
sp_serveroption(opzione connect_timeout)	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	N	N	S	S	S	S	S	S	S	N	N	N	N	N	N	No
sp_set_session_context	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	N	N	N	No

Stored procedure di sistema T-SQL	5	5	4	4	4	4	4	4	4	3	3	3	3	3	3	3	3	3	3	2	2	2	2	2	2	2	2	2	2	2	2	1.0
sp_special_columns	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	Sì
sp_sproc_columns	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	Sì
sp_sproc_columns_100	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	Sì
sp_statistics	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	Sì
sp_statistics_100	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	Sì
sp_stored_procedure	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	Sì
sp_table_privileges	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	Sì
sp_tablecollations_100	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	Sì
sp_tables	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	Sì
sp_testlinkedserver	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	N	N	S	S	S	S	S	S	S	N	N	N	N	N	No

Stored procedure di sistema T-SQL	5	5	4	4	4	4	4	4	4	3	3	3	3	3	3	3	3	3	2	2	2	2	2	2	2	2	2	2	2	2	1.0
sp_unprepare	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	Sì
sp_updateextendedproperty	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	N	N	S	S	S	S	S	S	N	N	N	N	N	No
sp_who	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	N	N	S	S	S	S	S	S	N	N	N	N	N	No
xp_qv	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	Sì

Proprietà T-SQL supportate nella funzione di sistema CONNECTIONPROPERTY

Nella tabella seguente è possibile trovare le proprietà T-SQL supportate nella funzione di sistema CONNECTIONPROPERTY supportata dalle diverse versioni di Babelfish.

Proprietà T-SQL supportate nella funzione di sistema CONNECTIONPROPERTY	5	5	4	4	4	4	4	4	4	3	3	3	3	3	3	3	3	3	2	2	2	2	2	2	2	2	2	2	2	2	1.0
auth_scheme	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	Sì
client_net_address	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	Sì
local_net_address																															

Proprietà T-SQL supportate nella funzione di sistema CONNECTIONPROPERTY	5	5	4	4	4	4	4	4	4	3	3	3	3	3	3	3	3	3	3	2	2	2	2	2	2	2	2	2	2	2	2	201.0
	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	Sì
local_tcp_port	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	Sì
net_transport	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	Sì
protocol_type	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	Sì
physical_net_transport	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	Sì

Proprietà T-SQL supportate nella funzione di sistema OBJECTPROPERTY

Nella tabella seguente è possibile trovare le proprietà T-SQL supportate nella funzione di sistema OBJECTPROPERTY supportata dalle diverse versioni di Babelfish.

Proprietà T-SQL supportate nella funzione di sistema OBJECTPROPERTY	5	5	4	4	4	4	4	4	4	3	3	3	3	3	3	3	3	3	3	2	2	2	2	2	2	2	2	2	2	2	2	201.0
IsInlineFunction	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	Sì
IsScalarFunction																																

Proprietà T-SQL supportate nella funzione di sistema OBJECTPROPERTY	5	5	4	4	4	4	4	4	4	3	3	3	3	3	3	3	3	3	3	2	2	2	2	2	2	2	2	2	2	2	2	2 1.0
	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	Sì
IsTableFunction	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	Sì

Proprietà T-SQL supportate nella funzione SERVERPROPERTY

Nella tabella seguente è possibile trovare le proprietà T-SQL supportate nella funzione di sistema SERVERPROPERTY supportata dalle diverse versioni di Babelfish.

Proprietà T-SQL supportate nella funzione SERVERPROPERTY	5	5	4	4	4	4	4	4	4	3	3	3	3	3	3	3	3	3	3	2	2	2	2	2	2	2	2	2	2	2	2	2 1.0
Babelfish	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	Sì
Collation (Regola di confronto)	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	Sì
CollationID	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	Sì
Edizione	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	Sì
EditionID																																

Proprietà T-SQL supportate nella funzione SERVERPROPERTY	5	5	4	4	4	4	4	4	4	3	3	3	3	3	3	3	3	3	3	2	2	2	2	2	2	2	2	2	2	2	2	1.0
	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	Sì
EngineEdition	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	Sì
InstanceName	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	No
IsAdvancedAnalyticsInstalled	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	Sì
IsBigDataCluster	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	Sì
IsFullTextInstalled	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	Sì
IsIntegratedSecurityOnly	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	Sì
IsLocalDB	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	Sì
IsPolyBaseInstalled	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	Sì

Proprietà T-SQL supportate nella funzione SERVERPROPERTY	5	5	4	4	4	4	4	4	4	3	3	3	3	3	3	3	3	3	3	2	2	2	2	2	2	2	2	2	2	2	2	2 1.0
IsSingleUser	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	Sì
IsXTPSupported	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	Sì
Japanese_CI_AI	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	Sì
Japanese_CI_AS	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	Sì
Japanese_CS_AS	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	Sì
LicenseType	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	Sì
MachineName	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	No
ProductLevel	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	N	N	No
ProductMajorVersion	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	Sì

Proprietà T-SQL supportate nella funzione SERVERPROPERTY	5	5	4	4	4	4	4	4	4	3	3	3	3	3	3	3	3	3	3	2	2	2	2	2	2	2	2	2	2	2	2	2.0	1.0
ProductMinorVersion	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	Sì	
ProductUpdateLevel	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	N	N	No	
ProductVersion	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	Sì	
ServerName (Nome server)	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	Sì	

Viste di SQL Server supportate da Babelfish

Nella tabella seguente è possibile trovare le viste di SQL Server supportate dalle diverse versioni di Babelfish.

Viste di SQL Server supportate da Babelfish	5	5	4	4	4	4	4	4	4	3	3	3	3	3	3	3	3	3	3	2	2	2	2	2	2	2	2	2	2	2	2	2.0	1.0
information_schema.key_column_usage	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	N	N	N	S	S	S	S	S	S	N	N	N	N	N	N	No	
information_schema.routines	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	N	N	No	

Viste di SQL Server supportate da Babelfish	5	5	4	4	4	4	4	4	4	3	3	3	3	3	3	3	3	3	3	2	2	2	2	2	2	2	2	2	2	2	2	2	1.0
information_schema.schemata	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	N	N	No	
information_schema.sequences	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	N	N	No	
sys.all_columns	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	Sì	
sys.all_objects	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	Sì	
sys.all_parameters	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	N	No		
sys.all_sql_modules	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	Sì	
sys.all_views	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	Sì	
sys.columns	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	Sì	
sys.configurations	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	Sì	

Viste di SQL Server supportate da Babelfish	5.5.4.4.4.4.4.4.4.3.3.3.3.3.3.3.3.3.2.2.2.2.2.2.2.2.2.2.2.2.2.2	1.0
sys.data_spaces	S...S	Sì
sys.database_files	S...S	Sì
sys.database_mirroring	S...S	Sì
sys.database_principals	S...S	Sì
sys.database_role_members	S...S	Sì
sys.databases	S...S	Sì
sys.destination_data_spaces	S...S.N...N	No
sys.dm_exec_connections	S...S	Sì
sys.dm_exec_session	S...S	Sì

Viste di SQL Server supportate da Babelfish	5	5	4	4	4	4	4	4	4	3	3	3	3	3	3	3	3	3	3	2	2	2	2	2	2	2	2	2	2	2	2	1.0
sys.dm_hard_database_replica_states	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	Sì
sys.dm_os_host_info	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	Sì
sys.dm_os_sys_info	S	S	S	S	S	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	No
sys.endpoint	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	Sì
sys.extended_properties	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	N	N	S	S	S	S	S	S	S	N	N	N	N	N	No
sys.indexes	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	Sì
sys.partitions	S	S	S	S	S	S	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	No
sys.partition_functions	S	S	S	S	S	S	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	No
sys.partition_parameters	S	S	S	S	S	S	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	No

Viste di SQL Server supportate da Babelfish	5.5	5.4	4.4	4.3	3.3	3.2	2.2	2.1.0
sys.partition_range_values	Sì	Sì	Sì	Sì	Sì	Sì	No	No
sys.partition_schemes	Sì	Sì	Sì	Sì	Sì	Sì	No	No
sys.schemas	Sì	Sì	Sì	Sì	Sì	Sì	Sì	Sì
sys.server_principals	Sì	Sì	Sì	Sì	Sì	Sì	Sì	Sì
sys.server_role_members	Sì	Sì	Sì	Sì	Sì	Sì	No	No
sys.sql_modules	Sì	Sì	Sì	Sì	Sì	Sì	Sì	Sì
sys.sysconfigures	Sì	Sì	Sì	Sì	Sì	Sì	Sì	Sì
sys.syscurconfigs	Sì	Sì	Sì	Sì	Sì	Sì	Sì	Sì
sys.syslogins	Sì	Sì	Sì	Sì	Sì	Sì	No	No

Viste di SQL Server supportate da Babelfish	5	5	4	4	4	4	4	4	4	3	3	3	3	3	3	3	3	3	3	2	2	2	2	2	2	2	2	2	2	2	2	201.0
sys.sysprocesses	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	Sì
sys.sysusers	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	N	S	S	S	S	S	S	S	N	N	N	N	N	No
sys.table_types	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	Sì
sys.tables	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	Sì
sys.types	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	N	N	No
sys.xml_schema_collections	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	Sì
syslanguages	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	Sì
sysobjects.crdate	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	N	No

Funzionalità non supportate in Babelfish

Negli elenchi seguenti è possibile trovare funzionalità attualmente non supportate in Babelfish. Gli aggiornamenti di Babelfish sono inclusi nelle versioni di Aurora PostgreSQL. Per ulteriori informazioni, consulta Release notes for Amazon Aurora PostgreSQL-Compatible Edition.

Argomenti

- Funzionalità attualmente non supportata
- Impostazioni non supportate
- Comandi non supportati
- Nomi di colonna o attributi non supportati
- Tipi di dati non supportati
- Tipi di oggetti non supportati
- Funzioni non supportate
- Sintassi non supportata

Funzionalità attualmente non supportata

Nella tabella è possibile trovare informazioni su determinate funzionalità attualmente non supportate.

Funzionalità o sintassi	Description
Moduli di assembly e routine SQL Common Language Runtime (CLR)	Le funzionalità relative ai moduli di assemblaggio e alle routine CLR non sono supportate.
Attributi cookie	ROWGUIDCOL, SPARSE, FILESTREAM e MASKED non sono supportati.
Database contenuti	I database contenuti con login autenticati a livello di database anziché a livello di server non sono supportati.
DDL tra database	L'esecuzione di istruzioni DDL che fanno riferimento o operano su oggetti su più database non è ancora supportata.
Cursori (aggiornabili)	I cursori aggiornabili non sono supportati.
Cursori (globali)	I cursori GLOBAL non sono supportati.
Cursore (comportamenti di recupero)	I seguenti comportamenti di recupero del cursore non sono supportati: FETCH PRIOR, FIRST, LAST, ABSOLUTE e RELATIVE

Funzionalità o sintassi	Description
Parametri di output di tipo cursore	Le variabili e i parametri di tipo cursore non sono supportati per i parametri di output (viene generato un errore).
Opzioni del cursore	SCROLL, KEYSET, DYNAMIC, FAST_FORWARD, SCROLL_LOCKS, OPTIMISTIC, TYPE_WARNING e FOR UPDATE
Crittografia dei dati	La crittografia dei dati non è supportata.
Applicazioni a livello di dati (DAC)	Le operazioni di importazione o esportazione di applicazioni a livello di dati (DAC) con file di pacchetto DAC (.dacpac) o di backup DAC (.bacpac) non sono supportate.
Comandi DBCC	I comandi Microsoft SQL Server Database Console Commands (DBCC) non sono supportati. DBCC CHECKIDENT è supportato in Babelfish 3.4.0 e versioni successive.
DROP IF EXISTS	Questa sintassi non è supportata per gli oggetti USER e SCHEMA. È supportato per gli oggetti TABLE, VIEW, PROCEDURE, FUNCTION e DATABASE.
Encryption (Crittografia)	Le funzioni e le istruzioni integrate non supportano la crittografia.
Connessioni ENCRYPT_CLIENT_CERT	Le connessioni ai certificati client non sono supportate.
Istruzione EXECUTE AS	Questa affermazione non è supportata.
Clausola EXECUTE AS SELF	Questa clausola non è supportata in funzioni, procedure o trigger.
Clausola EXECUTE AS USER	Questa clausola non è supportata in funzioni, procedure o trigger.
Vincoli di chiave esterna che fanno riferimento al nome del database	I vincoli di chiave esterna che fanno riferimento al nome del database non sono supportati.

Funzionalità o sintassi	Description
FORMAT	I tipi definiti dall'utente non sono supportati.
Dichiarazioni di funzione con più di 100 parametri	Le dichiarazioni di funzione che contengono più di 100 parametri non sono supportate.
Chiamate di funzione che includono DEFAULT come valore di parametro	DEFAULT non è un valore di parametro supportato per una chiamata di funzione. DEFAULT come valore di parametro per una chiamata di funzione è supportato da Babelfish 3.4.0 e versioni successive.
Funzioni, definite esternamente	Le funzioni esterne, incluse le funzioni SQL CLR, non sono supportate.
Tabelle temporanee globali (tabelle con nomi che iniziano con ##)	Le tabelle temporanee globali non sono supportate.
Funzionalità grafico	Tutte le funzionalità dei grafici SQL non sono supportate.
Stored procedure estese generali	Le stored procedure di sistema che forniscono un'interfaccia da un'istanza di SQL Server a programmi esterni non sono supportate per varie attività di manutenzione. Sono incluse `xp_cmdshell` e altre stored procedure di sistema. Per ulteriori informazioni, consulta Stored procedure estese generali.
Identificatori (variabili o parametri) con più caratteri @ iniziali	Gli identificatori che iniziano con più di un @ principale non sono supportati.
Identificatori, nomi di tabelle o colonne che contengono caratteri [@ o]	Nomi di tabelle o colonne che contengono un determinato segno @ o parentesi quadre non sono supportati.
Indici in linea	Gli indici in linea non sono supportati.
Richiamo di una procedura il cui nome è in una variabile	L'utilizzo di una variabile come nome di procedura non è supportato.

Funzionalità o sintassi	Description
Viste materializzate	Le viste materializzate non sono supportate.
Clausola NOT FOR REPLICATION	Questa sintassi è accettata e ignorata.
Funzioni di escape ODBC	Le funzioni di escape ODBC non sono supportate.
Chiamate di procedura che include DEFAULT come valore di parametro	DEFAULT non è un valore di parametro supportato. DEFAULT come valore di parametro per una chiamata di funzione è supportato da Babelfish 3.4.0 e versioni successive.
Dichiarazioni di procedura con più di 100 parametri	Le dichiarazioni con più di 100 parametri non sono supportate.
Procedure definite esternamente	Le procedure definite esternamente, incluse le procedure SQL CLR, non sono supportate.
Controllo delle versioni delle procedure	Il controllo delle versioni delle procedure non è supportato.
Procedure WITH RECOMPILE	WITH RECOMPILE (se usato in combinazione con le istruzioni DECLARE e EXECUTE) non è supportato.
Riferimenti agli oggetti remoti	L'esecuzione di stored procedure sui server collegati a Babelfish non è supportata. I nomi degli oggetti in quattro parti funzionano solo per la lettura e non per la modifica della tabella remota. UPDATE può fare riferimento a una tabella remota nella clausola FROM senza modificarla. Per ulteriori informazioni, consulta Babelfish supporta i server collegati.
Sicurezza a livello di riga	La sicurezza a livello di riga con CREATE SECURITY POLICY e le funzioni inline con valori di tabella non è supportata.
Funzionalità Service Broker	La funzionalità Service Broker non è supportata.

Funzionalità o sintassi	Description
SESSIONPROPERTY	Proprietà non supportate: ANSI_NULLS, ANSI_PADDING, ANSI_WARNINGS, ARITHABORT, CONCAT_NULL_YIELDS_NULL e NUMERIC_ROUNDABORT
IMPOSTA LINGUA	Questa sintassi non è supportata con alcun valore diverso da `english` o `us_english` .
SP_CONFIGURE	Questa procedura memorizzata di sistema non è supportata.
Parola chiave SQL `SPARSE`	La parola chiave SPARSE viene accettata e ignorata.
Sintassi del costruttore del valore della tabella (clausola FROM)	La sintassi non supportata è per una tabella derivata costruita con la clausola FROM.
Tabelle globali	Le tabelle temporali non sono supportate.
Le procedure temporanee non vengono interrate automatic amente	Questa funzionalità non è supportata.
Trigger, definiti esternamente	Questi trigger non sono supportati, incluso SQL Common Language Runtime (CLR).

Impostazioni non supportate

Le seguenti impostazioni non sono supportate:

- ATTIVA ANSI_NULL_DFLT_OFF
- DISATTIVA ANSI_NULL_DFLT_ON
- DISATTIVA IMPOSTA ANSI_PADDING
- DISATTIVA IMPOSTA ANSI_WARNINGS
- DISATTIVA ARITHABORT
- ATTIVA ARITHIGNORE
- SET CURSOR_CLOSE_ON_COMMIT SU

- SET NUMERIC_ROUNDABORT ON
- SET PARSEONLY ON (il comando non funziona come previsto)
- SET FMT-ONLY ON (il comando non funziona correttamente. Sopprime solo l'esecuzione delle istruzioni SELECT ma non di altre.)

Comandi non supportati

Alcune funzionalità per i seguenti comandi non sono supportate:

- ADD SIGNATURE
- ALTER DATABASE, ALTER DATABASE SET
- BACKUP/RESTORE DATABASE/LOG
- RIPRISTINO DEI FILE BACPAC e DACPAC
- CREATE, ALTER, DROP AUTHORIZATION. ALTER AUTHORIZATION è supportato per gli oggetti del database.
- CREATE, ALTER, DROP AVAILABILITY GROUP
- CREATE, ALTER, DROP BROKER PRIORITY
- CREATE, ALTER, DROP COLUMN ENCRYPTION KEY
- CREATE, ALTER, DROP DATABASE ENCRYPTION KEY
- CREATE, ALTER, DROP, BACKUP CERTIFICATE
- CREATE AGGREGATE
- CREATE CONTRACT
- CHECKPOINT

Nomi di colonna o attributi non supportati

I seguenti nomi di colonna non sono supportati:

- $IDENTITY
- $ROWGUID
- IDENTITYCOL

Tipi di dati non supportati

I seguenti tipi di dati non sono supportati:

- HIERARCHYID

Tipi di oggetti non supportati

I seguenti tipi di oggetti non sono supportati:

- COLUMN MASTER KEY
- CREATE, ALTER EXTERNAL DATA SOURCE
- CREATE, ALTER, DROP DATABASE AUDIT SPECIFICATION
- CREATE, ALTER, DROP EXTERNAL LIBRARY
- CREATE, ALTER, DROP SERVER AUDIT
- CREATE, ALTER, DROP SERVER AUDIT SPECIFICATION
- CREA, MODIFICA, RILASCIA, CHIAVE OPEN/CLOSE SIMMETRICA
- CREATE, DROP DEFAULT
- CREDENTIAL
- CRYPTOGRAPHIC PROVIDER
- DIAGNOSTIC SESSION
- Visualizzazioni indicizzate
- CHIAVE MASTER SERVICE
- SYNONYM

Funzioni non supportate

Le seguenti funzioni integrate non sono supportate:

Funzioni di aggregazione

- APPROX_COUNT_DISTINCT
- CHECKSUM_AGG
- GROUPING_ID
- STRING_AGG con la clausola WITHIN GROUP

Funzioni di crittografia

- Funzione CERTENCODED
- Funzione CERTID
- Funzione CERTPROPERTY

Funzioni sui metadati

- COLUMNPROPERTY
- TYPEPROPERTY
- Funzione SERVERPROPERTY - Le seguenti proprietà non sono supportate:
  - BuildClrVersion
  - ComparisonStyle
  - ComputerNamePhysicalNetBIOS
  - HadrManagerStatus
  - InstanceDefaultDataPath
  - InstanceDefaultLogPath
  - IsClustered
  - IsHadrEnabled
  - LCID
  - NumLicenses
  - ProcessID
  - ProductBuild
  - ProductBuildType
  - ProductUpdateReference
  - ResourceLastUpdateDateTime
  - ResourceVersion
  - ServerName
  - SqlCharSet
  - SqlCharSetName
  - SqlSortOrder
  - SqlSortOrderName

- FilestreamShareName
- FilestreamConfiguredLevel
- FilestreamEffectiveLevel

Funzioni di sicurezza

- CERTPRIVATEKEY
- LOGINPROPERTY

Istruzioni, operatori, altre funzioni

- Funzione EVENTDATA
- GET_TRANSMISSION_STATUS
- OPENXML

Sintassi non supportata

La seguente sintassi non è supportata:

- ALTER DATABASE
- ALTER DATABASE SCOPED CONFIGURATION
- ALTER DATABASE SCOPED CREDENTIAL
- ALTER DATABASE SET HADR
- ALTER INDEX
- ALTER PARTITION FUNCTION
- ALTER PARTITION SCHEME
- ALTER SCHEMA
- ALTER SERVER CONFIGURATION
- ALTER SERVICE, clausola BACKUP/RESTORE SERVICE MASTER KEY
- BEGIN CONVERSATION TIMER
- BEGIN DISTRIBUTED TRANSACTION
- BEGIN DIALOG CONVERSATION
- BULK INSERT

- CREATE COLUMNSTORE INDEX
- CREATE EXTERNAL FILE FORMAT
- CREATE EXTERNAL TABLE
- CREATE, ALTER, DROP APPLICATION ROLE
- CREATE, ALTER, DROP ASSEMBLY
- CREATE, ALTER, DROP ASYMMETRIC KEY
- CREATE, ALTER, DROP CREDENTIAL
- CREATE, ALTER, DROP CRYPTOGRAPHIC PROVIDER
- CREATE, ALTER, DROP ENDPOINT
- CREATE, ALTER, DROP EVENT SESSION
- CREATE, ALTER, DROP EXTERNAL LANGUAGE
- CREATE, ALTER, DROP EXTERNAL RESOURCE POOL
- CREATE, ALTER, DROP FULLTEXT CATALOG
- CREATE, ALTER, DROP FULLTEXT INDEX
- CREATE, ALTER, DROP FULLTEXT STOPLIST
- CREATE, ALTER, DROP MESSAGE TYPE
- CREA, MODIFICA, ELIMINA, CHIAVE OPEN/CLOSE, BACKUP/RESTORE PRINCIPALE
- CREATE, ALTER, DROP QUEUE
- CREATE, ALTER, DROP RESOURCE GOVERNOR
- CREATE, ALTER, DROP RESOURCE POOL
- CREATE, ALTER, DROP ROUTE
- CREATE, ALTER, DROP SEARCH PROPERTY LIST
- CREATE, ALTER, DROP SECURITY POLICY
- CREATE, ALTER, DROP SELECTIVE XML INDEX clause
- CREATE, ALTER, DROP SERVICE
- CREATE, ALTER, DROP SPATIAL INDEX
- CREATE, ALTER, DROP TYPE
- CREATE, ALTER, DROP XML INDEX
- CREATE, ALTER, DROP XML SCHEMA COLLECTION
- CREATE/DROP RULE

- CREATE, DROP WORKLOAD CLASSIFIER
- CREATE, ALTER, DROP WORKLOAD GROUP
- ALTER TRIGGER
- CREA TABELLA Clausola GRANT
- CREA TABELLA Clausola IDENTITY
- CREATE USER - Questa sintassi non è supportata. L'affermazione PostgreSQL CREATE USER non crea un utente equivalente alla sintassi di SQL Server CREATE USER. Per ulteriori informazioni, consulta Differenze T-SQL in Babelfish.
- DENY
- END, MOVE CONVERSATION
- EXECUTE with AS LOGIN or AT option
- GET CONVERSATION GROUP
- GROUP BY ALL clause
- GROUP BY CUBE clause
- GROUP BY ROLLUP clause
- INSERT… DEFAULT VALUES
- MERGE
- READTEXT
- REVERT
- SELECT TOP... WITH TIES
- SELECT... FOR BROWSE
- SELECT... FOR XML AUTO
- SELECT... FOR XML EXPLICIT
- SELECT... FOR XML PATH
- SEND
- SET DATEFORMAT
- SET DEADLOCK_PRIORITY
- SET FMTONLY
- SET FORCEPLAN
- SET NUMERIC_ROUNDABORT ON
- SET OFFSETS

- SET REMOTE_PROC_TRANSACTIONS
- SET SHOWPLAN_TEXT
- SET SHOWPLAN_XML
- SET STATISTICS
- SET STATISTICS PROFILE
- SET STATISTICS TIME
- SET STATISTICS XML
- SHUTDOWN statement
- UPDATE STATISTICS
- UPDATETEXT
- Using EXECUTE to call a SQL function
- VIEW... CHECK OPTION clause
- VIEW... VIEW_METADATA clause
- WAITFOR DELAY
- WAITFOR TIME
- WAITFOR, RECEIVE
- WITH XMLNAMESPACES construct
- WRITETEXT
- XPATH expressions

Utilizzo delle procedure di Babelfish per Aurora PostgreSQL

Panoramica

Puoi utilizzare la seguente procedura per le istanze database Amazon RDS che eseguono Babelfish per Aurora PostgreSQL per migliorare le prestazioni delle query:

- sp_babelfish_volatility
- sp_execute_postgresql

sp_babelfish_volatility

La volatilità delle funzioni PostgreSQL aiuta il sistema di ottimizzazione a migliorare l'esecuzione delle query, che, se utilizzata in alcune parti di determinate clausole, ha un impatto significativo sulle prestazioni delle query.

Sintassi

```
sp_babelfish_volatility 'function_name', 'volatility'
```

Argomenti

function_name (opzionale)

Puoi specificare il valore di questo argomento con un nome composto da due parti come `schema_name.function_name` o solo `function_name`. Se si specifica solo `function_name`, il nome dello schema è lo schema predefinito per l'utente corrente.

volatility (opzionale)

I valori PostgreSQL validi per la volatilità sono`stable`, `volatile` o `immutable`. Per ulteriori informazioni, consulta https://www.postgresql.org/docs/current/xfunc-volatility.html

> **Note**
>
> Quando `sp_babelfish_volatility` viene chiamato con `function_name` che ha più definizioni, verrà generato un errore.

Set di risultati

Se i parametri non sono menzionati, il set di risultati viene visualizzato nelle seguenti colonne: `schemaname`, `functionname`, `volatility`.

Note per l'utilizzo

La volatilità delle funzioni PostgreSQL aiuta il sistema di ottimizzazione a migliorare l'esecuzione delle query, che, se utilizzata in alcune parti di determinate clausole, ha un impatto significativo sulle prestazioni delle query.

Esempi

Gli esempi seguenti mostrano come creare funzioni semplici e in seguito spiegano come utilizzare `sp_babelfish_volatility` con metodi diversi.

```
1> create function f1() returns int as begin return 0 end
2> go
```

```
1> create schema test_schema
2> go
```

```
1> create function test_schema.f1() returns int as begin return 0 end
2> go
```

L'esempio seguente mostra la volatilità delle funzioni:

```
1> exec sp_babelfish_volatility
2> go

schemaname  functionname volatility
----------- ------------ ----------
dbo         f1           volatile
test_schema f1           volatile
```

L'esempio seguente mostra come modificare la volatilità delle funzioni:

```
1> exec sp_babelfish_volatility 'f1','stable'
2> go
1> exec sp_babelfish_volatility 'test_schema.f1','immutable'
```

```
2> go
```

Quando si specifica solo function_name, vengono visualizzati il nome dello schema, il nome della funzione e la volatilità di tale funzione. L'esempio seguente mostra la volatilità delle funzioni dopo la modifica dei valori:

```
1> exec sp_babelfish_volatility 'test_schema.f1'
2> go

schemaname  functionname volatility
----------- ------------ ----------
test_schema f1           immutable
```

```
1> exec sp_babelfish_volatility 'f1'
2> go

schemaname  functionname volatility
----------- ------------ ----------
dbo         f1           stable
```

Quando non specifichi alcun argomento, viene visualizzato un elenco di funzioni (nome dello schema, nome della funzione, volatilità delle funzioni) presenti nel database corrente:

```
1> exec sp_babelfish_volatility
2> go

schemaname  functionname volatility
----------- ------------ ----------
dbo         f1           stable
test_schema f1           immutable
```

sp_execute_postgresql

Permette di eseguire istruzioni PostgreSQL dall'endpoint T-SQL. In questo modo, non devi uscire dalla porta T-SQL per eseguire queste istruzioni, il che semplifica le tue applicazioni.

Sintassi

```
sp_execute_postgresql [ @stmt = ] statement
```

Argomenti

Istruzione [@stmt]

Il tipo di dati dell'argomento è varchar. Questo argomento accetta istruzioni in dialetto PG.

> **Note**
>
> È possibile passare come argomento solo un'istruzione in dialetto PG, in caso contrario viene generato l'errore seguente.

```
1>exec sp_execute_postgresql 'create extension pg_stat_statements; drop extension
 pg_stat_statements'
2>go
```

```
Msg 33557097, Level 16, State 1, Server BABELFISH, Line 1
expected 1 statement but got 2 statements after parsing
```

Note per l'utilizzo

CREATE EXTENSION

Crea e carica una nuova estensione nel database attuale.

```
1>EXEC sp_execute_postgresql 'create extension [ IF NOT EXISTS ] <extension name>
 [ WITH ] [SCHEMA schema_name] [VERSION version]';
2>go
```

Il seguente esempio illustra come creare un'estensione:

```
1>EXEC sp_execute_postgresql 'create extension pg_stat_statements with schema sys
 version "1.10"';
2>go
```

Per concedere l'accesso alla funzione Lambda, utilizza il seguente comando:

```
1>select * from pg_stat_statements;
2>go
```

> **Note**
>
> Se il nome dello schema non viene fornito esplicitamente durante la creazione dell'estensione, per impostazione predefinita le estensioni vengono installate nello schema public. È necessario fornire il qualificatore dello schema per accedere agli oggetti di estensione come indicato di seguito:

```
1>select * from [public].pg_stat_statements;
2>go
```

Estensioni supportate

Le seguenti estensioni disponibili con Aurora PostgreSQL funzionano con Babelfish.

- `pg_stat_statements`
- `tds_fdw`
- `fuzzystrmatch`

Limitazioni

- Per poter installare le estensioni, gli utenti devono avere il ruolo sysadmin su T-SQL e rds_superuser su postgres.
- Non è possibile installare estensioni in schemi creati dall'utente e nemmeno negli schemi dbo e guest per i database master, tempdb e msdb.
- L'opzione CASCADE non è supportata.

ALTER EXTENSION

È possibile eseguire l'aggiornamento a una nuova versione dell'estensione utilizzando l'istruzione ALTER extension.

```
1>EXEC sp_execute_postgresql 'alter extension <extension name> UPDATE TO
 <new_version>';
2>go
```

Limitazioni

- È possibile aggiornare la versione dell'estensione solo utilizzando l'istruzione ALTER Extension. Altre operazioni non sono supportate.

DROP EXTENSION

Rimuove l'estensione specificata. Per rimuovere l'estensione, è anche possibile usare le nostre opzioni `if exists` o `restrict`.

```
1>EXEC sp_execute_postgresql 'drop extension <extension name>';
2>go
```

Limitazioni

- L'opzione CASCADE non è supportata.

Babelfish supporta i metodi dei tipi di dati XML

A partire dalla versione 5.4.0, Babelfish supporta ora le stored procedure sp_xml_preparedocument e sp_xml_removedocument, la funzione rowset OPENXML () e il metodo xml dataype .VALUE (). Con queste funzioni e procedure, l'interrogazione su dati XML diventa molto più semplice.

Comprensione delle procedure e dei metodi XML

- sp_xml_preparedocument — La procedura sp_xml_preparedocument analizza un testo XML fornito come input e restituisce un handle a questo documento. Questo handle è valido durante la sessione o finché non viene rimosso da sp_xml_removedocument.
- sp_xml_removedocument — La procedura sp_xml_removedocument invalida l'handle creato dalla procedura sp_xml_preparedocument.
- OPENXML () — OPENXML fornisce una visualizzazione del set di righe su un documento XML. Poiché OPENXML è un fornitore di set di righe e restituisce un insieme di righe, possiamo usare OPENXML nella clausola FROM proprio come qualsiasi altra tabella, vista o funzione con valori di tabella.
- VALUE () — Il metodo del tipo di dati XML VALUE () viene utilizzato per estrarre un valore da un'istanza XML memorizzata in una colonna, un parametro o una variabile di tipo xml.

Limitazioni nelle procedure e nei metodi XML di Babelfish

- Babelfish supporta solo la sintassi XPATH 1.0 per il secondo argomento (cioè ROWPATTERN) di OPENXML ().
- Le meta-proprietà e il flag 8 non sono attualmente supportati in OPENXML ().
- Babelfish supporta la sintassi XPATH 1.0 solo per il primo argomento (cioè XQuery) del metodo del tipo di dati VALUE ().

Prestazioni e dimensionamento per Amazon Aurora PostgreSQL

La sezione seguente descrive come gestire le prestazioni e il dimensionamento per un cluster di database di Amazon Aurora PostgreSQL. Include anche informazioni su altre attività di manutenzione.

Argomenti

- Dimensionamento delle istanze database Aurora PostgreSQL

- Numero massimo di connessioni a un'istanza database Aurora PostgreSQL
- Limiti di storage temporaneo per Aurora PostgreSQL
- Huge Pages per Aurora PostgreSQL
- Test di Amazon Aurora PostgreSQL mediante query Fault Injection
- Visualizzazione dello stato del volume per un cluster di database Aurora PostgreSQL
- Specificare il disco RAM per stats_temp_directory
- Gestione dei file temporanei con PostgreSQL

Dimensionamento delle istanze database Aurora PostgreSQL

È possibile dimensionare le istanze database Aurora PostgreSQL in due diversi modi, ovvero tramite il dimensionamento delle istanze e il dimensionamento della lettura. Per ulteriori informazioni sul dimensionamento della lettura, consulta Dimensionamento della lettura.

Puoi dimensionare il cluster di database Aurora PostgreSQL in base alle necessità, modificando la classe di istanza database per ogni istanza database nel cluster di database. Aurora PostgreSQL supporta diverse classi di istanza database ottimizzate per Aurora. Non utilizzare classi di istanza db.t2 o db.t3 per cluster Aurora più grandi di dimensioni maggiori di 40 terabyte (TB).

> **Note**
>
> Consigliamo di utilizzare le classi di istanza database T solo per i server di sviluppo e test o altri server non di produzione. Per ulteriori informazioni sulle classi di istanza T, consulta Tipi di classi di istanza database.

Il dimensionamento non è immediato. Potrebbero essere necessari 15 minuti o più per completare la modifica a una classe di istanza database diversa. Se si utilizza questo approccio per modificare la classe di istanza database, la modifica viene applicata durante la prossima finestra di manutenzione pianificata (anziché immediatamente) per evitare di influenzare gli utenti.

In alternativa alla modifica diretta della classe di istanza database, è possibile ridurre al minimo i tempi di inattività utilizzando le funzionalità ad alta disponibilità di Amazon Aurora. Innanzitutto, aggiungi una replica Aurora al cluster. Quando crei la replica, scegli la dimensione della classe di istanza database da utilizzare per il cluster. Quando la replica Aurora è sincronizzata con il cluster, viene quindi eseguito il failover sulla replica appena aggiunta. Per ulteriori informazioni, consultare Repliche di Aurora e Failover rapido con Amazon Aurora PostgreSQL.

Per le specifiche dettagliate delle classi di istanza database supportate da Aurora PostgreSQL, consulta Motori di database supportati per classi di istanza database.

Numero massimo di connessioni a un'istanza database Aurora PostgreSQL

Un cluster di database Aurora PostgreSQL alloca le risorse in base alla classe dell'istanza database e alla memoria disponibile. Ogni connessione al cluster di database consuma quantità incrementali di queste risorse, come memoria e CPU. La memoria consumata per connessione varia in base al tipo di query, al conteggio e all'utilizzo delle tabelle temporanee. Anche una connessione inattiva consuma memoria e CPU. Questo perché quando le query vengono eseguite su una connessione, viene allocata più memoria per ogni query che non viene rilasciata completamente, anche quando l'elaborazione si arresta. Pertanto, ti consigliamo di assicurarti che le tue applicazioni non utilizzino connessioni inattive: ognuna di queste spreca risorse e influisce negativamente sulle prestazioni. Per ulteriori informazioni, consulta Resources consumed by idle PostgreSQL connections.

Il numero massimo di connessioni consentite a un'istanza database di Aurora PostgreSQL è determinato dal valore del parametro `max_connections` specificato nel gruppo di parametri per quell'istanza database. L'impostazione ideale per il parametro `max_connections` supporta tutte le connessioni client di cui l'applicazione ha bisogno, senza un eccesso di connessioni inutilizzate, oltre ad almeno altre 3 connessioni per supportare l'automazione AWS. Prima di modificare l'impostazione del parametro `max_connections`, è opportuno considerare quanto segue:

- Se il valore `max_connections` è troppo basso, l'istanza database di Aurora PostgreSQL potrebbe non avere connessioni sufficienti quando i client tentano di connettersi. Se ciò accade, tenta di connettersi utilizzando `psql` che genera messaggi di errore come il seguente:

  ```
  psql: FATAL: remaining connection slots are reserved for non-replication superuser
   connections
  ```

- Se il valore `max_connections` supera il numero di connessioni effettivamente necessarie, le connessioni inutilizzate possono causare un peggioramento delle prestazioni.

Il valore predefinito di `max_connections` è derivato dalla seguente funzione Aurora PostgreSQL `LEAST`:

`LEAST({DBInstanceClassMemory/9531392},5000)`.

Se desideri modificare il valore di `max_connections`, devi creare un gruppo di parametri cluster di database personalizzato e modificare il valore. Dopo aver applicato il gruppo di parametri database

personalizzato al cluster, assicurati di riavviare l'istanza primaria in modo che il nuovo valore diventi effettivo. Per ulteriori informazioni, consultare Amazon Aurora PostgreSQL parametri e Creazione di un gruppo di parametri del cluster di database in Amazon Aurora.

> **Tip**
>
> Se le applicazioni aprono e chiudono frequentemente connessioni o mantengono un numero elevato di connessioni di lunga durata aperte, ti consigliamo di utilizzare Server proxy per Amazon RDS. Proxy RDS è un proxy di database completamente gestito e ad alta disponibilità che utilizza il pooling di connessioni per condividere connessioni di database in modo sicuro ed efficiente. Per ulteriori informazioni sul Proxy RDS, consulta Server proxy per Amazon RDS per Aurora.

Per informazioni dettagliate su come le istanze Aurora Serverless v2 gestiscono questo parametro, vedi Numero massimo connessioni per Aurora Serverless v2.

Limiti di storage temporaneo per Aurora PostgreSQL

Aurora PostgreSQL memorizza tabelle e indici nel sottosistema di archiviazione Aurora. Aurora PostgreSQL utilizza l'archiviazione temporanea separata per i file temporanei non persistenti. Sono inclusi i file utilizzati per scopi quali l'ordinamento di set di dati di grandi dimensioni durante l'elaborazione delle query o per le operazioni di creazione dell'indice. Per ulteriori informazioni, consulta l'articolo Come posso risolvere i problemi di archiviazione locale nelle istanze compatibili con Aurora PostgreSQL?.

Questi volumi di archiviazione locale sono supportati da Amazon Elastic Block Store e possono essere estesi utilizzando una classe di istanza database più grande. Per ulteriori informazioni sullo storage, consultare Archiviazione Amazon Aurora. È inoltre possibile aumentare l'archiviazione locale per gli oggetti temporanei utilizzando un tipo di istanza abilitato per NVMe e oggetti temporanei abilitati per le letture ottimizzate per Aurora. Per ulteriori informazioni, consulta Prestazioni delle query migliorate per Aurora PostgreSQL con Letture ottimizzate per Aurora.

> **Note**
>
> Durante il dimensionamento delle istanze database, ad esempio da db.r5.2xlarge a db.r5.4xlarge, potrebbero essere visualizzati eventi `storage-optimization`.

La tabella seguente mostra la quantità massima di storage temporaneo disponibile per ogni classe di istanza database Aurora PostgreSQL. Per ulteriori informazioni sul supporto della classe di istanza database per Aurora, consultare Classi di istanze database Amazon Aurora.

DB instance class (Classe istanza database)	Storage temporaneo massimo disponibile (GiB)
db.x2g.16xlarge	1829
db.x2g.12xlarge	1606
db.x2g.8xlarge	1071
db.x2g.4xlarge	535
db.x2g.2xlarge	268
db.x2g.xlarge	134
db.x2g.large	67
db.r8g.48xlarge	3072
db.r8g.24xlarge	1536
db.r8g.16xlarge	998
db.r8g.12xlarge	749
db.r8g.8xlarge	499
db.r8g.4xlarge	250
db.r8g.2xlarge	125
db.r8g.xlarge	63
db.r8g.large	31
db.r7g.16xlarge	1008
db.r7g.12xlarge	756

DB instance class (Classe istanza database)	Storage temporaneo massimo disponibile (GiB)
db.r7g.8xlarge	504
db.r7g.4xlarge	252
db.r7g.2xlarge	126
db.r7g.xlarge	63
db.r7g.large	32
db.r7i.48xlarge	3072
db.r7i.24xlarge	1500
db.r7i.16xlarge	1008
db.r7i.12xlarge	748
db.r7i.8xlarge	504
db.r7i.4xlarge	249
db.r7i.2xlarge	124
db.r7i.xlarge	62
db.r7i.large	31
db.r6g.16xlarge	1008
db.r6g.12xlarge	756
db.r6g.8xlarge	504
db.r6g.4xlarge	252
db.r6g.2xlarge	126
db.r6g.xlarge	63

DB instance class (Classe istanza database)	Storage temporaneo massimo disponibile (GiB)
db.r6g.large	32
db.r6i.32xlarge	1829
db.r6i.24xlarge	1500
db.r6i.16xlarge	1008
db.r6i.12xlarge	748
db.r6i.8xlarge	504
db.r6i.4xlarge	249
db.r6i.2xlarge	124
db.r6i.xlarge	62
db.r6i.large	31
db.r5.24xlarge	1500
db.r5.16xlarge	1008
db.r5.12xlarge	748
db.r5.8xlarge	504
db.r5.4xlarge	249
db.r5.2xlarge	124
db.r5.xlarge	62
db.r5.large	31
db.r4.16xlarge	960
db.r4.8xlarge	480

DB instance class (Classe istanza database)	Storage temporaneo massimo disponibile (GiB)
db.r4.4xlarge	240
db.r4.2xlarge	120
db.r4.xlarge	60
db.r4.large	30
db.t4g.large	16,5
db.t4g.medium	8,13
db.t3.large	16
db.t3.medium	7,5

> **Note**
>
> I tipi di istanze abilitati per NVMe possono aumentare lo spazio temporaneo disponibile fino alla dimensione totale di NVMe. Per ulteriori informazioni, consulta Prestazioni delle query migliorate per Aurora PostgreSQL con Letture ottimizzate per Aurora.

È possibile monitorare lo spazio di archiviazione temporaneo disponibile per un'istanza database con il parametro `FreeLocalStorage` CloudWatch, come descritto in CloudWatch Parametri Amazon per Amazon Aurora. (Non valido per Aurora Serverless v2)

Per alcuni carichi di lavoro, è possibile ridurre la quantità di storage temporaneo allocando più memoria ai processi che stanno perfezionando l'operazione. Per aumentare la memoria disponibile per un'operazione, aumentando i valori dei parametri work_mem o maintenance _work_mem PostgreSQL.

Huge Pages per Aurora PostgreSQL

Le pagine di grandi dimensioni sono una caratteristica di gestione della memoria che riduce il sovraccarico quando un'istanza database lavora con grandi blocchi di memoria contigui, come quelli

utilizzati dai buffer condivisi. Questa caratteristica di PostgreSQL è supportata da tutte le versioni Aurora PostgreSQL attualmente disponibili.

Il parametro `Huge_pages` è attivato per impostazione predefinita per tutte le classi di istanza database diverse dalle classi di istanza t3.medium, db.t3.large, db.t4g.medium, db.t4g.large Non è possibile modificare il valore del parametro `huge_pages` o disattivare questa funzionalità nelle classi di istanza supportate di Aurora PostgreSQL.

Nelle istanze database Aurora PostgreSQL che non supportano la funzionalità di memoria per pagine di grandi dimensioni, l'utilizzo della memoria di processi specifici potrebbe aumentare senza le corrispondenti modifiche del carico di lavoro.

Il sistema alloca segmenti di memoria condivisa come la cache del buffer durante l'avvio del server. Quando non sono disponibili pagine di memoria di grandi dimensioni, il sistema non addebita queste allocazioni al processo postmaster. Al contrario, include la memoria nel processo che ha effettuato per primo l'accesso a ogni pagina da 4 KB nel segmento di memoria condivisa.

> **Note**
>
> Le connessioni attive condividono la memoria allocata in base alle esigenze, indipendentemente dal modo in cui l'utilizzo della memoria condivisa viene monitorato tra i processi.

Test di Amazon Aurora PostgreSQL mediante query Fault Injection

Puoi testare la tolleranza ai guasti del tuo cluster di database Aurora PostgreSQL utilizzando query fault injection. Le query di errore di iniezione vengono emesse come comandi SQL per un'istanza Amazon Aurora. Le query fault injection consentono di arrestare l'istanza in maniera anomala per poter testare il failover e il ripristino. È anche possibile simulare errore della replica di Aurora Replica, errore del disco e congestione del disco. Le query di fault injection sono supportate da tutte le versioni di Aurora PostgreSQL disponibili, come riportato di seguito.

- Aurora PostgreSQL versioni 12, 13, 14 e successive
- Aurora PostgreSQL versione 11.7 e versioni successive
- Aurora PostgreSQL versione 10.11 e versioni successive

Argomenti

- Test dell'arresto anomalo di un'istanza
- Test di un errore di replica Aurora
- Test di un errore del disco
- Test di una congestione del disco

Quando una query fault injection specifica un arresto anomalo, impone un arresto anomalo dell'istanza database di Aurora PostgreSQL. Le altre query fault injection generano simulazioni di eventi di errore, ma non causano l'evento. Quando invii una query fault injection, specifichi anche la durata della simulazione dell'evento di errore.

Puoi inviare una query fault injection a una delle istanze di replica Aurora eseguendo la connessione all'endpoint per la replica Aurora. Per ulteriori informazioni, consulta Connessioni degli endpoint Amazon Aurora.

Test dell'arresto anomalo di un'istanza

Puoi forzare un arresto anomalo di un'istanza Aurora PostgreSQL utilizzando la funzione query fault injection `aurora_inject_crash()`.

Per questa query fault injection, non si verifica alcun failover. Se desideri testare un failover, puoi scegliere l'operazione di istanza Failover per il cluster di database nella console di RDS, oppure utilizzare il comando failover-db-cluster della AWS CLI o l'operazione dell'API RDS FailoverDBCluster.

Sintassi

```
SELECT aurora_inject_crash ('instance' | 'dispatcher' | 'node');
```

Opzioni

Questa query fault injection accetta uno dei seguenti tipi di arresto anomalo. Il tipo di arresto anomalo non rileva la distinzione tra maiuscole e minuscole:

'instance'

Viene simulato un arresto anomalo del database compatibile con PostgreSQL per l'istanza Amazon Aurora.

'dispatcher'.

Viene simulato un arresto anomalo del dispatcher sull'istanza primaria per il cluster di database Aurora. Il dispatcher scrive gli aggiornamenti sul volume del cluster per un cluster di database Amazon Aurora.

'node'

Viene simulato un arresto anomalo del database compatibile con PostgreSQL e del dispatcher per l'istanza Amazon Aurora.

Test di un errore di replica Aurora

Puoi simulare l'errore di una replica Aurora mediante la funzione query fault injection `aurora_inject_replica_failure()`.

Un errore di replica Aurora blocca la replica nella replica Aurora o in tutte le repliche Aurora nel cluster di database in base alla percentuale specificata per l'intervallo di tempo specificato. Al termine di tale periodo, le repliche Aurora interessate sono automaticamente sincronizzate con l'istanza primaria.

Sintassi

```
SELECT aurora_inject_replica_failure(
   percentage_of_failure,
   time_interval,
   'replica_name'
);
```

Opzioni

Questa query fault injection accetta i seguenti parametri:

percentage_of_failure

La percentuale di repliche da bloccare durante l'evento di errore. Può essere un valore tra 0 e 100. Se specifichi 0, non viene bloccata alcuna replica. Se specifichi 100, vengono bloccate tutte le repliche.

time_interval

L'intervallo di tempo per simulare l'errore di replica Aurora. L’intervallo è in secondi. Ad esempio, se il valore è 20, la simulazione viene eseguita per 20 secondi.

> **Note**
>
> Fai attenzione quando specifichi l'intervallo di tempo per l'evento di errore della replica Aurora. Se specifichi un intervallo troppo lungo e l'istanza di scrittura scrive una grande quantità di dati durante l'evento di errore, il cluster database Aurora potrebbe ritenere che si è verificato un arresto anomalo della replica Aurora e quindi sostituirla.

replica_name

La replica Aurora in cui inserire la simulazione dell'errore. Specifica il nome della replica Aurora per simulare un errore di una singola replica Aurora. Specifica una stringa vuota per simulare errori per tutte le repliche Aurora nel cluster database.

Per identificare i nomi delle repliche, vedere la colonna `server_id` della funzione `aurora_replica_status()`. Per esempio:

```
postgres=> SELECT server_id FROM aurora_replica_status();
```

Test di un errore del disco

Puoi simulare un errore del disco per un cluster database Aurora PostgreSQL mediante la funzione query fault injection `aurora_inject_disk_failure()`.

Durante la simulazione di un errore del disco, il cluster di database Aurora PostgreSQL contrassegna in modo aleatorio i segmenti del disco come difettosi. Le richieste a tali segmenti sono bloccate per la durata della simulazione.

Sintassi

```
SELECT aurora_inject_disk_failure(
   percentage_of_failure,
   index,
   is_disk,
   time_interval
);
```

Opzioni

Questa query fault injection accetta i seguenti parametri:

percentage_of_failure

La percentuale del disco da contrassegnare come difettosa durante l'evento di errore. Può essere un valore tra 0 e 100. Se specifichi 0, nessuna parte del disco è contrassegnata come difettosa. Se specifichi 100, tutto il disco è contrassegnato come difettoso.

index

Un blocco di dati logico specifico nel quale simulare l'evento di errore. Se viene superato l'intervallo di blocchi logici o di dati dei nodi di storage disponibili, viene generato un errore che indica il valore di indice massimo che puoi specificare. Per evitare questo errore, consulta Visualizzazione dello stato del volume per un cluster di database Aurora PostgreSQL.

is_disk

Indica se l'errore di inserimento è in un blocco logico o in un nodo di storage. Se si specifica true, significa che gli errori di inserimento sono in un blocco logico. Se si specifica false, significa che gli errori di inserimento sono in un nodo di storage.

time_interval

L'intervallo di tempo per simulare l'errore del disco. L'intervallo è in secondi. Ad esempio, se il valore è 20, la simulazione viene eseguita per 20 secondi.

Test di una congestione del disco

È possibile simulare una congestione del disco per un cluster di database Aurora PostgreSQL mediante la funzione di query di iniezione di guasti `aurora_inject_disk_congestion()`.

Durante la simulazione della congestione del disco, il cluster di database Aurora PostgreSQL contrassegna in modo aleatorio i segmenti del disco come congestionati. Le richieste a tali segmenti vengono ritardate di un periodo di tempo compreso tra il ritardo minimo e quello massimo specificati per la durata della simulazione.

Sintassi

```
SELECT aurora_inject_disk_congestion(
   percentage_of_failure,
   index,
   is_disk,
   time_interval,
```

```
    minimum,
    maximum
);
```

Opzioni

Questa query fault injection accetta i seguenti parametri:

percentage_of_failure

La percentuale del disco da contrassegnare come congestionata durante l'evento di errore. Questo è un doppio valore compreso tra 0 e 100. Se specifichi 0, nessuna parte del disco è contrassegnata come congestionata. Se specifichi 100, tutto il disco è contrassegnato come congestionato.

index

Un blocco di dati logico o nodo storage specifico nel quale simulare l'evento di errore.

Se viene superato l'intervallo di blocchi logici o di dati dei nodi di storage disponibili, viene generato un errore che indica il valore di indice massimo che puoi specificare. Per evitare questo errore, consulta Visualizzazione dello stato del volume per un cluster di database Aurora PostgreSQL.

is_disk

Indica se l'errore di inserimento è in un blocco logico o in un nodo di storage. Se si specifica true, significa che gli errori di inserimento sono in un blocco logico. Se si specifica false, significa che gli errori di inserimento sono in un nodo di storage.

time_interval

L'intervallo di tempo per simulare la congestione del disco. L'intervallo è in secondi. Ad esempio, se il valore è 20, la simulazione viene eseguita per 20 secondi.

minimum, maximum

Il valore minimo e massimo del ritardo di congestione in millisecondi. I valori validi sono compresi tra 0,0 e 100,0 millisecondi. I segmenti del disco contrassegnati come congestionati vengono ritardati per un periodo di tempo aleatorio compreso nell'intervallo del valore minimo e massimo per la durata della simulazione. Il valore massimo deve essere maggiore del valore minimo.

Visualizzazione dello stato del volume per un cluster di database Aurora PostgreSQL

In Amazon Aurora, un volume di cluster di database è costituito da una raccolta di blocchi logici. Ognuno di questi rappresenta 10 gigabyte di storage allocato. Questi blocchi sono denominati gruppi di protezione.

I dati in ogni gruppo di protezione sono replicati su sei dispositivi di storage fisico denominati nodi di storage. Questi nodi sono allocati su tre zone di disponibilità (AZ) nella regione in cui si trova il cluster di database. Ogni nodo di storage contiene a sua volta uno o più blocchi di dati logici per il volume di cluster di database. Per ulteriori informazioni sui gruppi di protezione e i nodi di storage, consulta Introduzione del motore di storage Aurora nel blog AWS Database. Per ulteriori informazioni sui volumi del cluster Aurora, consulta Archiviazione Amazon Aurora.

Utilizza la funzione `aurora_show_volume_status()` per restituire le seguenti variabili di stato del server:

- `Disks` — Il numero totale di blocchi logici di dati per il volume del cluster di database.
- `Nodes` — Il numero totale di nodi di storage per il volume del cluster di database.

Puoi utilizzare la funzione `aurora_show_volume_status()` per evitare errori durante l'utilizzo la funzione fault injection `aurora_inject_disk_failure()`. La funzione fault injection `aurora_inject_disk_failure()` simula l'errore di un intero nodo di storage o di un singolo blocco logico di dati all'interno di un nodo di storage. Nella funzione, si specifica il valore di indice di un blocco logico di dati o nodo di storage specifico. Tuttavia, l'istruzione restituisce un errore se si specifica un valore di indice superiore al numero di blocchi di dati logici o nodi di storage utilizzati dal volume del cluster di database. Per ulteriori informazioni sulle query fault injection, consulta Test di Amazon Aurora PostgreSQL mediante query Fault Injection.

> **Note**
>
> La funzione `aurora_show_volume_status()` è disponibile per Aurora PostgreSQL versione 10.11. Per ulteriori informazioni sulle versioni di Aurora PostgreSQL, consulta versioni di Amazon Aurora PostgreSQL e versioni del motore.

Sintassi

```
SELECT * FROM aurora_show_volume_status();
```

Esempio

```
customer_database=> SELECT * FROM aurora_show_volume_status();
 disks | nodes
-------+-------
    96 |    45
```

Specificare il disco RAM per stats_temp_directory

Puoi utilizzare il parametro di Aurora PostgreSQL, `rds.pg_stat_ramdisk_size`, per specificare la memoria di sistema allocata a un disco RAM per l'archiviazione di `stats_temp_directory` PostgreSQL. Il parametro del disco RAM è disponibile in Aurora PostgreSQL 14 e versioni precedenti.

Per alcuni carichi di lavoro, l'impostazione di questo parametro può migliorare le prestazioni e ridurre i requisiti IO. Per ulteriori informazioni sul parametro `stats_temp_directory`, consulta Run-time Statistics nella documentazione di PostgreSQL. A partire dalla versione 15 di PostgreSQL, la community PostgreSQL ha iniziato a utilizzare la memoria condivisa dinamica. Quindi, non è necessario impostare `stats_temp_directory`.

Per abilitare un disco RAM per `stats_temp_directory`, imposta il parametro `rds.pg_stat_ramdisk_size` su un valore diverso da zero nel gruppo di parametri del cluster di database utilizzato dal tuo cluster di database. Questo parametro utilizza MB, quindi è necessario specificare un valore intero. Espressioni, formule e funzioni non sono valide per il parametro `rds.pg_stat_ramdisk_size`. Assicurati di riavviare il cluster di database in modo che la modifica abbia effetto. Per informazioni sull'estensione dei parametri consulta Gruppi di parametri per Amazon Aurora. Per ulteriori informazioni sul riavvio del cluster database, consulta Riavvio di un cluster Amazon Aurora DB o di un'istanza Amazon Aurora DB.

Ad esempio, il seguente comando AWS CLI imposta il parametro del disco RAM su 256 MB.

```
aws rds modify-db-cluster-parameter-group \
    --db-cluster-parameter-group-name db-cl-pg-ramdisk-testing \
    --parameters "ParameterName=rds.pg_stat_ramdisk_size, ParameterValue=256,
 ApplyMethod=pending-reboot"
```

Dopo il riavvio di cluster di database, esegui il seguente comando per visualizzare lo stato di `stats_temp_directory`:

```
postgres=> SHOW stats_temp_directory;
```

Il comando deve restituire i seguenti valori:

```
stats_temp_directory
----------------------------
/rdsdbramdisk/pg_stat_tmp
(1 row)
```

Gestione dei file temporanei con PostgreSQL

In PostgreSQL, una query complessa può eseguire più operazioni di ordinamento o hash contemporaneamente, ciascuna delle quali utilizza la memoria dell'istanza per archiviare i risultati fino al valore specificato nel parametro `work_mem`. Quando la memoria dell'istanza non è sufficiente, vengono creati file temporanei per archiviare i risultati. Questi vengono scritti su disco per completare l'esecuzione della query. Successivamente, questi file vengono rimossi automaticamente al completamento della query. In In Aurora PostgreSQL, questi file condividono l'archiviazione locale con altri file di log. Puoi monitorare lo spazio di archiviazione locale del tuo cluster database Aurora PostgreSQL osservando la metrica Amazon CloudWatch per `FreeLocalStorage`. Per ulteriori informazioni, consulta Risoluzione dei problemi relativi all'archiviazione locale.

Consigliamo di utilizzare cluster di lettura ottimizzata per Aurora per i carichi di lavoro che comportano più query simultanee che aumentano l'utilizzo di file temporanei. Questi cluster utilizzano l'archiviazione locale a blocchi SSD (Solid State Drive) basata su NVMe (Non-Volatile Memory Express) per posizionare i file temporanei. Per ulteriori informazioni, consulta Prestazioni delle query migliorate per Aurora PostgreSQL con Letture ottimizzate per Aurora.

È possibile utilizzare i seguenti parametri e funzioni per gestire i file temporanei nell'istanza.

- **`temp_file_limit`**: questo parametro annulla qualsiasi query che superi la dimensione definita in KB dal parametro temp_files. Questo limite impedisce a qualsiasi query di essere eseguita all'infinito e di consumare spazio su disco con file temporanei. È possibile stimare il valore utilizzando i risultati del parametro `log_temp_files`. È consigliabile esaminare il comportamento del carico di lavoro e impostare il limite in base alla stima. Gli esempi seguenti mostrano come viene annullata una query quando supera il limite.

```
postgres=>select * from pgbench_accounts, pg_class, big_table;
```

```
ERROR: temporary file size exceeds temp_file_limit (64kB)
```

- **log_temp_files**: questo parametro invia messaggi a postgresql.log quando i file temporanei di una sessione vengono rimossi. Questo parametro produce log dopo che una query è stata completata correttamente. Pertanto, potrebbe non essere utile nella risoluzione dei problemi delle query attive e con tempi di esecuzione lunghi.

  L'esempio seguente mostra che quando la query viene completata correttamente, le voci vengono registrate nel file postgresql.log mentre i file temporanei vengono eliminati.

```
2023-02-06 23:48:35 UTC:205.251.233.182(12456):adminuser@postgres:[31236]:LOG:
 temporary file: path "base/pgsql_tmp/pgsql_tmp31236.5", size 140353536
2023-02-06 23:48:35 UTC:205.251.233.182(12456):adminuser@postgres:[31236]:STATEMENT:
  select a.aid from pgbench_accounts a, pgbench_accounts b where a.bid=b.bid order by
 a.bid limit 10;
2023-02-06 23:48:35 UTC:205.251.233.182(12456):adminuser@postgres:[31236]:LOG:
 temporary file: path "base/pgsql_tmp/pgsql_tmp31236.4", size 180428800
2023-02-06 23:48:35 UTC:205.251.233.182(12456):adminuser@postgres:[31236]:STATEMENT:
  select a.aid from pgbench_accounts a, pgbench_accounts b where a.bid=b.bid order by
 a.bid limit 10;
```

- **pg_ls_tmpdir**: questa funzione disponibile in RDS per PostgreSQL 13 e versioni successive fornisce visibilità sull'attuale utilizzo dei file temporanei. La query completata non viene visualizzata nei risultati della funzione. Nell'esempio seguente, è possibile visualizzare i risultati di questa funzione.

```
postgres=>select * from pg_ls_tmpdir();
```

```
      name       |    size    |      modification
-----------------+------------+------------------------
```

```
 pgsql_tmp8355.1 | 1072250880 | 2023-02-06 22:54:56+00
 pgsql_tmp8351.0 | 1072250880 | 2023-02-06 22:54:43+00
 pgsql_tmp8327.0 | 1072250880 | 2023-02-06 22:54:56+00
 pgsql_tmp8351.1 |  703168512 | 2023-02-06 22:54:56+00
 pgsql_tmp8355.0 | 1072250880 | 2023-02-06 22:54:00+00
 pgsql_tmp8328.1 |  835031040 | 2023-02-06 22:54:56+00
 pgsql_tmp8328.0 | 1072250880 | 2023-02-06 22:54:40+00
(7 rows)
```

```
postgres=>select query from pg_stat_activity where pid = 8355;

query
-------------------------------------------------------------------------------------
select a.aid from pgbench_accounts a, pgbench_accounts b where a.bid=b.bid order by
 a.bid
(1 row)
```

Il nome del file include l'ID di elaborazione (PID) della sessione che ha generato il file temporaneo. Una query più avanzata, come nell'esempio seguente, esegue la somma dei file temporanei per ogni PID.

```
postgres=>select replace(left(name, strpos(name, '.')-1),'pgsql_tmp','') as pid,
 count(*), sum(size) from pg_ls_tmpdir() group by pid;
```

```
 pid  | count |   sum
------+-------+------------------
 8355 |     2 | 2144501760
 8351 |     2 | 2090770432
 8327 |     1 | 1072250880
 8328 |     2 | 2144501760
(4 rows)
```

- **pg_stat_statements**: se attivi il parametro pg_stat_statements, puoi visualizzare l'utilizzo medio dei file temporanei per chiamata. È possibile identificare il valore query_id della query e utilizzarlo per esaminare l'utilizzo dei file temporanei, come illustrato nell'esempio seguente.

```
postgres=>select queryid from pg_stat_statements where query like 'select a.aid from
 pgbench%';
```

```
       queryid
----------------------
 -7170349228837045701
(1 row)
```

```
postgres=>select queryid, substr(query,1,25), calls, temp_blks_read/calls
 temp_blks_read_per_call, temp_blks_written/calls temp_blks_written_per_call from
 pg_stat_statements where queryid = -7170349228837045701;
```

```
       queryid        |          substr           | calls | temp_blks_read_per_call |
 temp_blks_written_per_call
----------------------+---------------------------+-------+-------------------------
+----------------------------
 -7170349228837045701 | select a.aid from pgbench |    50 |                  239226 |
                    388678
(1 row)
```

- **`Performance Insights`**: nel pannello di controllo di Approfondimenti sulle prestazioni, puoi visualizzare l'utilizzo dei file temporanei attivando le metriche temp_bytes e temp_files. Puoi quindi vedere la media di entrambe queste metriche e verificare se corrispondono al carico di lavoro delle query. La visualizzazione all'interno di Approfondimenti sulle prestazioni non evidenzia in modo specifico le query che generano file temporanei. Tuttavia, combinando le informazioni di Approfondimenti sulle prestazioni con la query mostrata per il parametro `pg_ls_tmpdir`, è possibile definire, analizzare e risolvere eventuali problemi a livello di modifiche del carico di lavoro delle query.

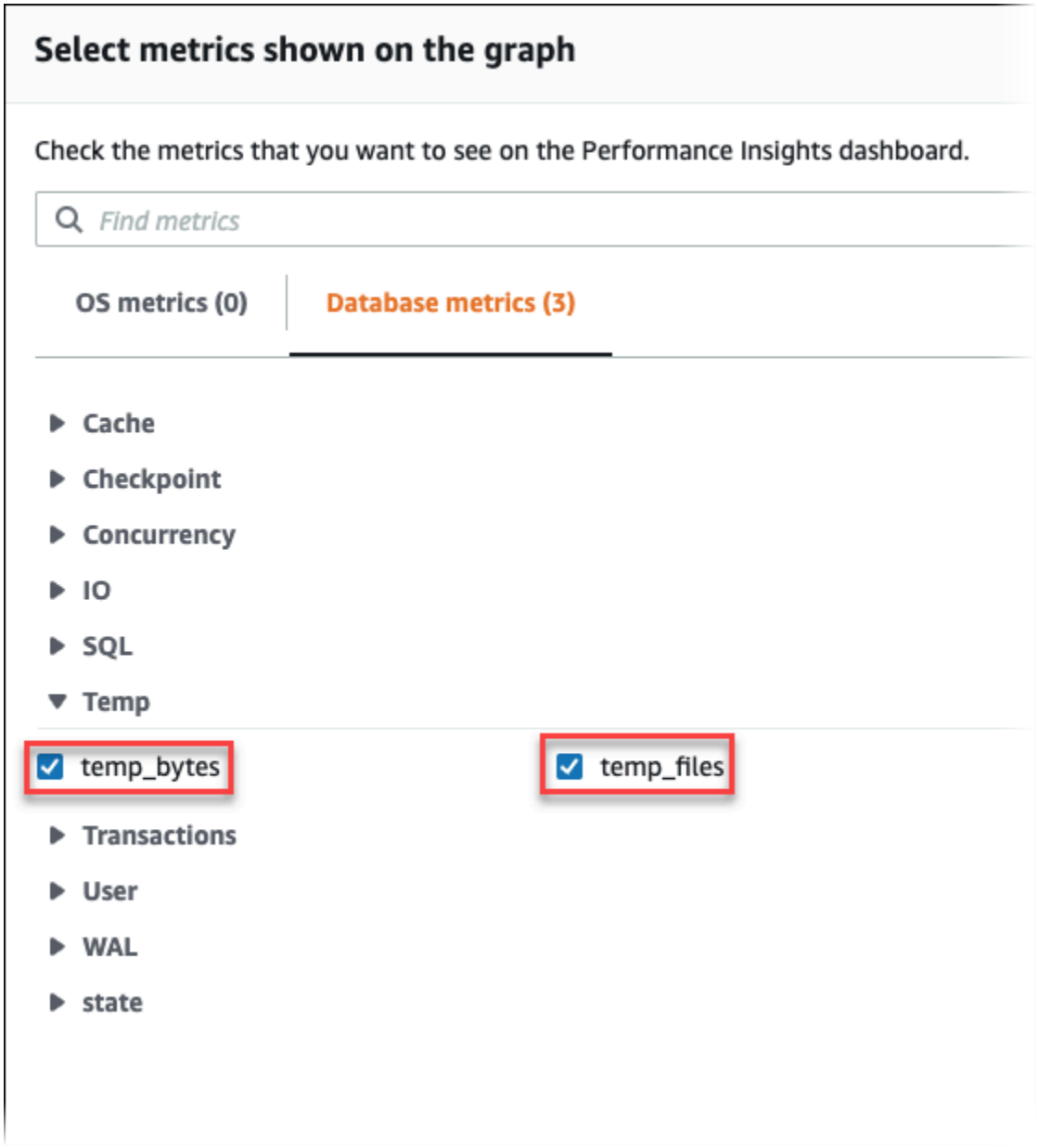
  Per ulteriori informazioni su come analizzare metriche e query con Approfondimenti sulle prestazioni, consulta Analisi delle metriche utilizzando il pannello di controllo Performance Insights.

  Per un esempio di visualizzazione dell'utilizzo dei file temporanei con Approfondimenti sulle prestazioni, consulta Visualizzazione dell'utilizzo dei file temporanei con Approfondimenti sulle prestazioni.

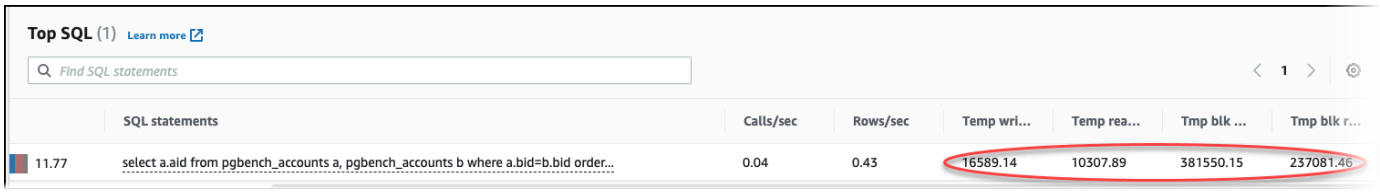
Visualizzazione dell’utilizzo dei file temporanei con Approfondimenti sulle prestazioni

È possibile utilizzare Approfondimenti sulle prestazioni per visualizzare l’utilizzo dei file temporanei attivando le metriche temp_bytes e temp_files. La visualizzazione in Approfondimenti sulle prestazioni non mostra le query specifiche che generano i file temporanei. Tuttavia, combinando Approfondimenti sulle prestazioni con la query mostrata per `pg_ls_tmpdir`, è possibile risolvere i problemi, analizzare e determinare le modifiche nel carico di lavoro delle query.

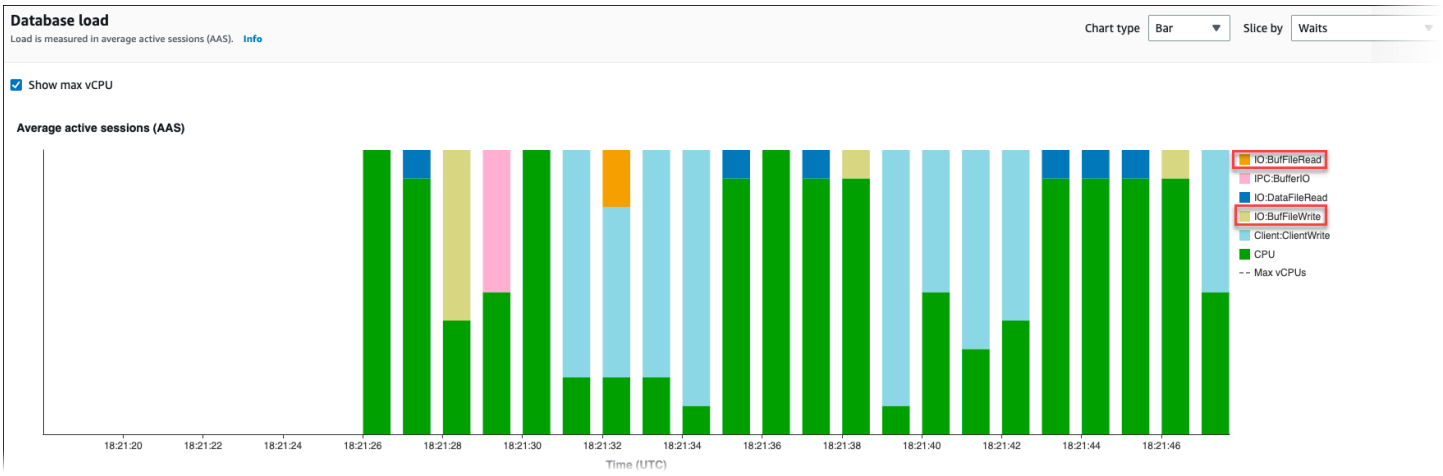
1. Nel pannello di controllo di Approfondimenti sulle prestazioni, scegli Gestisci parametri.
2. Seleziona Metriche del database e quindi seleziona le metriche temp_bytes e temp_files come illustrato nell'immagine seguente.



3. Nella scheda SQL principale, scegli l'icona Preferenze.
4. Nella finestra Preferenze, attiva le seguenti statistiche per visualizzarle nella scheda SQL principale e scegli Continua.
   - Scritture temporanee al secondo
   - Letture temporanee al secondo
   - Scritture temporanee in blocco a chiamata
   - Letture temporanee in blocco a chiamata
5. Il file temporaneo viene suddiviso quando viene combinato con la query visualizzata per `pg_ls_tmpdir`, come illustrato nell'esempio seguente.



Gli eventi `IO:BufFileRead` e `IO:BufFileWrite` si verificano quando le query principali del carico di lavoro creano spesso file temporanei. Puoi utilizzare Approfondimenti sulle prestazioni per identificare le query di livello superiore in attesa di `IO:BufFileRead` e `IO:BufFileWrite` esaminando la metrica Sessioni attive medie (AAS) nelle sezioni Caricamento del database e SQL principale.



Per ulteriori informazioni su come analizzare metriche principali ed eventi di attesa con Approfondimenti sulle prestazioni, consulta Panoramica della scheda Prime istruzioni SQL. Devi individuare e ottimizzare le query che causano un aumento dell'utilizzo dei file temporanei e dei

relativi eventi di attesa. Per ulteriori informazioni su questi eventi di attesa e sulla loro correzione, consulta IO:BufFileRead e IO:BufFileWrite.

> **Note**
>
> Il parametro `work_mem` controlla quando l'operazione di ordinamento esaurisce la memoria; i risultati vengono scritti in file temporanei. Si consiglia di non modificare l'impostazione di questo parametro specificando un valore superiore al valore predefinito perché ciò causerebbe un maggiore utilizzo della memoria da parte di ciascuna sessione del database. Inoltre, una sessione che esegue unioni e ordinamenti complessi può eseguire operazioni parallele in cui ogni operazione consuma memoria.
> Come best practice, in presenza di un report di grandi dimensioni con più unioni e ordinamenti, imposta questo parametro a livello di sessione utilizzando il comando `SET work_mem`. La modifica verrà quindi applicata solo alla sessione corrente e non comporterà la modifica del valore a livello globale.

Sintonizzazione degli eventi di attesa per Aurora PostgreSQL

Gli eventi di attesa sono un importante strumento di regolazione per Aurora PostgreSQL. Quando si scopre perché le sessioni sono in attesa di risorse e l'azione svolta, è possibile risolvere i colli di bottiglia in maniera più efficiente. È possibile utilizzare le informazioni contenute in questa sezione per trovare possibili cause e azioni correttive. Prima di approfondire questa sezione, è opportuno comprendere i concetti di base di Aurora, in particolare i seguenti argomenti:

- Archiviazione Amazon Aurora
- Gestione delle prestazioni e del dimensionamento dei cluster DB Aurora

> **Important**
>
> Gli eventi di attesa in questa sezione sono specifici di Aurora PostgreSQL. Utilizza le informazioni contenute in questa sezione per ottimizzare solo Amazon Aurora, non RDS per PostgreSQL.
> Alcuni eventi di attesa in questa sezione non hanno analoghi nelle versioni open source di questi motori di database. Altri eventi di attesa hanno lo stesso nome degli eventi nei motori open source, ma si comportano in modo diverso. Ad esempio, lo storage Amazon Aurora

funziona in modo diverso dallo storage open source, pertanto gli eventi di attesa relativi allo storage indicano condizioni di risorse diverse.

Argomenti

- Concetti essenziali per l'ottimizzazione di Aurora PostgreSQL
- Eventi di attesa Aurora PostgreSQL
- Client:ClientRead
- Client:ClientWrite
- CPU
- IO: BufFileRead e IO: BufFileWrite
- IO:DataFileRead
- IO:XactSync
- IPC:DamRecordTxAck
- Eventi di attesa IPC:parallel
- IPC: ProcArrayGroupUpdate
- Lock:advisory
- Lock:extend
- Lock:Relation
- Lock:transactionid
- Lock:tuple
- LWLock:buffer_content (BufferContent)
- LWLock:buffer_mapping
- LWLock:BufferIO (IPC:BufferIO)
- LWLock:lock_manager
- LWLock:MultiXact
- LWLock:pg_stat_statements
- Timeout: PG Sleep

Concetti essenziali per l'ottimizzazione di Aurora PostgreSQL

Prima di sintonizzare il database Aurora PostgreSQL, assicurati di sapere cosa sono gli eventi di attesa e perché si verificano. Esamina anche l'architettura di base della memoria e del disco di Aurora PostgreSQL. Per un utile diagramma architettonico, vedere il wikibook PostgreSQL.

Argomenti

- Eventi di attesa Aurora PostgreSQL
- Memoria Aurora PostgreSQL
- Processi Aurora PostgreSQL

Eventi di attesa Aurora PostgreSQL

Un evento di attesa indica una risorsa per la quale è in attesa di una sessione. Ad esempio, l'evento di attesa `Client:ClientRead` si verifica quando Aurora PostgreSQL è in attesa di ricevere dati dal client. Le risorse tipiche che una sessione attende includono quanto segue:

- Accesso a thread singolo a un buffer, ad esempio, quando una sessione sta tentando di modificare un buffer
- Una riga attualmente bloccata da un'altra sessione
- Un file di dati letto
- Scrittura di un file log

Ad esempio, per soddisfare una query, la sessione potrebbe eseguire una scansione completa della tabella. Se i dati non sono già in memoria, la sessione attende il completamento dell'I/O del disco. Quando i buffer vengono letti in memoria, potrebbe essere necessario attendere perché altre sessioni accedono agli stessi buffer. Il database registra le attese utilizzando un evento di attesa predefinito. Questi eventi sono raggruppati in categorie.

Un evento di attesa non mostra di per sé un problema di prestazioni. Ad esempio, se i dati richiesti non sono in memoria, è necessaria la lettura dei dati dal disco. Se una sessione blocca una riga per un aggiornamento, un'altra sessione attende che la riga venga sbloccata in modo che possa aggiornarla. Un commit richiede di attendere il completamento della scrittura su un file di registro. Le attese sono parte integrante del normale funzionamento di un database.

Un gran numero di eventi di attesa in genere mostrano un problema di prestazioni. In questi casi, è possibile utilizzare i dati degli eventi di attesa per determinare dove stanno trascorrendo il tempo delle

sessioni. Ad esempio, se un report che in genere viene eseguito in minuti ora viene eseguito per ore, è possibile identificare gli eventi di attesa che contribuiscono maggiormente al tempo di attesa totale. Se è possibile determinare le cause degli eventi di attesa principali, a volte è possibile apportare modifiche che migliorano le prestazioni. Ad esempio, se la sessione è in attesa su una riga bloccata da un'altra sessione, è possibile terminare la sessione di blocco.

Memoria Aurora PostgreSQL

La memoria Aurora PostgreSQL è divisa in condivisa e locale.

Argomenti

- Memoria condivisa in Aurora PostgreSQL
- Memoria locale in Aurora PostgreSQL

Memoria condivisa in Aurora PostgreSQL

Aurora PostgreSQL assegna memoria condivisa all'avvio dell'istanza. La memoria condivisa è divisa in più sottoaree. Di seguito è possibile trovare una descrizione dei più importanti.

Argomenti

- Buffer condivisi
- Buffer Write ahead log (WAL)

Buffer condivisi

Il pool buffer condiviso è un'area di memoria Aurora PostgreSQL che contiene tutte le pagine che sono o sono state utilizzate dalle connessioni delle applicazioni. Una pagina è la versione di memoria di un blocco disco. Il buffer pool condiviso memorizza nella cache i blocchi di dati letti dal disco. Il pool riduce la necessità di rileggere i dati dal disco, rendendo il database più efficiente.

Ogni tabella e indice vengono memorizzati come una matrice di pagine di dimensioni fisse. Ogni blocco contiene più tuple, che corrispondono alle righe. Una tupla può essere memorizzata in qualsiasi pagina.

Il buffer pool condiviso ha memoria finita. Se una nuova richiesta richiede una pagina che non è in memoria, e non esiste più memoria, Aurora PostgreSQL sfratterà una pagina utilizzata meno frequentemente per soddisfare la richiesta. La politica di sfratto è implementata da un algoritmo di sweep dell'orologio.

Il parametro `shared_buffers` determina la quantità di memoria che il server dedica alla memorizzazione nella cache dei dati.

Buffer Write ahead log (WAL)

Un Buffer write-ahead log (WAL) conserva i dati delle transazioni che Aurora PostgreSQL successivamente scrive sullo storage persistente. Utilizzando il meccanismo WAL, Aurora PostgreSQL può effettuare le seguenti operazioni:

- Recuperare i dati dopo un errore
- Ridurre l'I/O del disco evitando scritture frequenti su disco

Quando un client modifica i dati, Aurora PostgreSQL scrive le modifiche nel buffer WAL. Quando il client emette un `COMMIT`, il processo di scrittura WAL scrive i dati delle transazioni nel file WAL.

Il parametro `wal_level` determina la quantità di informazioni scritte sul WAL.

Memoria locale in Aurora PostgreSQL

Ogni processo di back-end assegna memoria locale per l'elaborazione delle query.

Argomenti

- Area di memoria di lavoro
- Area memoria di lavoro di manutenzione
- Area buffer temporanea

Area di memoria di lavoro

L'area di memoria di lavoro contiene dati temporanei per query che eseguono ordinamenti e hash. Ad esempio, una query con clausola `ORDER BY` esegue un ordinamento. Le query utilizzano tabelle hash nei join e nelle aggregazioni hash.

Il parametro `work_mem` indica la quantità di memoria da utilizzare dalle operazioni di ordinamento interno e dalle tabelle hash prima di scrivere su file di disco temporanei. Il valore predefinito è 4 MB. È possibile eseguire più sessioni contemporaneamente e ogni sessione può eseguire operazioni di manutenzione in parallelo. Per questo motivo, la memoria di lavoro totale utilizzata può essere costituita da multipli dell'impostazione `work_mem`.

Area memoria di lavoro di manutenzione

L'area di memoria di lavoro di manutenzione memorizza nella cache i dati per le operazioni di manutenzione. Queste operazioni includono l'aspirazione, la creazione di un indice e l'aggiunta di chiavi esterne.

Il parametro `maintenance_work_mem` specifica la quantità massima di memoria da utilizzare nelle operazioni di manutenzione. Il valore predefinito è 64 MB. Una sessione di database può eseguire solo un'operazione di manutenzione alla volta.

Area buffer temporanea

L'area buffer temporanea memorizza nella cache le tabelle temporanee per ciascuna sessione del database.

Ogni sessione assegna buffer temporanei secondo necessità fino al limite specificato. Quando la sessione scade, il server cancella i buffer.

Il parametro `temp_buffers` imposta il numero massimo di buffer temporanei utilizzati da ogni sessione. Prima del primo utilizzo di tabelle temporanee all'interno di una sessione, è possibile modificare il valore `temp_buffers`.

Processi Aurora PostgreSQL

Aurora PostgreSQL utilizza più processi.

Argomenti

- Processo postmaster
- Processi di back-end
- Processi in background

Processo postmaster

Il processo postmaster è il primo processo avviato quando si avvia Aurora PostgreSQL. Il processo postmaster ha le seguenti responsabilità principali:

- Forcella e monitoraggio dei processi in background
- Ricevere le richieste di autenticazione dai processi client e autenticarle prima di consentire al database di servire le richieste

Processi di back-end

Se il postmaster autentica una richiesta del cliente, il postmaster forcherà un nuovo processo di back-end, chiamato anche processo postgres. Un processo client si connette esattamente a un processo back-end. Il processo client e il processo di backend comunicano direttamente senza intervento da parte del processo postmaster.

Processi in background

Il processo postmaster forca diversi processi che eseguono diverse attività di back-end. Alcuni dei più importanti includono quanto segue:

- Scrittore WAL

  Aurora PostgreSQL scrive i dati nel buffer WAL (write ahead logging) nei file di log. Il principio della registrazione write ahead è che il database non può scrivere modifiche ai file di dati fino a quando il database ha scritto i record di log che descrivono tali modifiche su disco. Il meccanismo WAL riduce l'I/O del disco e consente ad Aurora PostgreSQL di utilizzare i log per recuperare il database dopo un errore.
- Background writer

  Questo processo scrive periodicamente pagine sporche (modificate) dai buffer di memoria ai file di dati. Una pagina diventa sporca quando un processo di back-end la modifica in memoria.
- Il daemon dell'Autovacuum

  Il daemon include i seguenti elementi:

  - Il lanciatore di autovacuum
  - I processi di autovacuum worker

  Se abilitata, verifica la presenza di tabelle con un numero elevato di tuple inserite, aggiornate o eliminate. Il daemon ha le seguenti responsabilità:

  - Recuperare o riutilizzare lo spazio su disco occupato da righe aggiornate o eliminate
  - Aggiornare le statistiche utilizzate dal planner
  - Proteggere contro la perdita di dati precedenti a causa di un involucro dell'ID transazione

  La funzione autovacuum automatizza l'esecuzione dei comandi `VACUUM` e `ANALYZE`. `VACUUM` ha le seguenti varianti: standard e full. Il vuoto standard funziona in parallelo con altre operazioni di database. `VACUUM FULL` richiede un blocco esclusivo sulla tabella su cui sta lavorando. Pertanto,

non può essere eseguito in parallelo con le operazioni che accedono alla stessa tabella. `VACUUM` crea una notevole quantità di traffico I/O, che può causare prestazioni scadenti per altre sessioni attive.

Eventi di attesa Aurora PostgreSQL

La seguente tabella elenca gli eventi di attesa per Aurora PostgreSQL che indicano più comunemente problemi di prestazioni, con cause più comuni e azioni correttive. I seguenti eventi di attesa sono un sottoinsieme dell'elenco in Eventi di attesa Amazon Aurora PostgreSQL.

Evento di attesa	Definizione
Client:ClientRead	Ad esempio, l'evento di attesa si verifica quando Aurora PostgreSQL è in attesa di ricevere dati dal client.
Client:ClientWrite	Ad esempio, l'evento di attesa si verifica quando Aurora PostgreSQL è in attesa di ricevere dati dal client.
CPU	Questo evento si verifica quando un thread è attivo nella CPU o è in attesa della CPU.
IO: BufFileRead e IO: BufFileWrite	Questi eventi si verificano quando Aurora PostgreSQL crea file temporanei.
IO:DataFileRead	Questo evento si verifica quando una connessione attende un processo di back-end per leggere una pagina richiesta dalla memoria perché la pagina non è disponibile nella memoria condivisa.
IO:XactSync	Questo evento si verifica quando il database è in attesa che il sottosistema di archiviazione Aurora riconosca il commit di una transazione regolare o il commit o il rollback di una transazione preparata.
IPC:DamRecordTxAck	Questo evento si verifica quando Aurora PostgreSQL in una sessione che utilizza flussi di

Evento di attesa	Definizione
	attività del database genera un evento di flusso di attività, quindi attende che tale evento diventi duraturo.
Lock:advisory	Questo evento si verifica quando un'applicazione PostgreSQL utilizza un blocco per coordinare l'attività su più sessioni.
Lock:extend	Questo evento si verifica quando un processo di back-end è in attesa di bloccare una relazione per estenderla mentre un altro processo ha un blocco su tale relazione per lo stesso scopo.
Lock:Relation	Questo evento si verifica quando una query è in attesa di acquisire un blocco su una tabella o vista attualmente bloccata da un'altra transazione.
Lock:transactionid	Questo evento si verifica quando una transazione è in attesa di un blocco a livello di riga.
Lock:tuple	Questo evento si verifica quando un processo di backend aspetta di acquisire un blocco su una tupla.
LWLock:buffer_content (BufferContent)	Questo evento si verifica quando una sessione è in attesa di accedere in lettura o scrittura a una pagina dati in memoria mentre un'altra sessione ha bloccato la pagina in scrittura.
LWLock:buffer_mapping	Questo evento si verifica quando un processo di backend è in attesa di associare un blocco di dati a un buffer nel pool di buffer condiviso.

Evento di attesa	Definizione
LWLock:BufferIO (IPC:BufferIO)	Questo evento si verifica quando Aurora PostgreSQL o RDS per PostgreSQL sono in attesa che altri processi finiscano le operazioni di input/output (I/O) quando si tenta contemporaneamente di accedere a una pagina.
LWLock:lock_manager	Questo evento si verifica quando il motore Aurora PostgreSQL mantiene l'area di memoria del blocco condiviso per allocare, controllare e dislocare un blocco quando non è possibile un blocco rapido del percorso.
LWLock:MultiXact	Questo tipo di evento si verifica quando Aurora PostgreSQL mantiene aperta una sessione per completare più transazioni che coinvolgono la stessa riga in una tabella. L'evento di attesa indica quale aspetto dell'elaborazione di più transazioni sta generando l'evento di attesa, ovvero LWLock:MultiXactOffsetSLRU, LWLock:MultiXactOffsetBuffer, LWLock:MultiXactMemberSLRU o LWLock:MultiXactMemberBuffer.
Timeout: PG Sleep	Questo evento si verifica quando un processo server ha chiamato la funzione `pg_sleep` e sta aspettando la scadenza del timeout del sonno.

Client:ClientRead

L'evento `Client:ClientRead` si verifica quando Aurora PostgreSQL è in attesa di ricevere dati dal client.

Argomenti

- Versioni del motore supportate
- Contesto
- Probabili cause di aumento delle attese

- Azioni

Versioni del motore supportate

Queste informazioni relative all'evento di attesa sono supportate per Aurora PostgreSQL versione 10 e successive.

Contesto

Un cluster Aurora PostgreSQL DB è in attesa di ricevere dati dal client. Il cluster Aurora PostgreSQL DB deve ricevere i dati dal client prima di poter inviare più dati al client. Il tempo in cui il cluster attende prima di ricevere i dati dal client è un evento `Client:ClientRead`.

Probabili cause di aumento delle attese

Le cause comuni della comparsa dell'evento `Client:ClientRead` che appare nelle prime attese includono:

Maggiore latenza di rete

Potrebbe esserci una maggiore latenza di rete tra il cluster Aurora PostgreSQL DB e il client. Una maggiore latenza di rete aumenta il tempo necessario per la ricezione dei dati dal client del cluster DB.

Aumento del carico sul client

Potrebbe esserci una pressione della CPU o una saturazione della rete sul client. Un aumento del carico sul client può ritardare la trasmissione dei dati dal client al cluster Aurora PostgreSQL DB.

Eccesso di viaggi di andata e ritorno in rete

Un gran numero di viaggi di andata e ritorno in rete tra il cluster Aurora PostgreSQL DB e il client possono ritardare la trasmissione dei dati dal client al cluster Aurora PostgreSQL DB.

Operazione copia di grandi dimensioni

Durante un'operazione di copia, i dati vengono trasferiti dal file system del client al cluster di database Aurora PostgreSQL. L'invio di una grande quantità di dati al cluster di database può ritardare la trasmissione dei dati dal client al cluster di database.

Connessione client inattiva

Una connessione a un'istanza database di Aurora PostgreSQL è inattiva in stato di transazione ed è in attesa che un client invii più dati o un comando. Questo stato può comportare un aumento degli eventi `Client:ClientRead`.

PGBouncer utilizzato per il connection pooling

PGBouncer ha un'impostazione di configurazione di rete di basso livello denominata `pkt_buf`, che è impostata su 4.096 per impostazione predefinita. Se il carico di lavoro invia pacchetti di query superiori a 4.096 byte tramite PGBouncer, consigliamo di aumentare l'impostazione `pkt_buf` a 8.192. Se la nuova impostazione non diminuisce il numero di eventi `Client:ClientRead`, consigliamo di aumentare l'impostazione `pkt_buf` su valori più grandi, come 16.384 o 32.768. Se il testo della query è grande, l'impostazione più grande può essere particolarmente utile.

Azioni

Consigliamo azioni diverse a seconda delle cause dell'evento di attesa.

Argomenti

- Posizionare i client nella stessa area di disponibilità e subnet VPC del cluster
- Ridimensionare il client
- Utilizza istanze di generazione corrente
- Aumentare la larghezza di banda di rete
- Monitora il massimo delle prestazioni di rete
- Monitorare le transazioni nello stato «inattivo nella transazione»

Posizionare i client nella stessa area di disponibilità e subnet VPC del cluster

Per ridurre la latenza di rete e aumentare il throughput di rete, posizionare i client nella stessa subnet di Availability Zone e Virtual Private Cloud (VPC) del cluster Aurora PostgreSQL DB. Assicurarsi che i client siano il più geograficamente vicini possibile al cluster DB.

Ridimensionare il client

Utilizzando Amazon CloudWatch o altri parametri host, stabilire se il proprio client è attualmente vincolato dalla CPU o dalla larghezza di banda di rete o entrambi. Se il client è vincolato, ridimensionare il client di conseguenza.

Utilizza istanze di generazione corrente

In alcuni casi, potresti non utilizzare una classe di istanza DB che supporta i frame jumbo. Se stai eseguendo l'applicazione su Amazon EC2, considera l'utilizzo di un'istanza di generazione corrente per il client. Inoltre, configura l'unità di trasmissione massima (MTU) sul sistema operativo client. Questa tecnica potrebbe ridurre il numero di round trip di rete e aumentare il throughput di rete. Per ulteriori informazioni, consulta Frame jumbo (9001 MTU) nella Guida per l'utente di Amazon EC2.

Per informazioni sulle classi di istanza database, consulta Classi di istanze database Amazon Aurora. Per determinare la classe di istanza DB equivalente a un tipo di istanza Amazon EC2, posizionare `db.` prima del nome del tipo di istanza Amazon EC2. Ad esempio, l'istanza Amazon EC2 `r5.8xlarge` è equivalente alla classe di istanza database `db.r5.8xlarge`.

Aumentare la larghezza di banda di rete

Utilizza i parametri Amazon CloudWatch `NetworkReceiveThroughput` e `NetworkTransmitThroughput` per monitorare il traffico di rete in entrata e in uscita sul cluster DB. Questi parametri possono aiutarti a determinare se la larghezza di banda della rete è sufficiente per il tuo carico di lavoro.

Se la larghezza di banda della rete non è sufficiente, aumentala. Se il file client AWS o l'istanza database sta raggiungendo i limiti di larghezza di banda di rete, l'unico modo per aumentare la larghezza di banda è aumentare la dimensione dell'istanza database.

Per ulteriori informazioni sui parametri CloudWatch, consultare CloudWatch Parametri Amazon per Amazon Aurora.

Monitora il massimo delle prestazioni di rete

Se utilizzi client Amazon EC2, Amazon EC2 fornisce il massimo per i parametri delle prestazioni di rete, inclusa la larghezza di banda aggregata in entrata e in uscita. Fornisce inoltre il monitoraggio della connessione per garantire che i pacchetti vengano restituiti come previsto e l'accesso ai servizi locali per servizi come il DNS (Domain Name System). Per monitorare questi massimi, utilizza un driver di rete avanzato e monitora le prestazioni di rete per il client.

Per ulteriori informazioni, consulta Monitoraggio delle prestazioni di rete per l'istanza EC2 nella Guida per l'utente di Amazon EC2 e Monitoraggio delle prestazioni di rete per l'istanza EC2 nella Guida per l'utente di Amazon EC2.

Monitorare le transazioni nello stato «inattivo nella transazione»

Controlla se hai un numero crescente di connessioni `idle in transaction`. Per fare ciò, monitora la colonna `state` nella tabella `pg_stat_activity`. Potrebbe essere possibile identificare l'origine della connessione eseguendo una query simile alla seguente.

```
select client_addr, state, count(1) from pg_stat_activity
where state like 'idle in transaction%'
group by 1,2
order by 3 desc
```

Client:ClientWrite

L'evento `Client:ClientWrite` si verifica quando Aurora PostgreSQL è in attesa di scrivere dati sul client.

Argomenti

- Versioni del motore supportate
- Contesto
- Probabili cause di aumento delle attese
- Azioni

Versioni del motore supportate

Queste informazioni relative all'evento di attesa sono supportate per Aurora PostgreSQL versione 10 e successive.

Contesto

Il processo client deve ricevere i dati da un cluster database Aurora PostgreSQL prima di poter inviare più dati. Il tempo in cui il cluster attende prima inviare altri dati al client è un evento `Client:ClientWrite`.

Il throughput di rete ridotto tra il cluster Aurora PostgreSQL DB e il client può causare questo evento. Anche la pressione della CPU e la saturazione della rete sul client possono causare questo evento.

Pressione CPU è quando la CPU è completamente utilizzata e ci sono attività in attesa del tempo della CPU. Saturazione rete è quando la rete tra il database e il client trasporta più dati di quelli che è in grado di gestire.

Probabili cause di aumento delle attese

Le cause comuni della comparsa dell'evento `Client:ClientWrite` che appare nelle prime attese includono:

Maggiore latenza di rete

Potrebbe esserci una maggiore latenza di rete tra il cluster Aurora PostgreSQL DB e il client. Una maggiore latenza di rete aumenta il tempo necessario per la ricezione dei dati dal client.

Aumento del carico sul client

Potrebbe esserci una pressione della CPU o una saturazione della rete sul client. Un aumento del carico sul client ritarda la ricezione dei dati dal cluster Aurora PostgreSQL DB.

Ampio volume di dati inviati al client

Il cluster Aurora PostgreSQL DB potrebbe inviare una grande quantità di dati al client. Un client potrebbe non essere in grado di ricevere i dati con la stessa rapidità dell'invio del cluster. Attività come una copia di una tabella di grandi dimensioni possono comportare un aumento degli eventi `Client:ClientWrite`.

Azioni

Consigliamo azioni diverse a seconda delle cause dell'evento di attesa.

Argomenti

- Posizionare i client nella stessa area di disponibilità e subnet VPC del cluster
- Utilizza istanze di generazione corrente
- Ridurre la quantità di dati inviati al client
- Ridimensionare il client

Posizionare i client nella stessa area di disponibilità e subnet VPC del cluster

Per ridurre la latenza di rete e aumentare il throughput di rete, posizionare i client nella stessa subnet di Availability Zone e Virtual Private Cloud (VPC) del cluster Aurora PostgreSQL DB.

Utilizza istanze di generazione corrente

In alcuni casi, potresti non utilizzare una classe di istanza DB che supporta i frame jumbo. Se stai eseguendo l'applicazione su Amazon EC2, considera l'utilizzo di un'istanza di generazione corrente per il client. Inoltre, configura l'unità di trasmissione massima (MTU) sul sistema operativo client. Questa tecnica potrebbe ridurre il numero di round trip di rete e aumentare il throughput di rete. Per ulteriori informazioni, consulta Frame jumbo (9001 MTU) nella Guida per l'utente di Amazon EC2.

Per informazioni sulle classi di istanza database, consulta Classi di istanze database Amazon Aurora. Per determinare la classe di istanza DB equivalente a un tipo di istanza Amazon EC2, posizionare `db.` prima del nome del tipo di istanza Amazon EC2. Ad esempio, l'istanza Amazon EC2 `r5.8xlarge` è equivalente alla classe di istanza database `db.r5.8xlarge`.

Ridurre la quantità di dati inviati al client

Quando possibile, regolare l'applicazione per ridurre la quantità di dati che il cluster Aurora PostgreSQL DB invia al client. Effettuare tali regolazioni allevia la contesa della CPU e della rete sul client.

Ridimensionare il client

Utilizzando Amazon CloudWatch o altri parametri host, stabilire se il proprio client è attualmente vincolato dalla CPU o dalla larghezza di banda di rete o entrambi. Se il client è vincolato, ridimensionare il client di conseguenza.

CPU

Questo evento si verifica quando un thread è attivo nella CPU o è in attesa della CPU.

Argomenti

- Versioni del motore supportate
- Context
- Probabili cause di aumento delle attese
- Azioni

Versioni del motore supportate

Queste informazioni relative all'evento di attesa sono supportate per Aurora PostgreSQL versione 9.6 e successive.

Context

L'unità di elaborazione centrale (CPU) è il componente di un computer che esegue le istruzioni. Ad esempio, le istruzioni della CPU eseguono operazioni aritmetiche e scambiano dati in memoria. Se una query aumenta il numero di istruzioni eseguite tramite il motore di database, il tempo impiegato per eseguire la query aumenta. Pianificazione CPU sta dando tempo alla CPU a un processo. La pianificazione viene orchestrata dal kernel del sistema operativo.

Argomenti

- Come sapere quando si verifica questa attesa
- Parametro DBLOADCPU
- Parametri di utilizzo di OS.CPU
- Probabile causa della pianificazione della CPU

Come sapere quando si verifica questa attesa

Questo evento di attesa `CPU` indica che un processo di backend è attivo nella CPU o è in attesa della CPU. Si verifica quando una query mostra le seguenti informazioni:

- La colonna `pg_stat_activity.state` ha il valore `active`.
- Le colonne `wait_event_type` e `wait_event` in `pg_stat_activity` sono entrambe `null`.

Per visualizzare i processi di backend in uso o in attesa sulla CPU, eseguire la seguente query.

```
SELECT *
FROM   pg_stat_activity
WHERE  state = 'active'
AND    wait_event_type IS NULL
AND    wait_event IS NULL;
```

Parametro DBLOADCPU

Il parametro Performance Insights per la CPU è `DBLoadCPU`. Il valore di `DBLoadCPU` può differire dal valore del parametro Amazon CloudWatch `CPUUtilization`. Quest'ultimo parametro viene raccolto dall'HyperVisor per un'istanza di database.

Parametri di utilizzo di OS.CPU

I parametri del sistema operativo Performance Insights forniscono informazioni dettagliate sull'utilizzo della CPU. Ad esempio, è possibile visualizzare i seguenti parametri:

- `os.cpuUtilization.nice.avg`
- `os.cpuUtilization.total.avg`
- `os.cpuUtilization.wait.avg`
- `os.cpuUtilization.idle.avg`

Performance Insights segnala l'utilizzo della CPU da parte del motore di database come `os.cpuUtilization.nice.avg`.

Probabile causa della pianificazione della CPU

Dal punto di vista del sistema operativo, la CPU è attiva quando non è in esecuzione il thread inattivo. La CPU è attiva mentre esegue un calcolo, ma è attiva anche in attesa di I/O di memoria. Questo tipo di I/O domina un carico di lavoro tipico del database.

È probabile che i processi attendano di essere pianificati su una CPU quando vengono soddisfatte le seguenti condizioni:

- Il parametro CloudWatch `CPUUtilization` è vicino al 100 percento.
- Il carico medio è superiore al numero di vCPU, indicando un carico pesante. Puoi trovare il parametro `loadAverageMinute` nella sezione parametri del sistema operativo in Performance Insights.

Probabili cause di aumento delle attese

Quando l'evento di attesa della CPU si verifica più del normale, probabilmente indicando un problema di prestazioni, le cause tipiche includono le seguenti.

Argomenti

- Probabili cause di picchi improvvisi
- Probabili cause di alta frequenza a lungo termine
- Casi particolari

Probabili cause di picchi improvvisi

Le cause più probabili di picchi improvvisi sono le seguenti:

- L'applicazione ha aperto troppe connessioni simultanee al database. Questo scenario è noto come «tempesta di connessione».
- Il carico di lavoro dell'applicazione è cambiato in uno dei seguenti modi:
  - Nuove query
  - Aumento delle dimensioni del set di dati
  - Manutenzione o creazione dell'indice
  - Nuove funzioni
  - Nuovi operatori
  - Un aumento dell'esecuzione parallela di query
- I piani di esecuzione delle query sono cambiati. In alcuni casi, un cambiamento può causare un aumento dei buffer. Ad esempio, la query ora utilizza una scansione sequenziale quando in precedenza utilizzava un indice. In questo caso, le query richiedono più CPU per raggiungere lo stesso obiettivo.

Probabili cause di alta frequenza a lungo termine

Le cause più probabili di eventi che si ripetono per un lungo periodo:

- Troppi processi backend sono in esecuzione contemporaneamente sulla CPU. Questi processi possono essere lavoratori paralleli.
- Le query vengono eseguite in modo subottimale perché necessitano di un numero elevato di buffer.

Casi particolari

Se nessuna delle cause probabili risulta essere una causa effettiva, potrebbero verificarsi le seguenti situazioni:

- La CPU sta scambiando i processi in entrata e in uscita.
- Il passaggio del contesto della CPU è aumentato.
- Nel codice Aurora PostgreSQL mancano gli eventi di attesa.

Azioni

Se l'evento di attesa CPU domina l'attività del database, non indica necessariamente un problema di prestazioni. Rispondi a questo evento solo quando le prestazioni diminuiscono.

Argomenti

- Indagare se il database sta causando l'aumento della CPU
- Determina se il numero di connessioni è aumentato
- Rispondere alle modifiche del carico di lavoro

Indagare se il database sta causando l'aumento della CPU

Esamina il parametro `os.cpuUtilization.nice.avg` in Performance Insights. Se questo valore è molto inferiore all'utilizzo della CPU, i processi nondatabase sono il principale contributore della CPU.

Determina se il numero di connessioni è aumentato

Esamina il parametro `DatabaseConnections` in Amazon CloudWatch. L'azione dipende dal fatto che il numero sia aumentato o diminuito durante il periodo di aumento degli eventi di attesa della CPU.

Le connessioni sono aumentate

Se il numero di connessioni è aumentato, confrontare il numero di processi back-end che consumano la vCPU con il numero di vCPU. Gli scenari possibili sono i seguenti:

- Il numero di processi backend che consumano vCPU è inferiore al numero di vCPU.

  In questo caso, il numero di connessioni non è un problema. Tuttavia, è comunque possibile provare a ridurre l'utilizzo della CPU.
- Il numero di processi backend che consumano vCPUs è inferiore al numero di vCPU.

  In questo caso, valuta le seguenti opzioni:
  - Riduci il numero di processi back-end collegati al database. Ad esempio, implementa una soluzione di connection pooling come RDS Proxy. Per ulteriori informazioni, consulta Server proxy per Amazon RDS per Aurora.
  - Aggiorna le dimensioni dell'istanza per ottenere un numero maggiore di vCPUs.
  - Reindirizza alcuni carichi di lavoro di sola lettura ai nodi del lettore, se applicabile.

Le connessioni sono aumentate

Esamina il parametro `blks_hit` in Performance Insights. Cerca una correlazione tra un aumento `blks_hit` e utilizzo CPU. Gli scenari possibili sono i seguenti:

- Utilizzo CPU e `blks_hit` sono correlati.

  In questo caso, trova le istruzioni SQL principali collegate all'utilizzo della CPU e cerca le modifiche al piano. Puoi utilizzare una delle seguenti tecniche:

  - Spiegare i piani manualmente e confrontarli con il piano di esecuzione previsto.
  - Cercare un aumento degli hit di blocco al secondo e dei blocchi locali al secondo. Nella sezione Top SQL della dashboard Performance Insights, scegli Preferenze.
- Utilizzo CPU e `blks_hit` non sono correlati.

  In questo caso, determinare se si verifica una delle seguenti condizioni:

  - L'applicazione si sta rapidamente connettendo e disconnettendo dal database.

    Diagnosticare questo comportamento attivando `log_connections` e `log_disconnections`, quindi analizzando i registri PostgreSQL. Considerare l'utilizzo dell'analizzatore di log `pgbadger`. Per ulteriori informazioni, consulta https://github.com/darold/pgbadger.
  - Il sistema operativo è sovraccarico.

    In questo caso, Performance Insights mostra che i processi back-end consumano la CPU per un tempo più lungo del solito. Cerca prove nei parametri `os.cpuUtilization` di Performance Insights o nel parametro CloudWatch `CPUUtilization`. Se il sistema operativo è sovraccarico, esamina i parametri di Enhanced Monitoring per effettuare ulteriori diagnosi. In particolare, guarda l'elenco dei processi e la percentuale di CPU consumata da ciascun processo.
  - Le istruzioni SQL principali consumano troppa CPU.

    Esaminare le istruzioni collegate all'utilizzo della CPU per verificare se possono utilizzare meno CPU. Esegui il comando `EXPLAIN` e concentrati sui nodi del piano che hanno il maggior impatto. Prendi in considerazione l'utilizzo di un visualizzatore del piano di esecuzione PostgreSQL. Per provare questo strumento, vedi http://explain.dalibo.com/.

Rispondere alle modifiche del carico di lavoro

Se il carico di lavoro è cambiato, cerca i seguenti tipi di modifiche:

Nuove query

Verifica se sono previste le nuove query. In tal caso, assicurarsi che siano previsti i piani di esecuzione e il numero di esecuzioni al secondo.

Aumento delle dimensioni del set di dati

Determina se il partizionamento, se non è già implementato, potrebbe essere d'aiuto. Questa strategia potrebbe ridurre il numero di pagine che una query deve recuperare.

Manutenzione o creazione dell'indice

Verificare se è previsto il programma per la manutenzione. Una best practice è pianificare le attività di manutenzione al di fuori delle attività di picco.

Nuove funzioni

Verifica se queste funzioni funzionano come previsto durante il test. In particolare, controlla se è previsto il numero di esecuzioni al secondo.

Nuovi operatori

Verifica se queste funzioni funzionano come previsto durante il test.

Un aumento dell'esecuzione di query parallele

Determina se si è verificata una delle seguenti situazioni:

- Le relazioni o gli indici coinvolti sono improvvisamente cresciuti di dimensioni in modo che differiscano in modo significativo da `min_parallel_table_scan_size` o `min_parallel_index_scan_size`.
- Cambiamenti recenti sono stati apportati a `parallel_setup_cost` o `parallel_tuple_cost`.
- Cambiamenti recenti sono stati apportati a `max_parallel_workers` o `max_parallel_workers_per_gather`.

IO: BufFileRead e IO: BufFileWrite

Gli eventi `IO:BufFileRead` e `IO:BufFileWrite` si verificano quando Aurora PostgreSQL crea file temporanei. Quando le operazioni richiedono più memoria rispetto ai parametri della memoria di lavoro attualmente definiti, scrivono dati temporanei sullo storage persistente. Questa operazione

viene talvolta chiamata «spilling to disk». Per ulteriori informazioni sui file temporanei e sul loro utilizzo, consulta Gestione dei file temporanei con PostgreSQL.

Argomenti

- Versioni del motore supportate
- Contesto
- Probabili cause di aumento delle attese
- Azioni

Versioni del motore supportate

Queste informazioni relative all'evento di attesa sono supportate per tutte le versioni di Aurora PostgreSQL.

Contesto

`IO:BufFileRead` e `IO:BufFileWrite` riguardano l'area di memoria di lavoro e l'area di memoria di lavoro di manutenzione. Per ulteriori informazioni su queste aree di memoria locale, consulta Area di memoria di lavoro e Area memoria di lavoro di manutenzione.

Il valore predefinito per `work_mem` è 4 MB. Se una sessione esegue operazioni in parallelo, ogni lavoratore che gestisce il parallelismo utilizza 4 MB di memoria. Per questo motivo, imposta `work_mem` con attenzione. Se si aumenta troppo il valore, un database che esegue molte sessioni potrebbe consumare troppa memoria. Se si imposta il valore troppo basso, Aurora PostgreSQL crea file temporanei nella memoria locale. Il disco I/O per questi file temporanei può ridurre le prestazioni.

Se si osserva la seguente sequenza di eventi, il database potrebbe generare file temporanei:

1. Riduzione improvvisa e brusca della disponibilità
2. Ripristino rapido per lo spazio libero

Potresti vedere anche un motivo a «motosega». Questo modello può indicare che il database sta creando costantemente file di piccole dimensioni.

Probabili cause di aumento delle attese

In generale, questi eventi di attesa sono causati da operazioni che consumano più memoria rispetto a quella allocata dai parametri `work_mem` o `maintenance_work_mem`. Per compensare, le operazioni

scrivono su file temporanei. Cause comuni degli eventi `IO:BufFileRead` e `IO:BufFileWrite` includono quanto segue:

Query che richiedono più memoria di quella esistente nell'area della memoria di lavoro

Le query con le seguenti caratteristiche utilizzano l'area di memoria di lavoro:

- Hash join
- `ORDER BY`Clausola
- `GROUP BY`Clausola
- `DISTINCT`
- Funzioni finestra
- `CREATE TABLE AS SELECT`
- Aggiornamento vista materializzata

Istruzioni che richiedono più memoria di quella esistente nell'area della memoria di lavoro di manutenzione

Le istruzioni seguenti utilizzano l'area di memoria di lavoro di manutenzione:

- `CREATE INDEX`
- `CLUSTER`

Azioni

Consigliamo azioni diverse a seconda delle cause dell’evento di attesa.

Argomenti

- Identificare il problema
- Esamina le tue query di join
- Esamina le query ORDER BY e GROUP BY
- Evitare di utilizzare l'operazione DISTINCT
- Considera l'utilizzo di funzioni finestra anziché le funzioni GROUP BY
- Indagare sulle viste materializzate e le istruzioni CTAS
- Usa pg_repack quando crei indici

- Aumenta maintenance_work_mem quando esegui cluster
- Ottimizza la memoria per prevenire IO: BufFileRead e IO: BufFileWrite

Identificare il problema

È possibile visualizzare l'utilizzo dei file temporanei direttamente in Performance Insights. Per ulteriori informazioni, consulta Visualizzazione dell'utilizzo dei file temporanei con Approfondimenti sulle prestazioni. Quando Performance Insights è disabilitato, potresti notare un aumento `IO:BufFileRead` `IO:BufFileWrite` delle operazioni. Per risolvere i problemi, procedi come segue:

1. Esamina la `FreeLocalStorage` metrica in Amazon CloudWatch.
2. Cerca un motivo a motosega, che è una serie di punte frastagliate.

Un motivo a motosega indica un consumo e un rilascio rapido dello storage, spesso associato a file temporanei. Se noti questo modello, attiva Performance Insights. Quando si utilizza Performance Insights, è possibile identificare quando si verificano gli eventi di attesa e quali query sono associate. La soluzione dipende dalla query specifica che causa gli eventi.

Oppure, imposta il parametro `log_temp_files`. Questo parametro registra tutte le query che generano più della soglia KB di file temporanei. Se il valore è `0`, Aurora PostgreSQL registra tutti i file temporanei. Se il valore è `1024`, Aurora PostgreSQL registra tutte le query che producono file temporanei di dimensioni superiori a 1 MB. Per ulteriori informazioni su `log_temp_files`, consulta Creazione di log e report di errore nella documentazione di PostgreSQL.

Esamina le tue query di join

Probabilmente la tua applicazione usa join. Ad esempio, la seguente query unisce quattro tabelle.

```
SELECT *
       FROM order
 INNER JOIN order_item
       ON (order.id = order_item.order_id)
 INNER JOIN customer
       ON (customer.id = order.customer_id)
 INNER JOIN customer_address
       ON (customer_address.customer_id = customer.id AND
           order.customer_address_id = customer_address.id)
 WHERE customer.id = 1234567890;
```

Una possibile causa di picchi nell'utilizzo temporaneo dei file è un problema nella query stessa. Ad esempio, una clausola interrotta potrebbe non filtrare correttamente i join. Considera il secondo inner join nell'esempio seguente.

```
SELECT *
       FROM order
 INNER JOIN order_item
       ON (order.id = order_item.order_id)
 INNER JOIN customer
       ON (customer.id = customer.id)
 INNER JOIN customer_address
       ON (customer_address.customer_id = customer.id AND
           order.customer_address_id = customer_address.id)
 WHERE customer.id = 1234567890;
```

La query precedente unisce erroneamente `customer.id` a `customer.id`, generando un prodotto cartesiano tra ogni cliente e ogni ordine. Questo tipo di join accidentale genera file temporanei di grandi dimensioni. A seconda delle dimensioni delle tabelle, una query cartesiana può anche riempire lo spazio di archiviazione. La domanda potrebbe avere join cartesiani quando vengono soddisfatte le seguenti condizioni:

- Si notano forti e nitide riduzioni della disponibilità dello storage, seguite da un rapido ripristino.
- Non sono in fase di creazione di indici.
- Non vengono rilasciate istruzioni `CREATE TABLE FROM SELECT`.
- Nessuna vista materializzata viene aggiornata.

Per verificare se le tabelle vengono unite utilizzando le chiavi appropriate, ispezionare le direttive di mappatura relazionale di query e oggetti. Tieni presente che alcune query della tua applicazione non vengono sempre chiamate e alcune query vengono generate dinamicamente.

Esamina le query ORDER BY e GROUP BY

In alcuni casi, una clausola `ORDER BY` può comportare file temporanei eccessivi. Considerare le linee guida seguenti:

- Includi solo colonne in una clausola `ORDER BY` quando devono essere ordinate. Questa linea guida è particolarmente importante per le query che restituiscono migliaia di righe e specificano molte colonne nella clausola `ORDER BY`.

- Considerando la creazione di indici per accelerare clausole ORDER BY quando corrispondono a colonne che hanno lo stesso ordine crescente o decrescente. Gli indici parziali sono preferibili perché sono più piccoli. Gli indici più piccoli vengono letti e attraversati più rapidamente.
- Se si creano indici per colonne che possono accettare valori nulli, considerare se si desidera che i valori nulli siano memorizzati alla fine o all'inizio degli indici.

  Se possibile, ridurre il numero di righe che devono essere ordinate filtrando il set di risultati. Se utilizzi istruzioni clausole o sottoquery WITH, ricorda che una query interna genera un set di risultati e lo passa alla query esterna. Maggiore è il numero di righe che una query può filtrare, minore è la necessità di ordinare la query.
- Se non è necessario ottenere il set completo di risultati, utilizzare la clausola LIMIT. Ad esempio, se si desidera solo le prime cinque righe, una query che utilizza la clausola LIMIT non continua a generare risultati. In questo modo, la query richiede meno memoria e file temporanei.

Una query che utilizza una clausola GROUP BY può richiedere anche file temporanei. Le query GROUP BY riepilogano i valori utilizzando funzioni come le seguenti:

- COUNT
- AVG
- MIN
- MAX
- SUM
- STDDEV

Per sintonizzare le query GROUP BY, segui i consigli per le query ORDER BY.

Evitare di utilizzare l'operazione DISTINCT

Se possibile, evitare di utilizzare l’operazione DISTINCT per rimuovere righe duplicate. Più righe non necessarie e duplicate restituite dalla tua query, più costosa diventa l’operazione DISTINCT. Se possibile, aggiungi filtri nella clausola WHERE anche se utilizzi gli stessi filtri per tabelle diverse. Filtrare la query e unirsi correttamente migliora le prestazioni e riduce l'utilizzo delle risorse. Previene inoltre report e risultati errati.

Se devi usare DISTINCT per più righe di una stessa tabella, prendi in considerazione la possibilità di creare un indice composito. Il raggruppamento di più colonne in un indice può migliorare il tempo

necessario per valutare righe distinte. Inoltre, se utilizzi Amazon Aurora PostgreSQL versione 10 o successiva, puoi correlare le statistiche tra più colonne utilizzando il comando `CREATE STATISTICS`.

Considera l'utilizzo di funzioni finestra anziché le funzioni GROUP BY

Utilizzando `GROUP BY` si modifica il set di risultati e quindi recupera il risultato aggregato. Utilizzando le funzioni della finestra, si aggregano i dati senza modificare il set di risultati. Una funzione finestra utilizza la clausola `OVER` per eseguire calcoli tra i set definiti dalla query, correlando una riga con un'altra. È possibile utilizzare tutte le funzioni `GROUP BY` nelle funzioni di finestra, ma anche funzioni come le seguenti:

- `RANK`
- `ARRAY_AGG`
- `ROW_NUMBER`
- `LAG`
- `LEAD`

Per ridurre al minimo il numero di file temporanei generati da una funzione di finestra, rimuovere le duplicazioni per lo stesso set di risultati quando sono necessarie due aggregazioni distinte. Considera la query seguente.

```
SELECT sum(salary) OVER (PARTITION BY dept ORDER BY salary DESC) as sum_salary
     , avg(salary) OVER (PARTITION BY dept ORDER BY salary ASC) as avg_salary
  FROM empsalary;
```

È possibile riscrivere la query con la clausola `WINDOW` come segue.

```
SELECT sum(salary) OVER w as sum_salary
         , avg(salary) OVER w as_avg_salary
    FROM empsalary
  WINDOW w AS (PARTITION BY dept ORDER BY salary DESC);
```

Per impostazione predefinita, il planner di esecuzione Aurora PostgreSQL consolida nodi simili in modo da non duplicare le operazioni. Tuttavia, utilizzando una dichiarazione esplicita per il blocco finestra, è possibile mantenere la query più facilmente. È inoltre possibile migliorare le prestazioni impedendo la duplicazione.

Indagare sulle viste materializzate e le istruzioni CTAS

Quando una vista materializzata si aggiorna, esegue una query. Questa query può contenere un'operazione come `GROUP BY`, `ORDER BY` oppure`DISTINCT`. Durante un aggiornamento, è possibile osservare un numero elevato di file temporanei e gli eventi di attesa `IO:BufFileWrite` e `IO:BufFileRead`. Allo stesso modo, quando crei una tabella basata su una dichiarazione `SELECT`, l'istruzione `CREATE TABLE` esegue una query. Per ridurre i file temporanei necessari, ottimizza la query.

Usa pg_repack quando crei indici

Quando crei un indice, il motore ordina il set di risultati. Man mano che le tabelle aumentano di dimensioni e man mano che i valori nella colonna indicizzata diventano più diversi, i file temporanei richiedono più spazio. Nella maggior parte dei casi, non è possibile impedire la creazione di file temporanei per tabelle di grandi dimensioni senza modificare l'area della memoria di lavoro di manutenzione. Per ulteriori informazioni, consulta Area memoria di lavoro di manutenzione.

Una possibile soluzione alternativa quando si ricrea un indice di grandi dimensioni consiste nell'utilizzare lo strumento pg_repack. Per ulteriori informazioni, consulta Riorganizzare le tabelle nei database PostgreSQL con blocchi minimi nella documentazione di pg_repack.

Aumenta maintenance_work_mem quando esegui cluster

Il comando `CLUSTER` raggruppa la tabella specificata da table_name in base a un indice esistente specificato da index_name. Aurora PostgreSQL ricrea fisicamente la tabella in modo che corrisponda all'ordine di un determinato indice.

Quando lo storage magnetico era prevalente, il clustering era comune perché il throughput di storage era limitato. Ora che lo storage basato su SSD è comune, il clustering è meno popolare. Tuttavia, se si raggruppano le tabelle, è comunque possibile aumentare leggermente le prestazioni a seconda delle dimensioni della tabella, dell'indice, della query e così via.

Se esegui il comando `CLUSTER` e osservi gli eventi di attesa `IO:BufFileWrite` e `IO:BufFileRead`, sintonizza `maintenance_work_mem`. Aumenta la dimensione della memoria a una quantità abbastanza grande. Un valore elevato significa che il motore può utilizzare più memoria per l'operazione di clustering.

Ottimizza la memoria per prevenire IO: BufFileRead e IO: BufFileWrite

In alcune situazioni, è necessario sintonizzare la memoria. Il tuo obiettivo è quello di bilanciare i seguenti requisiti:

- Il valore `work_mem` (vedi Area di memoria di lavoro)
- La memoria rimanente dopo aver scontato il valore `shared_buffers` (vedi Pool di buffer)
- Le connessioni massime aperte e in uso, limitate da `max_connections`

Aumentare le dimensioni dell'area di memoria di lavoro

In alcune situazioni, l'unica opzione è aumentare la memoria utilizzata dalla sessione. Se le tue query sono scritte correttamente e utilizzano i tasti corretti per i join, prendi in considerazione la possibilità di aumentare il valore `work_mem`. Per ulteriori informazioni, consulta Area di memoria di lavoro.

Per scoprire quanti file temporanei genera una query, imposta `log_temp_files` su `0`. Aumentando il valore `work_mem` al valore massimo identificato nei log, si impedisce alla query di generare file temporanei. Tuttavia, `work_mem` imposta il massimo per nodo piano per ogni connessione o operatore parallelo. Se il database ha 5.000 connessioni e se ciascuna utilizza una memoria di 256 MiB, il motore necessita di 1,2 TiB di RAM. Pertanto, la tua istanza potrebbe esaurirsi dalla memoria.

Riserva una memoria sufficiente per il buffer pool condiviso

Il database utilizza aree di memoria come il buffer pool condiviso, non solo l'area di memoria di lavoro. Considerare i requisiti di queste aree di memoria aggiuntive prima di aumentare `work_mem`. Per ulteriori informazioni sul pool di buffer, consulta Pool di buffer.

Ad esempio, supponiamo che la classe di istanza Aurora PostgreSQL sia db.r5.2xlarge. Questa classe ha 64 GiB di memoria. Per impostazione predefinita, il 75% della memoria è riservato al buffer pool condiviso. Dopo aver sottratto la quantità allocata all'area di memoria condivisa, rimangono 16.384 MB. Non allocare la memoria rimanente esclusivamente all'area della memoria di lavoro perché anche il sistema operativo e il motore richiedono memoria.

La memoria a cui è possibile allocare per `work_mem` dipende dalla classe di istanza. Se si utilizza una classe di istanza più grande, è disponibile più memoria. Tuttavia, nell'esempio precedente, non è possibile utilizzare più di 16 GiB. In caso contrario, la tua istanza diventa non disponibile quando esaurisce la memoria. Per ripristinare l'istanza dallo stato non disponibile, i servizi di automazione Aurora PostgreSQL si riavviano automaticamente.

Gestisci il numero di connessioni

Supponiamo che l'istanza del database disponga di 5.000 connessioni simultanee. Ogni connessione utilizza almeno 4 MiB di `work_mem`. L'elevato consumo di memoria delle connessioni rischia di peggiorare le prestazioni. Sono disponibili le seguenti opzioni:

- Eseguire l'aggiornamento a una classe di istanza più grande
- Diminuire il numero di connessioni simultanee al database utilizzando un proxy o un pool di connessioni.

Per i proxy, considera Amazon RDS Proxy, PGBouncer o un connection pooler basato sulla tua applicazione. Questa soluzione riduce il carico della CPU. Riduce inoltre il rischio quando tutte le connessioni richiedono l'area di memoria di lavoro. Quando esistono meno connessioni al database, è possibile aumentare il valore di `work_mem`. In questo modo, si riduce il verificarsi degli eventi di attesa `IO:BufFileRead` e `IO:BufFileWrite`. Inoltre, le query in attesa dell'area di memoria di lavoro accelerano in modo significativo.

IO:DataFileRead

L'evento `IO:DataFileRead` si verifica quando una connessione attende un processo di back-end per leggere una pagina richiesta dalla memoria perché la pagina non è disponibile nella memoria condivisa.

Argomenti

- Versioni del motore supportate
- Context
- Probabili cause di aumento delle attese
- Azioni

Versioni del motore supportate

Queste informazioni relative all'evento di attesa sono supportate per tutte el versioni di Aurora PostgreSQL.

Context

Tutte le query e le operazioni di manipolazione dei dati (DML) accedono alle pagine del buffer pool. Le dichiarazioni che possono indurre letture includono `SELECT`, `UPDATE` e `DELETE`. Ad esempio, un `UPDATE` può leggere pagine da tabelle o indici. Se la pagina richiesta o aggiornata non si trova nel buffer pool condiviso, questa lettura può portare all'evento `IO:DataFileRead`.

Poiché il buffer pool condiviso è finito, può essere riempito. In questo caso, le richieste di pagine che non sono in memoria impongono al database di leggere i blocchi dal disco. Se l'evento

`IO:DataFileRead` si verifica frequentemente, il buffer pool condiviso potrebbe essere troppo piccolo per adattarsi al carico di lavoro. Questo problema è particolarmente grave per le query `SELECT` che leggono un numero elevato di righe che non rientrano nel buffer pool. Per ulteriori informazioni sul pool di buffer, consulta Pool di buffer.

Probabili cause di aumento delle attese

Cause comuni dell'evento `IO:DataFileRead` includono quanto segue:

Picchi di connessione

Potresti trovare più connessioni che generano lo stesso numero di eventi di attesa IO:DataFileRead. In questo caso, può verificarsi un picco (aumento improvviso e grande) negli eventi `IO:DataFileRead`.

Le istruzioni SELECT e DML eseguono scansioni sequenziali

L'applicazione potrebbe aver eseguito una nuova operazione. Oppure un'operazione esistente potrebbe cambiare a causa di un nuovo piano di esecuzione. In questi casi, cerca tabelle (in particolare tabelle di grandi dimensioni) che abbiano un valore `seq_scan` maggiore. Puoi trovarli interrogando `pg_stat_user_tables`. Per tenere traccia delle query che generano più operazioni di lettura, utilizzare l'estensione `pg_stat_statements`.

CTAS e CREATE INDEX per set di dati di grandi dimensioni

Un CTAS è una `CREATE TABLE AS SELECT`dichiarazione. Se si esegue un CTAS utilizzando un set di dati di grandi dimensioni come origine o si crea un indice su una tabella di grandi dimensioni, l'evento `IO:DataFileRead` può verificarsi. Quando si crea un indice, il database potrebbe dover leggere l'intero oggetto utilizzando una scansione sequenziale. Un CTAS genera letture `IO:DataFile` quando le pagine non sono in memoria.

Diversi lavoratori sottovuoto in esecuzione contemporaneamente

Gli operatori del vuoto possono essere attivati manualmente o automaticamente. Raccomandiamo di adottare una strategia aggressiva per il vuoto. Tuttavia, quando una tabella contiene molte righe aggiornate o cancellate, l'attesa `IO:DataFileRead` aumenta. Dopo aver recuperato lo spazio, il tempo dedicato al vuoto su `IO:DataFileRead` diminuisce.

Ingresso di grandi quantità di dati

Quando l'applicazione acquisisce quantità di dati elevate, le operazioni `ANALYZE` potrebbero verificarsi più spesso. Il processo `ANALYZE` può essere attivato da un launcher automatico o richiamato manualmente.

L'operazione ANALYZE legge un sottoinsieme della tabella. Il numero di pagine che devono essere scansionate viene calcolato moltiplicando 30 per il valore `default_statistics_target`. Per ulteriori informazioni, consultare la documentazione di PostgreSQL. Il parametro `default_statistics_target` accetta valori compresi tra 1 e 10.000, dove il valore predefinito è 100.

Fame di risorse

Se si consuma la larghezza di banda di rete dell'istanza o la CPU, l'evento `IO:DataFileRead` potrebbe verificarsi più frequentemente.

Azioni

Consigliamo azioni diverse a seconda delle cause dell'evento di attesa.

Argomenti

- Controlla i filtri predicati per le query che generano attese
- Riduci al minimo l'effetto delle operazioni di manutenzione
- Rispondere a un numero elevato di connessioni

Controlla i filtri predicati per le query che generano attese

Supponiamo di identificare query specifiche che stanno generando eventi di attesa `IO:DataFileRead`. È possibile identificarli utilizzando le seguenti tecniche:

- Approfondimenti sulle prestazioni
- Viste catalogo come quella fornita dall'estensione `pg_stat_statements`
- La vista catalogo `pg_stat_all_tables`, se mostra periodicamente un numero maggiore di letture fisiche
- La vista `pg_statio_all_tables`, se lo mostra che i contatori `_read` sono in aumento

Si consiglia di determinare quali filtri vengono utilizzati nel predicato (clausola `WHERE`) di queste query. Seguire queste linee guida:

- Esegui il comando `EXPLAIN`. Nell'output, identificare quali tipi di scansioni vengono utilizzati. Una scansione sequenziale non indica necessariamente che ci sia un problema. Le query che utilizzano

scansioni sequenziali producono naturalmente più eventi `IO:DataFileRead` rispetto alle query che utilizzano filtri.

Scopri se la colonna elencata nella clausola `WHERE` è indicizzata. In caso contrario, prendi in considerazione la possibilità di creare un indice per questa colonna. Questo approccio evita le scansioni sequenziali e riduce gli eventi `IO:DataFileRead`. Se una query dispone di filtri restrittivi e continua a produrre scansioni sequenziali, valutare se vengono utilizzati gli indici appropriati.

- Scopri se la query sta accedendo a una tabella molto ampia. In alcuni casi, il partizionamento di una tabella può migliorare le prestazioni, consentendo alla query di leggere solo le partizioni necessarie.
- Esamina la cardinalità (numero totale di righe) dalle operazioni di join. Nota quanto sono restrittivi i valori che stai passando nei filtri per la tua clausola `WHERE`. Se possibile, sintonizza la query per ridurre il numero di righe passate in ogni fase del piano.

Riduci al minimo l'effetto delle operazioni di manutenzione

Operazioni di manutenzione come `VACUUM` e `ANALYZE` sono importanti. Si consiglia di non spegnerli qualora vengano trovati eventi di attesa `IO:DataFileRead` relativi a queste operazioni di manutenzione. I seguenti approcci possono ridurre al minimo l'effetto di queste operazioni:

- Eseguire manualmente le operazioni di manutenzione durante le ore non di punta. Questa tecnica impedisce al database di raggiungere la soglia per le operazioni automatiche.
- Per tabelle molto grandi, prendi in considerazione il partizionamento. Questa tecnica riduce il sovraccarico delle operazioni di manutenzione. Il database accede solo alle partizioni che richiedono manutenzione.
- Quando si acquisiscono grandi quantità di dati, prendere in considerazione la possibilità di disabilitare la funzione di analisi automatica.

La funzione autovacuum viene attivata automaticamente per una tabella quando la formula seguente è vera.

```
pg_stat_user_tables.n_dead_tup > (pg_class.reltuples x autovacuum_vacuum_scale_factor)
 + autovacuum_vacuum_threshold
```

La vista `pg_stat_user_tables` e il catalogo `pg_class` hanno più righe. Una riga può corrispondere a una riga della tabella. Questa formula presuppone che i `reltuples` sono per una tabella specifica. I parametri `autovacuum_vacuum_scale_factor` (0,20 per impostazione

predefinita) e `autovacuum_vacuum_threshold` (50 tuple per impostazione predefinita) sono generalmente impostate globalmente per l'intera istanza. Tuttavia, è possibile impostare valori diversi per una tabella specifica.

Argomenti

- Ricerca delle tabelle che consumano spazio inutile
- Trova tabelle che consumano spazio inutile
- Trova tabelle idonee per l'autovacuum

Ricerca delle tabelle che consumano spazio inutile

Per trovare le tabelle che consumano spazio più del necessario, esegui la query riportata di seguito. Quando questa query viene eseguita da un utente del database che non ha il ruolo `rds_superuser`, restituisce informazioni solo sulle tabelle per cui il ruolo utente ha le autorizzazioni di lettura. Questa query è supportata da PostgreSQL 12 e versioni successive.

```
WITH report AS (
   SELECT   schemaname
           ,tblname
           ,n_dead_tup
           ,n_live_tup
           ,block_size*tblpages AS real_size
           ,(tblpages-est_tblpages)*block_size AS extra_size
           ,CASE WHEN tblpages - est_tblpages > 0
              THEN 100 * (tblpages - est_tblpages)/tblpages::float
              ELSE 0
            END AS extra_ratio, fillfactor, (tblpages-est_tblpages_ff)*block_size AS
 bloat_size
           ,CASE WHEN tblpages - est_tblpages_ff > 0
              THEN 100 * (tblpages - est_tblpages_ff)/tblpages::float
              ELSE 0
            END AS bloat_ratio
           ,is_na
    FROM (
          SELECT  ceil( reltuples / ( (block_size-page_hdr)/tpl_size ) ) +
 ceil( toasttuples / 4 ) AS est_tblpages
                 ,ceil( reltuples / ( (block_size-page_hdr)*fillfactor/
(tpl_size*100) ) ) + ceil( toasttuples / 4 ) AS est_tblpages_ff
                 ,tblpages
                 ,fillfactor
                 ,block_size
```

```
                  ,tblid
                  ,schemaname
                  ,tblname
                  ,n_dead_tup
                  ,n_live_tup
                  ,heappages
                  ,toastpages
                  ,is_na
              FROM (
                     SELECT ( 4 + tpl_hdr_size + tpl_data_size + (2*ma)
                                - CASE WHEN tpl_hdr_size%ma = 0 THEN ma ELSE
tpl_hdr_size%ma END
                                - CASE WHEN ceil(tpl_data_size)::int%ma = 0 THEN ma ELSE
ceil(tpl_data_size)::int%ma END
                            ) AS tpl_size
                            ,block_size - page_hdr AS size_per_block
                            ,(heappages + toastpages) AS tblpages
                            ,heappages
                            ,toastpages
                            ,reltuples
                            ,toasttuples
                            ,block_size
                            ,page_hdr
                            ,tblid
                            ,schemaname
                            ,tblname
                            ,fillfactor
                            ,is_na
                            ,n_dead_tup
                            ,n_live_tup
                           FROM (
                                 SELECT  tbl.oid                       AS tblid
                                        ,ns.nspname                    AS schemaname
                                        ,tbl.relname                   AS tblname
                                        ,tbl.reltuples                 AS reltuples
                                        ,tbl.relpages                  AS heappages
                                        ,coalesce(toast.relpages, 0)   AS toastpages
                                        ,coalesce(toast.reltuples, 0)  AS toasttuples
                                        ,psat.n_dead_tup               AS n_dead_tup
                                        ,psat.n_live_tup               AS n_live_tup
                                        ,24                            AS page_hdr
                                        ,current_setting('block_size')::numeric AS
block_size
```

```
 ,coalesce(substring( array_to_string(tbl.reloptions, ' ') FROM
 'fillfactor=([0-9]+)')::smallint, 100) AS fillfactor
                                       ,CASE WHEN version()~'mingw32' OR version()~'64-
bit|x86_64|ppc64|ia64|amd64' THEN 8 ELSE 4 END        AS ma
                                       ,23 + CASE WHEN MAX(coalesce(null_frac,0)) > 0
 THEN ( 7 + count(*) ) / 8 ELSE 0::int END              AS tpl_hdr_size
                                       ,sum( (1-coalesce(s.null_frac, 0)) *
 coalesce(s.avg_width, 1024) )                                  AS tpl_data_size
                                       ,bool_or(att.atttypid =
 'pg_catalog.name'::regtype) OR count(att.attname) <> count(s.attname)         AS is_na
                                  FROM  pg_attribute       AS att
                                  JOIN  pg_class           AS tbl    ON (att.attrelid =
 tbl.oid)
                                  JOIN  pg_stat_all_tables AS psat   ON (tbl.oid =
 psat.relid)
                                  JOIN  pg_namespace       AS ns     ON (ns.oid =
 tbl.relnamespace)
                               LEFT JOIN  pg_stats         AS s      ON
 (s.schemaname=ns.nspname AND s.tablename = tbl.relname AND s.inherited=false AND
 s.attname=att.attname)
                               LEFT JOIN  pg_class         AS toast  ON
 (tbl.reltoastrelid = toast.oid)
                                   WHERE  att.attnum > 0
                                     AND  NOT att.attisdropped
                                     AND  tbl.relkind = 'r'
                                GROUP BY  tbl.oid, ns.nspname, tbl.relname,
 tbl.reltuples, tbl.relpages, toastpages, toasttuples, fillfactor, block_size, ma,
 n_dead_tup, n_live_tup
                                ORDER BY  schemaname, tblname
                              ) AS s
                  ) AS s2
       ) AS s3
 ORDER BY bloat_size DESC
)
  SELECT *
    FROM report
   WHERE bloat_ratio != 0
 -- AND schemaname = 'public'
 -- AND tblname = 'pgbench_accounts'
;

-- WHERE NOT is_na
```

```
--   AND tblpages*((pst).free_percent + (pst).dead_tuple_percent)::float4/100 >= 1
```

Puoi verificare l'aumento delle dimensioni della tabella e dell'indice nell'applicazione. Per ulteriori informazioni, consulta Diagnosi delle dimensioni della tabella e dell'indice.

Trova tabelle che consumano spazio inutile

Per trovare tabelle che richiedono spazio inutile, esegui la query riportata.

```
-- WARNING: run with a nonsuperuser role, the query inspects
-- only indexes on tables you have permissions to read.
-- WARNING: rows with is_na = 't' are known to have bad statistics ("name" type is not
 supported).
-- This query is compatible with PostgreSQL 8.2 and later.

SELECT current_database(), nspname AS schemaname, tblname, idxname,
 bs*(relpages)::bigint AS real_size,
  bs*(relpages-est_pages)::bigint AS extra_size,
  100 * (relpages-est_pages)::float / relpages AS extra_ratio,
  fillfactor, bs*(relpages-est_pages_ff) AS bloat_size,
  100 * (relpages-est_pages_ff)::float / relpages AS bloat_ratio,
  is_na
  -- , 100-(sub.pst).avg_leaf_density, est_pages, index_tuple_hdr_bm,
  -- maxalign, pagehdr, nulldatawidth, nulldatahdrwidth, sub.reltuples, sub.relpages
  -- (DEBUG INFO)
FROM (
  SELECT coalesce(1 +
       ceil(reltuples/floor((bs-pageopqdata-pagehdr)/(4+nulldatahdrwidth)::float)), 0
       -- ItemIdData size + computed avg size of a tuple (nulldatahdrwidth)
    ) AS est_pages,
    coalesce(1 +
       ceil(reltuples/floor((bs-pageopqdata-pagehdr)*fillfactor/
(100*(4+nulldatahdrwidth)::float))), 0
    ) AS est_pages_ff,
    bs, nspname, table_oid, tblname, idxname, relpages, fillfactor, is_na
    -- , stattuple.pgstatindex(quote_ident(nspname)||'.'||quote_ident(idxname)) AS
 pst,
    -- index_tuple_hdr_bm, maxalign, pagehdr, nulldatawidth, nulldatahdrwidth,
 reltuples
    -- (DEBUG INFO)
  FROM (
    SELECT maxalign, bs, nspname, tblname, idxname, reltuples, relpages, relam,
 table_oid, fillfactor,
      ( index_tuple_hdr_bm +
```

```
         maxalign - CASE -- Add padding to the index tuple header to align on MAXALIGN
           WHEN index_tuple_hdr_bm%maxalign = 0 THEN maxalign
           ELSE index_tuple_hdr_bm%maxalign
         END
       + nulldatawidth + maxalign - CASE -- Add padding to the data to align on
MAXALIGN
           WHEN nulldatawidth = 0 THEN 0
           WHEN nulldatawidth::integer%maxalign = 0 THEN maxalign
           ELSE nulldatawidth::integer%maxalign
         END
     )::numeric AS nulldatahdrwidth, pagehdr, pageopqdata, is_na
     -- , index_tuple_hdr_bm, nulldatawidth -- (DEBUG INFO)
   FROM (
     SELECT
       i.nspname, i.tblname, i.idxname, i.reltuples, i.relpages, i.relam, a.attrelid
AS table_oid,
       current_setting('block_size')::numeric AS bs, fillfactor,
       CASE -- MAXALIGN: 4 on 32bits, 8 on 64bits (and mingw32 ?)
         WHEN version() ~ 'mingw32' OR version() ~ '64-bit|x86_64|ppc64|ia64|amd64'
THEN 8
         ELSE 4
       END AS maxalign,
       /* per page header, fixed size: 20 for 7.X, 24 for others */
       24 AS pagehdr,
       /* per page btree opaque data */
       16 AS pageopqdata,
       /* per tuple header: add IndexAttributeBitMapData if some cols are null-able */
       CASE WHEN max(coalesce(s.null_frac,0)) = 0
         THEN 2 -- IndexTupleData size
         ELSE 2 + (( 32 + 8 - 1 ) / 8)
         -- IndexTupleData size + IndexAttributeBitMapData size ( max num filed per
index + 8 - 1 /8)
       END AS index_tuple_hdr_bm,
       /* data len: we remove null values save space using it fractionnal part from
stats */
       sum( (1-coalesce(s.null_frac, 0)) * coalesce(s.avg_width, 1024)) AS
nulldatawidth,
       max( CASE WHEN a.atttypid = 'pg_catalog.name'::regtype THEN 1 ELSE 0 END ) > 0
AS is_na
     FROM pg_attribute AS a
       JOIN (
         SELECT nspname, tbl.relname AS tblname, idx.relname AS idxname,
           idx.reltuples, idx.relpages, idx.relam,
           indrelid, indexrelid, indkey::smallint[] AS attnum,
```

```
            coalesce(substring(
              array_to_string(idx.reloptions, ' ')
               from 'fillfactor=([0-9]+)')::smallint, 90) AS fillfactor
          FROM pg_index
            JOIN pg_class idx ON idx.oid=pg_index.indexrelid
            JOIN pg_class tbl ON tbl.oid=pg_index.indrelid
            JOIN pg_namespace ON pg_namespace.oid = idx.relnamespace
          WHERE pg_index.indisvalid AND tbl.relkind = 'r' AND idx.relpages > 0
        ) AS i ON a.attrelid = i.indexrelid
        JOIN pg_stats AS s ON s.schemaname = i.nspname
          AND ((s.tablename = i.tblname AND s.attname =
 pg_catalog.pg_get_indexdef(a.attrelid, a.attnum, TRUE))
          -- stats from tbl
          OR  (s.tablename = i.idxname AND s.attname = a.attname))
          -- stats from functionnal cols
        JOIN pg_type AS t ON a.atttypid = t.oid
      WHERE a.attnum > 0
      GROUP BY 1, 2, 3, 4, 5, 6, 7, 8, 9
    ) AS s1
  ) AS s2
    JOIN pg_am am ON s2.relam = am.oid WHERE am.amname = 'btree'
) AS sub
-- WHERE NOT is_na
ORDER BY 2,3,4;
```

Trova tabelle idonee per l'autovacuum

Per trovare tabelle idonee per l'autovacuum, esegui la query riportata.

```
--This query shows tables that need vacuuming and are eligible candidates.
--The following query lists all tables that are due to be processed by autovacuum.
-- During normal operation, this query should return very little.
WITH  vbt AS (SELECT setting AS autovacuum_vacuum_threshold
              FROM pg_settings WHERE name = 'autovacuum_vacuum_threshold')
    , vsf AS (SELECT setting AS autovacuum_vacuum_scale_factor
              FROM pg_settings WHERE name = 'autovacuum_vacuum_scale_factor')
    , fma AS (SELECT setting AS autovacuum_freeze_max_age
              FROM pg_settings WHERE name = 'autovacuum_freeze_max_age')
    , sto AS (SELECT opt_oid, split_part(setting, '=', 1) as param,
                split_part(setting, '=', 2) as value
              FROM (SELECT oid opt_oid, unnest(reloptions) setting FROM pg_class) opt)
SELECT
    '"'||ns.nspname||'"."'||c.relname||'"' as relation
    , pg_size_pretty(pg_table_size(c.oid)) as table_size
```

```
    , age(relfrozenxid) as xid_age
    , coalesce(cfma.value::float, autovacuum_freeze_max_age::float)
 autovacuum_freeze_max_age
    , (coalesce(cvbt.value::float, autovacuum_vacuum_threshold::float) +
         coalesce(cvsf.value::float,autovacuum_vacuum_scale_factor::float) *
 c.reltuples)
         as autovacuum_vacuum_tuples
    , n_dead_tup as dead_tuples
FROM pg_class c
JOIN pg_namespace ns ON ns.oid = c.relnamespace
JOIN pg_stat_all_tables stat ON stat.relid = c.oid
JOIN vbt on (1=1)
JOIN vsf ON (1=1)
JOIN fma on (1=1)
LEFT JOIN sto cvbt ON cvbt.param = 'autovacuum_vacuum_threshold' AND c.oid =
 cvbt.opt_oid
LEFT JOIN sto cvsf ON cvsf.param = 'autovacuum_vacuum_scale_factor' AND c.oid =
 cvsf.opt_oid
LEFT JOIN sto cfma ON cfma.param = 'autovacuum_freeze_max_age' AND c.oid = cfma.opt_oid
WHERE c.relkind = 'r'
AND nspname <> 'pg_catalog'
AND (
    age(relfrozenxid) >= coalesce(cfma.value::float, autovacuum_freeze_max_age::float)
    or
    coalesce(cvbt.value::float, autovacuum_vacuum_threshold::float) +
      coalesce(cvsf.value::float,autovacuum_vacuum_scale_factor::float) * c.reltuples
 <= n_dead_tup
    -- or 1 = 1
)
ORDER BY age(relfrozenxid) DESC;
```

Rispondere a un numero elevato di connessioni

Quando monitori Amazon CloudWatch, potresti rilevare picchi nel parametro `DatabaseConnections`. Questo aumento indica un numero maggiore di connessioni al database. Consigliamo quanto segue:

- Limita il numero di connessioni che l'applicazione può aprire con ciascuna istanza. Se l'applicazione dispone di una funzione di connection pool incorporata, impostare un numero ragionevole di connessioni. Basa il numero su ciò che le vCPU nell'istanza possono parallelizzare efficacemente.

Se l’applicazione non utilizza una funzione di connection pool, considera l'utilizzo di Amazon RDS Proxy o un'alternativa. Questo approccio consente all'applicazione di aprire più connessioni con il bilanciamento del carico. Il bilanciatore può quindi aprire un numero limitato di connessioni con il database. Poiché un numero inferiore di connessioni sono in esecuzione in parallelo, l'istanza DB esegue meno commutazione di contesto nel kernel. Le query dovrebbero progredire più velocemente, causando un minor numero di eventi di attesa. Per ulteriori informazioni, consulta Server proxy per Amazon RDS per Aurora.

- Quando possibile, approfitta dei nodi di lettore per Aurora PostgreSQL e leggi le repliche per RDS per PostgreSQL. Quando l'applicazione esegue un'operazione di sola lettura, inviare queste richieste all'endpoint di sola lettura. Questa tecnica diffonde le richieste delle applicazioni su tutti i nodi del lettore, riducendo la pressione I/O sul nodo di scrittura.
- Prendi in considerazione la possibilità di scalare l'istanza database. Una classe di istanza a maggiore capacità fornisce più memoria, il che offre ad Aurora PostgreSQL un buffer pool condiviso più ampio per contenere le pagine. Le dimensioni maggiori conferiscono inoltre all'istanza database più vCPU per gestire le connessioni. Più vCPU sono particolarmente utili quando le operazioni che stanno generando gli eventi di attesa `IO:DataFileRead` sono scritture.

IO:XactSync

L’evento `IO:XactSync` si verifica quando un processo di backend è in attesa che il sottosistema di storage Aurora riconosca il commit di una transazione regolare o il commit o il rollback di una transazione preparata. Una transazione preparata fa parte del supporto di PostgreSQL per un commit in due fasi. Questo evento può verificarsi anche quando una query è in attesa del commit di un’altra transazione, in particolare nei casi in cui il commit automatico è disattivato. In tali scenari, potrebbe sembrare che gli aggiornamenti siano in attesa di XactSync anche se non sono ancora stati eseguiti.

Argomenti

- Versioni del motore supportate
- Contesto
- Probabili cause di aumento delle attese
- Azioni

Versioni del motore supportate

Queste informazioni relative all'evento di attesa sono supportate per tutte le versioni di Aurora PostgreSQL.

Contesto

L'evento `IO:XactSync` indica che l'istanza sta dedicando tempo ad aspettare che il sottosistema di storage Aurora confermi che i dati delle transazioni sono stati elaborati.

Probabili cause di aumento delle attese

Quando l'evento `IO:XactSync` si verifica più del normale, probabilmente indicando un problema di prestazioni, le cause tipiche includono le seguenti.

Saturazione di rete

Il traffico tra i client e l'istanza database o il traffico verso il sottosistema di storage potrebbe essere troppo pesante per la larghezza di banda della rete.

Pressione della CPU

Un carico di lavoro pesante potrebbe impedire al daemon di archiviazione Aurora di ottenere un tempo sufficiente per la CPU.

Azioni

Consigliamo azioni diverse a seconda delle cause dell'evento di attesa.

Argomenti

- Monitoraggio delle risorse
- Ridimensionamento della CPU
- Aumentare la larghezza di banda di rete
- Riduzione del numero di commit

Monitoraggio delle risorse

Per determinare la causa dell'aumento degli eventi `IO:XactSync`, controlla i seguenti parametri:

- `WriteThroughput` e `CommitThroughput` — Le modifiche del throughput di scrittura o del throughput di commit possono mostrare un aumento del carico di lavoro.
- `WriteLatency` e `CommitLatency` — Le modifiche alla latenza di scrittura o alla latenza di commit possono mostrare che al sottosistema di storage viene chiesto di svolgere più lavoro.
- `CPUUtilization` — Se l'utilizzo della CPU dell'istanza è superiore al 90%, il daemon di archiviazione Aurora potrebbe non avere tempo sufficiente per la CPU. In questo caso, le prestazioni I/O diminuiscono.

Per ulteriori informazioni su questi parametri, consulta Parametri a livello di istanza per Amazon Aurora.

Ridimensionamento della CPU

Per risolvere i problemi di fame della CPU, prendi in considerazione la possibilità di passare a un tipo di istanza con maggiore capacità della CPU. Per informazioni sulla capacità della CPU per una classe di istanza database, consulta Specifiche hardware per le classi di istanza database per Aurora.

Aumentare la larghezza di banda di rete

Per determinare se l'istanza sta raggiungendo i limiti di larghezza di banda di rete, verificare la presenza degli altri eventi di attesa seguenti:

- `IO:DataFileRead`, `IO:BufferRead`, `IO:BufferWrite` e `IO:XactWrite` — Le query che utilizzano grandi quantità di I/O possono generare più di questi eventi di attesa.
- `Client:ClientRead` e `Client:ClientWrite` — Le query con grandi quantità di comunicazioni client possono generare più di questi eventi di attesa.

Se la larghezza di banda di rete è un problema, considera la possibilità di passare a un tipo di istanza con maggiore larghezza di banda di rete. Per informazioni sulle prestazioni di rete per una classe di istanza database, consulta Specifiche hardware per le classi di istanza database per Aurora.

Riduzione del numero di commit

Per ridurre il numero di commit, combinare le dichiarazioni in blocchi di transazione.

IPC:DamRecordTxAck

L'evento `IPC:DamRecordTxAck` si verifica quando Aurora PostgreSQL in una sessione che utilizza flussi di attività del database genera un evento di flusso di attività, quindi attende che tale evento diventi duraturo.

Argomenti

- Versioni di motori pertinenti
- Context
- Cause
- Azioni

Versioni di motori pertinenti

Queste informazioni sugli eventi di attesa sono rilevanti per tutte le versioni di Aurora PostgreSQL 10.7 e successive 10, 11.4 e versioni successive 11 e tutte le versioni 12 e 13.

Context

In modalità sincrona, la durata degli eventi del flusso di attività è favorita rispetto alle prestazioni del database. In attesa di una scrittura duratura dell'evento, la sessione blocca le altre attività del database, causando l'evento di attesa `IPC:DamRecordTxAck`.

Cause

La causa più comune per l'evento `IPC:DamRecordTxAck` che appare nelle prime attese è che la funzione Database Activity Streams (DAS) è un audit olistico. L'attività SQL superiore genera eventi di flusso di attività che devono essere registrati.

Azioni

Consigliamo azioni diverse a seconda delle cause dell'evento di attesa:

- Riduci il numero di istruzioni SQL o disattiva i flussi di attività del database. In questo modo, è possibile ridurre il numero di eventi che richiedono scritture durevoli.
- Passa alla modalità asincrona. Ciò aiuta a ridurre le contese sull'evento di attesa `IPC:DamRecordTxAck`.

Tuttavia, la funzione DAS non può garantire la durata di ogni evento in modalità asincrona.

Eventi di attesa IPC:parallel

I seguenti `IPC:parallel wait events` indicano che una sessione è in attesa della comunicazione tra processi relativa alle operazioni di esecuzione di query parallele.

- `IPC:BgWorkerStartup`: un processo è in attesa che un processo worker parallelo completi la sequenza di avvio. Ciò si verifica durante l'inizializzazione dei worker per l'esecuzione di query parallele.
- `IPC:BgWorkerShutdown`: un processo è in attesa che un processo worker parallelo completi la sequenza di arresto. Ciò si verifica durante la fase di pulizia dell'esecuzione di query parallele.
- `IPC:ExecuteGather`: un processo è in attesa di ricevere dati da processi worker paralleli durante l'esecuzione di query. Ciò si verifica quando il processo leader deve raccogliere i risultati dai propri worker.
- `IPC:ParallelFinish`: un processo è in attesa che i worker paralleli completino l'esecuzione e riportino i risultati finali. Ciò avviene durante la fase di completamento dell'esecuzione di query parallele.

Argomenti

- Versioni del motore supportate
- Contesto
- Probabili cause di aumento delle attese
- Azioni

Versioni del motore supportate

Queste informazioni relative all'evento di attesa sono supportate per tutte le versioni di Aurora PostgreSQL.

Contesto

L'esecuzione parallela delle query in PostgreSQL implica la collaborazione di più processi per elaborare una singola query. Quando si determina che una query è adatta alla parallelizzazione,

un processo leader si coordina con uno o più processi worker paralleli in base all'impostazione del parametro `max_parallel_workers_per_gather`. Il processo leader divide il lavoro tra i worker, ciascuno dei quali elabora la propria porzione di dati in modo indipendente, e i risultati vengono raccolti dal processo leader.

> **Note**
>
> Ogni worker parallelo opera come un processo separato con requisiti di risorse simili a quelli di una sessione utente completa. Ciò significa che una query parallela con 4 worker può consumare fino a 5 volte le risorse (CPU, memoria, larghezza di banda I/O) rispetto a una query non parallela, poiché sia il processo leader che ogni processo worker mantengono le proprie allocazioni di risorse. Ad esempio, impostazioni come `work_mem` vengono applicate singolarmente a ciascun worker, moltiplicando potenzialmente l'utilizzo totale della memoria in tutti i processi.

L'architettura di query parallela è composta da tre componenti principali:

- Processo leader: il processo principale che avvia l'operazione parallela, divide il carico di lavoro e si coordina con i processi worker.
- Processi worker: processi in background che eseguono parti della query in parallelo.
- Gather/Gather merge: operazioni che combinano i risultati di più processi worker e li restituiscono al leader

Durante l'esecuzione parallela, i processi devono comunicare tra loro tramite meccanismi di comunicazione tra processi (IPC). Questi eventi di attesa IPC si verificano durante diverse fasi:

- Avvio del worker: quando i worker paralleli vengono inizializzati
- Scambio di dati: quando i worker elaborano i dati e inviano i risultati al leader
- Arresto del worker: quando l'esecuzione parallela viene completata e i worker vengono terminati
- Punti di sincronizzazione: quando i processi devono coordinarsi o attendere che altri processi completino le proprie attività

La comprensione di questi eventi di attesa è fondamentale per diagnosticare i problemi di prestazioni relativi all'esecuzione di query parallele, specialmente in ambienti ad alta concorrenza in cui più query parallele possono essere eseguite contemporaneamente.

Probabili cause di aumento delle attese

Diversi fattori possono contribuire a un aumento degli eventi di attesa IPC correlati alle query parallele:

Elevata concorrenza di query parallele

L'esecuzione simultanea di più query parallele può portare a un conflitto di risorse e a un aumento dei tempi di attesa per le operazioni IPC. Ciò è particolarmente comune nei sistemi con elevati volumi di transazioni o carichi di lavoro analitici.

Piani di query parallele non ottimali

Se il pianificatore di query sceglie piani paralleli inefficienti, ciò può comportare una parallelizzazione non necessaria o una scarsa distribuzione del lavoro tra i worker. Ciò può portare a un aumento delle attese IPC, in particolare per gli eventi IPC:ExecuteGather e IPC:ParallelFinish. Questi problemi di pianificazione spesso derivano da statistiche obsolete e dal sovraccarico di tabelle/indici.

Avvio e arresto frequenti dei worker paralleli

Le query di breve durata che avviano e terminano frequentemente i worker paralleli possono causare un aumento degli eventi `IPC:BgWorkerStartup` e `IPC:BgWorkerShutdown`. Ciò si verifica spesso nei carichi di lavoro OLTP con molte query di piccole dimensioni e parallelizzabili.

Vincoli delle risorse

Una capacità limitata di CPU, memoria o I/O può causare colli di bottiglia nell'esecuzione parallela, con conseguente aumento dei tempi di attesa in tutti gli eventi IPC. Ad esempio, se la CPU è satura, i processi worker potrebbero impiegare più tempo per avviarsi o elaborare la relativa parte di lavoro.

Strutture di query complesse

Le query con più livelli di parallelismo (ad esempio, join paralleli seguiti da aggregazioni parallele) possono portare a modelli IPC più complessi e tempi di attesa potenzialmente maggiori, soprattutto per gli eventi `IPC:ExecuteGather`.

Serie di risultati di grandi dimensioni

Le query che producono set di risultati di grandi dimensioni possono aumentare i tempi di attesa `IPC:ExecuteGather` poiché il processo leader dedica più tempo alla raccolta e all'elaborazione dei risultati dei processi worker.

La comprensione di questi fattori può aiutare a diagnosticare e risolvere i problemi di prestazioni relativi all'esecuzione di query parallele in Aurora PostgreSQL.

Azioni

Quando si verificano attese relative a query parallele, in genere significa che un processo di backend sta coordinando o attendendo processi worker paralleli. Queste attese sono comuni durante l'esecuzione di piani paralleli. È possibile analizzare e mitigare l'impatto di queste attese monitorando l'utilizzo dei worker paralleli, esaminando le impostazioni dei parametri e ottimizzando l'esecuzione delle query e l'allocazione delle risorse.

Argomenti

- Analizzare i piani di query per individuare eventuali parallelismi inefficienti
- Monitorare l'utilizzo delle query parallele
- Esaminare e modificare le impostazioni delle interrogazioni parallele
- Ottimizzare l'allocazione delle risorse
- Esaminare la gestione delle connessioni
- Esaminare e ottimizzare le operazioni di manutenzione
- Utilizzare Query Plan Management (QPM)

Analizzare i piani di query per individuare eventuali parallelismi inefficienti

L'esecuzione parallela delle query può spesso portare a instabilità del sistema, picchi di CPU e variazioni imprevedibili delle prestazioni delle query. È fondamentale analizzare a fondo se il parallelismo migliora effettivamente il carico di lavoro specifico. Utilizzare EXPLAIN ANALYZE per esaminare i piani di esecuzione delle query parallele.

Disattivare temporaneamente il parallelismo a livello di sessione per confrontare l'efficienza del piano:

```
SET max_parallel_workers_per_gather = 0;
EXPLAIN ANALYZE <your_query>;
```

Riattivare il parallelismo e confrontare:

```
RESET max_parallel_workers_per_gather;
```

```
EXPLAIN ANALYZE <your_query>;
```

Se la disabilitazione del parallelismo produce risultati migliori o più coerenti, prendere in considerazione la possibilità di disabilitarlo per query specifiche a livello di sessione utilizzando i comandi SET. Per un impatto più ampio, è possibile voler disabilitare il parallelismo a livello di istanza regolando i parametri pertinenti nel cluster o nel gruppo di parametri dell'istanza. Per ulteriori informazioni, consulta Amazon Aurora PostgreSQL parametri.

Monitorare l'utilizzo delle query parallele

Utilizzare le seguenti query per ottenere visibilità sull'attività e sulla capacità delle query parallele:

Controllare i processi worker paralleli attivi:

```
SELECT
    COUNT(*)
FROM
    pg_stat_activity
WHERE
    backend_type = 'parallel worker';
```

Questa query mostra il numero di processi worker paralleli attivi. Un valore elevato può indicare che `max_parallel_workers` è configurato con un valore elevato e potrebbe essere opportuno ridurlo.

Controllare le query parallele simultanee:

```
SELECT
    COUNT(DISTINCT leader_pid)
FROM
    pg_stat_activity
WHERE
    leader_pid IS NOT NULL;
```

Questa query restituisce il numero di processi leader distinti che hanno avviato query parallele. Un numero elevato indica che più sessioni eseguono query parallele contemporaneamente, il che può aumentare la richiesta di CPU e memoria.

Esaminare e modificare le impostazioni delle interrogazioni parallele

Controllare i seguenti parametri per assicurarsi che siano in linea con il carico di lavoro:

- `max_parallel_workers`: numero totale di worker paralleli in tutte le sessioni.
- `max_parallel_workers_per_gather`: numero massimo di worker per query.

Per i carichi di lavoro OLAP, l'aumento di questi valori può migliorare le prestazioni. Per i carichi di lavoro OLTP, in genere sono preferiti valori inferiori.

```
SHOW max_parallel_workers;
SHOW max_parallel_workers_per_gather;
```

Ottimizzare l'allocazione delle risorse

Monitorare l'utilizzo della CPU e valutare la possibilità di modificare il numero di vCPU se è costantemente elevato e se l'applicazione trae vantaggio dalle query parallele. Assicurarsi che sia disponibile una memoria adeguata per le operazioni parallele.

- Utilizzare le metriche di Approfondimenti sulle prestazioni per determinare se il sistema è legato alla CPU.
- Ogni worker parallelo utilizza il proprio `work_mem`. Assicurarsi che l'utilizzo totale della memoria rientri nei limiti delle istanze.

Le query parallele possono consumare molte più risorse rispetto alle query non parallele, poiché ogni processo worker è un processo completamente separato che ha all'incirca lo stesso impatto sul sistema di una sessione utente aggiuntiva. Questo deve essere tenuto in considerazione quando si sceglie un valore per questa impostazione, nonché quando si configurano altre impostazioni che controllano l'utilizzo delle risorse come `work_mem`. Per ulteriori informazioni, consultare la documentazione di PostgreSQL. I limiti delle risorse come `work_mem` vengono applicati individualmente a ciascun worker, il che significa che l'utilizzo totale può essere molto più elevato in tutti i processi rispetto a quanto sarebbe normalmente per ogni singolo processo.

Prendere in considerazione l'aumento delle vCPU o l'ottimizzazione dei parametri di memoria se il carico di lavoro è fortemente parallelizzato.

Esaminare la gestione delle connessioni

In caso di esaurimento delle connessioni, esaminare le strategie di pooling delle connessioni dell'applicazione. Prendere in considerazione l'implementazione del pooling delle connessioni a livello di applicazione, se non è già in uso.

Esaminare e ottimizzare le operazioni di manutenzione

Coordinare la creazione degli indici e le altre attività di manutenzione per prevenire il conflitto di risorse. Prendere in considerazione la possibilità di pianificare queste operazioni durante le ore non di picco. Evitare di pianificare una manutenzione intensa (ad esempio, build di indici paralleli) durante i periodi di elevato carico di query da parte degli utenti. Queste operazioni possono consumare worker paralleli e influire sulle prestazioni delle query regolari.

Utilizzare Query Plan Management (QPM)

In Aurora PostgreSQL, la funzionalità Query Plan Management (QPM) è progettata per garantire l'adattabilità e la stabilità del piano indipendentemente dalle modifiche apportate all'ambiente di database che potrebbero causare la regressione del piano di query. Per maggiori informazioni, consulta Panoramica della gestione del piano di query per Aurora PostgreSQL. QPM fornisce un certo controllo sull'ottimizzatore. Esaminare i piani approvati in QPM per assicurarsi che siano in linea con le attuali impostazioni di parallelismo. Aggiornare o rimuovere i piani obsoleti che potrebbero forzare un'esecuzione parallela non ottimale.

È anche possibile correggere i piani utilizzando pg_hint_plan. Per ulteriori informazioni, consulta Correzione dei piani mediante pg_hint_plan. È possibile utilizzare l'hint denominato `Parallel` per imporre l'esecuzione parallela. Per maggiori informazioni, consulta la pagina relativa agli hint per i piani paralleli.

IPC: ProcArrayGroupUpdate

L'`IPC:ProcArrayGroupUpdate`evento si verifica quando una sessione è in attesa che il capogruppo aggiorni lo stato della transazione al termine dell'operazione. Sebbene PostgreSQL generalmente associ eventi di attesa di tipo IPC a operazioni di query parallele, questo particolare evento di attesa non è specifico delle query parallele.

Argomenti

- Versioni del motore supportate
- Contesto

- Probabili cause di aumento delle attese
- Azioni

Versioni del motore supportate

Queste informazioni relative all'evento di attesa sono supportate per tutte le versioni di Aurora PostgreSQL.

Contesto

Comprendere l'array di processi — L'array process (proc) è una struttura di memoria condivisa in PostgreSQL. Contiene informazioni su tutti i processi in esecuzione, inclusi i dettagli delle transazioni. Durante il completamento della transazione (`COMMIToROLLBACK`), ProcArray deve essere aggiornato per riflettere la modifica e cancellare il TransactionID dall'array. La sessione che tenta di completare la transazione deve acquisire un blocco esclusivo su. ProcArray Ciò impedisce ad altri processi di ottenere blocchi condivisi o esclusivi su di esso.

Meccanismo di aggiornamento di gruppo: durante l'esecuzione di un COMMIT o ROLLBACK, se un processo di backend non riesce a ottenerne uno ProcArrayLock in modalità esclusiva, aggiorna un campo speciale chiamato. ProcArrayGroupMember Ciò aggiunge la transazione all'elenco delle sessioni che intendono terminare. Questo processo di backend quindi dorme e il tempo in cui dorme viene utilizzato come evento di attesa. ProcArrayGroupUpdate Il primo processo ProcArray con procArrayGroup Member, denominato processo leader, acquisisce la modalità esclusiva. ProcArrayLock Quindi cancella l'elenco dei processi in attesa della cancellazione del gruppo TransactionID. Una volta completata questa operazione, il leader rilascia ProcArrayLock e quindi riattiva tutti i processi in questo elenco, notificando loro che la transazione è stata completata.

Probabili cause di aumento delle attese

Maggiore è il numero di processi in esecuzione, più a lungo un leader rimarrà aggrappato procArrayLock a una modalità esclusiva. Di conseguenza, più transazioni di scrittura finiscono in uno scenario di aggiornamento di gruppo, causando un potenziale accumulo di processi in attesa dell'evento di `ProcArrayGroupUpdate` attesa. Nella visualizzazione SQL principale di Database Insights, vedrete che COMMIT è l'istruzione con la maggior parte di questo evento di attesa. Ciò è previsto, ma richiederà un'analisi più approfondita dello specifico SQL di scrittura in esecuzione per determinare l'azione appropriata da intraprendere.

Azioni

Consigliamo azioni diverse a seconda delle cause dell'evento di attesa. Identifica `IPC:ProcArrayGroupUpdate` gli eventi utilizzando Amazon RDS Performance Insights o interrogando la vista del sistema PostgreSQL. `pg_stat_activity`

Argomenti

- Monitoraggio delle operazioni di conferma e rollback delle transazioni
- Ridurre la concorrenza
- Implementazione del pool di connessioni

Monitoraggio delle operazioni di conferma e rollback delle transazioni

Monitora i commit e i rollback: un numero maggiore di commit e rollback può comportare una maggiore pressione su. ProcArray Ad esempio, se un'istruzione SQL inizia a fallire a causa di un aumento delle violazioni di chiavi duplicate, è possibile che si verifichi un aumento dei rollback che può aumentare la contesa e il sovraccarico delle tabelle. ProcArray

Amazon RDS Database Insights fornisce i `xact_commit` parametri di PostgreSQL e riporta il numero di commit `xact_rollback` e rollback al secondo.

Ridurre la concorrenza

Transazioni in batch: ove possibile, operazioni in batch in singole transazioni per ridurre commit/rollback le operazioni.

Limita la concorrenza: riduci il numero di transazioni attive contemporaneamente per alleviare la contesa tra blocchi su. ProcArray Sebbene richiederà alcuni test, la riduzione del numero totale di connessioni simultanee può ridurre i conflitti e mantenere la velocità effettiva.

Implementazione del pool di connessioni

Soluzioni di pool di connessioni: utilizza il pool di connessioni per gestire in modo efficiente le connessioni al database, riducendo il numero totale di backend e quindi il carico di lavoro sul. ProcArray Sebbene richieda alcuni test, la riduzione del numero totale di connessioni simultanee può ridurre i conflitti e mantenere la velocità effettiva.

Per ulteriori informazioni, consulta Connection pooling per Aurora PostgreSQL.

Riduci le tempeste di connessione: allo stesso modo, uno schema di creazione e interruzione frequenti delle connessioni causa una pressione aggiuntiva su. ProcArray Riducendo questo modello, si riduce la contesa complessiva.

Lock:advisory

L'evento `Lock:advisory` si verifica quando un'applicazione PostgreSQL utilizza un blocco per coordinare l'attività su più sessioni.

Argomenti

- Versioni di motori pertinenti
- Context
- Cause
- Azioni

Versioni di motori pertinenti

Queste informazioni relative all'evento di attesa sono supportate per Aurora PostgreSQL versione 9.6 e successive.

Context

I blocchi di consulenza PostgreSQL sono blocchi cooperativi a livello di applicazione esplicitamente bloccati e sbloccati dal codice dell'applicazione dell'utente. Un'applicazione PostgreSQL può utilizzare un blocco per coordinare l'attività su più sessioni. A differenza dei normali blocchi a livello di oggetto o riga, l'applicazione ha il pieno controllo sulla durata del blocco. Per ulteriori informazioni consulta Blocchi di consulenza nella documentazione di PostgreSQL.

I blocchi di consulenza possono essere rilasciati prima della fine di una transazione o essere trattenuti da una sessione tra le transazioni. Ciò tuttavia non è vero per i blocchi impliciti e applicati al sistema, come un blocco esclusivo di accesso su una tabella acquisita da una dichiarazione `CREATE INDEX`.

Per una descrizione delle funzioni utilizzate per acquisire (bloccare) e rilasciare (sbloccare) i blocchi di consulenza, vedere Funzioni di Advisory Lock nella documentazione di PostgreSQL.

I blocchi di consulenza sono implementati sopra il normale sistema di blocco PostgreSQL e sono visibili nella visualizzazione di sistema `pg_locks`.

Cause

Questo tipo di blocco è controllato esclusivamente da un'applicazione che lo utilizza esplicitamente. I blocchi di consulenza acquisiti per ogni riga come parte di una query possono causare un picco di blocchi o un accumulo a lungo termine.

Questi effetti si verificano quando la query viene eseguita in un modo che acquisisce blocchi su più righe di quelle restituite dalla query. L'applicazione dovrà comunque rilasciare ogni blocco, ma se i blocchi vengono acquisiti su righe che non vengono restituite, l'applicazione non riesce a trovare tutti i blocchi.

L'esempio seguente è tratto da Blocchi di consulenza nella documentazione di PostgreSQL.

```
SELECT pg_advisory_lock(id) FROM foo WHERE id > 12345 LIMIT 100;
```

In questo esempio, la clausola `LIMIT` può arrestare l'output della query solo dopo che le righe sono già state selezionate internamente e i relativi valori ID bloccati. Ciò può accadere improvvisamente quando un volume di dati crescente fa sì che il pianificatore scelga un piano di esecuzione diverso che non è stato testato durante lo sviluppo. L'accumulo in questo caso avviene perché l'applicazione chiama esplicitamente `pg_advisory_unlock` per ogni valore ID bloccato. Tuttavia, in questo caso non è possibile trovare il set di blocchi acquisiti su righe che non sono state restituite. Poiché i blocchi vengono acquisiti a livello di sessione, non vengono rilasciati automaticamente alla fine della transazione.

Un'altra possibile causa di picchi nei tentativi di blocco bloccati sono i conflitti non intenzionali. In questi conflitti, parti non correlate dell'applicazione condividono per errore lo stesso spazio ID di blocco.

Azioni

Esaminare l'utilizzo delle applicazioni dei blocchi di consulenza e i dettagli su dove e quando nel flusso dell'applicazione viene acquisito e rilasciato ogni tipo di blocco consultivo.

Determina se una sessione sta acquisendo troppi blocchi o che una sessione di lunga durata non rilascia blocchi abbastanza presto, causando un lento accumulo di blocchi. È possibile correggere un lento accumulo di blocchi a livello di sessione terminando la sessione utilizzando `pg_terminate_backend(pid)`.

Viene visualizzato un client in attesa di un blocco di avviso in `pg_stat_activity` con `wait_event_type=Lock` e `wait_event=advisory`. È possibile ottenere valori di blocco

specifici eseguendo una query nella vista di sistema `pg_locks` per lo stesso `pid`, cercando `locktype=advisory` e `granted=f`.

È quindi possibile identificare la sessione di blocco interrogando `pg_locks` per lo stesso blocco consultivo `granted=t`, come mostrato nell'esempio seguente.

```
SELECT blocked_locks.pid AS blocked_pid,
         blocking_locks.pid AS blocking_pid,
         blocked_activity.usename AS blocked_user,
         blocking_activity.usename AS blocking_user,
         now() - blocked_activity.xact_start AS blocked_transaction_duration,
         now() - blocking_activity.xact_start AS blocking_transaction_duration,
         concat(blocked_activity.wait_event_type,':',blocked_activity.wait_event) AS
 blocked_wait_event,
         concat(blocking_activity.wait_event_type,':',blocking_activity.wait_event) AS
 blocking_wait_event,
         blocked_activity.state AS blocked_state,
         blocking_activity.state AS blocking_state,
         blocked_locks.locktype AS blocked_locktype,
         blocking_locks.locktype AS blocking_locktype,
         blocked_activity.query AS blocked_statement,
         blocking_activity.query AS blocking_statement
    FROM pg_catalog.pg_locks blocked_locks
    JOIN pg_catalog.pg_stat_activity blocked_activity ON blocked_activity.pid =
 blocked_locks.pid
    JOIN pg_catalog.pg_locks blocking_locks
        ON blocking_locks.locktype = blocked_locks.locktype
        AND blocking_locks.DATABASE IS NOT DISTINCT FROM blocked_locks.DATABASE
        AND blocking_locks.relation IS NOT DISTINCT FROM blocked_locks.relation
        AND blocking_locks.page IS NOT DISTINCT FROM blocked_locks.page
        AND blocking_locks.tuple IS NOT DISTINCT FROM blocked_locks.tuple
        AND blocking_locks.virtualxid IS NOT DISTINCT FROM blocked_locks.virtualxid
        AND blocking_locks.transactionid IS NOT DISTINCT FROM
 blocked_locks.transactionid
        AND blocking_locks.classid IS NOT DISTINCT FROM blocked_locks.classid
        AND blocking_locks.objid IS NOT DISTINCT FROM blocked_locks.objid
        AND blocking_locks.objsubid IS NOT DISTINCT FROM blocked_locks.objsubid
        AND blocking_locks.pid != blocked_locks.pid
    JOIN pg_catalog.pg_stat_activity blocking_activity ON blocking_activity.pid =
 blocking_locks.pid
    WHERE NOT blocked_locks.GRANTED;
```

Tutte le funzioni API di blocco consultivo hanno due serie di argomenti, un argomento `bigint` o due argomenti `integer`:

- Per le funzioni API con un argomento `bigint`, i 32 bit superiori sono in `pg_locks.classid` e i 32 bit inferiori sono in `pg_locks.objid`.
- Per le funzioni API con due argomenti `integer`, il primo argomento è `pg_locks.classid` e il secondo argomento è `pg_locks.objid`.

Il valore `pg_locks.objsubid` indica quale modulo API è stato utilizzato: 1 significa un argomento `bigint`; 2 significa due argomenti `integer`.

Lock:extend

L'evento `Lock:extend` si verifica quando un processo di back-end è in attesa di bloccare una relazione per estenderla mentre un altro processo ha un blocco su tale relazione per lo stesso scopo.

Argomenti

- Versioni del motore supportate
- Context
- Probabili cause di aumento delle attese
- Azioni

Versioni del motore supportate

Queste informazioni relative all'evento di attesa sono supportate per tutte el versioni di Aurora PostgreSQL.

Context

L'evento `Lock:extend` indica che un processo di back-end è in attesa di estendere una relazione su cui un altro processo di backend mantiene un blocco mentre sta estendendo tale relazione. Poiché solo un processo alla volta può estendere una relazione, il sistema genera un evento di attesa `Lock:extend`. Le operazioni `INSERT`, `COPY`, e `UPDATE` possono generare questo evento.

Probabili cause di aumento delle attese

Quando l'evento `Lock:extend` si verifica più del normale, probabilmente indicando un problema di prestazioni, le cause tipiche includono le seguenti.

Aumento degli inserti simultanei o degli aggiornamenti della stessa tabella

Potrebbe esserci un aumento del numero di sessioni simultanee con query che inseriscono o aggiornano la stessa tabella.

Larghezza di banda di rete insufficiente

La larghezza di banda di rete sull'istanza database potrebbe essere insufficiente per le esigenze di comunicazione di storage del carico di lavoro corrente. Ciò può contribuire alla latenza dello storage che causa un aumento degli eventi `Lock:extend`.

Azioni

Consigliamo azioni diverse a seconda delle cause dell'evento di attesa.

Argomenti

- Riduci gli inserti e gli aggiornamenti simultanei alla stessa relazione
- Aumentare la larghezza di banda di rete

Riduci gli inserti e gli aggiornamenti simultanei alla stessa relazione

Innanzitutto, determinare se c'è un aumento dei parametri `tup_inserted` e `tup_updated` e un aumento di questi eventi di attesa. In tal caso, verificare quali relazioni sono in forte contesa per le operazioni di inserimento e aggiornamento. Per determinarlo, interrogare la vista `pg_stat_all_tables` per i valori nei campi `n_tup_ins` e `n_tup_upd`. Per ulteriori informazioni sulla vista `pg_stat_all_tables`, consultare pg_stat_all_tables nella documentazione PostgreSQL.

Per ottenere ulteriori informazioni sul blocco e le query bloccate, eseguire una query `pg_stat_activity` come nel seguente esempio:

```
SELECT
    blocked.pid,
    blocked.usename,
    blocked.query,
    blocking.pid AS blocking_id,
    blocking.query AS blocking_query,
    blocking.wait_event AS blocking_wait_event,
    blocking.wait_event_type AS blocking_wait_event_type
FROM pg_stat_activity AS blocked
```

```
JOIN pg_stat_activity AS blocking ON blocking.pid = ANY(pg_blocking_pids(blocked.pid))
where
blocked.wait_event = 'extend'
and blocked.wait_event_type = 'Lock';

   pid  | usename  |            query             | blocking_id |
            blocking_query                          | blocking_wait_event |
 blocking_wait_event_type
  ------+----------+------------------------------+-------------
+-------------------------------------------------------------------
+--------------------+--------------------------
   7143 |  myuser  | insert into tab1 values (1); |        4600 | INSERT INTO tab1 (a)
 SELECT s FROM generate_series(1,1000000) s; | DataFileExtend      | IO
```

Dopo aver identificato le relazioni che contribuiscono ad aumentare gli eventi `Lock:extend`, utilizza le seguenti tecniche per ridurre la contesa:

- Scopri se è possibile utilizzare il partizionamento per ridurre le contese per la stessa tabella. La separazione delle tuple inserite o aggiornate in diverse partizioni può ridurre le contese. Per informazioni sulle partizioni, consulta Gestione delle partizioni PostgreSQL con l'estensione pg_partman.
- Se l'evento di attesa è dovuto principalmente all'attività di aggiornamento, considerare di ridurre il valore del fattore di riempimento della relazione. Ciò può ridurre le richieste di nuovi blocchi durante l'aggiornamento. Il fattore di riempimento è un parametro di archiviazione per una tabella che determina la quantità massima di spazio per l'imballaggio di una pagina di tabella. Viene espresso come percentuale dello spazio totale per una pagina. Per ulteriori informazioni sul parametro fillfactor, consulta CREA TABELLA nella documentazione di PostgreSQL.

  > **Important**
  >
  > Si consiglia vivamente di testare il sistema se si modifica il fattore di riempimento perché la modifica di questo valore può influire negativamente sulle prestazioni, a seconda del carico di lavoro.

Aumentare la larghezza di banda di rete

Per vedere se c'è un aumento della latenza di scrittura, controlla il parametro `WriteLatency` in CloudWatch. Se è presente, usa i parametri Amazon CloudWatch `WriteThroughput` e `ReadThroughput` per monitorare il traffico relativo allo storage sul cluster DB. Questi parametri

possono aiutarti a determinare se la larghezza di banda della rete è sufficiente per il tuo carico di lavoro.

Se la larghezza di banda della rete non è sufficiente, aumentala. Se il file client o l'istanza DB sta raggiungendo i limiti di larghezza di banda di rete, l'unico modo per aumentare la larghezza di banda è aumentare la dimensione dell'istanza DB.

Per ulteriori informazioni sui parametri CloudWatch, consultare CloudWatch Parametri Amazon per Amazon Aurora. Per informazioni sulle prestazioni di rete per una classe di istanza database, consulta Specifiche hardware per le classi di istanza database per Aurora.

Lock:Relation

L'evento `Lock:Relation` si verifica quando una query è in attesa di acquisire un blocco su una tabella o vista (relazione) attualmente bloccata da un'altra transazione.

Argomenti

- Versioni del motore supportate
- Contesto
- Probabili cause di aumento delle attese
- Azioni

Versioni del motore supportate

Queste informazioni relative all'evento di attesa sono supportate per tutte le versioni di Aurora PostgreSQL.

Contesto

La maggior parte dei comandi PostgreSQL utilizza implicitamente i blocchi per controllare l'accesso simultaneo ai dati nelle tabelle. È inoltre possibile utilizzare questi blocchi esplicitamente nel codice dell'applicazione con il comando `LOCK`. Molte modalità di blocco non sono compatibili tra loro e possono bloccare le transazioni quando cercano di accedere allo stesso oggetto. Quando ciò accade, Aurora PostgreSQL genera un evento `Lock:Relation`. Di seguito sono riportati alcuni esempi comuni:

- Blocchi esclusivi come `ACCESS EXCLUSIVE` possono bloccare tutti gli accessi simultanei. Le operazioni DDL (Data Definition Language) come `DROP TABLE`, `TRUNCATE`, `VACUUM FULL`,

e CLUSTER acquisiscono implicitamente i blocchi ACCESS EXCLUSIVE. ACCESS EXCLUSIVE è anche la modalità di blocco predefinita per le istruzioni LOCK TABLE che non specificano esplicitamente una modalità.

- L'uso di CREATE INDEX (without CONCURRENT) su una tabella è in conflitto con le istruzioni DML (Data Manipulation Language) UPDATE, DELETE, e INSERT, che acquisiscono i blocchi ROW EXCLUSIVE.

Per ulteriori informazioni sui blocchi a livello di tabella e sulle modalità di blocco in conflitto, vedere Blocco esplicito nella documentazione di PostgreSQL.

Il blocco di query e transazioni in genere si sblocca in uno dei seguenti modi:

- Query di blocco: l'applicazione può annullare la query o l'utente può terminare il processo. Il motore può anche forzare la fine della query a causa del timeout dell'istruzione di una sessione o di un meccanismo di rilevamento del deadlock.
- Blocco della transazione: una transazione smette di bloccarsi quando esegue un'istruzione ROLLBACK o COMMIT. I rollback si verificano automaticamente anche quando le sessioni vengono disconnesse da un client o da problemi di rete o terminano. Le sessioni possono essere terminate quando il motore di database è spento, quando il sistema è fuori memoria e così via.

Probabili cause di aumento delle attese

Quando l'evento Lock:Relation si verifica più frequentemente del normale, può indicare un problema di prestazioni. Le cause tipiche sono:

Sessioni simultanee aumentate con blocchi di tabella in conflitto

Potrebbe esserci un aumento del numero di sessioni simultanee con query che inseriscono o aggiornano la stessa tabella.

Operazioni di manutenzione

Le operazioni di manutenzione Health come VACUUM e ANALYZE possono aumentare significativamente il numero di blocchi in conflitto. VACUUM FULL acquisisce un blocco ACCESS EXCLUSIVE, e ANALYZE acquisisce un blocco SHARE UPDATE EXCLUSIVE. Entrambi i tipi di blocchi possono causare un evento di attesa Lock:Relation. Le operazioni di manutenzione dei dati delle applicazioni, come l'aggiornamento di una vista materializzata, possono anche aumentare le query e le transazioni bloccate.

Blocchi sulle istanze del lettore

È possibile che si verifichi un conflitto tra i blocchi di relazione tenuti dallo scrittore e dai lettori. Attualmente solo i blocchi di relazione `ACCESS EXCLUSIVE` vengono replicati nelle istanze del lettore. Tuttavia, il blocco di relazione `ACCESS EXCLUSIVE` sarà in conflitto con qualsiasi blocco di relazione `ACCESS SHARE` tenuto dal lettore. Questo conflitto può causare un aumento degli eventi di attesa della relazione di blocco sul lettore.

Azioni

Consigliamo azioni diverse a seconda delle cause dell'evento di attesa.

Argomenti

- Riduci l'impatto del blocco delle istruzioni SQL
- Riduci al minimo l'effetto delle operazioni di manutenzione
- Verifica la presenza di blocchi del lettore

Riduci l'impatto del blocco delle istruzioni SQL

Per ridurre l'impatto del blocco delle istruzioni SQL, modificare il codice dell'applicazione laddove possibile. Di seguito sono riportate due tecniche comuni per ridurre i blocchi:

- Utilizzo dell'opzione `NOWAIT` — Alcuni comandi SQL, come le istruzioni `SELECT` e `LOCK`, supportano questa opzione. La direttiva `NOWAIT` annulla la query richiedente il blocco se il blocco non può essere acquisito immediatamente. Questa tecnica può aiutare a impedire che una sessione di blocco provochi un accumulo di sessioni bloccate dietro di essa.

  Ad esempio: si supponga che la transazione A sia in attesa di un blocco trattenuto dalla transazione B. Ora, se B richiede un blocco su una tabella bloccata dalla transazione C, la transazione A potrebbe essere bloccata fino al completamento della transazione C. Ma se la transazione B utilizza un `NOWAIT` quando richiede il blocco su C, può fallire rapidamente e garantire che la transazione A non debba attendere indefinitamente.
- Utilizza `SET lock_timeout` — Imposta un valore `lock_timeout` per limitare il tempo in cui un'istruzione SQL attende di acquisire un blocco su una relazione. Se il blocco non viene acquisito entro il timeout specificato, la transazione che richiede il blocco viene annullata. Impostare questo valore a livello di sessione.

Riduci al minimo l'effetto delle operazioni di manutenzione

Operazioni di manutenzione come `VACUUM` e `ANALYZE` sono importanti. Si consiglia di non spegnerli qualora vengano trovati eventi di attesa `Lock:Relation` relativi a queste operazioni di manutenzione. I seguenti approcci possono ridurre al minimo l'effetto di queste operazioni:

- Eseguire manualmente le operazioni di manutenzione durante le ore non di punta.
- Per ridurre le attese `Lock:Relation` causate da attività autovacuum, eseguire qualsiasi sintonizzazione automatica necessaria. Per informazioni sulla sintonizzazione autovacuum, fai riferimento a Funzionamento di PostgreSQL Autovacuum in Amazon RDS nella Guida per l'utente di Amazon RDS.

Verifica la presenza di blocchi del lettore

Puoi vedere come le sessioni simultanee su uno scrittore e sui lettori potrebbero tenere blocchi che si bloccano a vicenda. A questo scopo, puoi eseguire le query che restituiscono il tipo di blocco e la relazione. Nella tabella è possibile trovare una sequenza di query su due sessioni simultanee, una sessione di scrittura e una sessione di lettura.

Il processo di riproduzione attende la durata impostata per `max_standby_streaming_delay` prima di annullare la query dell'istanza di lettura. Come mostrato nell'esempio, il timeout di blocco di 100 ms è ben al di sotto del valore di default di `max_standby_streaming_delay` di 30 secondi. Il timeout di blocco si verifica prima che si verifichi un problema.

Evento di sequenza	Sessione	Comando o output

Imposta una variabile di ambiente denominata READER con il valore specificato e tenta di connetter si all'istanza database con questo endpoint.	Sessione lettore	Comando della CLI:
		`export READER=au rorapg2.1234567891 0.us-west-1.rds.am azonaws.com`
		`psql -h $READER`
		Output:
		`psql (15devel, server 10.14)`

Evento di sequenza	Sessione	Comando o output
		`Type "help" for help.`

Imposta una variabile di ambiente denominata WRITER e tenta di connetter si all'istanza database con questo endpoint.	Sessione di scrittura	Comando della CLI:
		`export WRITER=au rorapg1.1234567891 0.us-west-1.rds.am azonaws.com`
		`psql -h $WRITER`
		Output:
		`psql (15devel, server 10.14)`
		`Type "help" for help.`

La sessione di scrittura crea la tabella t1 sull'istanza di scrittura.	Sessione di scrittura	Query PostgreSQL:
		`postgres=> CREATE TABLE t1(b integer);`
		`CREATE TABLE`

Se non ci sono query in conflitto sullo scrittore, il blocco ACCESS EXCLUSIVE viene acquisito immediata mente sull'istanza di scrittura.	Sessione di scrittura	Blocco `ACCESS EXCLUSIVE` abilitato

La sessione del lettore imposta un intervallo di timeout di blocco di 100 millisecondi.	Sessione lettore	Query PostgreSQL:
		`postgres=> SET lock_timeout=100;`
		`SET`

Evento di sequenza	Sessione	Comando o output

La sessione dell'istanza di lettura tenta di leggere i dati dalla tabella t1 sull'istanza di lettura.	Sessione lettore	Query PostgreSQL:
		`postgres=> SELECT * FROM t1;`
		Output di esempio:
		`b ---(0 rows)`

La sessione di scrittura elimina t1.	Sessione di scrittura	Query PostgreSQL: `postgres=> BEGIN;BEGINpostgres=> DROP TABLE t1;DROP TABLEpostgres=>`

La query scade e viene annullata sul lettore.	Sessione lettore	Query PostgreSQL:
		`postgres=> SELECT * FROM t1;`
		Output di esempio:
		`ERROR: canceling statement due to lock timeoutLINE 1: SELECT * FROM t1; ^`

Evento di sequenza	Sessione	Comando o output
Per determinare la causa dell'errore, la sessione dell'istanza di lettura interroga `pg_locks` e `pg_stat_activity` .	Sessione lettore	Query PostgreSQL: `postgres=> SELECT locktype, relation, mode, backend_type postgres=> FROM pg_locks l, pg_stat_activity t1 postgres=> WHERE l.pid=t1.pid AND relation = 't1'::regclass::oid;`
Il risultato indica che il processo `aurora wal replay` è in possesso di un blocco `ACCESS EXCLUSIVE` sulla tabella t1.	Sessione lettore	Risultato della query: `locktype | relation | mode | backend_type ----------+---------- +------------------ --+--------------- ---- relation | 68628525 | AccessExclusiveLock | aurora wal replay (1 row)`

Lock:transactionid

L'evento `Lock:transactionid` si verifica quando una transazione è in attesa di un blocco a livello di riga.

Argomenti

- Versioni del motore supportate
- Context
- Probabili cause di aumento delle attese

- Azioni

Versioni del motore supportate

Queste informazioni relative all'evento di attesa sono supportate per tutte el versioni di Aurora PostgreSQL.

Context

L'evento `Lock:transactionid` si verifica quando una transazione sta tentando di acquisire un blocco a livello di riga già concesso a una transazione in esecuzione contemporaneamente. La sessione che mostra il l'evento di attesa `Lock:transactionid` è bloccato a causa di questo blocco. Dopo che la transazione di blocco termina in un'istruzione `COMMIT` o `ROLLBACK`, la transazione bloccata può procedere.

La semantica multiversione di controllo della concorrenza di Aurora PostgreSQL garantisce che i lettori non bloccano gli scrittori e gli scrittori non bloccano i lettori. Affinché si verifichino conflitti a livello di riga, le transazioni bloccate e bloccate devono emettere dichiarazioni in conflitto dei seguenti tipi:

- `UPDATE`
- `SELECT … FOR UPDATE`
- `SELECT … FOR KEY SHARE`

L'istruzione `SELECT … FOR KEY SHARE` è un caso speciale. Il database utilizza la clausola `FOR KEY SHARE` per ottimizzare le prestazioni dell'integrità referenziale. Un blocco a livello di riga su una fila può bloccare i comandi `INSERT`, `UPDATE`, e `DELETE` su altre tabelle che fanno riferimento alla riga.

Probabili cause di aumento delle attese

Quando questo evento appare più del normale, la causa è in genere un'istruzione `UPDATE`, `SELECT … FOR UPDATE`, oppure `SELECT … FOR KEY SHARE` combinate con le seguenti condizioni.

Argomenti

- Elevata concorrenza
- Inattivo in transazione

- Transazioni di lunga durata

Elevata concorrenza

Aurora PostgreSQL può utilizzare la semantica di blocco granulare a livello di riga. La probabilità di conflitti a livello di riga aumenta quando vengono soddisfatte le seguenti condizioni:

- Un carico di lavoro altamente simultaneo è conteso per le stesse righe.
- Aumenta la concorrenza.

Inattivo in transazione

A volte la colonna `pg_stat_activity.state` mostra il valore `idle in transaction`. Questo valore viene visualizzato per le sessioni che hanno avviato una transazione, ma non hanno ancora emesso un `COMMIT` o `ROLLBACK`. Se il valore `pg_stat_activity.state` non è `active`, la query mostrata in `pg_stat_activity` è la versione più recente a terminare l'esecuzione. La sessione di blocco non sta elaborando attivamente una query perché una transazione aperta contiene un blocco.

Se una transazione inattiva ha acquisito un blocco a livello di riga, potrebbe impedire ad altre sessioni di acquisirlo. Questa condizione porta al frequente verificarsi dell'evento di attesa `Lock:transactionid`. Per diagnosticare il problema, esaminare l'output da `pg_stat_activity` e `pg_locks`.

Transazioni di lunga durata

Le transazioni che vengono eseguite a lungo ricevono blocchi per un lungo periodo di tempo. Questi blocchi a tenuta lunga possono impedire l'esecuzione di altre transazioni.

Azioni

Il blocco delle righe è un conflitto tra le istruzioni `UPDATE`, `SELECT … FOR UPDATE`, oppure `SELECT … FOR KEY SHARE`. Prima di tentare una soluzione, scopri quando queste istruzioni sono in esecuzione sulla stessa riga. Utilizzare queste informazioni per scegliere una strategia descritta nelle sezioni seguenti.

Argomenti

- Rispondere a un'elevata concorrenza
- Rispondere alle transazioni inattive
- Rispondere alle transazioni di lunga durata

Rispondere a un'elevata concorrenza

Se il problema è la concorrenza, prova una delle seguenti tecniche:

- Riduci la concorrenza nell'applicazione. Ad esempio, diminuisci il numero di sessioni attive.
- Implementa un pool di connessioni. Per informazioni su come mettere in pool le connessioni con RDS Proxy, vedere Server proxy per Amazon RDS per Aurora.
- Progettare l'applicazione o il modello di dati per evitare di contendere le istruzioni `UPDATE` e `SELECT … FOR UPDATE`. È inoltre possibile ridurre il numero di chiavi esterne a cui accedono le istruzioni `SELECT … FOR KEY SHARE`.

Rispondere alle transazioni inattive

Se `pg_stat_activity.state` mostra `idle in transaction`, utilizza le seguenti strategie:

- Attiva autocommit laddove possibile. Questo approccio impedisce alle transazioni di bloccare altre transazioni durante l'attesa di un `COMMIT` o `ROLLBACK`.
- Cerca percorsi di codice a cui manca `COMMIT`, `ROLLBACK`, oppure `END`.
- Assicurati che la logica di gestione delle eccezioni nell'applicazione abbia sempre un percorso per un `end of transaction` valido.
- Assicurati che l'applicazione elabori i risultati delle query dopo aver terminato la transazione con `COMMIT` o `ROLLBACK`.

Rispondere alle transazioni di lunga durata

Se le transazioni di lunga durata causano il frequente verificarsi di `Lock:transactionid`, prova le seguenti strategie:

- Tieni i blocchi di riga fuori dalle transazioni di lunga durata.
- Limita la durata delle query implementando autocommit quando possibile.

Lock:tuple

L'evento `Lock:tuple` si verifica quando un processo di backend aspetta di acquisire un blocco su una tupla.

Argomenti

- Versioni del motore supportate
- Context
- Probabili cause di aumento delle attese
- Azioni

Versioni del motore supportate

Queste informazioni relative all'evento di attesa sono supportate per tutte el versioni di Aurora PostgreSQL.

Context

L'evento `Lock:tuple` indica che un backend è in attesa di acquisire un blocco su una tupla mentre un altro backend tiene un blocco in conflitto sulla stessa tupla. Nella tabella seguente viene illustrato uno scenario in cui le sessioni generano l'evento `Lock:tuple`.

Orari	Sessione 1	Sessione 2	Sessione 3
t1	Inizia una transazione.		
t2	Aggiorna la riga 1.		
t3		Aggiorna la riga 1. La sessione acquisisce un blocco esclusivo sulla tupla e quindi attende che la sessione 1 rilasci il blocco eseguendo il commit o il rollback.	
t4			Aggiorna la riga 1. La sessione attende che la sessione 2 rilasci il blocco esclusivo sulla tupla.

Oppure puoi simulare questo evento di attesa utilizzando lo strumento di benchmarking `pgbench`. Configurare un numero elevato di sessioni simultanee per aggiornare la stessa riga in una tabella con un file SQL personalizzato.

Per ulteriori informazioni sulle modalità di blocco in conflitto, vedere Blocco esplicito nella documentazione di PostgreSQL. Per ulteriori informazioni su `pgbench`, consulta pgbench nella documentazione di PostgreSQL.

Probabili cause di aumento delle attese

Quando l'evento si verifica più del normale, probabilmente indicando un problema di prestazioni, le cause tipiche includono le seguenti:

- Un numero elevato di sessioni simultanee sta cercando di acquisire un blocco in conflitto per la stessa tupla eseguendo istruzioni `UPDATE` o `DELETE`.
- Sessioni altamente simultanee stanno eseguendo un'istruzione `SELECT` usando le modalità di blocco `FOR UPDATE` o `FOR NO KEY UPDATE`.
- Diversi fattori spingono le applicazioni o i connection pool ad aprire più sessioni per eseguire le stesse operazioni. Mentre le nuove sessioni stanno tentando di modificare le stesse righe, il carico del DB può aumentare e `Lock:tuple` può apparire.

Per ulteriori informazioni consulta Blocchi a livello di riga nella documentazione di PostgreSQL.

Azioni

Consigliamo azioni diverse a seconda delle cause dell'evento di attesa.

Argomenti

- Indagare la logica dell'applicazione
- Trova la sessione responsabile del blocco
- Riduci la concorrenza quando è alta
- Risoluzione dei problemi dei colli di bottiglia

Indagare la logica dell'applicazione

Scopri se una sessione bloccante è rimasta in stato `idle in transaction` per lungo tempo. In tal caso, considera di terminare la sessione di blocco come soluzione a breve termine. È anche possibile

usare la funzione `pg_terminate_backend`. Per ulteriori informazioni su questa funzione, consulta Funzioni di segnalazione server nella documentazione di PostgreSQL.

Per una soluzione a lungo termine, fai quanto seguente:

- Regola la logica dell'applicazione.
- Utilizzo del parametro `idle_in_transaction_session_timeout`. Questo parametro termina qualsiasi sessione con una transazione aperta che è rimasta inattiva per un periodo di tempo superiore al periodo di tempo specificato. Per ulteriori informazioni, consulta la pagina Errori connessione client nella documentazione di PostgreSQL.
- Usa autocommit il più possibile. Per ulteriori informazioni, consulta la pagina CONFIGURA AUTOCOMMIT nella documentazione di PostgreSQL.

Trova la sessione responsabile del blocco

Mentre si verifica l'evento di attesa `Lock:tuple`, identifica il blocco e la sessione bloccata scoprendo quali blocchi dipendono l'uno dall'altro. Per ulteriori informazioni, consulta Informazioni sulle dipendenze dei blocchi nel wiki di PostgreSQL. Per analizzare gli eventi `Lock:tuple` passati, usa la funzione Aurora `aurora_stat_backend_waits`.

L'esempio seguente esegue una query su tutte le sessioni, filtrando su `tuple` e ordinando per `wait_time`.

```
--AURORA_STAT_BACKEND_WAITS
      SELECT a.pid,
             a.usename,
             a.app_name,
             a.current_query,
             a.current_wait_type,
             a.current_wait_event,
             a.current_state,
             wt.type_name AS wait_type,
             we.event_name AS wait_event,
             a.waits,
             a.wait_time
        FROM (SELECT pid,
                     usename,
                     left(application_name,16) AS app_name,
                     coalesce(wait_event_type,'CPU') AS current_wait_type,
                     coalesce(wait_event,'CPU') AS current_wait_event,
```

```
                    state AS current_state,
                    left(query,80) as current_query,
                    (aurora_stat_backend_waits(pid)).*
                FROM pg_stat_activity
               WHERE pid <> pg_backend_pid()
                 AND usename<>'rdsadmin') a
NATURAL JOIN aurora_stat_wait_type() wt
NATURAL JOIN aurora_stat_wait_event() we
WHERE we.event_name = 'tuple'
    ORDER BY a.wait_time;

  pid  | usename | app_name |                current_query                  |
 current_wait_type | current_wait_event | current_state | wait_type | wait_event |
 waits | wait_time
-------+---------+----------+-----------------------------------------------
+-------------------+--------------------+--------------+-----------+------------
+-------+-----------
 32136 | sys     | psql     | /*session3*/ update tab set col=1 where col=1; | Lock
           | tuple              | active        | Lock      | tuple      |     1 |
 1000018
 11999 | sys     | psql     | /*session4*/ update tab set col=1 where col=1; | Lock
           | tuple              | active        | Lock      | tuple      |     1 |
 1000024
```

Riduci la concorrenza quando è alta

L'evento `Lock:tuple` potrebbe verificarsi costantemente, soprattutto in un tempo di carico di lavoro occupato. In questa situazione, si consideri di ridurre l'elevata concorrenza per le file molto occupate. Spesso, solo poche righe controllano una coda o la logica booleana, il che rende queste righe molto occupate.

È possibile ridurre la concorrenza utilizzando approcci diversi in base ai requisiti aziendali, alla logica dell'applicazione e al tipo di carico di lavoro. Ad esempio, puoi eseguire le operazioni seguenti:

- Riprogetta la tua tabella e la logica dei dati per ridurre la concorrenza elevata.
- Modificare la logica dell'applicazione per ridurre la concorrenza elevata a livello di riga.
- Sfrutta e riprogetta le query con i blocchi a livello di riga.
- Utilizzo della clausola `NOWAIT` con operazioni di riprova.
- Prendi in considerazione l'utilizzo di un controllo della concorrenza logico ottimistico e ibrido.
- Valuta la possibilità di modificare il livello di isolamento del database.

Risoluzione dei problemi dei colli di bottiglia

`Lock:tuple` può verificarsi con colli di bottiglia come la fame di CPU o il massimo utilizzo della larghezza di banda Amazon EBS. Per ridurre i colli di bottiglia, valuta i seguenti approcci:

- Ridimensiona il tipo di classe di istanza.
- Ottimizza le query a uso intensivo di risorse.
- Modificare la logica dell'applicazione.
- Archivia i dati a cui si accede raramente.

LWLock:buffer_content (BufferContent)

L'evento `LWLock:buffer_content` si verifica quando una sessione è in attesa di accedere in lettura o scrittura a una pagina dati in memoria mentre un'altra sessione ha bloccato la pagina in scrittura. In Aurora PostgreSQL 13 e versioni successive, questo evento di attesa viene chiamato `BufferContent`.

Argomenti

- Versioni del motore supportate
- Contesto
- Probabili cause di aumento delle attese
- Azioni

Versioni del motore supportate

Queste informazioni relative all'evento di attesa sono supportate per tutte le versioni di Aurora PostgreSQL.

Contesto

Per leggere o manipolare i dati, PostgreSQL vi accede tramite buffer di memoria condivisa. Per leggere dal buffer, un processo ottiene un blocco leggero (LWLock) sul contenuto del buffer in modalità condivisa. Per scrivere sul buffer, ottiene quel blocco in modalità esclusiva. I blocchi condivisi consentono ad altri processi di acquisire contemporaneamente blocchi condivisi su quel contenuto. I blocchi esclusivi impediscono ad altri processi di ottenere qualsiasi tipo di blocco.

L'evento `LWLock:buffer_content` (`BufferContent`) indica che più processi stanno tentando di ottenere blocchi leggeri sul contenuto di un buffer specifico.

Probabili cause di aumento delle attese

Quando l'evento `LWLock:buffer_content` (`BufferContent`) si verifica più del normale, probabilmente indicando un problema di prestazioni, le cause tipiche includono le seguenti.

Aggiornamenti simultanei aumentati degli stessi dati

Potrebbe esserci un aumento del numero di sessioni simultanee con query che inseriscono o aggiornano la stessa tabella. Questa contesa può essere più marcata sulle tabelle con molti indici.

I dati del carico di lavoro non sono in memoria

Quando i dati elaborati dal carico di lavoro attivo non sono in memoria, questi eventi di attesa possono aumentare. Questo effetto è dovuto al fatto che i processi che contengono blocchi possono mantenerli più a lungo mentre eseguono operazioni di I/O su disco.

Uso eccessivo di vincoli di chiave esterna

I vincoli di chiave esterna possono aumentare la quantità di tempo che un processo mantiene su un blocco del contenuto del buffer. Questo effetto è dovuto al fatto che le operazioni di lettura richiedono un blocco del contenuto del buffer condiviso sulla chiave di riferimento mentre quella chiave viene aggiornata.

Azioni

Consigliamo azioni diverse a seconda delle cause dell'evento di attesa. Potresti identificare eventi `LWLock:buffer_content` (`BufferContent`) utilizzando Amazon RDS Performance Insights o interrogando la vista `pg_stat_activity`.

Argomenti

- Migliora l'efficienza in memoria
- Riduzione dell'utilizzo di vincoli di chiave esterna
- Rimuovere gli indici inutilizzati
- Rimuovere gli indici duplicati
- Eseguire eliminazione o REINDEX per gli indici non validi

- Utilizzare gli indici parziali
- Rimuovere il bloat di tabelle e indici

Migliora l'efficienza in memoria

Per aumentare la probabilità che i dati del carico di lavoro attivo si trovino in memoria, partizionare tabelle o scalare la classe di istanza. Per informazioni sulle classi di istanza database, consulta Classi di istanze database Amazon Aurora.

Monitorare la metrica `BufferCacheHitRatio`, che misura la percentuale di richieste gestite dalla cache del buffer di un'istanza database nel cluster di database. Questa metrica fornisce informazioni sulla quantità di dati gestiti dalla memoria. Una percentuale di riscontri elevata indica che l'istanza database dispone di memoria sufficiente per il set di dati di lavoro, mentre una percentuale bassa suggerisce che le query accedono frequentemente ai dati dalla risorsa di archiviazione.

I riscontri di lettura nella cache per tabella e i riscontri di lettura nella cache per indice nella sezione Impostazioni memoria del report PG Collector possono fornire informazioni dettagliate sul rapporto di riscontri nella cache di tabelle e indici.

Riduzione dell'utilizzo di vincoli di chiave esterna

Indagare sui carichi di lavoro con un numero elevato di eventi di attesa `LWLock:buffer_content` (`BufferContent`) per l'utilizzo di vincoli di chiave esterna. Rimuovere i vincoli di chiave esterna non necessari.

Rimuovere gli indici inutilizzati

Per carichi di lavoro con un numero elevato di eventi di attesa `LWLock:buffer_content` (`BufferContent`), identificare gli indici inutilizzati e rimuoverli.

La sezione degli indici inutilizzati del report PG Collector può fornire informazioni dettagliate sugli indici inutilizzati nel database.

Rimuovere gli indici duplicati

Identificare gli indici duplicati e rimuoverli.

La sezione degli indici duplicati del report PG Collector può fornire informazioni dettagliate sugli indici duplicati nel database.

Eseguire eliminazione o REINDEX per gli indici non validi

Gli indici non validi si verificano in genere quando si utilizza `CREATE INDEX CONCURRENTLY` o `REINDEX CONCURRENTLY` e il comando ha esito negativo o viene interrotto.

Gli indici non validi non possono essere utilizzati per le query, ma verranno comunque aggiornati e occuperanno spazio su disco.

La sezione degli indici non validi del report PG Collector può fornire informazioni dettagliate sugli indici non validi nel database.

Utilizzare gli indici parziali

Gli indici parziali possono essere utilizzati per migliorare le prestazioni delle query e ridurre le dimensioni dell'indice. Un indice parziale è un indice creato su un sottoinsieme di una tabella, con il sottoinsieme definito da un'espressione condizionale. Come descritto nella documentazione sugli indici parziali, gli indici parziali possono ridurre il sovraccarico di gestione degli indici, poiché PostgreSQL non ha bisogno di aggiornare l'indice in tutti i casi.

Rimuovere il bloat di tabelle e indici

Un eccessivo bloat di tabelle e indici può influire negativamente sulle prestazioni del database. Un bloat delle tabelle e degli indici aumenta le dimensioni del set di lavoro attivo, riducendo l'efficienza della memoria. Inoltre, il bloat aumenta i costi di archiviazione e rallenta l'esecuzione delle query. Per diagnosticare un bloat, consulta Diagnosi delle dimensioni della tabella e dell'indice. Inoltre, la sezione Fragmentation (Bloat) del report PG Collector può fornire informazioni dettagliate sul bloat di tabelle e indici.

Per risolvere il problema del bloat di tabelle e indici, sono disponibili alcune opzioni:

VACUUM FULL

> `VACUUM FULL` crea una nuova copia della tabella, copiando solo le tuple attive, quindi sostituisce la vecchia tabella con quella nuova mantenendo un blocco `ACCESS EXCLUSIVE`. Ciò impedisce qualsiasi lettura o scrittura sulla tabella, che può causare un'interruzione. Inoltre, `VACUUM FULL` richiederà più tempo se la tabella è di grandi dimensioni.

pg_repack

> `pg_repack` è utile in situazioni in cui `VACUUM FULL` potrebbe non essere adatto. Crea una nuova tabella che contiene i dati della tabella con bloat, tiene traccia delle modifiche rispetto alla tabella originale e quindi sostituisce la tabella originale con quella nuova. Non blocca la tabella

originale per le operazioni di lettura o scrittura durante la creazione della nuova tabella. Per ulteriori informazioni su come utilizzare `pg_repack`, consulta Rimozione del bloat con pg_repack e pg_repack.

REINDEX

Il comando `REINDEX` può essere utilizzato per risolvere il problema del bloat dell'indice. `REINDEX` scrive una nuova versione dell'indice senza le pagine inattive o le pagine vuote o quasi vuote, riducendo così il consumo di spazio dell'indice. Per informazioni dettagliate sul comando `REINDEX`, consulta la documentazione di REINDEX.

Dopo aver rimosso il bloat dalle tabelle e dagli indici, potrebbe essere necessario aumentare la frequenza dell'autovacuum su tali tabelle. L'implementazione di impostazioni aggressive dell'autovacuum a livello di tabella può aiutare a prevenire il verificarsi di futuri bloat. Per ulteriori informazioni, consulta la documentazione su `Vacuuming and analyzing tables automatically`.

LWLock:buffer_mapping

Questo evento si verifica quando un processo di backend è in attesa di associare un blocco di dati a un buffer nel pool di buffer condiviso.

> **Note**
>
> Questo evento appare come `LWLock:buffer_mapping` in Aurora PostgreSQL versione 12 e versioni successive e `LWLock:BufferMapping` nella versione 13 e successive.

Argomenti

- Versioni del motore supportate
- Context
- Cause
- Azioni

Versioni del motore supportate

Queste informazioni relative all'evento di attesa sono supportate per Aurora PostgreSQL versione 9.6 e successive.

Context

Il pool buffer condiviso è un'area di memoria Aurora PostgreSQL che contiene tutte le pagine che sono o sono state utilizzate dai processi. Quando un processo ha bisogno di una pagina, legge la pagina nel buffer pool condiviso. Il parametro `shared_buffers` imposta le dimensioni del buffer condiviso e riserva un'area di memoria per memorizzare la tabella e le pagine indice. Se si modifica questo parametro, assicurarsi di riavviare il database. Per ulteriori informazioni, consulta Buffer condivisi.

L'evento di attesa `LWLock:buffer_mapping` si verifica nei seguenti scenari:

- Un processo ricerca nella tabella buffer una pagina e acquisisce un blocco di mappatura buffer condiviso.
- Un processo carica una pagina nel buffer pool e acquisisce un esclusivo blocco di mappatura del buffer.
- Un processo rimuove una pagina dal pool e acquisisce un blocco esclusivo di mappatura del buffer.

Cause

Quando questo evento appare più del normale, probabilmente indicando un problema di prestazioni, il database sta eseguendo il paging in entrata e in uscita dal buffer pool condiviso. Le cause tipiche sono:

- Query di grandi dimensioni
- Indici e tabelle gonfie
- Scansioni complete della tabella
- Dimensioni del pool condiviso più piccole del set di lavoro

Azioni

Consigliamo azioni diverse a seconda delle cause dell'evento di attesa.

Argomenti

- Monitora i parametri relativi al buffer
- Valuta la tua strategia di indicizzazione
- Riduci il numero di buffer che devono essere allocati rapidamente

Monitora i parametri relativi al buffer

Quando `LWLock:buffer_mapping` aspetta il picco, indaga il rapporto di hit del buffer. È possibile utilizzare questi parametri per comprendere meglio cosa sta accadendo nella cache del buffer. Esamina i seguenti parametri:

`BufferCacheHitRatio`

Il parametro Amazon CloudWatch misura la percentuale di richieste gestite dalla cache del buffer di un'istanza database nel cluster di database. È possibile che questo parametro diminuisca in anticipo all'evento di attesa `LWLock:buffer_mapping`.

`blks_hit`

Questo parametro contatore Performance Insights indica il numero di blocchi recuperati dal buffer pool condiviso. Dopo che appare l'evento di attesa `LWLock:buffer_mapping`, potresti osservare un picco in `blks_hit`.

`blks_read`

Questo parametro del contatore Performance Insights indica il numero di blocchi che richiedevano la lettura di I/O nel buffer pool condiviso. Potresti osservare un picco in `blks_read` in vista dell'evento di attesa `LWLock:buffer_mapping`.

Valuta la tua strategia di indicizzazione

Per confermare che la strategia di indicizzazione non sta peggiorando le prestazioni, verifica quanto segue:

Index bloat

Assicurati che il bloat di indice e tabella non porti alla lettura di pagine non necessarie nel buffer condiviso. Se le tabelle contengono righe inutilizzate, considera l'archiviazione dei dati e la rimozione delle righe dalle tabelle. È quindi possibile ricostruire gli indici per le tabelle ridimensionate.

Indici per query utilizzate di frequente

Per determinare se disponi degli indici ottimali, monitora i parametri del motore DB in Performance Insights. Il parametro `tup_returned` mostra il numero di righe lette. Il parametro `tup_fetched` mostra il numero di righe restituite al client. Se `tup_returned` è significativamente più grande di

`tup_fetched`, i dati potrebbero non essere indicizzati correttamente. Inoltre, le statistiche della tabella potrebbero non essere aggiornate.

Riduci il numero di buffer che devono essere allocati rapidamente

Per ridurre gli eventi di attesa `LWLock:buffer_mapping`, cercare di ridurre il numero di buffer che devono essere allocati rapidamente. Una strategia consiste nell'eseguire operazioni di batch di dimensioni ridotte. Potresti essere in grado di ottenere batch più piccoli partizionando le tabelle.

LWLock:BufferIO (IPC:BufferIO)

L'evento `LWLock:BufferIO` si verifica quando Aurora PostgreSQL o RDS per PostgreSQL sono in attesa che altri processi finiscano le operazioni di input/output (I/O) quando si tenta contemporaneamente di accedere a una pagina. Il suo scopo è quello di leggere la stessa pagina nel buffer condiviso.

Argomenti

- Versioni di motori pertinenti
- Context
- Cause
- Azioni

Versioni di motori pertinenti

Queste informazioni relative all'evento di attesa sono rilevanti per tutte le versioni di Aurora PostgreSQL. Per Aurora PostgreSQL 12 e versioni precedenti questo evento di attesa è denominato lwlock:buffer_io mentre in Aurora PostgreSQL versione 13 è denominato lwlock:bufferio. In Aurora PostgreSQL versione 14, l'evento di attesa BufferIO è stato spostato da `LWLock` al tipo di evento di attesa `IPC` (IPC:BufferIO).

Context

Ogni buffer condiviso ha un blocco I/O associato all'evento di attesa `LWLock:BufferIO`, ogni volta che un blocco (o una pagina) deve essere recuperato all'esterno del buffer pool condiviso.

Questo blocco viene utilizzato per gestire più sessioni che richiedono tutte l'accesso allo stesso blocco. Questo blocco deve essere letto dall'esterno del buffer pool condiviso, definito dal parametro `shared_buffers`.

Non appena la pagina viene letta all'interno del buffer pool condiviso, il blocco `LWLock:BufferIO` viene rilasciato.

> **Note**
>
> L'evento di attesa `LWLock:BufferIO` precede l'evento di attesa IO:DataFileRead. L'evento di attesa `IO:DataFileRead` si verifica mentre i dati vengono letti dallo storage.

Per ulteriori informazioni sui blocchi leggeri, consultaPanoramica dei blocchi.

Cause

Le cause comuni della comparsa dell'evento `LWLock:BufferIO` che appare nelle prime attese includono:

- Più backend o connessioni che tentano di accedere alla stessa pagina in attesa di un'operazione di I/O
- Il rapporto tra le dimensioni del buffer pool condiviso (definito dal parametro `shared_buffers`) e il numero di buffer necessari per il carico di lavoro corrente
- La dimensione del buffer pool condiviso non è ben bilanciata con il numero di pagine sfrattate da altre operazioni
- Indici grandi o gonfi che richiedono al motore di leggere più pagine del necessario nel buffer pool condiviso
- Mancanza di indici che costringe il motore DB a leggere più pagine dalle tabelle del necessario
- Picchi improvvisi per le connessioni al database che tentano di eseguire operazioni sulla stessa pagina

Azioni

Consigliamo azioni diverse a seconda delle cause dell'evento di attesa:

- Osservare i parametri di Amazon CloudWatch per la correlazione tra forti diminuzioni degli eventi di attesa `BufferCacheHitRatio` e `LWLock:BufferIO`. Questo effetto può significare che hai una piccola impostazione dei buffer condivisi. Potrebbe essere necessario aumentarlo o scalare la classe di istanza DB. È possibile dividere il carico di lavoro in più nodi di lettore.
- Verifica se hai indici inutilizzati, quindi rimuovili.

- Utilizzare tabelle partizionate (che hanno anche indici partizionati). Ciò aiuta a mantenere basso il riordino dell'indice e ne riduce l'impatto.
- Evitare di indicizzare inutilmente le colonne.
- Evita improvvisi picchi di connessione al database utilizzando un connection pool.
- Limitare il numero massimo di connessioni al database come best practice.

LWLock:lock_manager

Questo evento si verifica quando il motore Aurora PostgreSQL mantiene l'area di memoria del blocco condiviso per allocare, controllare e dislocare un blocco quando non è possibile un blocco rapido del percorso.

Argomenti

- Versioni del motore supportate
- Context
- Probabili cause di aumento delle attese
- Azioni

Versioni del motore supportate

Queste informazioni relative all'evento di attesa sono supportate per Aurora PostgreSQL versione 9.6 e successive.

Context

Quando si emette un'istruzione SQL, Aurora PostgreSQL registra i blocchi per proteggere la struttura, i dati e l'integrità del database durante le operazioni simultanee. Il motore può raggiungere questo obiettivo utilizzando un blocco veloce del percorso o un blocco del percorso che non è veloce. Un blocco del percorso che non è veloce è più costoso e crea più sovraccarico di un blocco veloce del percorso.

Blocco rapido del percorso

Per ridurre il sovraccarico dei blocchi che vengono presi e rilasciati frequentemente, ma che raramente sono in conflitto, i processi di backend possono utilizzare il blocco rapido del percorso. Il database utilizza questo meccanismo per i blocchi che soddisfano i seguenti criteri:

- Usano il metodo di blocco DEFAULT.
- Rappresentano un blocco su una relazione di database anziché una relazione condivisa.
- Sono blocchi deboli che difficilmente entrino in conflitto.
- Il motore può verificare rapidamente che non siano presenti blocchi in conflitto.

Il motore non può utilizzare il blocco rapido del percorso quando una di queste condizioni è vera:

- Il blocco non soddisfa i criteri precedenti.
- Non sono disponibili più slot per il processo di backend.

Per ulteriori informazioni sul blocco rapido del percorso, consulta percorso rapido nel gestore di blocco PostgreSQL README e pg-locks nella documentazione di PostgreSQL.

Esempio di problema di scalabilità per il blocco manager

In questo esempio, una tabella denominata `purchases` memorizza cinque anni di dati, partizionati per giorno. Ogni partizione ha due indici. Si verifica la seguente sequenza di eventi:

1. Si interrogano dati su diversi giorni, il che richiede al database di leggere molte partizioni.
2. Il database crea una voce di blocco per ogni partizione. Se gli indici di partizione fanno parte del percorso di accesso dell'ottimizzatore, il database crea anche una voce di blocco per loro.
3. Quando il numero di voci di blocco richieste per lo stesso processo di back-end è superiore a 16, ovvero il valore di `FP_LOCK_SLOTS_PER_BACKEND`, il gestore di blocco utilizza il metodo di blocco del percorso non veloce.

Le applicazioni moderne potrebbero avere centinaia di sessioni. Se le sessioni simultanee eseguono una query sul genitore senza un'adeguata scrematura delle partizioni, il database potrebbe creare centinaia o addirittura migliaia di blocchi di percorso non veloci. In genere, quando questa concorrenza è superiore al numero di vCPU, appare l'evento di attesa `LWLock:lock_manager`.

> **Note**
>
> L'evento di attesa `LWLock:lock_manager` non è correlato al numero di partizioni o indici in uno schema di database. Al contrario, è correlato al numero di blocchi di percorso non veloci che il database deve controllare.

Probabili cause di aumento delle attese

Quando l'evento di attesa `LWLock:lock_manager` si verifica più del normale, probabilmente indicando un problema di prestazioni, le cause più probabili di picchi improvvisi sono le seguenti:

- Le sessioni attive simultanee eseguono query che non utilizzano blocchi di percorso veloci. Queste sessioni superano anche la vCPU massima.
- Un gran numero di sessioni attive simultanee sta accedendo a una tabella fortemente partizionata. Ogni partizione ha più indici.
- Il database sta vivendo una tempesta di connessione. Per impostazione predefinita, alcune applicazioni e software del connection pool creano più connessioni quando il database è lento. Questa pratica peggiora il problema. Ottimizza il software del pool di connessioni in modo che non si verifichino tempeste di connessione.
- Un numero elevato di sessioni esegue una query su una tabella padre senza potare le partizioni.
- Un DDL (Data Definition Language), DML (Data Manipolation Language) o un comando di manutenzione blocca esclusivamente una relazione occupata o tuple a cui si accede frequentemente o modificate.

Azioni

Consigliamo azioni diverse a seconda delle cause dell'evento di attesa.

Argomenti

- Usare la potatura delle partizioni
- Rimuovere indici non necessari
- Ottimizza le tue query per bloccare rapidamente i percorsi
- Sintonizzati per altri eventi di attesa
- Riduzione dei colli di bottiglia hardware
- Utilizzare un connection pooler
- Aggiorna la versione Aurora PostgreSQL

Usare la potatura delle partizioni

Potatura delle partizioni è una strategia di ottimizzazione delle query che esclude le partizioni non necessarie dalle scansioni di tabelle, migliorando così le prestazioni. La potatura delle partizioni è attivata per impostazione predefinita. Se è spento, accenderlo come segue.

```
SET enable_partition_pruning = on;
```

Le query possono trarre vantaggio dalla potatura delle partizioni quando la clausola `WHERE` contiene la colonna utilizzata per il partizionamento. Per ulteriori informazioni, consulta Potatura delle partizioni nella documentazione di PostgreSQL.

Rimuovere indici non necessari

Il database potrebbe contenere indici inutilizzati o usati raramente. In tal caso, considera la possibilità di eliminarli. Esegui una delle operazioni seguenti:

- Scopri come trovare indici non necessari consultanto Indici non utilizzati nel wiki di PostgreSQL.
- Esegui PG Collector. Questo script SQL raccoglie le informazioni del database e le presenta in un report HTML consolidato. Controlla la sezione «Indici non utilizzati». Per ulteriori informazioni, consulta pg-collector nel repository AWS Labs GitHub.

Ottimizza le tue query per bloccare rapidamente i percorsi

Per scoprire se le tue query utilizzano il blocco rapido dei percorsi, esegui una query nella colonna `fastpath` della tabella `pg_locks`. Se le query non utilizzano il blocco rapido dei percorsi, provare a ridurre il numero di relazioni per query a meno di 16.

Sintonizzati per altri eventi di attesa

Se `LWLock:lock_manager` è il primo o il secondo nell'elenco delle prime attese, controlla se nell'elenco vengono visualizzati anche i seguenti eventi di attesa:

- `Lock:Relation`
- `Lock:transactionid`
- `Lock:tuple`

Se gli eventi precedenti appaiono in alto nell'elenco, prendi in considerazione prima l'ottimizzazione di questi eventi di attesa. Questi eventi possono essere un driver per `LWLock:lock_manager`.

Riduzione dei colli di bottiglia hardware

Potresti avere un collo di bottiglia hardware, come la fame di CPU o il massimo utilizzo della larghezza di banda Amazon EBS. In questi casi, valuta la riduzione dei colli di bottiglia hardware. Prendi in considerazione le seguenti azioni:

- Ridimensiona la tua classe di istanza.
- Ottimizza le query che consumano grandi quantità di CPU e memoria.
- Cambia la logica dell'applicazione.
- Archivia i tuoi dati.

Per ulteriori informazioni su CPU, memoria e larghezza di banda di rete EBS, vedere Tipi di istanza Amazon RDS.

Utilizzare un connection pooler

Se il numero totale di connessioni attive supera la vCPU massima, più processi del sistema operativo richiedono CPU di quella supportata dal tipo di istanza. In questo caso, valuta l'utilizzo o l'ottimizzazione di un connection pool. Per ulteriori informazioni sul numero di vCPU per ogni tipo di istanza, consulta Tipi di istanza di Amazon RDS.

Per ulteriori informazioni sul connection pool, consulta le risorse seguenti:

- Server proxy per Amazon RDS per Aurora
- pgbouncer
- Connection pool e origini dati nella Documentazione di PostgreSQL

Aggiorna la versione Aurora PostgreSQL

Se la versione attuale di Aurora PostgreSQL è inferiore a 12, esegui l'aggiornamento alla versione 12 o superiore. Le versioni 12 e 13 di PostgreSQL hanno un meccanismo di partizione migliorato. Per ulteriori informazioni sui miglioramenti nella versioni 12, consulta PostgreSQL 12 Release Notes. Per ulteriori informazioni sulle versioni di Aurora PostgreSQL, consulta Aggiornamenti del motore di database Amazon Aurora PostgreSQL per 02/06/2020.

LWLock:MultiXact

Gli eventi di attesa `LWLock:MultiXactMemberBuffer`, `LWLock:MultiXactOffsetBuffer`, `LWLock:MultiXactMemberSLRU` e `LWLock:MultiXactOffsetSLRU` indicano che una sessione è in attesa di recuperare un elenco di transazioni che modifica la stessa riga di una determinata tabella.

- `LWLock:MultiXactMemberBuffer`: un processo è in attesa di I/O su un buffer semplice utilizzato meno di recente (SLRU) per un membro multixact.
- `LWLock:MultiXactMemberSLRU`: un processo è in attesa di accedere alla cache semplice utilizzata meno di recente (SLRU) per un membro multixact.
- `LWLock:MultiXactOffsetBuffer`: un processo è in attesa di I/O su un buffer semplice utilizzato meno di recente (SLRU) per un offset multixact.
- `LWLock:MultiXactOffsetSLRU`: un processo è in attesa di accedere alla cache semplice utilizzata meno di recente (SLRU) per un offset multixact.

Argomenti

- Versioni del motore supportate
- Contesto
- Probabili cause di aumento delle attese
- Azioni

Versioni del motore supportate

Queste informazioni relative all'evento di attesa sono supportate per tutte le versioni di Aurora PostgreSQL.

Contesto

Un multixact è una struttura dati che memorizza un elenco di ID transazione (XID) che modificano la stessa riga della tabella. Quando una singola transazione fa riferimento alla riga di una tabella, l'ID transazione viene memorizzato nella riga di intestazione della tabella. Quando più transazioni fanno riferimento alla stessa riga di una tabella, l'elenco degli ID transazione viene memorizzato nella struttura dati multixact. Gli eventi di attesa multixact indicano che una sessione sta recuperando dalla struttura dei dati l'elenco delle transazioni che fanno riferimento a una determinata riga di una tabella.

Probabili cause di aumento delle attese

Di seguito sono indicate tre cause comuni dell'uso di multixact:

- Sottotransazioni da punti di salvataggio espliciti: la creazione esplicita di un punto di salvataggio nelle transazioni genera nuove transazioni per la stessa riga. Ad esempio, utilizzando `SELECT FOR UPDATE`, `SAVEPOINT` e quindi `UPDATE`.

  Alcuni driver, ORM (Object-Relational Mapper) e livelli di astrazione dispongono di opzioni di configurazione per eseguire automaticamente il wrapping di tutte le operazioni con punti di salvataggio. Questo comportamento può generare molti eventi di attesa multixact in alcuni carichi di lavoro. L'opzione `autosave` del driver PostgreSQL JDBC ne è un esempio. Per ulteriori informazioni, consulta pgJDBC nella documentazione di PostgreSQL JDBC. Un altro esempio è il driver PostgreSQL ODBC e l'opzione `protocol`. Per ulteriori informazioni, consulta psqlODBC Configuration Options (Opzioni di configurazione di psqlODBC) nella documentazione del driver PostgreSQL ODBC.
- Sottotransazioni da clausole PL/pgSQL EXCEPTION: ogni clausola `EXCEPTION` scritta nelle funzioni o procedure PL/pgSQL crea internamente un oggetto `SAVEPOINT`.
- Chiavi esterne. Più transazioni acquisiscono un blocco di condivisione nel record padre.

Quando una determinata riga è inclusa in un'operazione con transazioni multiple, l'elaborazione della riga richiede il recupero degli ID transazione dagli elenchi `multixact`. Se le ricerche non riescono a ottenere il multixact dalla cache di memoria, la struttura dei dati deve essere letta dal livello di archiviazione di Aurora. Questa operazione I/O dall'archiviazione significa che le query SQL possono richiedere più tempo. Gli errori nella cache di memoria possono iniziare a verificarsi con l'utilizzo intensivo, a causa del numero elevato di transazioni multiple. Tutti questi fattori contribuiscono ad un aumento di questo evento di attesa.

Azioni

Consigliamo azioni diverse a seconda delle cause dell'evento di attesa. Alcune di queste azioni possono aiutare a ridurre immediatamente gli eventi di attesa. Tuttavia, altre potrebbero richiedere indagini e correzioni per scalare il carico di lavoro.

Argomenti

- Eseguire il congelamento del vacuum sulle tabelle con questo evento di attesa
- Aumentare la frequenza di autovacuum sulle tabelle con questo evento di attesa

- Aumentare i parametri di memoria
- Ridurre le transazioni di lunga durata
- Azioni a lungo termine

Eseguire il congelamento del vacuum sulle tabelle con questo evento di attesa

Se questo evento di attesa si verifica all'improvviso e influisce sull'ambiente di produzione, è possibile utilizzare uno dei seguenti metodi temporanei per ridurne il numero.

- Utilizzare VACUUM FREEZE sulla tabella o sulla partizione di tabella interessata per risolvere immediatamente il problema. Per ulteriori informazioni, consulta VACUUM.
- Utilizzare la clausola VACUUM (FREEZE, INDEX_CLEANUP FALSE) per eseguire un processo di vacuum rapido saltando gli indici. Per ulteriori informazioni, consulta Vacuum di una tabella il più rapidamente possibile.

Aumentare la frequenza di autovacuum sulle tabelle con questo evento di attesa

Dopo aver scansionato tutte le tabelle in tutti i database, VACUUM rimuoverà infine i multixact e i relativi valori multixact meno recenti verranno avanzati. Per ulteriori informazioni, consulta Multixact e wraparound. Per ridurre al minimo gli eventi di attesa LWLock:MultiXact, è necessario eseguire VACUUM tutte le volte che è necessario. A tale scopo, assicurarsi che VACUUM nel cluster di database Aurora PostgreSQL sia configurato in modo ottimale.

Se l'utilizzo di VACUUM FREEZE sulla tabella o sulla partizione di tabella interessata risolve il problema dell'evento di attesa, è consigliabile utilizzare un pianificatore, ad esempio `pg_cron`, per eseguire VACUUM invece di regolare l'autovacuum a livello di istanza.

Per aumentare la frequenza di autovacuum, è possibile ridurre il valore del parametro di archiviazione `autovacuum_multixact_freeze_max_age` nella tabella interessata. Per ulteriori informazioni, consulta autovacuum_multixact_freeze_max_age.

Aumentare i parametri di memoria

È possibile ottimizzare l'utilizzo della memoria per le cache multixact regolando i seguenti parametri. Queste impostazioni controllano la quantità di memoria riservata a queste cache, il che può aiutare a ridurre gli eventi di attesa multixact nel carico di lavoro. Per iniziare, è consigliabile utilizzare i seguenti valori:

Per Aurora PostgreSQL 17 e versioni successive:

- `multixact_offset_buffers` = 128
- `multixact_member_buffers` = 256

Per Aurora PostgreSQL 16 e versioni precedenti:

- `multixact_offsets_cache_size` = 128
- `multixact_members_cache_size` = 256

> **Note**
>
> In Aurora PostgreSQL 17, i nomi dei parametri vengono modificati da `multixact_offsets_cache_size` a `multixact_offset_buffers` e da `multixact_members_cache_size` a `multixact_member_buffers` per allinearli alla versione Community PostgreSQL 17.

È possibile impostare questi parametri a livello di cluster in modo che tutte le istanze del cluster rimangano coerenti. È consigliabile testare e modificare i valori per adattarli al meglio ai requisiti specifici del carico di lavoro e alla classe di istanza. È necessario riavviare l'istanza di scrittura affinché le modifiche ai parametri abbiano effetto.

I parametri sono espressi in termini di voci di cache multixact. Ogni voce di cache utilizza `8 KB` di memoria. Per calcolare la memoria totale riservata, moltiplicare il valore di ogni parametro per `8 KB`. Ad esempio, se si imposta un parametro su 128, la memoria riservata totale sarà `128 * 8 KB = 1 MB`.

Ridurre le transazioni di lunga durata

Le transazioni di lunga durata fanno sì che le informazioni sulle transazioni vengano mantenute nel vacuum fino al commit della transazione o fino alla chiusura della transazione di sola lettura. Ti consigliamo di monitorare e gestire in modo proattivo le transazioni di lunga durata. Per ulteriori informazioni, consulta Il database ha una connessione di transazione inattiva da molto tempo. Provare a modificare l'applicazione per evitare o ridurre al minimo l'uso di transazioni di lunga durata.

Azioni a lungo termine

Esaminare il carico di lavoro per scoprire la causa dello spillover multixact. È necessario risolvere il problema per scalare il carico di lavoro e ridurre l'evento di attesa.

- È necessario analizzare il DDL (Data Definition Language) utilizzato per creare le tabelle. Assicurarsi che le strutture e gli indici delle tabelle siano ben progettati.
- Se le tabelle interessate hanno chiavi esterne, stabilire se sono necessarie o se esiste un altro modo per applicare l'integrità referenziale.
- Quando una tabella ha indici inutilizzati di grandi dimensioni, l'autovacuum può non adattarsi al carico di lavoro e potrebbe impedirne l'esecuzione. Per evitare ciò, verificare la presenza di indici non utilizzati e rimuoverli completamente. Per ulteriori informazioni, consulta Gestione dell'autovacuum con indici di grandi dimensioni.
- Ridurre l'uso di punti di salvataggio nelle transazioni.

LWLock:pg_stat_statements

L'evento di attesa LWLock:pg_stat_statements si verifica quando l'estensione `pg_stat_statements` blocca in modo esclusivo la tabella hash che tiene traccia delle istruzioni SQL. Ciò accade nei seguenti scenari:

- Quando il numero di istruzioni tracciate raggiunge il valore del parametro `pg_stat_statements.max` configurato ed è necessario lasciare spazio a ulteriori voci, l'estensione esegue un ordinamento in base al numero di chiamate, rimuove il 5% delle istruzioni meno eseguite e ripopola l'hash con le voci rimanenti.
- Quando `pg_stat_statements` esegue un'operazione `garbage collection` sul file `pgss_query_texts.stat` su disco e riscrive il file.

Argomenti

- Versioni del motore supportate
- Context
- Probabili cause di aumento delle attese
- Azioni

Versioni del motore supportate

Queste informazioni relative all'evento di attesa sono supportate per tutte le versioni di Aurora PostgreSQL.

Context

Comprendere l'estensione pg_stat_statements: l'estensione pg_stat_statements tiene traccia delle statistiche di esecuzione delle istruzioni SQL in una tabella hash. L'estensione tiene traccia delle istruzioni SQL fino al limite definito dal parametro `pg_stat_statements.max`. Questo parametro determina il numero massimo di istruzioni che possono essere tracciate che corrisponde al numero massimo di righe nella vista pg_stat_statements.

Persistenza delle statistiche delle istruzioni: l'estensione mantiene le statistiche delle istruzioni tra i riavvii dell'istanza mediante:

- Scrittura di dati in un file denominato pg_stat_statements.stat
- Utilizzo del parametro pg_statements.save per controllare il comportamento di persistenza

Quando pg_statements.save è impostato su:

- on (impostazione predefinita): le statistiche vengono salvate all'arresto e ricaricate all'avvio del server
- off: le statistiche non vengono salvate all'arresto né ricaricate all'avvio del server

Archiviazione del testo delle query: l'estensione archivia il testo delle query tracciate in un file denominato `pgss_query_texts.stat`. Questo file può aumentare fino a raddoppiare la dimensione media di tutte le istruzioni SQL tracciate prima che si verifichi la rimozione di oggetti inutili. L'estensione richiede un blocco esclusivo sulla tabella hash durante le operazioni di pulizia e riscrittura del file `pgss_query_texts.stat`.

Processo di deallocazione delle istruzioni: quando il numero di istruzioni tracciate raggiunge il limite `pg_stat_statements.max` ed è necessario tenere traccia delle nuove istruzioni, l'estensione:

- Acquisisce un blocco esclusivo (LWLock:pg_stat_statements) sulla tabella hash.
- Carica i dati esistenti nella memoria locale.
- Esegue un ordinamento rapido in base al numero di chiamate.
- Rimuove le istruzioni meno chiamate (inferiore al 5%).
- Ripopola la tabella hash con le voci rimanenti.

Monitoraggio della deallocazione delle istruzioni: in PostgreSQL 14 e versioni successive, è possibile monitorare la deallocazione delle istruzioni utilizzando la vista pg_stat_statements_info. Questa visualizzazione include una colonna dealloc che mostra quante volte le istruzioni sono state deallocate per fare spazio a quelle nuove

Se la deallocazione delle istruzioni si verifica frequentemente, ciò comporterà una rimozione di oggetti inutili più frequente del file `pgss_query_texts.stat` su disco.

Probabili cause di aumento delle attese

Le cause tipiche dell'aumento delle attese di `LWLock:pg_stat_statements` includono:

- Un aumento del numero di query univoche utilizzate dall'applicazione.
- Il valore del parametro `pg_stat_statements.max` è piccolo rispetto al numero di query univoche utilizzate.

Azioni

Consigliamo azioni diverse a seconda delle cause dell'evento di attesa. È possibile identificare gli eventi `LWLock:pg_stat_statements` utilizzando Approfondimenti sulle prestazioni di Amazon RDS o eseguendo una query sulla vista `pg_stat_activity`.

Modificare i seguenti parametri `pg_stat_statements` per controllare il comportamento di tracciamento e ridurre gli eventi di attesa LWLock:pg_stat_statements.

Argomenti

- Disabilitare il parametro pg_stat_statements.track
- Aumentare il parametro pg_stat_statements.max
- Disabilitare il parametro pg_stat_statements.track_utility

Disabilitare il parametro pg_stat_statements.track

Se l'evento di attesa LWLock:pg_stat_statements ha un impatto negativo sulle prestazioni del database ed è necessaria una soluzione rapida prima di un'ulteriore analisi della vista `pg_stat_statements` per identificare la causa principale, il parametro `pg_stat_statements.track` può essere disabilitato impostandolo su `none`. In questo modo verrà disabilitata la raccolta delle statistiche sulle istruzioni.

Aumentare il parametro pg_stat_statements.max

Per ridurre la deallocazione e minimizzare la rimozione di oggetti inutili del file `pgss_query_texts.stat` su disco, aumentare il valore del parametro `pg_stat_statements.max`. Il valore predefinito è `5,000`.

> **Note**
>
> Il parametro `pg_stat_statements.max` è statico. È necessario riavviare l'istanza database per applicare le modifiche a questo parametro.

Disabilitare il parametro pg_stat_statements.track_utility

È possibile analizzare la vista pg_stat_statements per determinare quali comandi di utilità stanno consumando la maggior parte delle risorse monitorate da `pg_stat_statements`.

Il parametro `pg_stat_statements.track_utility` controlla se il modulo tiene traccia dei comandi di utilità, che includono tutti i comandi tranne SELECT, INSERT, UPDATE, DELETE e MERGE. Questo parametro è impostato su on per impostazione predefinita.

Ad esempio, quando l'applicazione utilizza molte query sui punti di salvataggio, che sono intrinsecamente univoche, può aumentare la deallocazione delle istruzioni. Per risolvere questo problema, è possibile disabilitare il parametro `pg_stat_statements.track_utility` per impedire a `pg_stat_statements` di monitorare le query sui punti di salvataggio.

> **Note**
>
> Il parametro `pg_stat_statements.track_utility` è un parametro dinamico. È possibile modificarne il valore senza riavviare l'istanza database.

Example Esempio di query univoche sui punti di salvataggio in pg_stat_statements

```
                    query                         |       queryid
--------------------------------------------------+---------------------
 SAVEPOINT JDBC_SAVEPOINT_495701                  | -7249565344517699703
 SAVEPOINT JDBC_SAVEPOINT_1320                    | -1572997038849006629
 SAVEPOINT JDBC_SAVEPOINT_26739                   |  54791337410474486
```

```
SAVEPOINT JDBC_SAVEPOINT_1294466                | 8170064357463507593
ROLLBACK TO SAVEPOINT JDBC_SAVEPOINT_65016      | -33608214779996400
SAVEPOINT JDBC_SAVEPOINT_14185                  | -2175035613806809562
SAVEPOINT JDBC_SAVEPOINT_45837                  | -6201592986750645383
SAVEPOINT JDBC_SAVEPOINT_1324                   | 6388797791882029332
```

PostgreSQL 17 introduce diversi miglioramenti per il tracciamento dei comandi di utilità:

- I nomi dei punti di salvataggio vengono ora visualizzati come costanti.
- Gli ID di transazione globali (GID) dei comandi di commit in due fasi vengono ora visualizzati come costanti.
- I nomi delle istruzioni DEALLOCATE vengono visualizzati come costanti.
- I parametri CALL vengono ora visualizzati come costanti.

Timeout: PG Sleep

L'evento `Timeout:PgSleep` si verifica quando un processo server ha chiamato la funzione `pg_sleep` e sta aspettando la scadenza del timeout del sonno.

Argomenti

- Versioni del motore supportate
- Probabili cause di aumento delle attese
- Azioni

Versioni del motore supportate

Queste informazioni relative all'evento di attesa sono supportate per tutte el versioni di Aurora PostgreSQL.

Probabili cause di aumento delle attese

Questo evento di attesa si verifica quando un'applicazione, una funzione memorizzata o un utente emette un'istruzione SQL che chiama una delle seguenti funzioni:

- `pg_sleep`
- `pg_sleep_for`
- `pg_sleep_until`

Le funzioni precedenti ritardano l'esecuzione fino a quando non è trascorso il numero di secondi specificato. Ad esempio: `SELECT pg_sleep(1)` si ferma per 1 secondo. Per ulteriori informazioni, consulta Ritardo dell'esecuzione nella documentazione di PostgreSQL.

Azioni

Identificare la dichiarazione che stava eseguendo la funzione `pg_sleep`. Determina se l'uso della funzione è appropriato.

Ottimizzazione di DevOps

DevOpsGuru proactive Insights rileva le condizioni sui cluster causare problemi e ti informa prima che si verifichino. Gli approfondimenti proattivi possono avvisare di una connessione di transazione inattiva da molto tempo. Per ulteriori informazioni sulla risoluzione dei problemi di inattività prolungata nelle connessioni di transazione, consulta Il database ha una connessione di transazione inattiva da molto tempo.

DevOpsGuru può fare quanto segue:

- Evitare molti problemi comuni relativi al database controllando la configurazione del database rispetto alle impostazioni consigliate comuni.
- Ricevere gli avvisi per le criticità relative al parco istanze che, se non controllate, possono portare a problemi più gravi in seguito.
- Ricevere gli avvisi per i nuovi problemi individuati.

Ogni approfondimento proattivo contiene un'analisi della causa del problema e i suggerimenti per le azioni correttive.

Per ulteriori informazioni su Amazon DevOps Guru for Amazon RDS, consulta. Analisi delle anomalie prestazionali di Aurora con DevOps Amazon Guru per Amazon RDS

Il database ha una connessione di transazione inattiva da molto tempo

Una connessione al database è nello stato `idle in transaction` da più di 1800 secondi.

Argomenti

- Versioni del motore supportate
- Contesto

- Probabili cause di questo problema
- Azioni
- Metriche pertinenti

Versioni del motore supportate

Queste informazioni approfondite sono supportate per tutte le versioni di Aurora PostgreSQL.

Contesto

Una transazione nello stato `idle in transaction` può contenere blocchi che impediscono l'esecuzione di altre query. Può anche impedire al `VACUUM` (incluso l'autovacuum) di cancellare le righe inutilizzate, con conseguente aumento delle dimensioni dell'indice o della tabella o del wraparound dell'ID della transazione.

Probabili cause di questo problema

Una transazione avviata in una sessione interattiva con BEGIN o START TRANSACTION non è stata terminata utilizzando un comando COMMIT, ROLLBACK o END. Lo stato della transazione diventa pertanto `idle in transaction`.

Azioni

Puoi individuare le transazioni inattive eseguendo la query `pg_stat_activity`.

Nel client SQL, esegui la query riportata di seguito per elencare tutte le connessioni nello stato `idle in transaction` e ordinarle in base alla durata:

```
SELECT now() - state_change as idle_in_transaction_duration, now() - xact_start as
 xact_duration,*
FROM  pg_stat_activity
WHERE state  = 'idle in transaction'
AND   xact_start is not null
ORDER BY 1 DESC;
```

Consigliamo azioni diverse a seconda delle cause degli approfondimenti.

Argomenti

- Terminare la transazione

- Interrompere la connessione
- Configurare il parametro idle_in_transaction_session_timeout
- Controllare lo stato di AUTOCOMMIT
- Controllare la logica delle transazioni nel codice dell'applicazione

Terminare la transazione

Quando si avvia una transazione in una sessione interattiva con BEGIN o START TRANSACTION, lo stato della transazione diventa `idle in transaction`. Rimane in questo stato finché non si termina la transazione con un comando COMMIT, ROLLBACK, END o si disconnette completamente la connessione per eseguire il rollback della transazione.

Interrompere la connessione

Interrompi la connessione con una transazione inattiva utilizzando la seguente query:

```
SELECT pg_terminate_backend(pid);
```

pid è l'ID di processo della connessione.

Configurare il parametro idle_in_transaction_session_timeout

Configura il parametro `idle_in_transaction_session_timeout` nel gruppo di parametri. Il vantaggio della configurazione di questo parametro è che non richiede un intervento manuale per terminare la transazione inattiva da tempo. Per ulteriori informazioni, consulta la documentazione di PostgreSQL.

Il seguente messaggio verrà riportato nel file di log di PostgreSQL dopo l'interruzione della connessione, quando lo stato di una transazione è idle_in_transaction per un periodo superiore al tempo specificato.

```
FATAL: terminating connection due to idle in transaction timeout
```

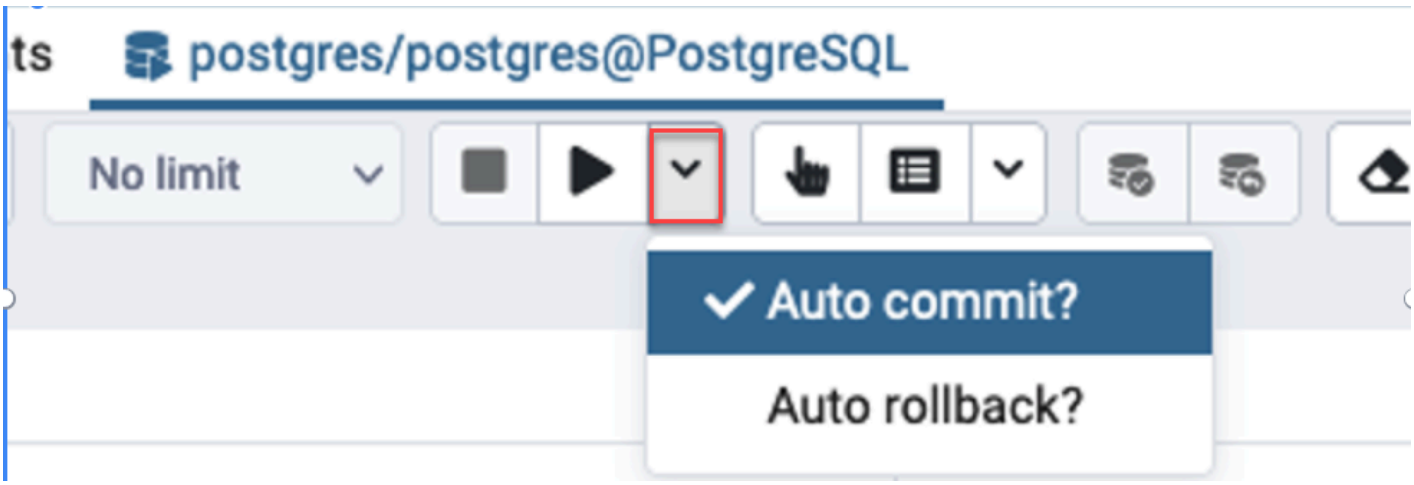
Controllare lo stato di AUTOCOMMIT

AUTOCOMMIT è attivato per impostazione predefinita. Tuttavia accidentalmente se viene disattivato nel client, assicurati di riattivarlo.

- Nel client psql, esegui il comando seguente:

```
postgres=> \set AUTOCOMMIT on
```

- In pgadmin, attivalo scegliendo l'opzione AUTOCOMMIT selezionando la freccia rivolta verso il basso.



Controllare la logica delle transazioni nel codice dell'applicazione

Controlla la logica dell'applicazione per individuare eventuali problemi. Prendi in considerazione le seguenti azioni:

- Controlla se il commit automatico JDBC è impostato su true nell'applicazione. Inoltre, considera l'utilizzo di comandi `COMMIT` espliciti nel codice.
- Controlla la logica di gestione degli errori per vedere se chiude una transazione dopo gli errori.
- Controlla se l'applicazione impiega molto tempo per elaborare le righe restituite da una query mentre la transazione è aperta. In tal caso, valuta la possibilità di codificare l'applicazione per chiudere la transazione prima di elaborare le righe.
- Controlla se una transazione contiene molte operazioni di lunga durata. In tal caso, dividi la singola transazione in più transazioni.

Metriche pertinenti

A questo approfondimento sono correlati i seguenti parametri PI:

- idle_in_transaction_count - Numero di sessioni nello stato `idle in transaction`.
- idle_in_transaction_max_time - La durata della transazione in esecuzione più lunga nello stato `idle in transaction`.

Best practice con Amazon Aurora PostgreSQL

Di seguito, sono disponibili diverse best practice per gestire il cluster database Amazon Aurora PostgreSQL. Assicurati di esaminare anche le attività di manutenzione di base. Per ulteriori informazioni, consulta Prestazioni e dimensionamento per Amazon Aurora PostgreSQL.

Argomenti

- Evitare il rallentamento delle prestazioni, il riavvio automatico e il failover per le istanze database Aurora PostgreSQL
- Diagnosi delle dimensioni della tabella e dell'indice
- Migliore gestione della memoria in Aurora PostgreSQL
- Failover rapido con Amazon Aurora PostgreSQL
- Ripristino rapido dopo il failover con Cluster Cache Management per Aurora PostgreSQL
- Gestione dell’abbandono della connessione Aurora PostgreSQL con pooling
- Gestione delle connessioni interrotte in PostgreSQL
- Ottimizzazione dei parametri di memoria per Aurora PostgreSQL
- Utilizzo dei CloudWatch parametri di Amazon per analizzare l'utilizzo delle risorse per Aurora PostgreSQL
- Utilizzo della replica logica per eseguire l'aggiornamento a una versione principale per Aurora PostgreSQL
- Risoluzione dei problemi di archiviazione in Aurora PostgreSQL

Evitare il rallentamento delle prestazioni, il riavvio automatico e il failover per le istanze database Aurora PostgreSQL

Se si eseguono uno o più carichi di lavoro pesanti che eccedono le risorse allocate dell'istanza database, è possibile che le risorse su cui si esegue l'applicazione e il database Aurora si esauriscano. Per ottenere metriche sull'istanza database quali utilizzo di CPU, utilizzo di memoria e il numero di connessioni di database utilizzate, è possibile fare riferimento alle metriche fornite da Amazon CloudWatch, Approfondimenti sulle prestazioni ed Enhanced Monitoring. Per ulteriori informazioni sul monitoraggio dell'istanza database, consultare Monitoraggio dei parametri in un cluster di database Amazon Aurora.

Se il carico di lavoro esaurisce le risorse in uso, è possibile che l'istanza database rallenti, venga riavviata o esegua un failover su un'altra istanza database. Per evitare che questo si verifichi, occorre

monitorare l'utilizzo delle risorse, esaminare il carico di lavoro in esecuzione sull'istanza database ed eseguire le ottimizzazioni dove necessario. Se le ottimizzazioni non migliorano le metriche dell'istanza e non mitigano l'esaurimento delle risorse, valutare la possibilità di aumentare l'istanza database prima che raggiunga i suoi limiti. Per ulteriori informazioni sulle classi di istanza database disponibili e le relative specifiche, consultare Classi di istanze database Amazon Aurora.

Diagnosi delle dimensioni della tabella e dell'indice

Puoi usare PostgreSQL Multiversion Concurrency Control (MVCC) per preservare l'integrità dei dati. PostgreSQL MVCC funziona salvando una copia interna delle righe aggiornate o eliminate (chiamate anche tuple) fino al commit o al rollback di una transazione. Questa copia interna salvata è invisibile agli utenti. Tuttavia, le dimensioni della tabella possono aumentare quando le copie invisibili non vengono pulite regolarmente dalle utilità VACUUM o AUTOVACUUM. Se non controllato, l'aumento delle dimensioni della tabella può comportare un aumento dei costi di archiviazione e rallentare la velocità di elaborazione.

In molti casi, le impostazioni predefinite per VACUUM o AUTOVACUUM su Aurora sono sufficienti per gestire l'aumento indesiderato delle dimensioni della tabella. Tuttavia, verifica la presenza di un aumento delle dimensioni se l'applicazione presenta le seguenti condizioni:

- Elabora un gran numero di transazioni in un tempo relativamente breve tra i processi VACUUM.
- Funziona male e esaurisce lo spazio di archiviazione.

Per iniziare, raccogli le informazioni accurate su quanto spazio viene utilizzato dalle tuple inattive e su quanto puoi recuperare ripulendo le dimensioni della tabella e dell'indice. Per farlo, usa l'estensione `pgstattuple` per raccogliere statistiche sul cluster Aurora. Per ulteriori informazioni, consulta pgstattuple. I privilegi di utilizzo dell'estensione `pgstattuple` sono limitati al ruolo `pg_stat_scan_tables` e ai superuser del database.

Per creare l'estensione `pgstattuple` su Aurora, connetti una sessione client al cluster, ad esempio psql o pgAdmin, e usa il seguente comando:

```
CREATE EXTENSION pgstattuple;
```

Crea l'estensione in ogni database da profilare. Dopo aver creato l'estensione, usa l'interfaccia della linea di comando (CLI) per misurare la quantità di spazio inutilizzato che puoi recuperare. Prima di raccogliere le statistiche, modifica il gruppo di parametri del cluster impostando AUTOVACUUM su

0. Un'impostazione pari a 0 impedisce ad Aurora di eliminare automaticamente tutte le tuple inattive dell'applicazione, il che può influire sulla precisione dei risultati. Immetti il seguente comando per creare una tabella semplice:

```
postgres=> CREATE TABLE lab AS SELECT generate_series (0,100000);
SELECT 100001
```

Nell'esempio seguente, eseguiamo la query con AUTOVACUUM attivato per il cluster di database. Il valore di `dead_tuple_count` è pari a 0, che indica che AUTOVACUUM ha cancellato i dati obsoleti o le tuple dal database PostgreSQL.

Per utilizzare `pgstattuple` per raccogliere informazioni sulla tabella, specifica il nome di una tabella o un identificatore di oggetto (OID) nella query:

```
postgres=> SELECT * FROM pgstattuple('lab');
```

```
table_len  | tuple_count | tuple_len | tuple_percent | dead_tuple_count |
 dead_tuple_len | dead_tuple_percent | free_space | free_percent
-----------+-------------+-----------+---------------+------------------
+----------------+--------------------+------------+--------------
3629056    | 100001      | 2800028   | 77.16         | 0                | 0
  | 0                  | 16616      | 0.46
(1 row)
```

Nella seguente query, disattiviamo AUTOVACUUM ed eseguiamo un comando che elimina 25.000 righe dalla tabella. Di conseguenza, il valore di `dead_tuple_count` aumenta a 25000.

```
postgres=> DELETE FROM lab WHERE generate_series < 25000;

DELETE 25000
```

```
SELECT * FROM pgstattuple('lab');
```

```
table_len | tuple_count | tuple_len | tuple_percent | dead_tuple_count | dead_tuple_len
 | dead_tuple_percent | free_space | free_percent
-----------+-------------+-----------+---------------+------------------
+----------------+--------------------+------------+--------------
3629056 | 75001 | 2100028 | 57.87 | 25000 | 700000 | 19.29 | 16616 | 0.46
(1 row)
```

Per recuperare le tuple inattive, avvia un processo VACUUM.

Osservazione delle dimensioni senza interrompere l'applicazione

Le impostazioni su un cluster Aurora sono ottimizzate per fornire le best practice per la maggior parte dei carichi di lavoro. Tuttavia, potresti voler ottimizzare un cluster per adattarlo meglio alle tue applicazioni e ai tuoi modelli di utilizzo. In questo caso, puoi utilizzare l'estensione `pgstattuple` senza interrompere un'applicazione in esecuzione. A tale scopo, esegui i seguenti passaggi:

1. Clona la tua istanza Aurora.
2. Modifica il file dei parametri per disattivare AUTOVACUUM nel clone.
3. Esegui una query `pgstattuple` durante il test del clone con un carico di lavoro di esempio o con pgbench, che è un programma per eseguire test di benchmark su PostgreSQL. Per ulteriori informazioni, consulta pgbench.

Dopo aver eseguito le applicazioni e visualizzato il risultato, usa pg_repack o VACUUM FULL sulla copia ripristinata e confronta le differenze. Se noti un calo significativo dei valori di dead_tuple_count, dead_tuple_len o dead_tuple_percent, modifica il programma di vacuum sul cluster di produzione per ridurre al minimo l'aumento delle dimensioni.

Impedimento dell'aumento delle dimensioni nelle tabelle temporanee

Se l'applicazione crea tabelle temporanee, assicurati che vengano rimosse quando non sono più necessarie. I processi Autovacuum non individuano le tabelle temporanee. Se non selezionate, le tabelle temporanee possono creare rapidamente un aumento delle dimensioni del database. Inoltre,

l'aumento delle dimensioni può estendersi alle tabelle di sistema, che sono le tabelle interne che tracciano gli oggetti e gli attributi di PostgreSQL, come pg_attribute e pg_depend.

Quando una tabella temporanea non è più necessaria, è possibile utilizzare un'istruzione TRUNCATE per svuotarla e liberare spazio. Quindi, esegui manualmente il vacuum delle tabelle pg_attribute e pg_depend. Il vacuum di queste tabelle garantisce che la creazione e il troncamento o l'eliminazione delle tabelle temporanee in modo continuo non comporti l'aggiunta di tuple né contribuisca all'aumento delle dimensioni del sistema.

Puoi evitare questo problema durante la creazione di una tabella temporanea includendo la seguente sintassi che elimina le nuove righe quando il contenuto viene sottoposto a commit:

```
CREATE TEMP TABLE IF NOT EXISTS table_name(table_description) ON COMMIT DELETE ROWS;
```

La clausola `ON COMMIT DELETE ROWS` tronca la tabella temporanea quando la transazione viene sottoposta a commit.

Impedimento dell'aumento delle dimensioni negli indici

Quando si modifica un campo indicizzato in una tabella, l'aggiornamento dell'indice genera una o più tuple inattive in quell'indice. Per impostazione predefinita, il processo autovacuum elimina l'aumento delle dimensioni negli indici, ma tale pulizia richiede una notevole quantità di tempo e risorse. Per specificare le preferenze di pulizia dell'indice quando crei una tabella, includi la clausola vacuum_index_cleanup. Per impostazione predefinita, al momento della creazione della tabella, la clausola è impostata su AUTO, il che significa che il server decide se l'indice deve essere pulito quando esegue il vacuum della tabella. È possibile impostare la clausola su ON per attivare la pulizia dell'indice per una tabella specifica o su OFF per disattivare la pulizia dell'indice per la tabella. Tieni presente che la disattivazione della pulizia dell'indice può far risparmiare tempo, ma può potenzialmente portare a un aumento delle dimensioni dell'indice.

È possibile controllare manualmente la pulizia dell'indice quando esegui il VACUUM di una tabella dalla linea di comando. Per eseguire il vacuum di una tabella e rimuovere le tuple inattive dagli indici, includi la clausola INDEX_CLEANUP con il valore ON e il nome della tabella:

```
acctg=> VACUUM (INDEX_CLEANUP ON) receivables;

INFO: aggressively vacuuming "public.receivables"
VACUUM
```

Per eseguire il vacuum di una tabella senza pulire gli indici, specifica il valore OFF:

```
acctg=> VACUUM (INDEX_CLEANUP OFF) receivables;

INFO: aggressively vacuuming "public.receivables"
VACUUM
```

Migliore gestione della memoria in Aurora PostgreSQL

Aurora PostgreSQL ora include funzionalità avanzate di gestione della memoria per ottimizzare le prestazioni e la resilienza del database con carichi di lavoro diversi. Questi miglioramenti aiutano Aurora PostgreSQL a mantenere disponibilità e reattività costanti, anche durante i periodi di elevata richiesta di memoria.

Questa funzionalità è disponibile e abilitata per impostazione predefinita nelle seguenti versioni di Aurora PostgreSQL per le istanze con provisioning:

- 15.3 e tutte le versioni secondarie successive
- 14.8 e versioni secondarie successive
- 13.11 e versioni secondarie successive
- 12.15 e versioni secondarie successive
- 11.20 e versioni secondarie successive

Questa funzionalità è disponibile e abilitata per impostazione predefinita nelle seguenti versioni di Aurora PostgreSQL per le istanze Aurora Serverless:

- 16.3 e tutte le versioni secondarie successive
- 15.7 e tutte le versioni secondarie successive
- 14.12 e versioni secondarie successive
- 13.15 e versioni secondarie successive

Quando i carichi di lavoro dei clienti utilizzano tutta la memoria libera disponibile, il sistema operativo può riavviare il database per proteggere le risorse, con conseguente indisponibilità temporanea. I nuovi miglioramenti alla gestione della memoria in Aurora PostgreSQL annullano in modo proattivo determinate transazioni quando il sistema subisce un'elevata pressione della memoria, contribuendo a mantenere la stabilità del database.

Le funzionalità principali di una migliore gestione della memoria sono le seguenti:

- Annullamento delle transazioni del database che richiedono più memoria quando il sistema sta raggiungendo un carico critico della memoria.
- Si dice che il sistema sia sottoposto a un carico critico della memoria quando esaurisce tutta la memoria fisica e sta per esaurire lo spazio di scambio. In queste circostanze, qualsiasi transazione che richiede memoria verrà annullata nel tentativo di ridurre immediatamente il carico della memoria nell'istanza database.
- Le utilità di avvio di PostgreSQL essenziali e i worker in background come i worker autovacuum sono sempre protetti.

Gestione dei parametri di gestione della memoria

Per attivare la gestione della memoria

Questa funzionalità è attivata per impostazione predefinita. Viene visualizzato un messaggio di errore, come quello illustrato nell'esempio seguente, quando una transazione viene annullata a causa di memoria insufficiente.

```
ERROR: out of memory Detail: Failed on request of size 16777216.
```

Per disattivare la gestione della memoria

Per disattivare questa funzionalità, connettersi al cluster di database Aurora PostgreSQL con psql e utilizzare l’istruzione SET per i valori dei parametri come indicato di seguito.

> **Note**
>
> È consigliabile mantenere abilitata la gestione della memoria. Questo aiuta a prevenire potenziali errori di memoria insufficiente che possono portare al riavvio del database indotto dal carico di lavoro a causa dell’esaurimento della memoria.

La tabella seguente mostra come disattivare la funzionalità di gestione della memoria per le diverse versioni di Aurora PostgreSQL:

Versioni di Aurora PostgreSQL	Parametro	Default	Comando per disattivare la gestione della memoria a livello di sessione
11.20, 11.21, 12.15, 12.16, 13.11, 13.12, 14.8, 14.9, 15.3, 15.4	`rds.memory_allocation_guard`	`false`	`SET rds.memory_allocation_guard = true;`
12.17, 13.13, 14.10, 15.5 e versioni successive	`rds.enable_memory_management`	`true`	`SET rds.enable_memory_management = false;`

> **Note**
>
> Il parametro `rds.memory_allocation_guard` è obsoleto in Aurora PostgreSQL 12.17, 13.13, 14.10, 15.5 e versioni successive.

Impostando i valori di questi parametri nel gruppo di parametri del cluster di database si impedisce l'annullamento delle query. Per ulteriori informazioni sui gruppi di parametri del cluster di database, consulta Gruppi di parametri per Amazon Aurora.

Limitazione

- Questa funzionalità non è supportata nella classe di istanze db.t3.medium.

Failover rapido con Amazon Aurora PostgreSQL

Di seguito, vengono fornite informazioni su come garantire che il failover si verifichi il più rapidamente possibile. Per ripristinare rapidamente dopo il failover, è possibile utilizzare la gestione della cache del cluster per il cluster di database Aurora PostgreSQL. Per ulteriori informazioni, consulta Ripristino rapido dopo il failover con Cluster Cache Management per Aurora PostgreSQL.

Alcuni dei passaggi da eseguire affinché il failover venga eseguito rapidamente includono i seguenti:

- Impostare i keepalive TCP (Transmission Control Protocol) con intervalli temporali brevi, per interrompere le query in esecuzione più lunghe prima che scada il timeout di lettura in caso di errore.
- Impostare i timeout per la memorizzazione nella cache DNS (Domain Name System) Java in modo aggressivo. Questo garantisce che l'endpoint di sola lettura di Aurora possa scorrere correttamente nodi di sola lettura nei successivi tentativi di connessione.
- Impostare le variabili di timeout usate nella stringa di connessione JDBC più basse possibile. Utilizzare oggetti di connessione separati per query di breve e lunga durata.
- Utilizzare gli endpoint Aurora di lettura e scrittura forniti per connetterti al cluster.
- Utilizzare le operazioni API RDS per testare la risposta dell'applicazione in caso di errori lato server. Inoltre, utilizzare uno strumento di eliminazione pacchetti per testare la risposta dell'applicazione per gli errori lato client.
- Utilizzare il driver JDBC di AWS per sfruttare al massimo le funzionalità di failover di Aurora PostgreSQL. Per ulteriori informazioni sul driver JDBC di AWS e le istruzioni complete per utilizzarlo, consulta il repository GitHub del driver JDBC per Amazon Web Services (AWS).

Questi sono descritti in modo dettagliato di seguito.

Argomenti

- Impostazione dei parametri di keepalive TCP
- Configurare l'applicazione per il Failover rapido
- Verifica del Failover
- Esempio di failover rapido in Java

Impostazione dei parametri di keepalive TCP

Quando si imposta una connessione TCP, ad essa viene associato un set di timer. Quando il timer keepalive raggiunge lo zero, un pacchetto di esplorazione keepalive viene inviato all'endpoint di connessione. Se la sonda riceve una risposta, si può presumere che la connessione sia ancora attiva e in esecuzione.

Attivare i parametri keepalive TCP e impostarli in modo aggressivo garantisce che se il client non è in grado di connettersi al database, le eventuali connessioni attive vengono chiuse rapidamente. L'applicazione può quindi connettersi a un nuovo endpoint.

I seguenti parametri keepalive TCP devono essere impostati:

- `tcp_keepalives_idle` controlla il tempo, in secondi, dopo il quale viene inviato un pacchetto keepalive quando nessun dato è stato inviato dal socket. Le ACK non sono considerate dati. Consigliamo la seguente impostazione:

  ```
  tcp_keepalives_idle = 1
  ```

- `tcp_keepalives_interval` controlla il tempo, in secondi, tra l'invio di successivi pacchetti keepalive dopo l'invio del pacchetto iniziale. Imposta questo tempo utilizzando il parametro `tcp_keepalives_idle`. Consigliamo la seguente impostazione:

  ```
  tcp_keepalives_interval = 1
  ```

- `tcp_keepalives_count` è il numero di esplorazioni keepalive non riconosciute che si verificano prima che l'applicazione venga notificata. Consigliamo la seguente impostazione:

  ```
  tcp_keepalives_count = 5
  ```

Queste impostazioni dovrebbero notificare l'applicazione entro cinque secondi quando il database smette di rispondere. Se i pacchetti keepalive vengono spesso eliminati all'interno della rete dell'applicazione, è possibile impostare un valore `tcp_keepalives_count` più elevato. Questo consente di avere più buffer in reti meno affidabili, sebbene aumenti il tempo necessario per rilevare un errore effettivo.

Per impostare parametri keepalive TCP su Linux

1. Esegui il test della modalità di configurazione dei parametri keepalive TCP.

   A questo scopo, utilizza la riga di comando con i seguenti comandi. Questa è una configurazione consigliata a livello di sistema. In altre parole, influenza anche tutte le altre applicazioni che creano socket con l'opzione `SO_KEEPALIVE` attivata.

   ```
   sudo sysctl net.ipv4.tcp_keepalive_time=1
   sudo sysctl net.ipv4.tcp_keepalive_intvl=1
   sudo sysctl net.ipv4.tcp_keepalive_probes=5
   ```

2. Una volta trovata una configurazione che funziona per l'applicazione, mantenere le impostazioni aggiungendo le seguenti righe a `/etc/sysctl.conf`, comprese le eventuali modifiche apportate:

   ```
   tcp_keepalive_time = 1
   tcp_keepalive_intvl = 1
   ```

```
tcp_keepalive_probes = 5
```

Configurare l'applicazione per il Failover rapido

Di seguito, è disponibile una discussione su diverse modifiche di configurazione per Aurora PostgreSQL che è possibile apportare per un failover rapido. Per ulteriori informazioni sulla configurazione e la configurazione del driver JDBC PostgreSQL, vedere la documentazione PostgreSQL JDBC Driver.

Argomenti

- Ridurre i timeout della cache DNS
- Impostare una stringa di connessione Aurora PostgreSQL per un Failover rapido
- Altre opzioni per ottenere la stringa host

Ridurre i timeout della cache DNS

Quando l'applicazione prova a stabilire una connessione dopo un failover, il nuovo scrittore Aurora PostgreSQL sarà un lettore precedente Per individuarlo, utilizzare l'endpoint di sola lettura Aurora prima della completa propagazione degli aggiornamenti DNS. Impostare Time to Live (TTL) java DNS su un valore basso, ad esempio meno di 30 secondi, consente di effettuare il ciclo tra i nodi del lettore nei successivi tentativi di connessione.

```
// Sets internal TTL to match the Aurora RO Endpoint TTL
java.security.Security.setProperty("networkaddress.cache.ttl" , "1");
// If the lookup fails, default to something like small to retry
java.security.Security.setProperty("networkaddress.cache.negative.ttl" , "3");
```

Impostare una stringa di connessione Aurora PostgreSQL per un Failover rapido

Per usare il failover rapido di Aurora PostgreSQL, accertarsi che la stringa di connessione dell'applicazione contenga un elenco di host anziché un singolo host. Di seguito è riportata una stringa di connessione di esempio che può essere utilizzata per connettersi a un cluster Aurora PostgreSQL: In questo esempio, gli host sono in grassetto.

```
jdbc:postgresql://myauroracluster.cluster-c9bfei4hjlrd.us-east-1-
beta.rds.amazonaws.com:5432,
myauroracluster.cluster-ro-c9bfei4hjlrd.us-east-1-beta.rds.amazonaws.com:5432
/postgres?user=<primaryuser>&password=<primarypw>&loginTimeout=2
```

```
&connectTimeout=2&cancelSignalTimeout=2&socketTimeout=60
&tcpKeepAlive=true&targetServerType=primary
```

Per una migliore disponibilità e per evitare una dipendenza dall'API RDS, è opportuno mantenere un file con cui eseguire la connessione. Questo file contiene una stringa host che viene letta dall'applicazione quando si stabilisce una connessione al database. Questa stringa host contiene tutti gli endpoint Aurora disponibili per il cluster. Per altre informazioni sugli endpoint Aurora, consultare Connessioni degli endpoint Amazon Aurora.

Ad esempio, potrebbe essere necessario archiviare gli endpoint in un file locale come mostrato di seguito.

```
myauroracluster.cluster-c9bfei4hjlrd.us-east-1-beta.rds.amazonaws.com:5432,
myauroracluster.cluster-ro-c9bfei4hjlrd.us-east-1-beta.rds.amazonaws.com:5432
```

L'applicazione legge da questo file per compilare la sezione host della stringa di connessione JDBC. Rinominare il cluster DB fa sì che questi endpoint siano modificati. Assicurati che l'applicazione gestisca questo evento nel caso in cui si verifichi.

Un'altra opzione è usare un elenco di nodi di istanze database, come illustrato di seguito.

```
my-node1.cksc6xlmwcyw.us-east-1-beta.rds.amazonaws.com:5432,
my-node2.cksc6xlmwcyw.us-east-1-beta.rds.amazonaws.com:5432,
my-node3.cksc6xlmwcyw.us-east-1-beta.rds.amazonaws.com:5432,
my-node4.cksc6xlmwcyw.us-east-1-beta.rds.amazonaws.com:5432
```

Il vantaggio di questo approccio è che il driver di connessione PostgreSQL JDBC esegue un ciclo di tutti i nodi in questo elenco per individuare una connessione valida. Al contrario, quando si usano gli endpoint Aurora, vengono provati solo due nodi per ogni tentativo di connessione. Tuttavia, l'utilizzo dei nodi di istanza database presenta uno svantaggio. Se si aggiungono o si rimuovono nodi dal cluster e l'elenco degli endpoint di istanza diventa obsoleto, il driver di connessione potrebbe non trovare mai l'host corretto cui connettersi.

Per garantire che l'applicazione non rimanga in attesa troppo a lungo prima di connettersi a un host, imposta i seguenti parametri in modo aggressivo.

- `targetServerType`: controlla se il driver si connette a un nodo di scrittura o lettura. Per garantire che le applicazioni si riconnetteranno solo a un nodo di scrittura, imposta il valore `targetServerType` su `primary`.

I valori per il parametro `targetServerType` includono `primary`, `secondary`, `any` e `preferSecondary`. Il valore `preferSecondary` esegue prima un tentativo per stabilire una connessione a un lettore. Se non è possibile stabilire una connessione con il lettore, si connette allo scrittore.

- `loginTimeout`: controlla per quanto tempo l'applicazione rimane in attesa prima di accedere al database dopo che è stata stabilita una connessione socket.
- `connectTimeout` – Controlla quanto tempo il connettore attende per stabilire una connessione con il database.

In base al livello di aggressività desiderato per l'applicazione, puoi modificare altri parametri dell'applicazione per accelerare il processo di connessione:

- `cancelSignalTimeout`: in alcune applicazioni, potrebbe essere necessario inviare un segnale di annullamento "massimo sforzo" su una query scaduta. Se questo segnale di annullamento si trova nel percorso di failover, valuta se impostarlo in modo aggressivo per evitare che venga inviato a un host morto.
- `socketTimeout` – Questo parametro controlla quanto tempo il connettore attende per le operazioni di lettura. Questo parametro può essere usato come un "timeout di query" globale per assicurare che nessuna query attenda più a lungo di questo valore. Una buona pratica è quella di disporre di due gestori di connessione. Un gestore di connessione che esegue query di breve durata e imposta questo valore più basso. Un altro gestore di connessione, per query a esecuzione prolungata, con questo valore impostato molto più alto. Con questo approccio, è possibile fare affidamento sui parametri keepalive TCP per interrompere le query a esecuzioni prolungata se il server non funziona.
- `tcpKeepAlive`: attivare questo parametro per garantire che i parametri keepalive TCP impostati siano rispettati.
- `loadBalanceHosts` – Quando è impostato su `true`, l'applicazione del parametro si connette a un host casuale scelto da un elenco di host candidati.

Altre opzioni per ottenere la stringa host

È possibile ottenere la stringa host da diverse fonti, compresa la funzione `aurora_replica_status` e usando l'API di Amazon RDS.

In molti casi, occorre determinare chi è lo scrittore del cluster o trovare altri nodi di lettura nel cluster. A questo scopo, l'applicazione può connettersi a qualsiasi istanza database nel cluster database ed eseguire la query della funzione `aurora_replica_status`. Questa funzione può essere utilizzata per ridurre il tempo necessario per trovare un host cui connettersi. Tuttavia, in determinati scenari di errore di rete la funzione `aurora_replica_status` potrebbe mostrare informazioni obsolete o incomplete.

Un buon modo per assicurarsi che l'applicazione possa individuare un nodo cui connettersi è provare a connettersi all'endpoint di scrittura del cluster e quindi all'endpoint di lettura del cluster, fino a quando non è possibile stabilire una connessione leggibile. Questi endpoint non cambiano a meno che il cluster database non venga rinominato. Quindi, in genere possono essere lasciati come membri statici dell'applicazione o archiviati in un file di risorsa che viene letto dall'applicazione.

Una volta stabilita una connessione usando uno di questi endpoint, è possibile ottenere informazioni sul resto del cluster. A questo scopo, richiama la funzione `aurora_replica_status`. Ad esempio, il comando seguente recupera informazioni con `aurora_replica_status`.

```
postgres=> SELECT server_id, session_id, highest_lsn_rcvd, cur_replay_latency_in_usec,
 now(), last_update_timestamp
FROM aurora_replica_status();

server_id | session_id | highest_lsn_rcvd | cur_replay_latency_in_usec | now |
 last_update_timestamp
-----------+--------------------------------------+-----------------
+---------------------------+-------------------------------+-----------------------
mynode-1 | 3e3c5044-02e2-11e7-b70d-95172646d6ca | 594221001 | 201421 | 2017-03-07
 19:50:24.695322+00 | 2017-03-07 19:50:23+00
mynode-2 | 1efdd188-02e4-11e7-becd-f12d7c88a28a | 594221001 | 201350 | 2017-03-07
 19:50:24.695322+00 | 2017-03-07 19:50:23+00
mynode-3 | MASTER_SESSION_ID | | | 2017-03-07 19:50:24.695322+00 | 2017-03-07
 19:50:23+00
(3 rows)
```

Ad esempio, la sezione host della stringa di connessione potrebbe iniziare con entrambi gli endpoint del cluster di scrittura e di lettura, come mostrato di seguito.

```
myauroracluster.cluster-c9bfei4hjlrd.us-east-1-beta.rds.amazonaws.com:5432,
myauroracluster.cluster-ro-c9bfei4hjlrd.us-east-1-beta.rds.amazonaws.com:5432
```

In questo scenario, l'applicazione prova a stabilire una connessione a qualsiasi tipo di nodo, primario o secondario. Una volta connessa, una buona prassi è quella di esaminare innanzitutto lo stato di lettura-scrittura del nodo. A questo scopo, occorre eseguire la query del risultato del comando `SHOW transaction_read_only`.

Se il valore restituito della query è `OFF`, allora la connessione al nodo primario è andata a buon fine. Tuttavia, si supponga che il valore restituito sia `ON` e che l'applicazione richieda una connessione di lettura/scrittura. In questo caso, si può richiamare la funzione `aurora_replica_status` per determinare il `server_id` che ha `session_id='MASTER_SESSION_ID'`. Questa funzione fornisce il nome del nodo primario. Ciò è possibile il `endpointPostfix` descritto di seguito.

È importante sapere quando si esegue una connessione a una replica con dati obsoleti. Quando ciò avviene, la funzione `aurora_replica_status` potrebbe mostrare informazioni aggiornate. Una soglia di obsolescenza può essere impostata a livello di applicazione. Per la verifica, osservare la differenza tra l'ora del server e il valore `last_update_timestamp`. In generale, l'applicazione dovrebbe evitare di alternarsi tra due host a causa delle informazioni in conflitto restituite dalla funzione `aurora_replica_status`. L'applicazione dovrebbe provare prima tutti gli host conosciuti anziché seguire i dati restituiti da `aurora_replica_status`.

Elencare istanze utilizzando l'operazione API DescribeDBClusters, esempio in Java

L'elenco delle istanze può essere ricavato in maniera programmatica utilizzando l'AWS SDK per Java, in particolare l'operazione API DescribeDBClusters.

Di seguito è riportato un piccolo esempio di come eseguire questa operazione in Java 8:

```
AmazonRDS client = AmazonRDSClientBuilder.defaultClient();
DescribeDBClustersRequest request = new DescribeDBClustersRequest()
   .withDBClusterIdentifier(clusterName);
DescribeDBClustersResult result =
rdsClient.describeDBClusters(request);

DBCluster singleClusterResult = result.getDBClusters().get(0);

String pgJDBCEndpointStr =
singleClusterResult.getDBClusterMembers().stream()
   .sorted(Comparator.comparing(DBClusterMember::getIsClusterWriter)
   .reversed()) // This puts the writer at the front of the list
   .map(m -> m.getDBInstanceIdentifier() + endpointPostfix + ":" +
 singleClusterResult.getPort()))
   .collect(Collectors.joining(","));
```

Qui, `pgJDBCEndpointStr` contiene un elenco formattato di endpoint, come mostrato di seguito.

```
my-node1.cksc6xlmwcyw.us-east-1-beta.rds.amazonaws.com:5432,
my-node2.cksc6xlmwcyw.us-east-1-beta.rds.amazonaws.com:5432
```

La variabile `endpointPostfix` può essere una costante impostata dall'applicazione. Oppure, può essere ottenuta dall'applicazione eseguendo la query dell'operazione API `DescribeDBInstances` per una singola istanza del cluster. Questo valore rimane costante all'interno di Regione AWS e per un singolo cliente. Pertanto, salva una chiamata API per mantenere semplicemente questa costante in un file di risorsa letto dall'applicazione. Nell'esempio precedente, è impostato sul valore seguente.

```
.cksc6xlmwcyw.us-east-1-beta.rds.amazonaws.com
```

Ai fini della disponibilità, una buona norma è quella di usare per impostazione predefinita gli endpoint Aurora del cluster database se l'API non risponde o impiega troppo tempo a rispondere. Viene garantito l'aggiornamento degli endpoint nel tempo necessario per aggiornare il record DNS. L'aggiornamento del record DNS con un endpoint richiede in genere meno di 30 secondi. L'endpoint può essere memorizzato in un file di risorse utilizzato dall'applicazione.

Verifica del Failover

In tutti i casi è necessario disporre di un cluster di database contenente due o più istanze database.

Dal lato server, alcune operazioni API possono causare un'interruzione che può essere usata per testare la risposta delle applicazioni:

- FailoverDBCluster: questa operazione tenta di promuovere una nuova istanza database nel cluster database per la scrittura

  Il seguente esempio di codice mostra come è possibile utilizzare `failoverDBCluster` per causare un'interruzione. Per ulteriori informazioni sulla configurazione di un client Amazon RDS, consulta Utilizzo di AWS SDK per Java.

  ```
  public void causeFailover() {

      final AmazonRDS rdsClient = AmazonRDSClientBuilder.defaultClient();

      FailoverDBClusterRequest request = new FailoverDBClusterRequest();
      request.setDBClusterIdentifier("cluster-identifier");
  ```

```
    rdsClient.failoverDBCluster(request);
}
```

- RebootDBInstance: il failover non è garantito con questa operazione API. Tuttavia, arresta il database dello scrittore. Può essere utilizzata per testare la risposta dell'applicazione all'eliminazione delle connessioni. Il parametro `ForceFailover` non si applica ai motori Aurora. Utilizzare invece l'operazione API `FailoverDBCluster`.
- ModifyDBCluster: la modifica del parametro `Port` causa un'interruzione quando i nodi del cluster iniziano ad ascoltare su una nuova porta. In generale, l'applicazione può rispondere innanzitutto a questo errore assicurandosi che solo l'applicazione controlli le modifiche alle porte. Inoltre, accertarsi che possa aggiornare in modo appropriato gli endpoint da cui dipende. A questo scopo, è necessario che qualcuno aggiorni manualmente la porta quando vengono apportate modifiche a livello di API. Oppure, è possibile utilizzare l'API RDS nell'applicazione per determinare se la porta è cambiata.
- ModifyDBInstance: la modifica del parametro `DBInstanceClass`causa un'interruzione.
- DeleteDBInstance: l'eliminazione del primario (scrittore) causa la promozione di una nuova istanza database a scrittore nel cluster database.

Dal lato applicazione o client, se si usa Linux, è possibile testare in che modo l'applicazione risponde a improvvise eliminazioni di pacchetti. Ciò può essere fatto a seconda che porta, host o se i pacchetti keepalive TCP vengono inviati o ricevuti utilizzando il comando iptables.

Esempio di failover rapido in Java

Il seguente esempio di codice mostra come un'applicazione potrebbe configurare un gestore di driver di Aurora PostgreSQL.

L'applicazione richiama la funzione `getConnection` quando è richiesta una connessione. Una chiamata a `getConnection` può non riuscire a trovare un host valido. Un esempio è quando non viene trovato alcuno scrittore ma il parametro `targetServerType` è impostato su `primary`. In questo caso, l'applicazione chiamante deve semplicemente riprovare a richiamare la funzione.

Per evitare di spingere il comportamento dei tentativi continui nell'applicazione, puoi impacchettare questo tentativo di chiamata in un pool di connessioni. Con la maggior parte dei pool di connessioni, puoi specificare una stringa di connessione JDBC. Pertanto l'applicazione può richiamare `getJdbcConnectionString` e passarla al pool di connessioni. Ciò significa che con Aurora PostgreSQL si può utilizzare un failover più rapido.

```
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.SQLException;
import java.sql.Statement;
import java.util.ArrayList;
import java.util.List;
import java.util.stream.Collectors;
import java.util.stream.IntStream;

import org.joda.time.Duration;

public class FastFailoverDriverManager {
   private static Duration LOGIN_TIMEOUT = Duration.standardSeconds(2);
   private static Duration CONNECT_TIMEOUT = Duration.standardSeconds(2);
   private static Duration CANCEL_SIGNAL_TIMEOUT = Duration.standardSeconds(1);
   private static Duration DEFAULT_SOCKET_TIMEOUT = Duration.standardSeconds(5);

   public FastFailoverDriverManager() {
       try {
            Class.forName("org.postgresql.Driver");
       } catch (ClassNotFoundException e) {
            e.printStackTrace();
       }

       /*
         * RO endpoint has a TTL of 1s, we should honor that here. Setting this
 aggressively makes sure that when
         * the PG JDBC driver creates a new connection, it will resolve a new different
 RO endpoint on subsequent attempts
         * (assuming there is > 1 read node in your cluster)
         */
        java.security.Security.setProperty("networkaddress.cache.ttl" , "1");
       // If the lookup fails, default to something like small to retry
       java.security.Security.setProperty("networkaddress.cache.negative.ttl" , "3");
   }

   public Connection getConnection(String targetServerType) throws SQLException {
       return getConnection(targetServerType, DEFAULT_SOCKET_TIMEOUT);
   }

   public Connection getConnection(String targetServerType, Duration queryTimeout)
 throws SQLException {
```

```
       Connection conn =
DriverManager.getConnection(getJdbcConnectionString(targetServerType, queryTimeout));

      /*
        * A good practice is to set socket and statement timeout to be the same thing
since both
        * the client AND server will stop the query at the same time, leaving no
running queries
        * on the backend
        */
       Statement st = conn.createStatement();
       st.execute("set statement_timeout to " + queryTimeout.getMillis());
       st.close();

      return conn;
  }

  private static String urlFormat = "jdbc:postgresql://%s"
          + "/postgres"
          + "?user=%s"
          + "&password=%s"
          + "&loginTimeout=%d"
          + "&connectTimeout=%d"
          + "&cancelSignalTimeout=%d"
          + "&socketTimeout=%d"
          + "&targetServerType=%s"
          + "&tcpKeepAlive=true"
          + "&ssl=true"
          + "&loadBalanceHosts=true";
  public String getJdbcConnectionString(String targetServerType, Duration
queryTimeout) {
      return String.format(urlFormat,
                getFormattedEndpointList(getLocalEndpointList()),
                CredentialManager.getUsername(),
                CredentialManager.getPassword(),
                LOGIN_TIMEOUT.getStandardSeconds(),
                CONNECT_TIMEOUT.getStandardSeconds(),
                CANCEL_SIGNAL_TIMEOUT.getStandardSeconds(),
                queryTimeout.getStandardSeconds(),
                targetServerType
      );
  }

  private List<String> getLocalEndpointList() {
```

```
        /*
          * As mentioned in the best practices doc, a good idea is to read a local
 resource file and parse the cluster endpoints.
          * For illustration purposes, the endpoint list is hardcoded here
          */
         List<String> newEndpointList = new ArrayList<>();
         newEndpointList.add("myauroracluster.cluster-c9bfei4hjlrd.us-east-1-
beta.rds.amazonaws.com:5432");
         newEndpointList.add("myauroracluster.cluster-ro-c9bfei4hjlrd.us-east-1-
beta.rds.amazonaws.com:5432");

        return newEndpointList;
    }

    private static String getFormattedEndpointList(List<String> endpoints) {
        return IntStream.range(0, endpoints.size())
                .mapToObj(i -> endpoints.get(i).toString())
                .collect(Collectors.joining(","));
    }
}
```

Ripristino rapido dopo il failover con Cluster Cache Management per Aurora PostgreSQL

Per il ripristino rapido dell'istanza database di scrittura nei cluster Aurora PostgreSQL in caso di failover, utilizzare la gestione della cache del cluster per Amazon Aurora PostgreSQL. La gestione della cache del cluster garantisce il mantenimento delle prestazioni dell'applicazione in caso di failover.

In una tipica situazione di failover, è possibile che si verifichi un degrado delle prestazioni temporaneo ma di grandi dimensioni dopo il failover. Questo degrado si verifica perché all'avvio dell'istanza database di failover, la cache del buffer è vuota. Una cache vuota è anche nota col nome di cache fredda. Una cache fredda degrada le prestazioni perché l'istanza database deve leggere dal disco a una velocità minore, invece di sfruttare i valori memorizzati nella cache del buffer.

Con la gestione della cache del cluster, si imposta una specifica istanza database di lettura come destinazione del failover. La gestione della cache del cluster garantisce che i dati nella cache dell'istanza di lettura designata siano mantenuti sincronizzati con i dati nella cache dell'istanza database di scrittura. La cache dell'istanza di lettura designata pre-popolata con i valori prende il nome di cache calda. Se si verifica un failover, l'istanza di lettura designata utilizza immediatamente

i valori presenti nella sua cache calda nel momento in cui viene promossa al ruolo di nuova istanza database di scrittura. Questo approccio garantisce all'applicazione prestazioni di recupero decisamente migliori.

La gestione della cache del cluster richiede che l'istanza del lettore designata disponga dello stesso tipo di classe di istanza e dimensione (`db.r5.2xlarge` o `db.r5.xlarge`, ad esempio) di scrittura. Tieni presente questo aspetto quando crei cluster database Aurora PostgreSQL in modo che il cluster possa essere ripristinato durante un failover. Per un elenco dei tipi e delle dimensioni delle classi di istanza, consulta Specifiche hardware per le classi di istanza database per Aurora.

> **Note**
>
> La gestione della cache del cluster non è supportata per i cluster di database Aurora PostgreSQL secondari che fanno parte dei database globali Aurora. Per i cluster primari in cui questa funzionalità è supportata, si consiglia di non eseguire alcun carico di lavoro sull'istanza di lettura di livello 0 designata.

Indice

- Configurazione della gestione della cache del cluster
  - Abilitazione della gestione della cache del cluster
  - Impostazione della priorità del livello di promozione per l'istanza database di scrittura
  - Impostazione della priorità del livello di promozione per un'istanza database di lettura
- Monitoraggio della cache
- Risoluzione degli errori di configurazione CCM

Configurazione della gestione della cache del cluster

Per configurare la gestione della cache del cluster, eseguire i processi seguenti in ordine.

Argomenti

- Abilitazione della gestione della cache del cluster
- Impostazione della priorità del livello di promozione per l'istanza database di scrittura
- Impostazione della priorità del livello di promozione per un'istanza database di lettura

> **Note**
>
> Dopo aver completato queste fasi, attendi almeno 1 minuto affinché la gestione della cache del cluster sia completamente operativa.

Abilitazione della gestione della cache del cluster

Per abilitare la gestione della cache del cluster, attenersi alla procedura descritta di seguito.

Console

Per abilitare la gestione della cache del cluster

1. Accedi alla Console di gestione AWS e apri la console Amazon RDS all'indirizzo https://console.aws.amazon.com/rds/.
2. Nel pannello di navigazione, scegli Parameter groups (Gruppi di parametri).
3. Nell'elenco, selezionare il gruppo di parametri per il cluster di database di Aurora PostgreSQL.

   Il cluster di database deve utilizzare un gruppo di parametri diverso da quello predefinito, poiché non è possibile modificare i valori in un gruppo di parametri predefinito.
4. Per Parameter group actions (Operazioni del gruppo di parametri), scegliere Edit (Modifica).
5. Impostare il valore del parametro del cluster `apg_ccm_enabled` su 1.
6. Scegli Save changes (Salva modifiche).

AWS CLI

Per abilitare la gestione della cache del cluster per un cluster database di Aurora PostgreSQL, utilizzare il comando AWS CLI della `modify-db-cluster-parameter-group` con i seguenti parametri obbligatori:

- `--db-cluster-parameter-group-name`
- `--parameters`

Example

Per Linux, macOS o Unix:

```
aws rds modify-db-cluster-parameter-group \
    --db-cluster-parameter-group-name my-db-cluster-parameter-group \
    --parameters "ParameterName=apg_ccm_enabled,ParameterValue=1,ApplyMethod=immediate"
```

Per Windows:

```
aws rds modify-db-cluster-parameter-group ^
    --db-cluster-parameter-group-name my-db-cluster-parameter-group ^
    --parameters "ParameterName=apg_ccm_enabled,ParameterValue=1,ApplyMethod=immediate"
```

Impostazione della priorità del livello di promozione per l'istanza database di scrittura

Per la gestione della cache del cluster, assicurarsi che la priorità della promozione sia tier-0 per l'istanza database di scrittura del cluster di database di Aurora PostgreSQL. Il livello della priorità di promozione è un valore che specifica l'ordine in cui un'istanza di lettura di Aurora viene promossa al ruolo di istanza database di scrittura dopo un errore. I valori validi sono compresi nell'intervallo tra 0 e 15, dove 0 è la priorità massima e 15 è la priorità minima. Per ulteriori informazioni sul livello di promozione, consultare Tolleranza ai guasti di un cluster DB Aurora.

Console

Per impostare la priorità di promozione per l'istanza database di scrittura su tier-0

1. Accedi alla Console di gestione AWS e apri la console Amazon RDS all'indirizzo https://console.aws.amazon.com/rds/.
2. Nel pannello di navigazione, scegliere Databases (Database).
3. Scegliere l'istanza database Writer (di scrittura) del cluster di database di Aurora PostgreSQL.
4. Scegliere Modify (Modifica). Viene visualizzata la pagina Modify DB Instance (Modifica istanza database).
5. Nel pannello Additional configuration (Configurazione aggiuntiva) scegliere tier-0 (livello 0) per la Failover priority (Priorità failover).
6. Scegliere Continue (Continua) e controllare il riepilogo delle modifiche.
7. Per applicare immediatamente le modifiche dopo il salvataggio, scegliere Apply immediately (Applica immediatamente).
8. Scegliere Modify DB Instance (Modifica istanza database) per salvare le modifiche.

AWS CLI

Per impostare la priorità del livello di promozione su 0 per l'istanza database di scrittura utilizzando la AWS CLI, invocare il comando modify-db-instance con i seguenti parametri obbligatori:

- `--db-instance-identifier`
- `--promotion-tier`
- `--apply-immediately`

Example

Per Linux, macOS o Unix:

```
aws rds modify-db-instance \
    --db-instance-identifier writer-db-instance \
    --promotion-tier 0 \
    --apply-immediately
```

Per Windows:

```
aws rds modify-db-instance ^
    --db-instance-identifier writer-db-instance ^
    ---promotion-tier 0  ^
    --apply-immediately
```

Impostazione della priorità del livello di promozione per un'istanza database di lettura

È necessario impostare una sola istanza database di lettura per la gestione della cache del cluster. Per farlo, scegli un lettore dal cluster Aurora PostgreSQL che sia la stessa classe di istanza e che abbia le stesse dimensioni dell'istanza database di scrittura. Ad esempio, se lo scrittore utilizza `db.r5.xlarge`, scegliere un lettore che utilizzi lo stesso tipo di classe di istanza e la stessa dimensione. Quindi impostare il livello della priorità di promozione su 0.

Il livello della priorità di promozione è un valore che specifica l'ordine in cui un'istanza di lettura di Aurora viene promossa al ruolo di istanza database di scrittura dopo un errore. I valori validi sono compresi nell'intervallo tra 0 e 15, dove 0 è la priorità massima e 15 è la priorità minima.

Console

Per impostare la priorità di promozione per l'istanza database di lettura su tier-0

1. Accedi alla Console di gestione AWS e apri la console Amazon RDS all'indirizzo https://console.aws.amazon.com/rds/.
2. Nel pannello di navigazione, scegliere Databases (Database).
3. Scegliere l'istanza database Reader (Di lettura) del cluster di database di Aurora PostgreSQL che sia della stessa classe di istanza dell'istanza database di scrittura.
4. Scegliere Modify (Modifica). Viene visualizzata la pagina Modify DB Instance (Modifica istanza database).
5. Nel pannello Additional configuration (Configurazione aggiuntiva) scegliere tier-0 (livello 0) per la Failover priority (Priorità failover).
6. Scegliere Continue (Continua) e controllare il riepilogo delle modifiche.
7. Per applicare immediatamente le modifiche dopo il salvataggio, scegliere Apply immediately (Applica immediatamente).
8. Scegliere Modify DB Instance (Modifica istanza database) per salvare le modifiche.

AWS CLI

Per impostare la priorità del livello di promozione su 0 per l'istanza database di lettura utilizzando la AWS CLI, invocare il comando modify-db-instance con i seguenti parametri obbligatori:

- `--db-instance-identifier`
- `--promotion-tier`
- `--apply-immediately`

Example

Per Linux, macOS o Unix:

```
aws rds modify-db-instance \
    --db-instance-identifier reader-db-instance \
    --promotion-tier 0 \
    --apply-immediately
```

Per Windows:

```
aws rds modify-db-instance ^
    --db-instance-identifier reader-db-instance ^
    ---promotion-tier 0  ^
    --apply-immediately
```

Monitoraggio della cache

Dopo aver configurato la gestione della cache del cluster, è possibile monitorare lo stato di sincronizzazione tra la cache del buffer dell'istanza database di scrittura e la cache del buffer calda di scrittura designata. Per esaminare il contenuto della cache del buffer sia sull'istanza database di scrittura sia sull'istanza database di lettura designata, utilizza il modulo PostgreSQL `pg_buffercache`. Per ulteriori informazioni, consulta la documentazione di PostgreSQL `pg_buffercache`.

Utilizzo della funzione **`aurora_ccm_status`**

La gestione della cache del cluster mette a disposizione anche la funzione `aurora_ccm_status`. Utilizza la funzione `aurora_ccm_status` sull'istanza database di scrittura per ottenere le seguenti informazioni sull'avanzamento del popolamento della cache sull'istanza database di lettura designata:

- `buffers_sent_last_minute` – Quanti buffer sono stati inviati all'istanza database di lettura designata nell'ultimo minuto.
- `buffers_found_last_minute`: il numero di buffer con accesso frequente identificati durante l'ultimo minuto.
- `buffers_sent_last_scan` – Quanti buffer sono stati inviati all'istanza database di lettura designata durante l'ultima scansione completa della cache.
- `buffers_found_last_scan` – Quanti buffer sono stati identificati come soggetti ad accesso frequente ed è stato necessario inviare durante l'ultima scansione completa della cache. I buffer già memorizzati sull'istanza database di lettura designata non vengono inviati.
- `buffers_sent_current_scan` – Quanti buffer sono stati inviati finora durante la scansione corrente.
- `buffers_found_current_scan` – Quanti buffer sono stati identificati come soggetti ad accesso frequente nella scansione corrente.
- `current_scan_progress` – Quanti buffer sono stati analizzati finora durante la scansione corrente.

L'esempio seguente mostra come utilizzare la funzione `aurora_ccm_status` per convertire parte del suo output in velocità e percentuale di popolamento della cache calda.

```
SELECT buffers_sent_last_minute*8/60 AS warm_rate_kbps,
   100*(1.0-buffers_sent_last_scan::float/buffers_found_last_scan) AS warm_percent
   FROM aurora_ccm_status();
```

Risoluzione degli errori di configurazione CCM

Quando si abilita il parametro `apg_ccm_enabled` del cluster, la gestione della cache del cluster viene attivata automaticamente a livello di istanza sull'istanza database di scrittura e su una sola istanza database di lettura sul cluster di database Aurora PostgreSQL. L'istanza di scrittura e quella di lettura devono utilizzare lo stesso tipo e dimensione della classe di istanza. La priorità del livello di promozione è impostata su `0`. Le altre istanze di lettura nel cluster di database devono avere un livello di promozione diverso da zero e la gestione della cache del cluster è disabilitata per tali istanze.

I seguenti motivi possono causare problemi nella configurazione e disabilitare la gestione della cache del cluster:

- Quando non esiste una singola istanza database di lettura impostata sul livello di promozione 0.
- Quando l'istanza database di scrittura non è impostata sul livello di promozione 0.
- Quando più di un'istanza database di lettura è impostata sul livello di promozione 0.
- Quando le istanze database di scrittura e un'istanza database di lettura con livello di promozione 0 non hanno le stesse dimensioni.

Gestione dell'abbandono della connessione Aurora PostgreSQL con pooling

In caso di connessione e disconnessione frequente delle applicazioni client, è possibile che il tempo di risposta del cluster database Aurora PostgreSQL rallenti. In questo caso si dice che nel cluster si verifica un abbandono della connessione. Ogni nuova connessione all'endpoint del cluster database Aurora PostgreSQL consuma risorse, riducendo pertanto quelle che possono essere utilizzate per elaborare il carico di lavoro effettivo. È consigliabile gestire il problema dell'abbandono della connessione seguendo alcune delle best practice descritte di seguito.

Per prima cosa, i tempi di risposta sui cluster database Aurora PostgreSQL che hanno frequenze elevate di abbandono della connessione possono essere migliorati. A questo scopo, puoi utilizzare un pool di connessioni come RDS Proxy. Un pool di connessioni fornisce una cache di connessioni

pronte all'uso per i client. Quasi tutte le versioni di Aurora PostgreSQL supportano RDS Proxy. Per ulteriori informazioni, consulta Amazon RDS Proxy con Aurora PostgreSQL.

Se la versione specifica di Aurora PostgreSQL non supporta RDS Proxy, puoi utilizzare un altro pool di connessioni compatibile con PostgreSQL, ad esempio PGBouncer. Per ulteriori informazioni, consulta il sito Web PgBouncer.

Per verificare se il cluster database Aurora PostgreSQL può trarne vantaggio dal pooling delle connessioni, puoi controllare il file `postgresql.log` per verificare la presenza di connessioni e disconnessioni. Puoi anche utilizzare Performance Insights per individuare il numero di abbandoni della connessione che si verificano nel cluster database Aurora PostgreSQL. Di seguito sono disponibili informazioni su entrambi gli argomenti.

Registrazione di connessioni e disconnessioni

I parametri `log_connections` e `log_disconnections` PostgreSQL possono acquisire connessioni e disconnessioni all'istanza di scrittura del cluster database Aurora PostgreSQL. Per impostazione predefinita, questi parametro sono disattivati. Per attivare questi parametri, utilizza un gruppo di parametri personalizzati e attivali modificando il valore in 1. Per ulteriori informazioni sui gruppi di parametri, consulta Gruppi di parametri del cluster di database per i cluster di database Amazon Aurora. Per verificare le impostazioni, esegui la connessione all'endpoint del cluster database per Aurora PostgreSQL utilizzando psql ed esegui le query come segue.

```
labdb=> SELECT setting FROM pg_settings
  WHERE name = 'log_connections';
 setting
---------
on
(1 row)
labdb=> SELECT setting FROM pg_settings
  WHERE name = 'log_disconnections';
setting
---------
on
(1 row)
```

Con entrambi questi parametri attivati, il registro acquisisce tutte le nuove connessioni e disconnessioni. Per ogni nuova connessione autorizzata, vengono visualizzati utente e database. Al momento della disconnessione, viene registrata anche la durata della sessione, come mostrato nell'esempio seguente.

```
2022-03-07 21:44:53.978 UTC [16641] LOG: connection authorized: user=labtek
 database=labdb application_name=psql
2022-03-07 21:44:55.718 UTC [16641] LOG: disconnection: session time: 0:00:01.740
 user=labtek database=labdb host=[local]
```

Per verificare l'abbandono della connessione nell'applicazione, attiva questi parametri se non è già stato fatto. Quindi raccogli i dati nel registro PostgreSQL per l'analisi eseguendo l'applicazione con un carico di lavoro e un periodo di tempo realistici. Puoi visualizzare i file di registro nella console RDS. Scegli l'istanza di scrittura del cluster database Aurora PostgreSQL, quindi seleziona la scheda Logs & events (Log ed eventi). Per ulteriori informazioni, consulta Visualizzazione ed elenco dei file di log del database.

Oppure, puoi scaricare il file di registro dalla console e utilizzare la seguente sequenza di comandi. Questa sequenza individua il numero totale di connessioni autorizzate e interrotte al minuto.

```
grep "connection authorized\|disconnection: session time:"
 postgresql.log.2022-03-21-16|\
awk {'print $1,$2}' |\
sort |\
uniq -c |\
sort -n -k1
```

Nell'output di esempio, è possibile visualizzare un picco nelle connessioni autorizzate seguito da disconnessioni a partire dalle 16:12:10.

```
.....
,.......
.........
5 2022-03-21 16:11:55 connection authorized:
9 2022-03-21 16:11:55 disconnection: session
5 2022-03-21 16:11:56 connection authorized:
5 2022-03-21 16:11:57 connection authorized:
5 2022-03-21 16:11:57 disconnection: session
32 2022-03-21 16:12:10 connection authorized:
30 2022-03-21 16:12:10 disconnection: session
31 2022-03-21 16:12:11 connection authorized:
27 2022-03-21 16:12:11 disconnection: session
27 2022-03-21 16:12:12 connection authorized:
27 2022-03-21 16:12:12 disconnection: session
41 2022-03-21 16:12:13 connection authorized:
47 2022-03-21 16:12:13 disconnection: session
```

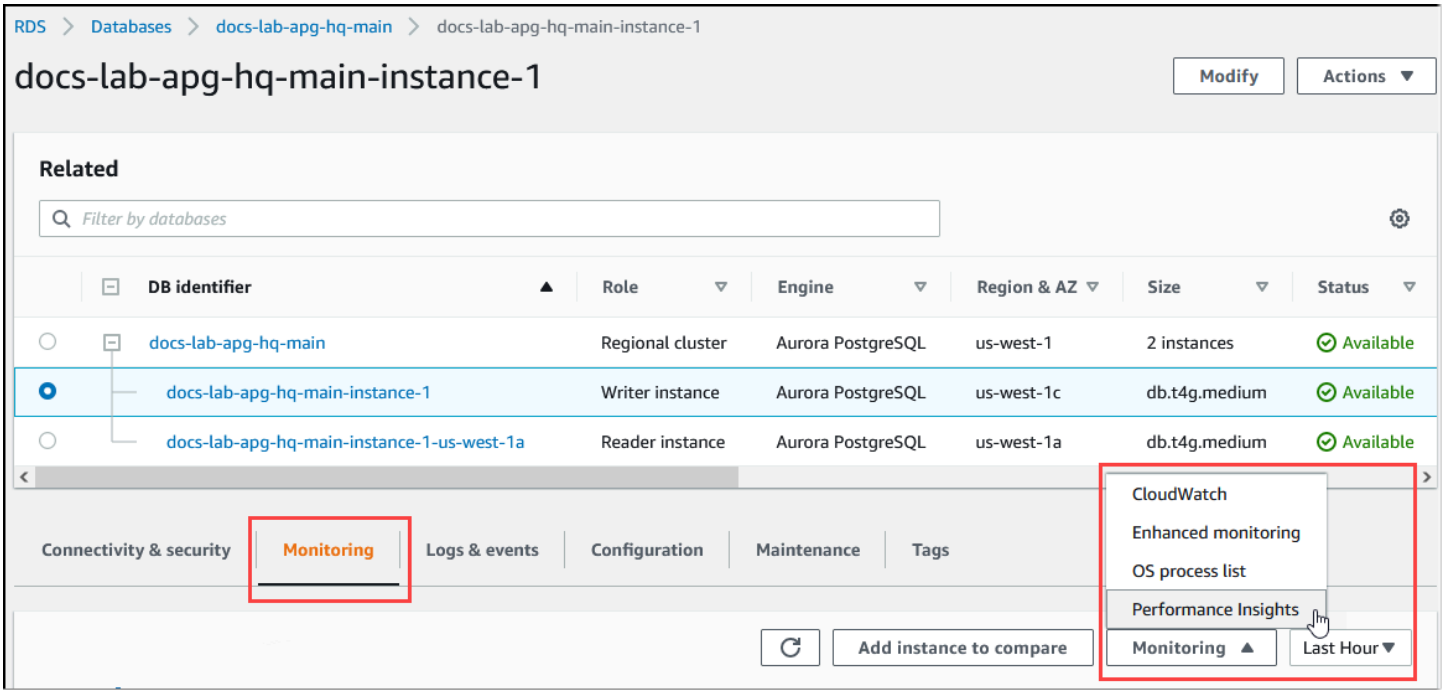
```
46 2022-03-21 16:12:14 connection authorized:
41 2022-03-21 16:12:14 disconnection: session
24 2022-03-21 16:12:15 connection authorized:
29 2022-03-21 16:12:15 disconnection: session
28 2022-03-21 16:12:16 connection authorized:
24 2022-03-21 16:12:16 disconnection: session
40 2022-03-21 16:12:17 connection authorized:
42 2022-03-21 16:12:17 disconnection: session
40 2022-03-21 16:12:18 connection authorized:
40 2022-03-21 16:12:18 disconnection: session
.....
,.......
.........
1 2022-03-21 16:14:10 connection authorized:
1 2022-03-21 16:14:10 disconnection: session
1 2022-03-21 16:15:00 connection authorized:
1 2022-03-21 16:16:00 connection authorized:
```

Con queste informazioni, puoi decidere se il carico di lavoro può trarre vantaggio da un pool di connessioni. Per analisi più dettagliate, puoi utilizzare Performance Insights.

Rilevamento dell’abbandono della connessione con Performance Insights

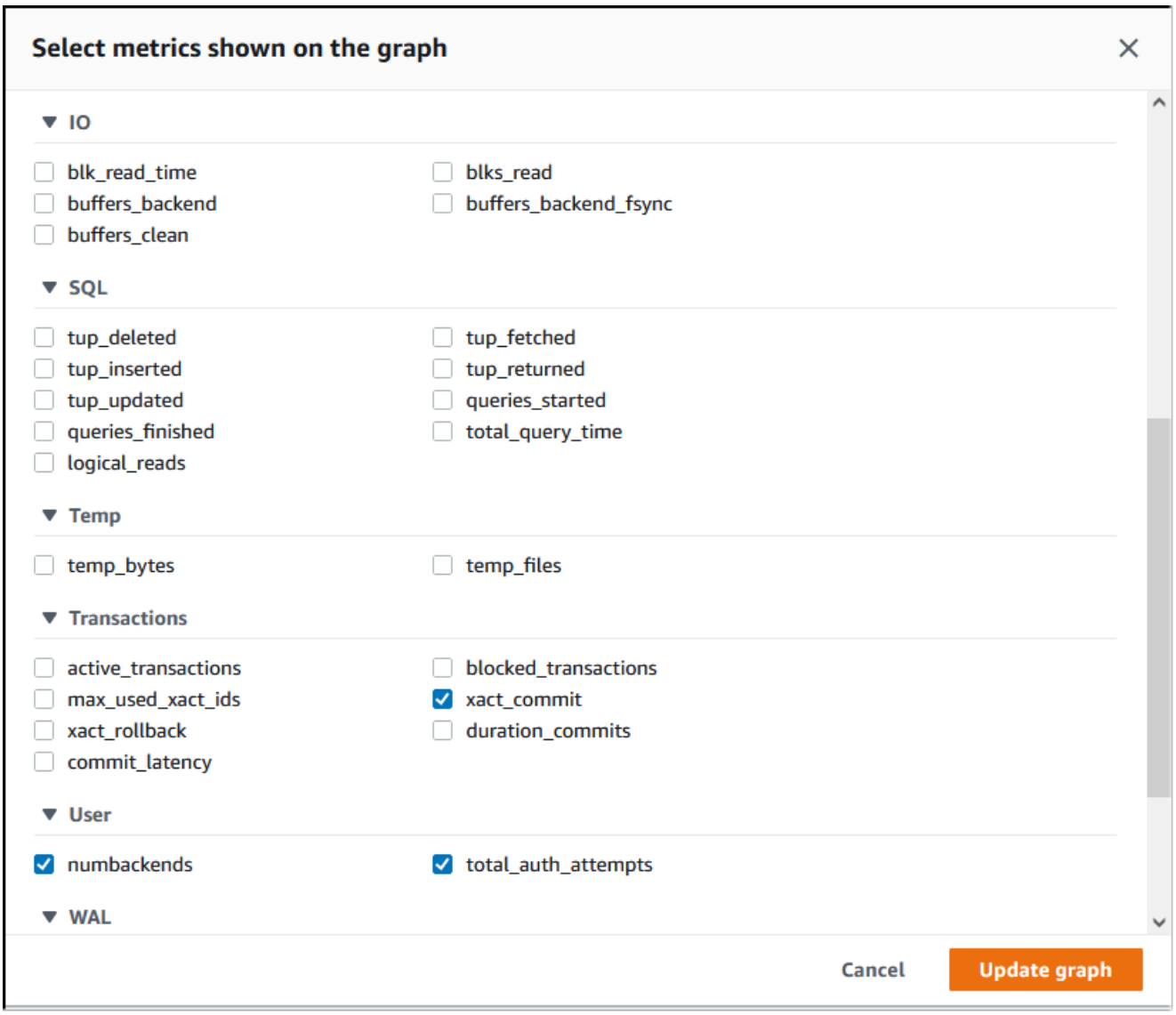
Puoi utilizzare Performance Insights per valutare il numero di abbandoni della connessione sul cluster database Aurora edizione compatibile con PostgreSQL. Durante la creazione di un cluster database Aurora PostgreSQL, l'impostazione per Performance Insights è attivata per impostazione predefinita. Se questa scelta è stata deselezionata al momento della creazione del cluster database, modifica il cluster per attivare la funzionalità. Per ulteriori informazioni, consulta Modifica di un cluster database Amazon Aurora.

Con Performance Insights in esecuzione sul cluster database Aurora PostgreSQL, puoi scegliere le metriche che desideri monitorare. Puoi accedere a Performance Insights dal riquadro di navigazione nella console. Puoi anche accedere a Performance Insights dalla scheda Monitoring (Monitoraggio) dell'istanza di scrittura per il cluster database Aurora PostgreSQL, come mostrato nell'immagine seguente.



Dalla console Performance Insights, scegli Manage metrics (Gestisci metriche). Per analizzare l'attività di connessione e disconnessione del cluster database Aurora PostgreSQL, scegli le seguenti metriche. Queste sono tutte metriche di PostgreSQL.

- `xact_commit`: il numero di transazioni confermate.
- `total_auth_attempts`: il numero di tentativi di connessione dell'utente autenticate al minuto.
- `numbackends`: il numero di backend attualmente connessi al database.

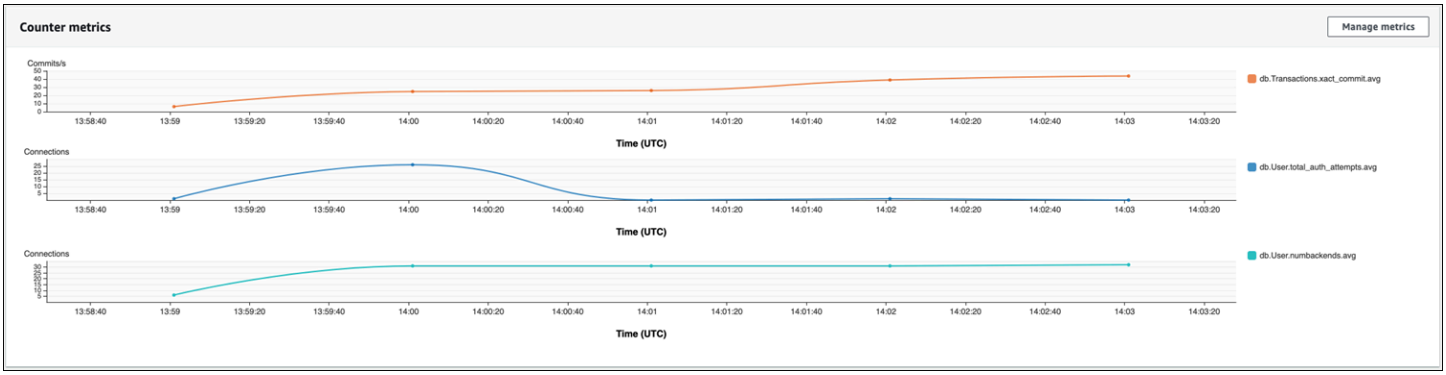


Per salvare le impostazioni e visualizzare l'attività di connessione, scegli Update graph (Aggiorna grafico).

Nell'immagine seguente è visibile l'impatto dell'esecuzione di pgbench con 100 utenti. La linea che mostra le connessioni ha una pendenza costante verso l'alto. Per ulteriori informazioni su pgbench e su come utilizzarlo, consulta pgbench nella documentazione di PostgreSQL.

L'immagine mostra che l'esecuzione di un carico di lavoro con un minimo di 100 utenti senza un pool di connessioni può causare un aumento significativo del numero di `total_auth_attempts` per tutta la durata dell'elaborazione del carico di lavoro. Notare che è meglio mantenere `total_auth_attempts` più vicino possibile allo zero.

Con il pooling delle connessioni RDS Proxy, i tentativi di connessione aumentano all'inizio del carico di lavoro. Dopo aver impostato il pool di connessioni, la media diminuisce. Le risorse utilizzate dall'uso di transazioni e backend rimangono coerenti per tutta l'elaborazione del carico di lavoro.



Per ulteriori informazioni sull'utilizzo di Performance Insights con il cluster database Aurora PostgreSQL, consulta Monitoraggio del carico DB con Performance Insights su Amazon Aurora. Per analizzare le metriche, consulta Analisi delle metriche utilizzando il pannello di controllo Performance Insights.

Dimostrazione dei vantaggi del pooling delle connessioni

Come accennato in precedenza, se si determina che il cluster database Aurora PostgreSQL DB presenta un problema di abbandono della connessione, è possibile utilizzare RDS Proxy per migliorare le prestazioni. Di seguito, è disponibile un esempio che mostra le differenze nell'elaborazione di un carico di lavoro con e senza il pooling delle connessioni. L'esempio utilizza pgbench per modellare il carico di lavoro transazionale.

Analogamente a psql, pgbench è un'applicazione client PostgreSQL che può essere installata ed eseguita dal computer client locale. Può anche essere installata ed eseguita dall'istanza Amazon EC2 utilizzata per gestire il cluster database PostgreSQL Aurora. Per ulteriori informazioni, consulta pgbench nella documentazione di PostgreSQL.

Per procedere con questo esempio, occorre creare innanzitutto l'ambiente pgbench nel database. Il seguente comando è il modello base per inizializzare le tabelle pgbench nel database specificato. Questo esempio utilizza l'account utente principale predefinito,`postgres`, per l'accesso. Modificarlo secondo necessità per il cluster database Aurora PostgreSQL. L'ambiente pgbench viene creato in un database sull'istanza di scrittura del cluster.

> **Note**
>
> Il processo di inizializzazione di pgbench elimina e ricrea le tabelle denominate `pgbench_accounts`, `pgbench_branches`, `pgbench_history` e `pgbench_tellers`. Assicurarsi che il database scelto per *dbname* durante l'inizializzazione di pgbench non utilizzi questi nomi.

```
pgbench -U postgres -h db-cluster-instance-1.111122223333.aws-region.rds.amazonaws.com
 -p 5432 -d -i -s 50 dbname
```

Per pgbench, specifica i seguenti parametri.

-d

Genera un report di debug durante l'esecuzione di pgbench.

-h

Specifica l'endpoint dell'istanza di scrittura del cluster database Aurora PostgreSQL.

-i

Inizializza l'ambiente pgbench nel database per i test di benchmark.

-p

Identifica la porta utilizzata per le connessioni al database. L'impostazione predefinita per Aurora PostgreSQL è in genere 5432 o 5433.

-s

Specifica il fattore di dimensionamento da utilizzare per compilare le tabelle con righe. Il fattore di dimensionamento predefinito è 1, che genera 1 riga nella tabella `pgbench_branches`, 10 righe nella tabella `pgbench_tellers` e 100000 righe nella tabella `pgbench_accounts`.

-U

Specifica l'account utente per l'istanza di scrittura del cluster database Aurora PostgreSQL.

Dopo aver impostato l'ambiente pgbench, puoi eseguire test di benchmarking con e senza pooling delle connessioni. Il test predefinito consiste in una serie di cinque comandi SELECT, UPDATE e INSERT per transazione che vengono eseguiti ripetutamente per il tempo specificato. Puoi specificare il fattore di dimensionamento, il numero di client e altri dettagli per modellare i tuoi casi d'uso.

Ad esempio, il comando seguente esegue il benchmark per 60 secondi (opzione -T, per il tempo) con 20 connessioni simultanee (l'opzione -c). L'opzione -C determina l'esecuzione del test utilizzando una nuova connessione ogni volta, anziché una volta per sessione client. Questa impostazione fornisce un'indicazione del sovraccarico delle connessioni.

```
pgbench -h docs-lab-apg-133-test-instance-1.c3zr2auzukpa.us-west-1.rds.amazonaws.com -U
 postgres -p 5432 -T 60 -c 20 -C labdb
Password:**********
pgbench (14.3, server 13.3)
  starting vacuum...end.
  transaction type: <builtin: TPC-B (sort of)>
    scaling factor: 50
    query mode: simple
    number of clients: 20
    number of threads: 1
    duration: 60 s
    number of transactions actually processed: 495
    latency average = 2430.798 ms
    average connection time = 120.330 ms
    tps = 8.227750 (including reconnection times)
```

L'esecuzione di pgbench sull'istanza di scrittura di un cluster database Aurora PostgreSQL senza riutilizzare le connessioni mostra che vengono elaborate solo circa 8 transazioni al secondo. Ciò fornisce un totale di 495 transazioni durante il test di 1 minuto.

Se si riutilizzano le connessioni, la risposta del cluster database Aurora PostgreSQL per il numero di utenti è quasi 20 volte più veloce. Con il riutilizzo, viene elaborato un totale di 9.042 transazioni rispetto alle 495 nello stesso periodo di tempo e per lo stesso numero di connessioni utente. La differenza è che in quello seguente, ogni connessione viene riutilizzata.

```
pgbench -h docs-lab-apg-133-test-instance-1.c3zr2auzukpa.us-west-1.rds.amazonaws.com -U
 postgres -p 5432 -T 60 -c 20 labdb
Password:*********
pgbench (14.3, server 13.3)
      starting vacuum...end.
      transaction type: <builtin: TPC-B (sort of)>
      scaling factor: 50
      query mode: simple
      number of clients: 20
      number of threads: 1
      duration: 60 s
      number of transactions actually processed: 9042
      latency average = 127.880 ms
      initial connection time = 2311.188 ms
      tps = 156.396765 (without initial connection time)
```

Questo esempio mostra che il pooling delle connessioni può migliorare significativamente i tempi di risposta. Per informazioni sulla configurazione di RDS Proxy per il cluster database Aurora PostgreSQL, consulta Server proxy per Amazon RDS per Aurora.

Gestione delle connessioni interrotte in PostgreSQL

Le connessioni interrotte si verificano quando una sessione di database rimane attiva sul server nonostante l'applicazione client sia stata abbandonata o terminata in modo anomalo. Questa situazione si verifica in genere quando i processi client vanno in crash o vengono terminati in modo imprevisto senza chiudere correttamente le connessioni al database o annullare le richieste in corso.

PostgreSQL esegue efficacemente l'identificazione e la pulizia delle connessioni interrotte quando i processi del server sono inattivi o tentano di inviare dati ai client. Tuttavia il rilevamento è complesso per le sessioni inattive, in attesa dell'input del client, o che eseguono attivamente le query. Per gestire questi scenari, PostgreSQL fornisce i parametri `tcp_keepalives_*`, `tcp_user_timeout` e `client_connection_check_interval`.

Argomenti

- Informazioni sui keepalive TCP

- Principali parametri keepalive TCP in Aurora PostgreSQL
- Casi d’uso per le impostazioni keepalive TCP
- Best practice

Informazioni sui keepalive TCP

Keepalive TCP è un meccanismo a livello di protocollo che aiuta a mantenere e verificare l’integrità della connessione. Ogni connessione TCP mantiene le impostazioni a livello di kernel che regolano il comportamento di keepalive. Quando il timer keepalive scade, il sistema effettua le seguenti operazioni:

- Invia un pacchetto di esplorazione senza dati e con il flag ACK impostato.
- Aspetta una risposta dall’endpoint remoto in base alle specifiche TCP/IP.
- Gestisce lo stato della connessione in base alla risposta o alla sua mancanza.

Principali parametri keepalive TCP in Aurora PostgreSQL

Parametro	Descrizione	Valori predefiniti
`tcp_keepalives_idle`	Specifies number of seconds of inactivity before sending keepalive message.	300
`tcp_keepalives_interval`	Specifies number of seconds between retransmissions of unacknowledged keepalive messages.	30
`tcp_keepalives_count`	Maximum lost keepalive messages before declaring connection dead	2
`tcp_user_timeout`	Specifies how long (in Milliseconds) unacknowledged data can remain before forcibly closing the connection.	0

Parametro	Descrizione	Valori predefiniti
`client_connection_check_interval`	Sets the interval (in Milliseconds) for checking client connection status during long-running queries. This ensures quicker detection of closed connections.	0

Casi d’uso per le impostazioni keepalive TCP

Keepalive per le sessioni inattive

Per evitare che le connessioni inattive vengano terminate da firewall o router a causa dell’inattività, procedi come segue:

- Configura `tcp_keepalives_idle` per inviare pacchetti keepalive a intervalli regolari.

Rilevamento di connessioni interrotte

Per rilevare tempestivamente le connessioni interrotte, procedi come segue:

- Regola `tcp_keepalives_idle`, `tcp_keepalives_interval` e `tcp_keepalives_count`. Ad esempio, con le impostazioni predefinite di Aurora PostgreSQL, è necessario circa un minuto (2 sonde × 30 secondi) per rilevare una connessione interrotta. Riducendo questi valori è possibile velocizzare il rilevamento.
- Utilizza `tcp_user_timeout` per specificare il tempo di attesa massimo per un riconoscimento.

Le impostazioni keepalive TCP aiutano il kernel a rilevare le connessioni interrotte, ma PostgreSQL potrebbe non agire finché non viene utilizzato il socket. Se una sessione esegue una query di lunga durata, le connessioni interrotte potrebbero essere rilevate solo dopo il completamento della query. In PostgreSQL 14 e versioni successive, `client_connection_check_interval` può accelerare il rilevamento delle connessioni interrotte tramite polling periodico del socket durante l’esecuzione della query.

Best practice

- Imposta intervalli keepalive ragionevoli: ottimizza `tcp_user_timeout`, `tcp_keepalives_idle`, `tcp_keepalives_count` e `tcp_keepalives_interval` per bilanciare la velocità di rilevamento e l'utilizzo delle risorse.
- Ottimizza per il tuo ambiente: allinea le impostazioni al comportamento della rete, alle policy del firewall e alle esigenze delle sessioni.
- Sfrutta le funzionalità di PostgreSQL: utilizza `client_connection_check_interval` in PostgreSQL 14 e versioni successive per controlli di connessione efficienti.

Ottimizzazione dei parametri di memoria per Aurora PostgreSQL

In Amazon Aurora PostgreSQL, puoi utilizzare diversi parametri che controllano la quantità di memoria utilizzata per varie attività di elaborazione. Se un'attività richiede più memoria della quantità impostata per un determinato parametro, Aurora PostgreSQL utilizza altre risorse per l'elaborazione, ad esempio scrivendo su disco. Ciò può causare il rallentamento o il potenziale arresto del cluster database Aurora PostgreSQL, con un errore di memoria insufficiente.

L'impostazione predefinita per ogni parametro di memoria può in genere gestire le attività di elaborazione previste. Tuttavia, è anche possibile ottimizzare i parametri correlati alla memoria del cluster database Aurora PostgreSQL . Questa ottimizzazione garantisce che venga allocata una quantità di memoria sufficiente per l'elaborazione del carico di lavoro specifico.

Di seguito sono disponibili informazioni sui parametri che controllano la gestione della memoria. Sono incluse anche informazioni su come valutare l'utilizzo della memoria.

Controllo e impostazione dei valori dei parametri

I parametri che è possibile impostare per gestire la memoria e valutare l'utilizzo della memoria del cluster database Aurora PostgreSQL includono i seguenti:

- `work_mem`: specifica la quantità di memoria utilizzata dal cluster database Aurora PostgreSQL per le operazioni di ordinamento interno e le tabelle hash prima di scrivere su file del disco temporanei.
- `log_temp_files`: registra la data di creazione di file temporanei, i nomi file e le dimensioni. Quando questo parametro è attivato, una voce di registro viene memorizzata per ogni file temporaneo creato. Attivarlo per visualizzare la frequenza di scrittura nel disco del cluster database Aurora PostgreSQL. Dopo aver raccolto informazioni sulla generazione di file temporanei del

cluster database Aurora PostgreSQL, disattivarlo nuovamente per evitare una registrazione eccessiva.

- `logical_decoding_work_mem`: specifica la quantità di memoria (in kilobyte) che deve essere utilizzata da ogni buffer di riordino interno prima di essere trasferita su disco. Questa memoria viene utilizzata per la decodifica logica, ovvero il processo per creare una replica. Tale processo viene eseguito convertendo i dati dal file WAL (Write-Ahead Log) nell'output di streaming logico richiesto dal target.

  Il valore di questo parametro crea un singolo buffer della dimensione specificata per ogni connessione di replica. Per impostazione predefinita, è 65536 KB. Dopo aver riempito questo buffer, l'eccesso viene scritto su disco come un file. Per ridurre al minimo l'attività del disco, puoi impostare il valore di questo parametro su un valore molto più alto di quello di `work_mem`.

Questi sono tutti parametri dinamici, quindi puoi modificarli per la sessione corrente. A questo scopo, esegui la connessione al cluster database Aurora PostgreSQL con psql e usando l'istruzione `SET`, come mostrato di seguito.

```
SET parameter_name TO parameter_value;
```

Le impostazioni della sessione vengono mantenute solo per la durata della sessione. Quando la sessione scade, l'impostazione del parametro viene ripristinata nel gruppo di parametri cluster database. Prima di modificare qualsiasi parametro, controlla innanzitutto i valori correnti eseguendo una query della tabella `pg_settings`, come riportato di seguito.

```
SELECT unit, setting, max_val
   FROM pg_settings WHERE name='parameter_name';
```

Ad esempio, per trovare il valore del parametro `work_mem`, esegui la connessione all'istanza di scrittura del cluster database Aurora PostgreSQL ed esegui la query seguente.

```
SELECT unit, setting, max_val, pg_size_pretty(max_val::numeric)
  FROM pg_settings WHERE name='work_mem';
unit  | setting  | max_val   | pg_size_pretty
------+----------+-----------+----------------
 kB   | 1024     | 2147483647| 2048 MB
(1 row)
```

La modifica delle impostazioni dei parametri in modo che vengano mantenuti richiede l'utilizzo di un gruppo di parametri cluster DB personalizzato. Dopo aver provato il cluster database Aurora PostgreSQL con valori diversi per questi parametri utilizzando l'istruzione `SET`, puoi creare un gruppo di parametri personalizzati e applicarlo al cluster database Aurora PostgreSQL. Per ulteriori informazioni, consulta Gruppi di parametri per Amazon Aurora.

Comprendere il parametro memoria di lavoro

Il parametro memoria di lavoro (`work_mem`) specifica la quantità massima di memoria utilizzabile da Aurora PostgreSQL per elaborare query complesse. Le query complesse includono quelle che implicano operazioni di ordinamento o raggruppamento, in altre parole, le query che utilizzano le seguenti clausole:

- ORDER BY
- DISTINCT
- GROUP BY
- JOIN (MERGE e HASH)

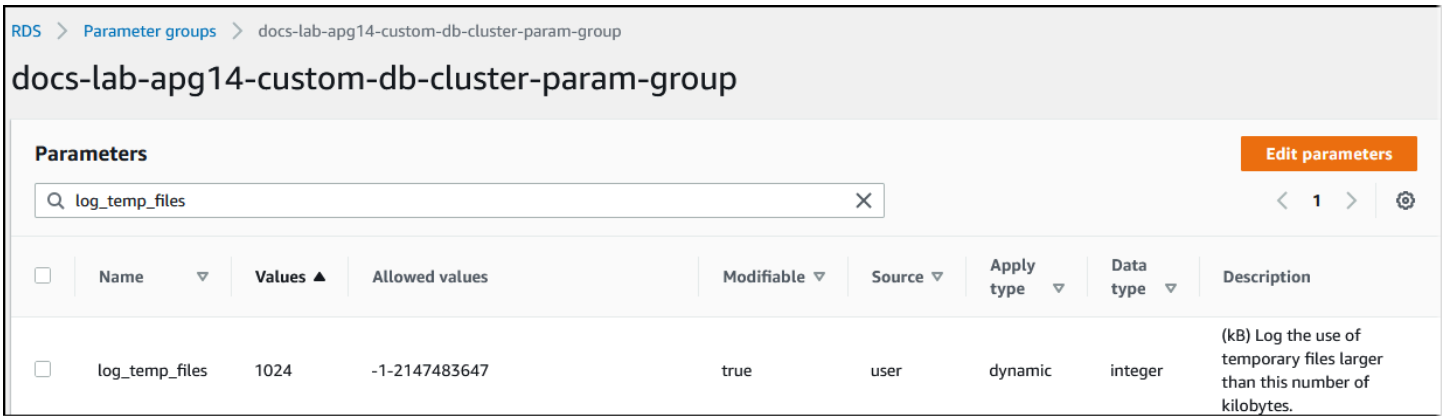
Il pianificatore query influenza indirettamente la modalità di utilizzo della memoria di lavoro nel cluster database Aurora PostgreSQL. Il pianificatore query genera piani di esecuzione per l'elaborazione delle istruzioni SQL. Un piano specifico potrebbe suddividere una query complessa in più unità di lavoro che possono essere eseguite in parallelo. Quando possibile, Aurora PostgreSQL utilizza la quantità di memoria specificata nel parametro `work_mem` per ogni sessione prima della scrittura su disco per ogni processo parallelo.

Più utenti di database che eseguono più operazioni simultaneamente e generano più unità di lavoro in parallelo possono esaurire la memoria di lavoro allocata del cluster database Aurora PostgreSQL. Ciò può portare alla creazione di un numero eccessivo di file temporanei e di I/O del disco o, peggio, può causare un errore di memoria insufficiente.

Identificazione dell'uso di file temporanei

Ogni volta che la memoria richiesta per l'elaborazione delle query supera il valore specificato nel parametro `work_mem`, i dati di lavoro vengono scaricati su disco in un file temporaneo. Per visualizzare la frequenza di questo evento, attiva il parametro `log_temp_files`. Il parametro è disattivato (è impostato su -1) per impostazione predefinita. Per acquisire tutte le informazioni sui file temporanei, imposta questo parametro su 0. Imposta `log_temp_files` su qualsiasi altro numero

intero positivo per acquisire informazioni sui file temporanei per file uguali o superiori a tale quantità di dati (in kilobyte). Nell'immagine seguente, è riportato un esempio da Console di gestione AWS.



Dopo aver configurato la registrazione dei file temporanei, puoi effettuare un test con il carico di lavoro per vedere se l'impostazione della memoria di lavoro è sufficiente. Puoi anche simulare un carico di lavoro usando pgbench, una semplice applicazione di benchmarking della community PostgreSQL.

L'esempio seguente inizializza (`-i`) `pgbench` creando le tabelle e le righe necessarie per eseguire i test. In questo esempio, il fattore di dimensionamento (`-s` 50) crea 50 righe nella tabella `pgbench_branches`, 500 righe in `pgbench_tellers` e 5.000.000 di righe nella tabella `pgbench_accounts` nel database `labdb`.

```
pgbench -U postgres -h your-cluster-instance-1.111122223333.aws-regionrds.amazonaws.com
 -p 5432 -i -s 50 labdb
Password:
dropping old tables...
NOTICE:  table "pgbench_accounts" does not exist, skipping
NOTICE:  table "pgbench_branches" does not exist, skipping
NOTICE:  table "pgbench_history" does not exist, skipping
NOTICE:  table "pgbench_tellers" does not exist, skipping
creating tables...
generating data (client-side)...
5000000 of 5000000 tuples (100%) done (elapsed 15.46 s, remaining 0.00 s)
vacuuming...
creating primary keys...
done in 61.13 s (drop tables 0.08 s, create tables 0.39 s, client-side generate 54.85
 s, vacuum 2.30 s, primary keys 3.51 s)
```

Dopo aver inizializzato l'ambiente, puoi eseguire il benchmark per un periodo di tempo specifico (`-T`) e il numero di client (`-c`). In questo esempio viene utilizzata anche l'opzione `-d` per generare

informazioni di debug mentre le transazioni vengono elaborate dal cluster database Aurora PostgreSQL.

```
pgbench -h -U postgres your-cluster-instance-1.111122223333.aws-regionrds.amazonaws.com
 -p 5432 -d -T 60 -c 10 labdb
Password:*******
pgbench (14.3)
starting vacuum...end.
transaction type: <builtin: TPC-B (sort of)>
scaling factor: 50
query mode: simple
number of clients: 10
number of threads: 1
duration: 60 s
number of transactions actually processed: 1408
latency average = 398.467 ms
initial connection time = 4280.846 ms
tps = 25.096201 (without initial connection time)
```

Per ulteriori informazioni su pgbench, consulta pgbench nella documentazione di PostgreSQL.

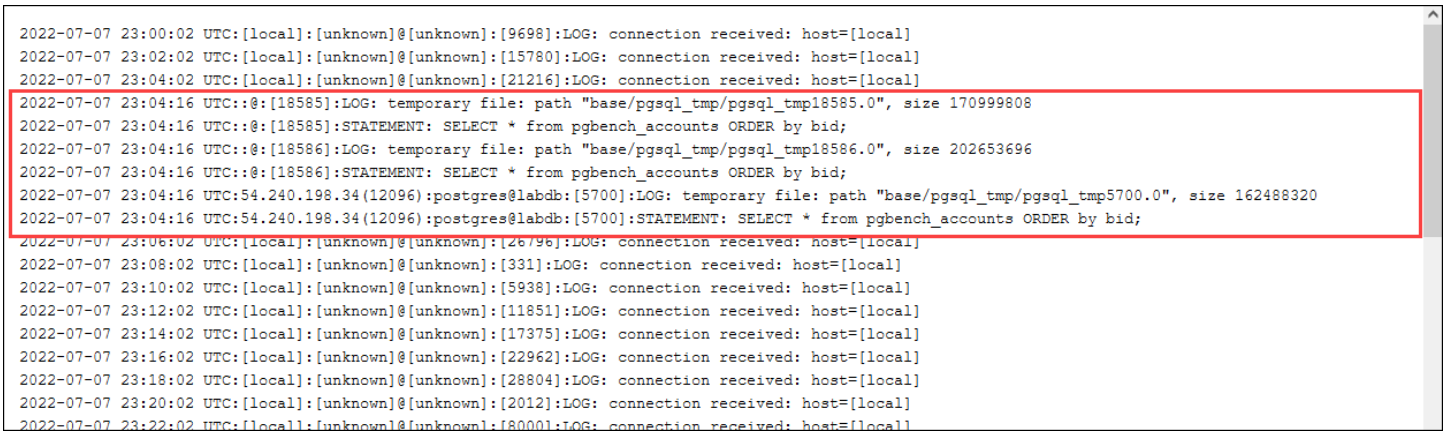
Puoi usare il metacomando psql (\d) per elencare le relazioni quali tabelle, viste e indici creati da pgbench.

```
labdb=>  \d pgbench_accounts
 Table "public.pgbench_accounts"
  Column  |     Type      | Collation | Nullable | Default
----------+---------------+-----------+----------+---------
 aid      | integer       |           | not null |
 bid      | integer       |           |          |
 abalance | integer       |           |          |
 filler   | character(84) |           |          |
Indexes:
    "pgbench_accounts_pkey" PRIMARY KEY, btree (aid)
```

Come mostrato nell'output, la tabella `pgbench_accounts` è indicizzata sulla colonna `aid`. Per garantire che questa query successiva utilizzi la memoria di lavoro, esegui una query su qualsiasi colonna non indicizzata, come quella mostrata nell'esempio seguente.

```
postgres=> SELECT * FROM pgbench_accounts ORDER BY bid;
```

Controlla se il registro contiene file temporanei. A questo scopo, apri la Console di gestione AWS, scegli l'istanza cluster database Aurora PostgreSQL, quindi seleziona la scheda Registri ed eventi. Visualizza i registri nella console o scaricali per ulteriori analisi. Come mostrato nell'immagine seguente, la dimensione dei file temporanei necessari per elaborare la query indica che è opportuno valutare se aumentare la quantità specificata per il parametro `work_mem`.



Puoi configurare questo parametro in modo diverso per singoli e gruppi, in base alle esigenze operative. Ad esempio, puoi impostare il parametro `work_mem` su 8 GB per il ruolo denominato `dev_team`.

```
postgres=> ALTER ROLE dev_team SET work_mem='8GB';
```

Con questa impostazione per `work_mem`, a qualsiasi ruolo che è un membro del ruolo `dev_team` sono assegnati fino a 8 GB di memoria di lavoro.

Utilizzo degli indici per tempo di risposta più rapido

Se le query richiedono troppo tempo per restituire i risultati, puoi verificare che l'uso degli indici sia quello previsto. Innanzitutto, attiva `\timing`, il metacomando psql, come segue.

```
postgres=>  \timing on
```

Dopo aver attivato la tempistica, utilizza una semplice istruzione SELECT.

```
postgres=> SELECT COUNT(*) FROM
  (SELECT * FROM pgbench_accounts
  ORDER BY bid)
  AS accounts;
count
-------
```

```
5000000
(1 row)
Time: 3119.049 ms (00:03.119)
```

Come mostrato nell'output, questa query ha richiesto poco più di 3 secondi per il completamento. Per migliorare il tempo di risposta, crea un indice su `pgbench_accounts`, come riportato di seguito.

```
postgres=> CREATE INDEX ON pgbench_accounts(bid);
CREATE INDEX
```

Esegui nuovamente la query e osserva che tempo di risposta è inferiore. In questo esempio, la query è stata completata all'incirca 5 volte più velocemente, in circa mezzo secondo.

```
postgres=>  SELECT COUNT(*) FROM (SELECT * FROM pgbench_accounts ORDER BY bid) AS
 accounts;
 count
-------
 5000000
(1 row)
Time: 567.095 ms
```

Regolazione della memoria di lavoro per la decodifica logica

La replica logica è disponibile in tutte le versioni di Aurora PostgreSQL sin dalla sua introduzione in PostgreSQL versione 10. Quando si configura la replica logica, è anche possibile impostare il parametro `logical_decoding_work_mem` per specificare la quantità di memoria utilizzabile dal processo di decodifica logica per il processo di decodifica e streaming.

Durante la decodifica logica, i record WAL (Write-Ahead Log) vengono convertiti in istruzioni SQL che vengono quindi inviate a un'altra destinazione per la replica logica o un'altra attività. Quando una transazione viene scritta nel WAL e quindi convertita, l'intera transazione deve rientrare nel valore specificato per `logical_decoding_work_mem`. Questo parametro è impostato su 65536 KB per impostazione predefinita. L'eventuale overflow viene scritto su disco. Ciò significa che deve essere riletto dal disco prima di poter essere inviato alla destinazione, rallentando così il processo complessivo.

È possibile valutare l'entità dell'overflow di transazione nel carico di lavoro corrente in un momento specifico utilizzando la funzione `aurora_stat_file` come mostrato nell'esempio seguente.

```
SELECT split_part (filename, '/', 2)
```

```
   AS slot_name, count(1) AS num_spill_files,
   sum(used_bytes) AS slot_total_bytes,
   pg_size_pretty(sum(used_bytes)) AS slot_total_size
   FROM aurora_stat_file()
   WHERE filename like '%spill%'
   GROUP BY 1;
  slot_name | num_spill_files | slot_total_bytes | slot_total_size
------------+-----------------+------------------+-----------------
 slot_name  |       590       |      411600000   | 393 MB
(1 row)
```

Questa query restituisce il numero e la dimensione dei file spill sul cluster database Aurora PostgreSQL quando la query viene richiamata. Per i carichi di lavoro a esecuzione più lunga, è possibile che sul disco non siano ancora presenti file spill. Per profilare i carichi di lavoro con esecuzione a lungo termine, è opportuno creare una tabella per acquisire le informazioni sui file spill durante l'esecuzione del carico di lavoro. È possibile creare la tabella come indicato di seguito.

```
CREATE TABLE spill_file_tracking AS
    SELECT now() AS spill_time,*
    FROM aurora_stat_file()
    WHERE filename LIKE '%spill%';
```

Per vedere come vengono utilizzati i file spill durante la replica logica, configura un autore e un sottoscrittore e quindi avvia una semplice replica. Per ulteriori informazioni, consulta Configurazione della replica logica per il cluster database Aurora PostgreSQL. Con la replica in corso, puoi creare un processo che acquisisce il set di risultati dalla funzione file spill `aurora_stat_file()`, come riportato di seguito.

```
INSERT INTO spill_file_tracking
  SELECT now(),*
  FROM aurora_stat_file()
  WHERE filename LIKE '%spill%';
```

Usa il seguente comando psql per eseguire il processo una volta al secondo.

```
\watch 0.5
```

Mentre il processo è in esecuzione, esegui la connessione all'istanza di scrittura da un'altra sessione psql. Utilizza la seguente serie di istruzioni per eseguire un carico di lavoro che supera la configurazione di memoria e causa la creazione di file spill in Aurora PostgreSQL.

```
labdb=> CREATE TABLE my_table (a int PRIMARY KEY, b int);
CREATE TABLE
labdb=> INSERT INTO my_table SELECT x,x FROM generate_series(0,10000000) x;
INSERT 0 10000001
labdb=> UPDATE my_table SET b=b+1;
UPDATE 10000001
```

Queste istruzioni richiedono diversi minuti per il completamento. Al termine, premi contemporaneamente i tasti Ctrl e C per interrompere la funzione di monitoraggio. Quindi utilizza il seguente comando per creare una tabella che contenga le informazioni relative all'utilizzo dei file spill del cluster database Aurora PostgreSQL.

```
SELECT spill_time, split_part (filename, '/', 2)
    AS slot_name, count(1)
    AS spills, sum(used_bytes)
    AS slot_total_bytes, pg_size_pretty(sum(used_bytes))
    AS slot_total_size FROM spill_file_tracking
  GROUP BY 1,2 ORDER BY 1;
                  spill_time | slot_name             | spills | slot_total_bytes |
 slot_total_size
-------------------------------+-----------------------+--------+------------------
+-----------------
2022-04-15 13:42:52.528272+00 | replication_slot_name | 1      | 142352280        | 136
 MB
2022-04-15 14:11:33.962216+00 | replication_slot_name | 4      | 467637996        | 446
 MB
2022-04-15 14:12:00.997636+00 | replication_slot_name | 4      | 569409176        | 543
 MB
2022-04-15 14:12:03.030245+00 | replication_slot_name | 4      | 569409176        | 543
 MB
2022-04-15 14:12:05.059761+00 | replication_slot_name | 5      | 618410996        | 590
 MB
2022-04-15 14:12:07.22905+00  | replication_slot_name | 5      | 640585316        | 611
 MB
(6 rows)
```

L'output mostra che l'esecuzione dell'esempio ha creato cinque file spill che utilizzano 611 MB di memoria. Per evitare di scrivere su disco, è consigliabile impostare il parametro `logical_decoding_work_mem` per la dimensione di memoria successiva più elevata, 1024.

Utilizzo dei CloudWatch parametri di Amazon per analizzare l'utilizzo delle risorse per Aurora PostgreSQL

Aurora invia automaticamente i dati metrici a CloudWatch in periodi di 1 minuto. È possibile analizzare l'utilizzo delle risorse per Aurora PostgreSQL utilizzando le metriche. CloudWatch valutare la velocità di trasmissione effettiva e l'utilizzo della rete.

Valutazione del throughput di rete con CloudWatch

Quando l'utilizzo del sistema è prossimo ai limiti delle risorse per il tipo di istanza, l'elaborazione può subire un rallentamento. È possibile utilizzare CloudWatch Logs Insights per monitorare l'utilizzo delle risorse di archiviazione e garantire che siano disponibili risorse sufficienti. Se necessario, puoi modificare l'istanza database in una classe di istanza più grande.

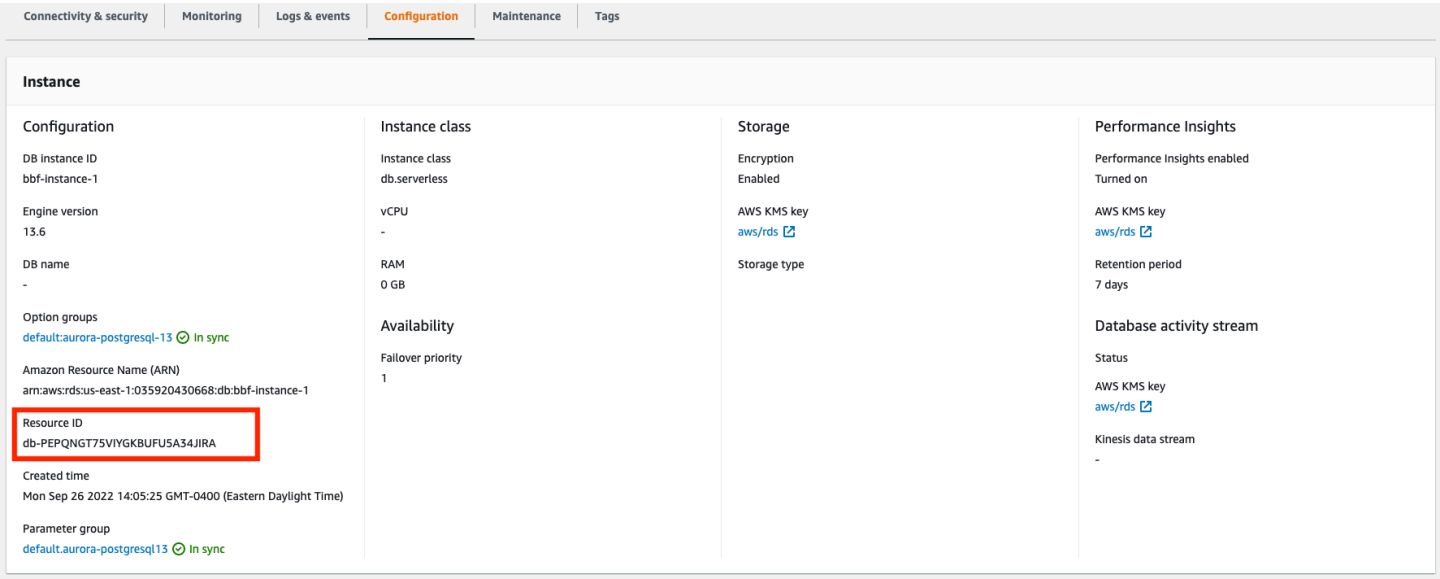
L'elaborazione dell'archiviazione di Aurora può subire un rallentamento perché:

- La larghezza di banda della rete è insufficiente tra il client e l'istanza database.
- La larghezza di banda della rete è insufficiente per il sottosistema di archiviazione.
- Un carico di lavoro è troppo elevato per il tipo di istanza in uso.

È possibile interrogare CloudWatch Logs Insights per generare un grafico dell'utilizzo delle risorse di archiviazione Aurora per monitorare le risorse. Il grafico mostra l'utilizzo della CPU e i parametri per permetterti di decidere se aumentare le dimensioni dell'istanza. Per informazioni sulla sintassi delle query per CloudWatch Logs Insights, vedere CloudWatch Sintassi delle query di Logs Insights

Per utilizzarli CloudWatch, è necessario esportare i file di registro di Aurora PostgreSQL in. CloudWatch È inoltre possibile modificare il cluster esistente in cui esportare i log. CloudWatch Per informazioni sull'esportazione dei log in, vedere CloudWatch. Attivazione dell'opzione per pubblicare i registri in Amazon CloudWatch

È necessario il Resource ID dell'istanza DB per interrogare CloudWatch Logs Insights. Il valore Resource ID (ID risorsa) è disponibile nella scheda Configuration (Configurazione) della console:



Per eseguire le query sui file di log per i parametri di archiviazione delle risorse:

1. Apri la CloudWatch console all'indirizzo https://console.aws.amazon.com/cloudwatch/.

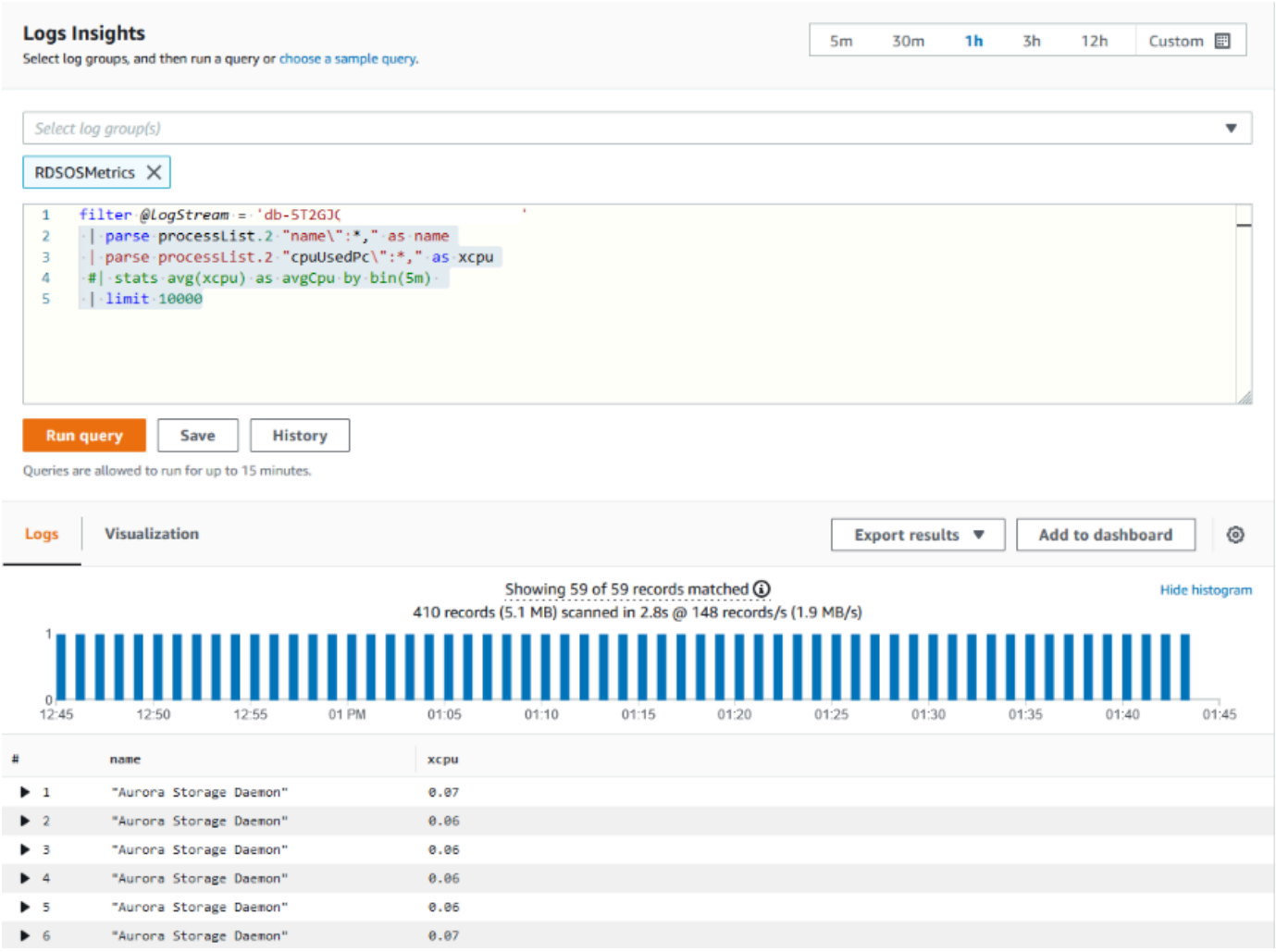
   Viene visualizzata la home page CloudWatch panoramica.
2. Se necessario, modifica Regione AWS. Nella barra di navigazione, scegli Regione AWS dove si trovano AWS le tue risorse. Per maggiori informazioni, consulta Regioni ed endpoint di .
3. Nel pannello di navigazione scegli Logs (Log), quindi Logs Insights (Approfondimenti sui file di log).

   Viene visualizzata la pagina Approfondimenti sui file di log.
4. Seleziona nell'elenco a discesa i file di log da analizzare.
5. Immetti la seguente query nel campo, sostituendo `<resource ID>` con l'ID risorsa del tuo cluster database:

   ```
   filter @logStream = <resource ID> | parse @message "\"Aurora Storage Daemon\"*memoryUsedPc\":*,\"cpuUsedPc\":*," as a,memoryUsedPc,cpuUsedPc | display memoryUsedPc,cpuUsedPc #| stats avg(xcpu) as avgCpu by bin(5m) | limit 10000
   ```
6. Fai clic su Run query (Esegui query).

   Viene visualizzato il grafico dell'utilizzo dell'archiviazione.

L'immagine seguente mostra la pagina Approfondimenti sui file di log e la visualizzazione del grafico.



Valutazione dell'utilizzo delle istanze DB per Aurora CloudWatch PostgreSQL con metriche

Puoi utilizzare le CloudWatch metriche per monitorare il throughput dell'istanza e scoprire se la classe di istanza fornisce risorse sufficienti per le tue applicazioni. Per informazioni sui limiti della classe di istanza database, consulta Specifiche hardware per le classi di istanza database per Aurora e individua le specifiche della classe di istanza database in uso per esaminare le prestazioni della rete.

Se l’utilizzo dell’istanza database è prossimo al limite della classe di istanza, le prestazioni potrebbero iniziare a rallentare. Le CloudWatch metriche possono confermare questa situazione in modo da poter pianificare la scalabilità manuale a una classe di istanze più ampia.

Combina i seguenti valori CloudWatch delle metriche per scoprire se ti stai avvicinando al limite della classe di istanza:

- NetworkThroughput— La quantità di throughput di rete ricevuta e trasmessa dai client per ogni istanza nel cluster Aurora DB. La velocità di trasmissione effettiva non include il traffico di rete tra le istanze nel cluster database e il volume del cluster.
- StorageNetworkThroughput— La quantità di throughput di rete ricevuta e inviata al sottosistema di archiviazione Aurora da ciascuna istanza del cluster Aurora DB.

Aggiungi NetworkThroughputa per trovare il StorageNetworkThroughputthroughput di rete ricevuto e inviato al sottosistema di archiviazione Aurora da ciascuna istanza del cluster Aurora DB. Il limite della classe dell’istanza deve essere maggiore della somma di questi due parametri combinati.

È possibile utilizzare i seguenti parametri per esaminare ulteriori dettagli del traffico di rete proveniente dalle applicazioni client durante l’invio e la ricezione:

- NetworkReceiveThroughput— La quantità di throughput di rete ricevuta dai client da ciascuna istanza nel cluster Aurora PostgreSQL DB. Questo throughput non include il traffico di rete tra le istanze nel cluster di database e il volume del cluster.
- NetworkTransmitThroughput: la quantità di throughput di rete inviato ai client da ogni istanza nel cluster di database Aurora. Questo throughput non include il traffico di rete tra le istanze nel cluster di database e il volume del cluster.
- StorageNetworkReceiveThroughput: la quantità di throughput di rete ricevuto dal sottosistema di archiviazione Aurora mediante ogni istanza nel cluster di database.
- StorageNetworkTransmitThroughput— La quantità di throughput di rete inviata al sottosistema di archiviazione Aurora da ciascuna istanza del cluster DB.

Somma tutti questi parametri per confrontare l’utilizzo della rete e il limite della classe di istanza. Il limite della classe di istanza deve essere maggiore della somma di questi parametri combinati.

I limiti della rete e l’utilizzo della CPU per l’archiviazione sono reciproci, pertanto quando la velocità di trasmissione effettiva della rete aumenta, cresce anche l’utilizzo della CPU. Il monitoraggio

dell'utilizzo della CPU e della rete fornisce informazioni su come e perché le risorse vengono esaurite.

Per ridurre al minimo l'utilizzo della rete, puoi prendere in considerazione:

- L'uso di una classe di istanza più grande.
- L'uso di strategie di partizionamento `pg_partman`.
- La suddivisione delle richieste di scrittura in batch per ridurre le transazioni in generale.
- L'instradamento del carico di lavoro di sola lettura a un'istanza di sola lettura.
- L'eliminazione degli indici non utilizzati.
- La verifica della presenza di oggetti di grandi dimensioni e VACUUM. In caso di dimensioni estremamente elevate, utilizza l'estensione PostgreSQL `pg_repack`. Per ulteriori informazioni su `pg_repack`, consulta Reorganize tables in PostgreSQL databases with minimal locks (Riorganizzazione delle tabelle nei database PostgreSQL con blocchi minimi).

Utilizzo della replica logica per eseguire l'aggiornamento a una versione principale per Aurora PostgreSQL

L'utilizzo della replica logica e della clonazione rapida Aurora consente di eseguire un aggiornamento della versione principale che utilizza la versione corrente del database Aurora PostgreSQL mentre si esegue gradualmente la migrazione dei dati modificati nel nuovo database della versione principale. Questo processo di aggiornamento con tempi di inattività ridotti viene definito aggiornamento blu/verde. La versione corrente del database è denominata ambiente "blu" e la nuova versione del database è denominata ambiente "verde".

La clonazione rapida di Aurora carica completamente i dati esistenti generando uno snapshot del database di origine. La clonazione rapida utilizza un protocollo copy-on-write creato sul livello di archiviazione di Aurora che consente di creare rapidamente un clone del database. Questo metodo è molto efficace quando si esegue l'aggiornamento a un database di grandi dimensioni.

La replica logica in PostgreSQL monitora e trasferisce le modifiche ai dati dall'istanza iniziale a una nuova istanza in esecuzione in parallelo fino al passaggio alla versione più recente di PostgreSQL. La replica logica utilizza un modello di pubblicazione e sottoscrizione. Per ulteriori informazioni sulla replica logica di Aurora PostgreSQL, consulta Replica con Amazon Aurora PostgreSQL.

> **Tip**
>
> È possibile ridurre al minimo i tempi di inattività necessari per l'aggiornamento a una versione principale utilizzando la funzionalità gestita dell'implementazione blu/verde di Amazon RDS. Per ulteriori informazioni, consulta Utilizzo di Amazon Blue/Green Aurora Deployments per gli aggiornamenti del database.

Argomenti

- Requisiti
- Limitazioni
- Impostazione e controllo dei valori dei parametri
- Aggiornamento di Aurora PostgreSQL a una nuova versione principale
- Esecuzione di attività successive all'aggiornamento

Requisiti

È necessario soddisfare i seguenti requisiti per eseguire questo processo di aggiornamento con tempi di inattività ridotti:

- È necessario disporre delle autorizzazioni rds_superuser.
- Il cluster database Aurora PostgreSQL da aggiornare deve eseguire una versione supportata in grado di eseguire aggiornamenti di versione principali utilizzando la replica logica. Assicurarsi di applicare eventuali aggiornamenti e patch della versione secondaria al cluster database in uso. La funzione `aurora_volume_logical_start_lsn` utilizzata in questa tecnica è supportata nelle seguenti versioni di Aurora PostgreSQL:
  - 15.2 o versioni successive alla 15
  - 14.3 o versioni successive alla 14
  - 13.6 o versioni successive alla 13
  - 12.10 e versioni successive alla 12
  - 11.15 e versioni successive alla 11
  - 10.20 e versioni successive alla 10

  Per ulteriori informazioni sulla funzione `aurora_volume_logical_start_lsn` consulta aurora_volume_logical_start_lsn.

- Tutte le tabelle devono avere una chiave primaria o includere una colonna di identità PostgreSQL.
- Configura il gruppo di sicurezza per il VPC in modo da consentire l'accesso in entrata e in uscita tra i due cluster di database Aurora PostgreSQL, vecchio e nuovo. Puoi autorizzare l'accesso a un intervallo di instradamento interdominio senza classi (CIDR) specifico o a un altro gruppo di sicurezza nel tuo VPC o in un VPC in peering. Il VPC in peering richiede una connessione peering VPC.

> **Note**
>
> Per informazioni dettagliate sulle autorizzazioni necessarie per configurare e gestire uno scenario di replica logica in esecuzione, consulta la documentazione principale di PostgreSQL.

Limitazioni

Quando si esegue l'aggiornamento con tempi di inattività ridotti del cluster database Aurora PostgreSQL a una nuova versione principale, si utilizza la funzionalità di replica logica nativa di PostgreSQL. Ha le stesse capacità e limitazioni della replica logica di PostgreSQL. Per ulteriori informazioni, consulta la pagina relativa alla replica logica di PostgreSQL.

- I comandi DDL (Data Definition Language) non vengono replicati.
- La replica non supporta le modifiche dello schema in un database attivo. Lo schema viene ricreato nel suo formato originale durante il processo di clonazione. Se si modifica lo schema dopo la clonazione, ma prima di completare l'aggiornamento, la modifica non si riflette nell'istanza aggiornata.
- Gli oggetti di grandi dimensioni non vengono replicati, ma è possibile archiviare i dati in tabelle normali.
- La replica è supportata solo dalle tabelle, comprese le tabelle partizionate. La replica su altri tipi di relazioni, come viste, viste materializzate o tabelle esterne, non è supportata.
- I dati della sequenza non vengono replicati e richiedono l'aggiornamento manuale dopo il failover.

> **Note**
>
> Questo aggiornamento non supporta l'elaborazione automatica di script. È necessario eseguire tutti i passaggi manualmente.

Impostazione e controllo dei valori dei parametri

Prima dell'aggiornamento, configura l'istanza di scrittura del cluster di database Aurora PostgreSQL in modo che agisca da server di pubblicazione. L'istanza deve utilizzare un gruppo di parametri del·cluster di database personalizzato con le seguenti impostazioni:

- `rds.logical_replication`: imposta questo parametro su 1. Il parametro `rds.logical_replication` ha lo stesso scopo del parametro `wal_level` di un server PostgreSQL autonomo e di altri parametri che controllano la gestione dei file WAL (write-ahead log).
- `max_replication_slots`: imposta questo parametro sul numero totale di sottoscrizioni che intendi creare. Se stai usando AWS DMS, imposta questo parametro sul numero di attività AWS DMS che intendi utilizzare per l'acquisizione dei dati modificati da questo cluster di database.
- `max_wal_senders`: imposta il numero di connessioni simultanee, più qualcuna extra, da rendere disponibili per le attività di gestione e le nuove sessioni. Se usi AWS DMS, il numero di max_wal_senders deve essere uguale al numero di sessioni simultanee più il numero di attività AWS DMS che possono essere in esecuzione in un dato momento.
- `max_logical_replication_workers`: imposta il numero di worker di replica logica e di sincronizzazione delle tabelle previsti. In genere è sicuro impostare il numero di worker di replica sullo stesso valore utilizzato per max_wal_senders. I worker vengono presi dal pool dei processi in background (max_worker_processes) allocati per il server.
- `max_worker_processes`: imposta il numero di processi in background per il server. Questo numero deve essere sufficientemente grande da assegnare i worker alla replica, ai processi di pulizia automatica e ad altri processi di manutenzione che possono avvenire contemporaneamente.

Quando si esegue l'aggiornamento a una versione più recente di Aurora PostgreSQL, è necessario duplicare tutti i parametri modificati nella versione precedente del gruppo di parametri. Questi parametri vengono applicati alla versione aggiornata. Puoi eseguire query sulla tabella

`pg_settings` per ottenere l'elenco delle impostazioni dei parametri in modo da poterle ricreare nel tuo nuovo cluster di database Aurora PostgreSQL.

Ad esempio, per ottenere le impostazioni per i parametri di replica, esegui la query riportata di seguito:

```
SELECT name, setting FROM pg_settings WHERE name in
('rds.logical_replication', 'max_replication_slots',
'max_wal_senders', 'max_logical_replication_workers',
'max_worker_processes');
```

Aggiornamento di Aurora PostgreSQL a una nuova versione principale

Per preparare l'entità di pubblicazione (blu)

1. Nell'esempio che segue, l'istanza di scrittura di origine (blu) è un cluster di database Aurora PostgreSQL che esegue PostgreSQL versione 11.15. È il nodo di pubblicazione nel nostro scenario di replica. Per questa dimostrazione, l'istanza di scrittura di origine ospita una tabella di esempio che contiene una serie di valori:

   ```
   CREATE TABLE my_table (a int PRIMARY KEY);
   INSERT INTO my_table VALUES (generate_series(1,100));
   ```

2. Per creare una pubblicazione nell'istanza di origine, connettiti al nodo di scrittura dell'istanza con psql (l'interfaccia della linea di comando per PostgreSQL) o con il client di tua scelta. Immetti il comando seguente in ogni database:

   ```
   CREATE PUBLICATION publication_name FOR ALL TABLES;
   ```

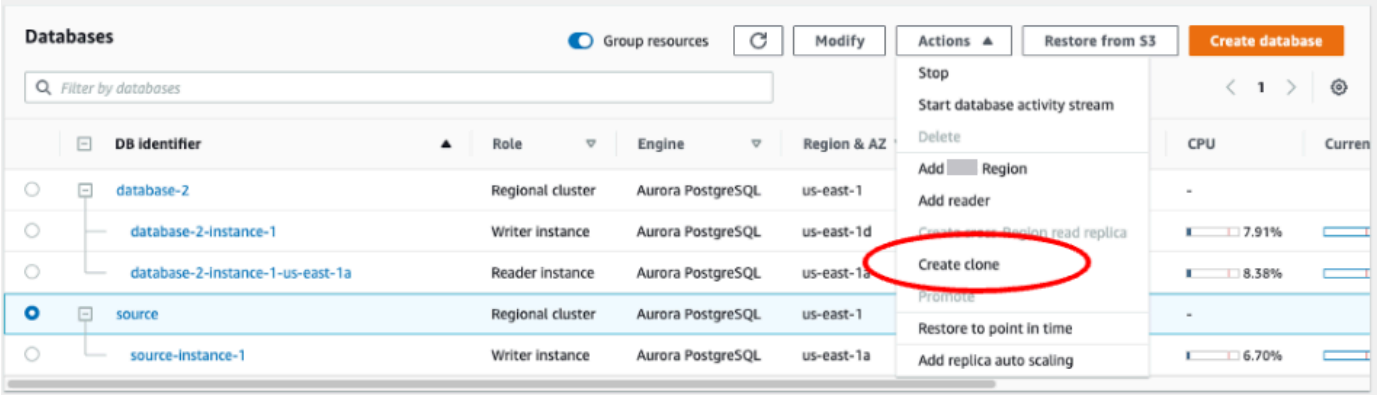
   L'indicazione publication_name specifica il nome della pubblicazione.

3. È inoltre necessario creare uno slot di replica nell'istanza. Il comando seguente crea uno slot di replica e carica il plug-in di decodifica logica `pgoutput`. Il plug-in modifica il contenuto letto da WAL (write-ahead log) al protocollo di replica logica e filtra i dati in base alle specifiche di pubblicazione.

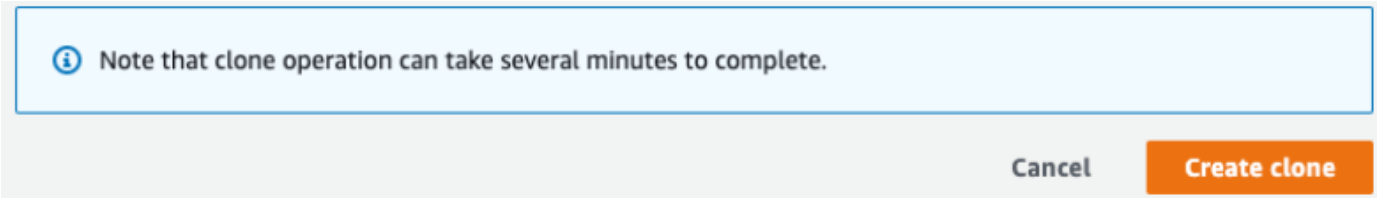
   ```
   SELECT pg_create_logical_replication_slot('replication_slot_name', 'pgoutput');
   ```

Per clonare l'entità di pubblicazione

1. Utilizza la console Amazon RDS per creare un clone dell'istanza di origine. Evidenzia il nome dell'istanza nella console Amazon RDS, quindi scegli Create clone (Crea clone) nel menu Actions (Operazioni).



2. Specifica un nome univoco per l'istanza. La maggior parte delle impostazioni sono quelle predefinite dell'istanza di origine. Dopo aver apportato le modifiche necessarie per la nuova istanza, scegli Create clone (Crea clone).



3. Durante l'avvio dell'istanza di destinazione, la colonna Status (Stato) del nodo di scrittura visualizza Creating (Creazione) nella colonna Status (Stato). Quando l'istanza è pronta, lo stato diventa Available (Disponibile).

Per preparare il clone per un aggiornamento

1. Il clone è l'istanza "verde" nel modello di implementazione. È l'host del nodo di sottoscrizione della replica. Quando il nodo diventa disponibile, connettiti con psql ed esegui una query sul nuovo nodo di scrittura per ottenere il numero di sequenza del log che identifica l'inizio di un record nel flusso WAL.

```
SELECT aurora_volume_logical_start_lsn();
```

2. Nella risposta alla query, trovi il numero di sequenza del log. Questo numero ti servirà più avanti nel processo, quindi prendine nota.

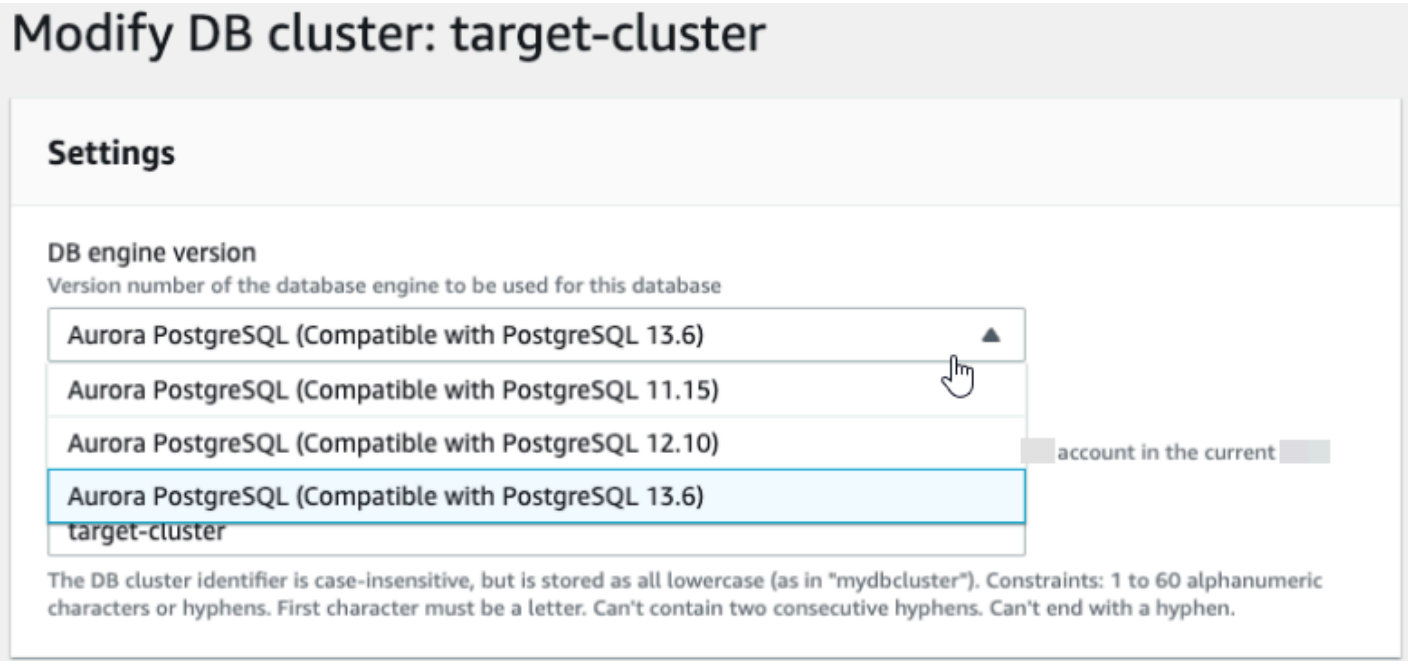
```
postgres=> SELECT aurora_volume_logical_start_lsn();
aurora_volume_logical_start_lsn
---------------
0/402E2F0
(1 row)
```

3. Prima di aggiornare il clone, elimina lo slot di replica del clone.

```
SELECT pg_drop_replication_slot('replication_slot_name');
```

Per aggiornare il cluster a una nuova versione principale

- Dopo aver clonato il nodo provider, utilizza la console Amazon RDS per avviare l'aggiornamento della versione principale sul nodo di sottoscrizione. Evidenzia il nome dell'istanza nella console RDS e seleziona il pulsante Modify (Modifica). Seleziona la versione aggiornata e i gruppi di parametri aggiornati e applica immediatamente le impostazioni per aggiornare l'istanza di destinazione.



- È anche possibile utilizzare l'interfaccia della linea di comando per eseguire un aggiornamento:

```
aws rds modify-db-cluster –db-cluster-identifier $TARGET_Aurora_ID –engine-version
 13.6 –allow-major-version-upgrade –apply-immediately
```

Per preparare il sottoscrittore (verde)

1. Quando il clone diventa disponibile, connettersi a psql e definire la sottoscrizione. Per farlo, è necessario specificare le seguenti opzioni nel comando `CREATE SUBSCRIPTION`:

   - `subscription_name`: il nome della sottoscrizione.
   - `admin_user_name`: il nome di un utente amministrativo con autorizzazioni rds_superuser.
   - `admin_user_password`: la password associata all'utente amministrativo.
   - `source_instance_URL`: l'URL dell'istanza del server di pubblicazione.
   - `database`: il database a cui si connette il server di sottoscrizione.
   - `publication_name`: il nome del server di pubblicazione.
   - `replication_slot_name`: il nome dello slot di replica.

   ```
   CREATE SUBSCRIPTION subscription_name
   CONNECTION 'postgres://admin_user_name:admin_user_password@source_instance_URL/
   database' PUBLICATION publication_name
   WITH (copy_data = false, create_slot = false, enabled = false, connect = true,
    slot_name = 'replication_slot_name');
   ```

2. Dopo aver creato la sottoscrizione, esegui la query sulla vista pg_replication_origin per recuperare il valore roname, che è l'identificatore dell'origine della replica. Ogni istanza ha un solo `roname`:

   ```
   SELECT * FROM pg_replication_origin;
   ```

   Per esempio:

   ```
   postgres=>
   SELECT * FROM pg_replication_origin;

   roident | roname
   ---------+----------
   1 | pg_24586
   ```

3. Specifica il numero di sequenza del log che hai salvato dalla precedente query del nodo di pubblicazione e il `roname` restituito dall'[ISTANZA] del nodo di sottoscrizione nel comando. Questo comando utilizza la funzione `pg_replication_origin_advance` per specificare il punto iniziale nella sequenza del log per la replica.

```
SELECT pg_replication_origin_advance('roname', 'log_sequence_number');
```

`roname` è l'identificatore restituito dalla vista pg_replication_origin.

`log_sequence_number` è il valore restituito dalla precedente query della funzione `aurora_volume_logical_start_lsn`.

4. Quindi, utilizza la clausola `ALTER SUBSCRIPTION... ENABLE` per attivare la replica logica.

```
ALTER SUBSCRIPTION subscription_name ENABLE;
```

5. A questo punto, puoi verificare se la replica funziona. Aggiungi un valore all'istanza di pubblicazione, quindi verifica che il valore sia stato replicato nel nodo di sottoscrizione.

   Quindi, utilizza il comando seguente per monitorare il ritardo di replica sul nodo di pubblicazione:

```
SELECT now() AS CURRENT_TIME, slot_name, active, active_pid,
 pg_size_pretty(pg_wal_lsn_diff(pg_current_wal_lsn(),
confirmed_flush_lsn)) AS diff_size, pg_wal_lsn_diff(pg_current_wal_lsn(),
confirmed_flush_lsn) AS diff_bytes FROM pg_replication_slots WHERE slot_type =
 'logical';
```

   Per esempio:

```
postgres=> SELECT now() AS CURRENT_TIME, slot_name, active, active_pid,
 pg_size_pretty(pg_wal_lsn_diff(pg_current_wal_lsn(),
confirmed_flush_lsn)) AS diff_size, pg_wal_lsn_diff(pg_current_wal_lsn(),
 confirmed_flush_lsn) AS diff_bytes FROM pg_replication_slots WHERE slot_type =
 'logical';

current_time                   | slot_name             | active | active_pid |
 diff_size | diff_bytes
-------------------------------+-----------------------+--------+------------
+-----------+------------
2022-04-13 15:11:00.243401+00  | replication_slot_name | t      | 21854      | 136
 bytes | 136
(1 row)
```

   È possibile monitorare il ritardo di replica utilizzando i valori `diff_size` e `diff_bytes`. Quando questi valori sono pari a 0, la replica ha raggiunto l'istanza database di origine.

Esecuzione di attività successive all'aggiornamento

Al termine dell'aggiornamento, lo stato dell'istanza viene visualizzato come Available (Disponibile) nella colonna Status (Stato) del dashboard della console. Nella nuova istanza, ti consigliamo di effettuare le seguenti operazioni:

- Reindirizza le tue applicazioni in modo che puntino al nodo di scrittura.
- Aggiungi i nodi di lettura per gestire il contenitore e garantire un'elevata disponibilità in caso di problemi con il nodo di scrittura.
- I cluster di database Aurora PostgreSQL richiedono occasionalmente gli aggiornamenti del sistema operativo. Questi aggiornamenti a volte includono una nuova versione della libreria glibc. Durante gli aggiornamenti, ti consigliamo di seguire le linee guida descritte in Regole di confronto supportate in Aurora PostgreSQL.
- Aggiorna le autorizzazioni utente sulla nuova istanza per garantire l'accesso.

Dopo aver testato l'applicazione e i dati sulla nuova istanza, ti consigliamo di eseguire un backup finale dell'istanza iniziale prima di rimuoverla. Per ulteriori informazioni sull'uso della replica logica in un host Aurora, consulta Configurazione della replica logica per il cluster database Aurora PostgreSQL.

Risoluzione dei problemi di archiviazione in Aurora PostgreSQL

Se la quantità di memoria di lavoro necessaria per le operazioni di ordinamento o creazione dell'indice supera la quantità allocata dal parametro `work_mem`, Aurora PostgreSQL scrive i dati in eccesso in file temporanei su disco. Quando scrive i dati, Aurora PostgreSQL usa lo stesso spazio utilizzato per l'archiviazione dei log di errori e messaggi, ossia lo spazio di archiviazione locale. Ogni istanza del cluster database Aurora PostgreSQL ha una quantità di spazio di archiviazione locale disponibile che si basa sulla relativa classe di istanza database. Per aumentare la quantità di spazio di archiviazione locale, è necessario modificare l'istanza per utilizzare una classe di istanza database più grande. Per le specifiche per la classe di istanza database, consulta Specifiche hardware per le classi di istanza database per Aurora.

Puoi monitorare lo spazio di archiviazione locale del tuo cluster database Aurora PostgreSQL osservando la metrica Amazon CloudWatch per `FreeLocalStorage`. Questa metrica segnala la quantità di spazio di archiviazione disponibile per ogni istanza database nel cluster database Aurora per le tabelle e i log temporanei. Per ulteriori informazioni, consulta Monitoraggio dei parametri di Amazon Aurora con Amazon CloudWatch.

Le operazioni di ordinamento, indicizzazione e raggruppamento iniziano nella memoria di lavoro, ma spesso devono essere sottoposte a offload nello spazio di archiviazione locale. Se il cluster Aurora PostgreSQL DB esaurisce la memoria locale a causa di questi tipi di operazioni, è possibile risolvere il problema eseguendo una delle seguenti azioni.

- Aumenta la quantità di memoria di lavoro. In tal modo si riduce la necessità di utilizzare lo spazio di archiviazione locale. Per impostazione predefinita, PostgreSQL alloca 4 MB per ogni operazione di ordinamento, raggruppamento e indicizzazione. Per verificare il valore corrente della memoria di lavoro per l'istanza di scrittura del cluster database Aurora PostgreSQL, esegui la connessione all'istanza utilizzando `psql` ed esegui il comando indicato di seguito.

  ```
  postgres=> SHOW work_mem;
  work_mem
  ----------
   4MB
  (1 row)
  ```

  Puoi aumentare la memoria di lavoro a livello di sessione prima di ordinare, raggruppare o eseguire altre operazioni, come indicato di seguito.

  ```
  SET work_mem TO '1 GB';
  ```

  Per ulteriori informazioni sulla memoria di lavoro, consulta Resource Consumption (Consumo delle risorse) nella documentazione di PostgreSQL.
- Modifica il periodo di conservazione dei log in modo che vengano archiviati per periodi di tempo più brevi. Per scoprire come, consulta File di log del database Aurora PostgreSQL.

Se il cluster database Aurora PostgreSQL è più grande di 40 TB, non utilizzare le classi di istanza db.t2, db.t3 or db.t4g. Consigliamo di utilizzare le classi di istanza database T solo per i server di sviluppo e test o altri server non di produzione. Per ulteriori informazioni, consulta Tipi di classi di istanza database.

Replica con Amazon Aurora PostgreSQL

Di seguito sono riportate le informazioni sulla replica con Amazon Aurora PostgreSQL, incluso il monitoraggio e l'uso della replica logica.

Argomenti

- Utilizzo delle repliche di Aurora
- Miglioramento della disponibilità di lettura delle repliche Aurora
- Monitoraggio della replica Aurora PostgreSQL.
- Panoramica della replica logica di PostgreSQL con Aurora
- Configurazione della replica logica per il cluster database Aurora PostgreSQL
- Disattivazione della replica logica
- Monitoraggio della cache di scrittura e degli slot logici per la replica logica di Aurora PostgreSQL
- Esempio: Utilizzo della replica logica di PostgreSQL con Aurora
- Esempio: replica logica con Aurora PostgreSQL e AWS Database Migration Service
- Configurazione dell'autenticazione IAM per le connessioni di replica logica

Utilizzo delle repliche di Aurora

Le repliche Aurora sono endpoint indipendenti in un cluster database Aurora, utilizzati specialmente per operazioni di dimensionamento della lettura e maggiore disponibilità. Un cluster Aurora DB può includere fino a 15 repliche Aurora situate nelle zone di disponibilità della regione del cluster Aurora DB. AWS

Il volume del cluster DB si compone di più copie dei dati per il cluster DB. Tuttavia, i dati nel volume del cluster sono rappresentati come singolo volume logico all'istanza primaria di scrittura e alle repliche di Aurora nel cluster di database. Per ulteriori informazioni sulle repliche di Aurora, consulta Repliche di Aurora.

Le repliche di Aurora funzionano bene per il dimensionamento della lettura perché sono dedicate completamente a operazioni di lettura nel volume del cluster. L'istanza database di scrittura gestisce le operazioni di scrittura. Il volume del cluster è condiviso tra tutte le istanze del cluster di database Aurora PostgreSQL. Di conseguenza, non occorre fare altro per replicare una copia dei dati per ciascuna replica Aurora.

Con Aurora PostgreSQL, quando una replica Aurora viene eliminata, l'endpoint dell'istanza viene rimosso immediatamente e la replica Aurora viene rimossa dall'endpoint di lettura. Se vi sono istruzioni in esecuzione nella replica Aurora da eliminare, si ha un periodo di tolleranza di tre minuti. Le istruzioni esistenti possono finire correttamente durante un periodo di tolleranza. Quando finisce il periodo di tolleranza, la replica Aurora viene chiusa ed eliminata.

I cluster Aurora PostgreSQL DB supportano le repliche Aurora in diverse regioni, utilizzando il database globale Aurora. AWS Per ulteriori informazioni, consulta Utilizzo di Database globale Amazon Aurora.

> **Note**
>
> Se si utilizza la funzionalità della disponibilità di lettura e si desidera riavviare le repliche Aurora nel cluster di database, è necessario eseguire questa operazione manualmente. Per i cluster database creati prima di questa funzionalità, il riavvio dell'istanza database di scrittura comporta il riavvio automatico delle repliche Aurora. Il riavvio automatico ristabilisce un punto di ingresso che garantisce la coerenza all'interno del cluster DB. read/write

Miglioramento della disponibilità di lettura delle repliche Aurora

Aurora PostgreSQL migliora la disponibilità di lettura nel cluster database soddisfacendo continuamente le richieste di lettura quando l'istanza database di scrittura si riavvia o quando la replica Aurora non è in grado di gestire il traffico di scrittura.

La funzionalità della disponibilità di lettura è disponibile per impostazione predefinita nelle seguenti versioni di Aurora PostgreSQL:

- 16.1 e tutte le versioni successive
- 15.2 o versioni successive alla 15
- 14.7 o versioni successive alla 14
- 13.10 o versioni successive alla 13
- 12.14 e versioni successive alla 12

La funzionalità della disponibilità di lettura è supportata dal database globale Aurora nelle seguenti versioni:

- 16.1 e tutte le versioni successive
- 15.4 e versioni successive alla 15
- 14.9 e versioni successive alla 14
- 13.12 e versioni successive alla 13
- 12.16 e versioni successive alla 12

Per utilizzare la funzionalità della disponibilità di lettura per un cluster database creato in una di queste versioni prima del lancio, riavviare l'istanza di scrittura del cluster database.

Quando vengono modificati i parametri statici del cluster database Aurora PostgreSQL, è necessario riavviare l'istanza di scrittura per rendere effettive le modifiche apportate ai parametri. Ad esempio, è necessario riavviare l'istanza di scrittura quando si imposta il valore di `shared_buffers`. Con la funzionalità della disponibilità di lettura delle repliche Aurora, il cluster di database mantiene la disponibilità migliorata, il che riduce l'impatto al riavvio dell'istanza di scrittura. Le istanze di lettura non si riavviano e continuano a rispondere alle richieste di lettura. Per applicare modifiche statiche ai parametri, riavviare ogni singola istanza di lettura.

Una replica Aurora di un cluster database Aurora PostgreSQL è in grado di ripristinare gli errori di replica, come riavvii dell'istanza di scrittura, failover, replica lenta e problemi di rete, ripristinando rapidamente lo stato del database in memoria dopo la riconnessione all'istanza di scrittura. Questo approccio consente alle istanze delle repliche Aurora di raggiungere la coerenza con gli ultimi aggiornamenti dell'archiviazione mentre il database client è ancora disponibile.

Le transazioni in corso che sono in conflitto con il ripristino della replica potrebbero ricevere un errore, ma il client può provare a rieseguire queste transazioni, dopo che le istanze di lettura hanno raggiunto i livelli di prestazioni dell'istanza di scrittura.

Monitoraggio delle repliche Aurora

È possibile monitorare le repliche Aurora durante il ripristino dopo una disconnessione dell'istanza di scrittura. Utilizzare le metriche riportate di seguito per verificare le informazioni più recenti sull'istanza di lettura e per tenere traccia delle transazioni di sola lettura in corso.

- La `aurora_replica_status` funzione viene aggiornata per restituire la maggior parte delle up-to-date informazioni per l'istanza del lettore quando è ancora connessa. Il timestamp dell'ultimo aggiornamento in `aurora_replica_status` è sempre vuoto per la riga corrispondente all'istanza database su cui viene eseguita la query. Ciò indica che l'istanza di lettura include i dati più recenti.
- Quando la replica Aurora si disconnette dall'istanza di scrittura e si riconnette, viene emesso il seguente evento del database:

  ```
  Read replica has been disconnected from the writer instance and
  reconnected.
  ```
- Quando una query di sola lettura viene annullata a causa di un conflitto di ripristino, è possibile che venga visualizzato il seguente messaggio di errore una o più volte nel log degli errori del database:

```
Canceling statement due to conflict with recovery.

User query may not have access to page data to replica disconnect.

User query might have tried to access a file that no longer exists.

When the replica reconnects, you will be able to repeat your command.
```

Limitazioni

Le seguenti limitazioni sono valide per la repliche Aurora con la funzionalità della disponibilità di lettura:

- Le repliche Aurora del cluster di database secondario possono riavviarsi se i dati non possono essere trasmessi dall'istanza di scrittura durante il ripristino della replica.
- Le repliche Aurora non supportano il ripristino delle repliche online se una replica è già in corso e pertanto questa verrà riavviata.
- Le repliche Aurora verranno riavviate quando l'istanza database si avvicina al wraparound dell'ID delle transazioni. Per ulteriori informazioni sul wraparound dell'ID delle transazioni, consulta l'argomento relativo alla risoluzione degli errori di wraparound dell'ID delle transazioni.
- È possibile che le repliche Aurora vengano riavviate quando il processo di replica viene bloccato in determinate circostanze.

Monitoraggio della replica Aurora PostgreSQL.

Il dimensionamento di lettura e l'alta disponibilità dipendono da un periodo di ritardo minimo. Puoi monitorare il ritardo di una replica Aurora rispetto all'istanza Writer DB del tuo cluster Aurora PostgreSQL DB monitorando la metrica Amazon. CloudWatch `ReplicaLag` Poiché le repliche di Aurora eseguono la lettura dallo stesso volume del cluster dell'istanza database di scrittura, il parametro `ReplicaLag` assume un significato diverso per un cluster di database Aurora PostgreSQL. Il parametro `ReplicaLag` per una replica Aurora indica il ritardo per la cache di pagina della replica Aurora rispetto a quello dell'istanza database di scrittura.

Per ulteriori informazioni sul monitoraggio delle istanze e dei parametri RDS, consulta. CloudWatch Monitoraggio dei parametri in un cluster di database Amazon Aurora

Panoramica della replica logica di PostgreSQL con Aurora

Utilizzando la funzionalità di replica logica di PostgreSQL con il cluster database Aurora PostgreSQL, è possibile replicare e sincronizzare singole tabelle anziché l'intera istanza database. La replica logica utilizza un modello di pubblicazione e sottoscrizione per replicare le modifiche da un'origine in uno o più destinatari. Funziona utilizzando i record di modifica del WAL (write-ahead log) PostgreSQL. L'origine, o l'autore, invia i dati WAL per le tabelle specificate a uno o più destinatari (sottoscrittore), replicando così le modifiche e mantenendo sincronizzata la tabella di un sottoscrittore con la tabella dell'autore. L'insieme di modifiche apportate dall'editore vengono identificate mediante una pubblicazione. Gli abbonati ottengono le modifiche creando un abbonamento che definisce la connessione al database dell'autore e alle sue pubblicazioni. Uno slot di replica è il meccanismo utilizzato in questo schema per tenere traccia dell'avanzamento di una sottoscrizione.

Per i cluster database Aurora PostgreSQL, i record WAL vengono salvati nell'archiviazione Aurora. Il cluster database Aurora PostgreSQL che funge da autore in uno scenario di replica logica legge i dati WAL dall'archiviazione Aurora, li decodifica e li invia al sottoscrittore in modo da poter applicare le modifiche alla tabella su tale istanza. L'autore utilizza un decodificatore logico per decodificare i dati per l'uso da parte degli abbonati. Per impostazione predefinita, i cluster database Aurora PostgreSQL utilizzano il plug-in `pgoutput` PostgreSQL nativo per l'invio di dati. Sono disponibili altri decodificatori logici. Ad esempio, Aurora PostgreSQL supporta anche il plug-in `wal2json` che converte i dati WAL in JSON.

A partire dalle versioni 14.5, 13.8, 12.12 e 11.17, Aurora PostgreSQL aumenta il processo di replica logica PostgreSQL con una cache write-through per migliorare le prestazioni. I log delle transazioni WAL vengono memorizzati nella cache locale, in un buffer, per ridurre la quantità di I/O su disco, ovvero la lettura dell'archiviazione Aurora durante la decodifica logica. La cache di scrittura viene utilizzata per impostazione predefinita ogni volta che si usa la replica logica per il cluster database Aurora PostgreSQL. Aurora offre diverse funzioni che puoi utilizzare per gestire la cache. Per ulteriori informazioni, consulta Monitoraggio della cache di scrittura della replica logica di Aurora PostgreSQL.

La replica logica è supportata da tutte le versioni di Aurora PostgreSQL attualmente disponibili. Per ulteriori informazioni, consultare Aggiornamenti di Amazon Aurora PostgreSQL nelle Note di rilascio di Aurora PostgreSQL.

La replica logica è supportata da Babelfish per Aurora PostgreSQL a partire dalle seguenti versioni:

- 15.7 e versioni successive
- 16.3 e versioni successive

> **Note**
>
> Oltre alla funzionalità di replica logica nativa di PostgreSQL introdotta in PostgreSQL 10, Aurora PostgreSQL supporta anche l'estensione `pglogical`. Per ulteriori informazioni, consulta Utilizzo di pglogical per sincronizzare i dati tra le istanze.

Per ulteriori informazioni sulla replica logica PostgreSQL, consultare le sezioni relative alla replica logica e ai concetti di decodifica logica nella documentazione di PostgreSQL.

> **Note**
>
> PostgreSQL 16 ha aggiunto il supporto per la decodifica logica dalle repliche di lettura. Questa funzionalità non è supportata su Aurora PostgreSQL.

Configurazione della replica logica per il cluster database Aurora PostgreSQL

La configurazione della replica logica richiede privilegi `rds_superuser`. Il cluster database Aurora PostgreSQL deve essere configurato per utilizzare un gruppo di parametri cluster database personalizzati in modo da poter impostare i parametri necessari come descritto nella procedura seguente. Per ulteriori informazioni, consulta Gruppi di parametri del cluster di database per i cluster di database Amazon Aurora.

Per configurare la replica logica PostgreSQL per il cluster database Aurora PostgreSQL

1. Accedi a Console di gestione AWS e apri la console Amazon RDS all'indirizzo https://console.aws.amazon.com/rds/.
2. Nel riquadro di navigazione, scegli il cluster database Aurora PostgreSQL.
3. Apri la scheda Configurazione. Nei dettagli dell'istanza, cercare il collegamento Gruppo di parametri con l'opzione Tipo impostata su Gruppo di parametri del cluster DB.
4. Scegli il collegamento per aprire i parametri personalizzati associati al cluster database Aurora PostgreSQL.
5. Nel campo di ricerca Parametri, digita `rds` per trovare il parametro `rds.logical_replication`. Il valore predefinito per questo parametro è `0`, per indicare che è disattivato per impostazione predefinita.

6. Scegli Modifica parametri per accedere ai valori delle proprietà, quindi seleziona 1 dal selettore per attivare la funzione. A secondo dell'utilizzo previsto, potrebbe anche essere necessario modificare le impostazioni per i seguenti parametri. Tuttavia, in molti casi, i valori predefiniti sono sufficienti.

   - `max_replication_slots`: imposta questo parametro su un valore almeno uguale al numero totale pianificato di pubblicazioni e sottoscrizioni della replica logica. Se lo utilizzi AWS DMS, questo parametro deve corrispondere almeno alle attività di acquisizione dei dati di modifica pianificate dal cluster, più le pubblicazioni e gli abbonamenti relativi alla replica logica.
   - `max_wal_senderse max_logical_replication_workers` — Imposta questi parametri su un valore almeno uguale al numero di slot di replica logica che intendi rendere attivi o al numero di AWS DMS attività attive per l'acquisizione dei dati di modifica. Lasciando inattivo uno slot di replica logica si impedisce al vacuum di rimuovere le tuple obsolete dalle tabelle, pertanto ti consigliamo di monitorare gli slot di replica e rimuovere gli slot inattivi in base alle esigenze.
   - `max_worker_processes`: imposta questo parametro su un valore che sia almeno uguale al totale dei valori `max_logical_replication_workers`, `autovacuum_max_workers` e `max_parallel_workers`. Su classi di istanza database di piccole dimensioni, i processi dell'operatore in background potrebbero influire sui carichi di lavoro delle applicazioni, pertanto monitorare le prestazioni del database se si imposta `max_worker_processes` su un valore più elevato di quello predefinito. (Il valore predefinito è il risultato di `GREATEST(${DBInstanceVCPU*2},8}`, il che significa che, per impostazione predefinita, è 8 o il doppio dell'equivalente CPU della classe di istanza database, a seconda di quale valore è più grande).

   > **Note**
   >
   > È possibile modificare i valori dei parametri in un gruppo di parametri database creato dal cliente, ma non i valori dei parametri in un gruppo di parametri database predefinito.

7. Scegli Save changes (Salva modifiche).
8. Riavvia l'istanza di scrittura del cluster database Aurora PostgreSQL in modo da rendere effettiva le modifiche. Nella console Amazon RDS, scegli l'istanza database principale del cluster e seleziona Riavvia dal menu Azioni.
9. Quando l'istanza è disponibile, puoi verificare che la replica logica sia attivata, come riportato di seguito.

a. Utilizza `psql` per connetterti all'istanza di scrittura del cluster database Aurora PostgreSQL.

```
psql --host=your-db-cluster-instance-1.aws-region.rds.amazonaws.com --port=5432
 --username=postgres --password --dbname=labdb
```

b. Verifica che la replica logica sia stata abilitata utilizzando il seguente comando.

```
labdb=> SHOW rds.logical_replication;
 rds.logical_replication
-------------------------
 on
(1 row)
```

c. Verifica che `wal_level` sia impostato su `logical`.

```
labdb=> SHOW wal_level;
  wal_level
-----------
 logical
(1 row)
```

Per un esempio di utilizzo della replica logica per mantenere una tabella di database sincronizzata con le modifiche di un cluster database Aurora PostgreSQL di origine, consultare Esempio: Utilizzo della replica logica di PostgreSQL con Aurora.

Disattivazione della replica logica

Dopo aver completato le attività di replica, è necessario interrompere il processo di replica, eliminare gli slot di replica e disattivare la replica logica. Prima di eliminare gli slot, assicurati che non siano più necessari. Gli slot di replica attivi non possono essere eliminati.

Disattivazione della replica logica

1. Abbandonare tutti gli slot di replica.

   Per abbandonare tutti gli slot di replica, esegui la connessione all'editore ed esegui il seguente comando SQL

```
SELECT pg_drop_replication_slot(slot_name)
  FROM pg_replication_slots
```

```
WHERE slot_name IN (SELECT slot_name FROM pg_replication_slots);
```

Gli slot di replica non possono essere attivi quando si esegue questo comando.

2. Modifica il gruppo di parametri cluster database personalizzato associato all'editore come descritto in Configurazione della replica logica per il cluster database Aurora PostgreSQL, ma imposta il parametro `rds.logical_replication` su 0

   Per ulteriori informazioni sui gruppi di parametri, consulta Modifica dei parametri in un gruppo di parametri del cluster di database in Amazon Aurora.
3. Riavvia il cluster database Aurora PostgreSQL per rendere effettivo il parametro `rds.logical_replication`.

Monitoraggio della cache di scrittura e degli slot logici per la replica logica di Aurora PostgreSQL

Il monitoraggio della cache di scrittura della replica logica e la gestione degli slot logici migliorano le prestazioni del cluster di database Aurora PostgreSQL. Di seguito, sono riportate ulteriori informazioni sulla cache di scrittura e sugli slot logici.

Argomenti

- Monitoraggio della cache di scrittura della replica logica di Aurora PostgreSQL
- Gestione degli slot logici per Aurora PostgreSQL

Monitoraggio della cache di scrittura della replica logica di Aurora PostgreSQL

Per impostazione predefinita, le versioni 14.5, 13.8, 12.12 e 11.17 e successive di Aurora PostgreSQL utilizzano una cache write-through per migliorare le prestazioni per la replica logica. Senza la cache write-through, Aurora PostgreSQL utilizza il livello di archiviazione Aurora nell'implementazione del processo di replica logica nativa di PostgreSQL. Lo fa scrivendo i dati WAL nell'archivio e quindi leggendo i dati dall'archivio per decodificarli e inviarli (replicare) alle destinazioni (sottoscrittori). Questo comportamento può causare rallentamenti durante la replica logica per i cluster database Aurora PostgreSQL.

La cache di scrittura riduce al minimo la dipendenza dal livello di archiviazione Aurora. Invece di scrivere e leggere in modo coerente da questo livello, Aurora PostgreSQL utilizza un buffer per memorizzare nella cache il flusso logico WAL da utilizzare durante il processo di replica,

riducendo la necessità di accedere al disco. Questo buffer è la cache nativa di PostgreSQL utilizzata dalla replica logica, identificata nei parametri del cluster di database Aurora PostgreSQL come `rds.logical_wal_cache`.

Quando usi la replica logica con il cluster database Aurora PostgreSQL per le versioni che supportano la cache write-through, puoi monitorare la percentuale di riscontri della cache per vedere se funziona per il tuo caso d'uso. Per farlo, connettiti all'istanza di scrittura del cluster database Aurora PostgreSQL utilizzando `psql` e quindi usa la funzione Aurora `aurora_stat_logical_wal_cache`, come mostrato nell'esempio seguente.

```
SELECT * FROM aurora_stat_logical_wal_cache();
```

La funzione restituisce un output come il seguente:

```
name       | active_pid | cache_hit | cache_miss | blks_read | hit_rate |
 last_reset_timestamp
-----------+------------+-----------+------------+-----------+----------+--------------
test_slot1 | 79183      | 24        | 0          | 24        | 100.00%  | 2022-08-05
 17:39...
test_slot2 |            | 1         | 0          |  1        | 100.00%  | 2022-08-05
 17:34...
(2 rows)
```

I valori `last_reset_timestamp` sono stati abbreviati per garantire la leggibilità. Per ulteriori informazioni su questa funzione, consulta aurora_stat_logical_wal_cache.

Aurora PostgreSQL fornisce le seguenti due funzioni per il monitoraggio della cache write-through.

- La funzione `aurora_stat_logical_wal_cache`: per la documentazione di riferimento, consulta aurora_stat_logical_wal_cache.
- La funzione `aurora_stat_reset_wal_cache`: per la documentazione di riferimento, consulta aurora_stat_reset_wal_cache.

Se si ritiene che la dimensione della cache WAL regolata automaticamente non sia sufficiente per i carichi di lavoro, è possibile modificare il valore di `rds.logical_wal_cache` manualmente. Considera i seguenti aspetti:

- Quando il parametro `rds.logical_replication` è disabilitato, `rds.logical_wal_cache` è impostato su zero (0).

- Quando il parametro `rds.logical_replication` è abilitato, `rds.logical_wal_cache` ha il valore predefinito 16 MB.
- Il parametro `rds.logical_wal_cache` è statico e richiede il riavvio dell'istanza database per rendere effettive le modifiche. Questo parametro è definito in termini di blocchi da 8 KB. Qualsiasi valore positivo inferiore a 32 KB viene considerato come 32 KB. Per ulteriori informazioni su `wal_buffers`, consultare Write Ahead Log nella documentazione PostgreSQL.

Gestione degli slot logici per Aurora PostgreSQL

L'attività di streaming viene acquisita nella vista `pg_replication_origin_status`.
Per visualizzare il contenuto di questa vista, è possibile utilizzare la funzione `pg_show_replication_origin_status()`, come illustrato di seguito:

```
SELECT * FROM pg_show_replication_origin_status();
```

Puoi ottenere un elenco degli slot logici utilizzando la seguente query SQL.

```
SELECT * FROM pg_replication_slots;
```

Per eliminare uno slot logico, utilizza `pg_drop_replication_slot` con il nome dello slot, come illustrato nel seguente comando.

```
SELECT pg_drop_replication_slot('test_slot');
```

Esempio: Utilizzo della replica logica di PostgreSQL con Aurora

La seguente procedura mostra come avviare la replica logica tra due cluster database Aurora PostgreSQL. Sia l'editore che il sottoscrittore devono essere configurati per la replica logica, come descritto in Configurazione della replica logica per il cluster database Aurora PostgreSQL.

Il cluster database Aurora PostgreSQL, che è l'editore designato, deve inoltre consentire l'accesso allo slot di replica. Per farlo, occorre modificare il gruppo di sicurezza associato al cloud privato virtuale (VPC) del cluster database Aurora PostgreSQL basato sul servizio Amazon VPC. Consentire l'accesso in entrata aggiungendo il gruppo di sicurezza associato al VPC del sottoscrittore al gruppo di sicurezza dell'editore. Per ulteriori informazioni, consultare Controlla il traffico verso le risorse utilizzando gruppi di sicurezza nella Guida per l'utente di Amazon VPC.

Una volta completati questi passaggi preliminari, puoi usare i comandi PostgreSQL `CREATE PUBLICATION` sull'editore e `CREATE SUBSCRIPTION` sul sottoscrittore, come descritto nella procedura seguente.

Come avviare il processo di replica logica tra due cluster database Aurora PostgreSQL

Queste fasi presuppongono che i cluster database Aurora PostgreSQL dispongano di un'istanza di scrittura con un database in cui creare le tabelle di esempio.

1. Nel cluster database Aurora PostgreSQL dell'editore

    a. Crea una tabella utilizzando la seguente istruzione SQL.

    ```
    CREATE TABLE LogicalReplicationTest (a int PRIMARY KEY);
    ```

    b. Inserire dati nel database del publisher utilizzando la seguente istruzione SQL.

    ```
    INSERT INTO LogicalReplicationTest VALUES (generate_series(1,10000));
    ```

    c. Verifica che i dati siano presenti nella tabella utilizzando la seguente istruzione SQL.

    ```
    SELECT count(*) FROM LogicalReplicationTest;
    ```

    d. Crea una pubblicazione per questa tabella utilizzando l'istruzione `CREATE PUBLICATION`, come descritto di seguito.

    ```
    CREATE PUBLICATION testpub FOR TABLE LogicalReplicationTest;
    ```

2. Nel cluster database Aurora PostgreSQL del sottoscrittore

    a. Nel sottoscrittore, crea la stessa tabella `LogicalReplicationTest` che hai creato nell'editore, come descritto di seguito.

    ```
    CREATE TABLE LogicalReplicationTest (a int PRIMARY KEY);
    ```

    b. Verifica che questa tabella sia vuota.

    ```
    SELECT count(*) FROM LogicalReplicationTest;
    ```

    c. Crea una sottoscrizione per ottenere le modifiche dall'editore. È necessario utilizzare i seguenti dettagli sul cluster database Aurora PostgreSQL dell'editore.

- host: l'istanza database di scrittura del cluster database Aurora PostgreSQL dell'editore.
- porta: la porta di ascolto dell'istanza database di scrittura. L'impostazione predefinita per PostgreSQL è l5432.
- dbname: il nome del database.

```
CREATE SUBSCRIPTION testsub CONNECTION
   'host=publisher-cluster-writer-endpoint port=5432 dbname=db-name user=user
 password=password'
   PUBLICATION testpub;
```

> **Note**
>
> Specifica una password diversa dal prompt mostrato qui come best practice per la sicurezza.

Successivamente alla creazione della sottoscrizione, dal lato del publisher viene creato uno slot di replica logica.

d. Per verificare che nell'esempio i dati iniziali vengono replicati nel sottoscrittore, utilizzare la seguente istruzione SQL nel database del sottoscrittore.

```
SELECT count(*) FROM LogicalReplicationTest;
```

Ogni successiva modifica al publisher viene replicata nel sottoscrittore.

La replica logica influisce sulle prestazioni. Ti consigliamo di disattivare la replica logica al termine delle attività di replica.

Esempio: replica logica con Aurora PostgreSQL e AWS Database Migration Service

È possibile utilizzare AWS Database Migration Service (AWS DMS) per replicare un database o una parte di un database. AWS DMS Usalo per migrare i dati da un database Aurora PostgreSQL a un altro database open source o commerciale. Per ulteriori informazioni in merito AWS DMS, consulta la Guida per l'utente.AWS Database Migration Service

L'esempio seguente mostra come impostare la replica logica da un database Aurora PostgreSQL come editore e quindi utilizzarla per la migrazione. AWS DMS L'esempio utilizza lo stesso publisher e lo stesso sottoscrittore creati in Esempio: Utilizzo della replica logica di PostgreSQL con Aurora.

Per configurare la replica logica con AWS DMS, hai bisogno dei dettagli sul tuo editore e abbonato da Amazon RDS. In particolare, occorrono i dettagli sull'istanza database di scrittura del publisher e sull'istanza database del sottoscrittore.

Ottenere le seguenti informazioni per l'istanza database di scrittura del publisher:

- L'identificatore del cloud privato virtuale (virtual private cloud, VPC)
- Il gruppo di sottoreti
- La zona di disponibilità (Availability Zone, AZ)
- Il gruppo di sicurezza VPC
- L'ID dell'istanza database

Ottenere le seguenti informazioni per l'istanza database del sottoscrittore:

- L'ID dell'istanza database
- Il motore di origine

Da utilizzare AWS DMS per la replica logica con Aurora PostgreSQL

1. Preparare il database dell'editore con cui lavorare. AWS DMS

   Per far ciò, i database PostgreSQL 10.x e successivi richiedono l'applicazione di funzioni wrapper AWS DMS al database del publisher. Per i dettagli su questa fase e sulle successive, consulta le istruzioni riportate in Utilizzo di PostgreSQL versione 10.x e successive come origine per AWS DMS nella Guida per l'utente di AWS Database Migration Service .
2. Accedi a Console di gestione AWS e apri la AWS DMS console all'indirizzohttps://console.aws.amazon.com/dms/v2. In alto a destra, scegli la stessa AWS regione in cui si trovano l'editore e l'abbonato.
3. Crea un'istanza di AWS DMS replica.

   Selezionare valori che siano identici a quelli dell'istanza database di scrittura del publisher. Tali valori includono le seguenti impostazioni:

- Per VPC, selezionare lo stesso VPC dell'istanza database di scrittura.
- Per Replication Subnet Group (Gruppo di sottoreti di replica), selezionare lo stesso gruppo di sottoreti con gli stessi valori dell'istanza database di scrittura. Creane uno nuovo se necessario.
- Per Availability zone (Zona di disponibilità), selezionare la stessa zona dell'istanza database di scrittura.
- Per VPC Security Group (Gruppo di sicurezza VPC), selezionare lo stesso gruppo dell'istanza database di scrittura.

4. Crea un AWS DMS endpoint per l'origine.

   Specificare il publisher come endpoint di origine utilizzando le seguenti impostazioni:

   - Per Endpoint type (Tipo di endpoint), selezionare Source endpoint (Endpoint di origine).
   - Scegliere Select RDS DB Instance (Seleziona un'istanza database RDS).
   - Per RDS Instance (Istanza RDS), selezionare l'identificatore del database dell'istanza database di scrittura del publisher.
   - In Source engine (Motore di origine), scegliere postgres (postgres).

5. Crea un AWS DMS endpoint per l'obiettivo.

   Specificare il publisher come endpoint di destinazione utilizzando le seguenti impostazioni:

   - Per Endpoint type (Tipo di endpoint), scegliere Target endpoint (Endpoint di destinazione).
   - Scegliere Select RDS DB Instance (Seleziona un'istanza database RDS).
   - Per RDS Instance (Istanza RDS), selezionare l'identificatore del database dell'istanza database del sottoscrittore.
   - Selezionare un valore per Source engine (Motore di origine). Ad esempio, se il sottoscrittore è un database RDS PostgreSQL, selezionare postgres (postgres). Se il sottoscrittore è un database Aurora PostgreSQL, scegliere aurora-postgresql.

6. Crea un'attività di migrazione AWS DMS del database.

   Un'attività di migrazione del database serve a specificare le tabelle del database di cui effettuare la migrazione, a mappare i dati utilizzando lo schema di destinazione e a creare nuove tabelle nel database di destinazione. Utilizzare almeno le seguenti impostazioni per Task configuration (Configurazione attività):

- Per Replication instance (Istanza di replica), scegliere l'istanza di replica creata in precedenza.
- Per Source database endpoint (Endpoint del database di origine), selezionare l'origine del publisher creata in precedenza.
- Per Target database endpoint (Endpoint del database di destinazione), selezionare la destinazione del sottoscrittore creata in precedenza.

I rimanenti dettagli dell'attività dipendono dal progetto di migrazione. Per ulteriori informazioni sulla specificazione di tutti i dettagli per le attività DMS, consulta Lavorare con le attività AWS DMS nella Guida per l'AWS Database Migration Service utente.

Dopo aver AWS DMS creato l'attività, inizia la migrazione dei dati dall'editore al sottoscrittore.

Configurazione dell'autenticazione IAM per le connessioni di replica logica

A partire dalle versioni 11 e successive di Aurora PostgreSQL, puoi utilizzare l'autenticazione Identity and AWS Access Management (IAM) per le connessioni di replica. Questa funzionalità migliora la sicurezza consentendo di gestire l'accesso al database utilizzando i ruoli IAM anziché le password. Funziona a livello di cluster e segue lo stesso modello di sicurezza dell'autenticazione IAM standard.

L'autenticazione IAM per le connessioni di replica è una funzionalità opzionale. Per abilitarla, imposta il `rds.iam_auth_for_replication` parametro su `1` nel gruppo di parametri del cluster DB. Poiché si tratta di un parametro dinamico, non è necessario riavviare il cluster DB, il che consente di sfruttare l'autenticazione IAM con i carichi di lavoro esistenti senza tempi di inattività. Prima di abilitare questa funzionalità, è necessario soddisfare i requisiti elencati di seguito. Prerequisiti

Prerequisiti

Per utilizzare l'autenticazione IAM per le connessioni di replica, devi soddisfare tutti i seguenti requisiti:

- Il cluster Aurora PostgreSQL DB deve avere la versione 11 o successiva.
- Sul cluster Aurora PostgreSQL DB dell'editore:
  - Abilita l'autenticazione del database IAM.

    Per ulteriori informazioni, consulta Abilitazione e disabilitazione dell'autenticazione database IAM.
  - Abilita la replica logica impostando il `rds.logical_replication` parametro su`1`.

Per ulteriori informazioni, consulta Configurazione della replica logica per il cluster database Aurora PostgreSQL.

Nella replica logica, l'editore è il cluster Aurora PostgreSQL DB di origine che invia i dati ai cluster di abbonati. Per ulteriori informazioni, consulta Panoramica della replica logica di PostgreSQL con Aurora.

> **Note**
>
> Sia l'autenticazione IAM che la replica logica devono essere abilitate sul cluster Aurora PostgreSQL DB dell'editore. Se una delle due non è abilitata, non è possibile utilizzare l'autenticazione IAM per le connessioni di replica.

Abilitazione dell'autenticazione IAM per le connessioni di replica

Completa i seguenti passaggi per abilitare l'autenticazione IAM per la connessione di replica.

1. Verifica che il cluster Aurora PostgreSQL DB soddisfi tutti i prerequisiti per l'autenticazione IAM con connessioni di replica. Per informazioni dettagliate, vedi Prerequisiti.
2. Configura il `rds.iam_auth_for_replication` parametro modificando il gruppo di parametri del cluster DB:
   - Imposta il parametro `rds.iam_auth_for_replication` su `1`. Questo parametro dinamico non richiede un riavvio.
3. Connect al database e assegna i ruoli necessari all'utente di replica:

   I seguenti comandi SQL garantiscono i ruoli necessari per abilitare l'autenticazione IAM per le connessioni di replica:

```
-- Grant IAM authentication role
GRANT rds_iam TO replication_user_name;
-- Grant replication privileges
ALTER USER replication_user_name WITH REPLICATION;
```

Dopo aver completato questi passaggi, l'utente specificato deve utilizzare l'autenticazione IAM per le connessioni di replica.

> **Important**
>
> Quando abiliti la funzionalità, gli utenti con entrambi i `rds_iam` `rds_replication` ruoli devono utilizzare l'autenticazione IAM per le connessioni di replica. Ciò vale sia che i ruoli vengano assegnati direttamente all'utente o ereditati da altri ruoli.

Disabilitazione dell'autenticazione IAM per le connessioni di replica

Puoi disabilitare l'autenticazione IAM per le connessioni di replica utilizzando uno dei seguenti metodi:

- Imposta il `rds.iam_auth_for_replication` parametro su `0` nel gruppo di parametri del cluster DB
- In alternativa, puoi disabilitare una di queste funzionalità sul tuo cluster Aurora PostgreSQL DB:
  - Disabilita la replica logica impostando il parametro su `rds.logical_replication` `0`
  - Disabilita l'autenticazione IAM

Quando disabiliti la funzionalità, le connessioni di replica possono utilizzare le password del database per l'autenticazione, se configurate.

> **Note**
>
> Le connessioni di replica per gli utenti senza il `rds_iam` ruolo possono utilizzare l'autenticazione tramite password anche quando la funzionalità è abilitata.

Considerazioni e limitazioni

Le seguenti limitazioni e considerazioni si applicano all'utilizzo dell'autenticazione IAM per le connessioni di replica.

- L'autenticazione IAM per le connessioni di replica è disponibile solo per Aurora PostgreSQL versione 11 e successive.
- L'editore deve supportare l'autenticazione IAM per le connessioni di replica.
- Per impostazione predefinita, il token di autenticazione IAM scade dopo 15 minuti. Potrebbe essere necessario aggiornare le connessioni di replica di lunga durata prima della scadenza del token.

Inoltro di scrittura locale in Aurora PostgreSQL

L'inoltro di scrittura locale (all'interno del cluster) consente alle applicazioni di eseguire transazioni di lettura/scrittura direttamente su una replica di Aurora. I comandi di scrittura vengono quindi inoltrati all'istanza database di scrittura in attesa che venga eseguito il commit. È possibile utilizzare l'inoltro di scrittura locale per le applicazioni con scritture occasionali e che richiedono la coerenza lettura dopo scrittura, ovvero la capacità di leggere l'ultima scrittura in una transazione.

Senza inoltro di lettura, le applicazioni devono suddividere completamente tutto il traffico in lettura e in scrittura, mantenendo due gruppi di connessioni al database per inviare il traffico all'endpoint appropriato. Le repliche di lettura ricevono aggiornamenti in modo asincrono dall'istanza di scrittura. Inoltre, poiché il ritardo di replica può variare tra le repliche di lettura, è difficile ottenere una coerenza di lettura globale tra tutte le repliche. È necessario eseguire la transazione delle eventuali letture che richiedono la coerenza lettura dopo scrittura sull'istanza database di scrittura. In alternativa, è necessario sviluppare una complessa logica dell'applicazione personalizzata per sfruttare più repliche di lettura per la scalabilità, garantendo al tempo stesso la coerenza.

Con l'inoltro di scrittura, non è necessario suddividere tali transazioni o inviarle esclusivamente all'istanza di scrittura. Inoltre, non è necessario sviluppare una logica dell'applicazione complessa per ottenere la coerenza lettura dopo scrittura.

L'inoltro di scrittura locale è disponibile in tutte le Regioni in cui è disponibile Aurora PostgreSQL. Questa funzionalità è supportata nelle seguenti versioni Aurora PostgreSQL:

- 16.4 e versioni successive
- 15.8 e versioni successive
- 14.13 e versioni successive

L'inoltro di scrittura locale viene utilizzato per inoltrare le scritture da repliche nella Regione. Per inoltrare le scritture da una replica globale, consultare Utilizzo dell'inoltro di scrittura in un database globale Amazon Aurora.

Argomenti

- Limitazioni e considerazioni sull'inoltro di scrittura locale in Aurora PostgreSQL
- Configurazione di Aurora PostgreSQL per l'inoltro di scrittura locale
- Utilizzo dell'inoltro di scrittura locale per Aurora PostgreSQL
- Monitoraggio dell'inoltro di scrittura locale in Aurora PostgreSQL

Limitazioni e considerazioni sull’inoltro di scrittura locale in Aurora PostgreSQL

Al momento, le seguenti limitazioni si applicano all’inoltro di scrittura locale in Aurora PostgreSQL:

- L’inoltro di scrittura locale non è supportato con Server proxy per RDS.
- Alcune istruzioni non sono consentite o possono produrre risultati non aggiornati quando vengono utilizzate in Aurora PostgreSQL con inoltro di scrittura. Inoltre, le funzioni e le procedure definite dall’utente non sono supportate. Pertanto, l’impostazione `EnableLocalWriteForwarding` è disattivata per impostazione predefinita per i cluster di database. Prima di attivarlo, verificare che il codice dell'applicazione non sia interessato da nessuna di queste restrizioni.
- I seguenti tipi di istruzioni SQL non sono supportati con l'inoltro di scrittura:

  > **Note**
  >
  > Queste istruzioni possono essere utilizzate implicitamente dall’utente nella propria applicazione o dedotte dal protocollo PostgreSQL. Ad esempio, la gestione delle eccezioni PL/SQL può comportare l’uso di SAVEPOINT, che non è un’istruzione supportata.

  - `ANALYZE`
  - `CLUSTER`
  - `COPY`
  - Cursori: i cursori non sono supportati. Verificare che siano chiusi prima di utilizzare l’inoltro di scrittura locale.
  - Istruzioni DDL (Data Definition Language)
  - `GRANT|REVOKE|REASSIGN OWNED|SECURITY LABEL`
  - `LISTEN / NOTIFY`
  - `LOCK`
  - `SAVEPOINT`
  - `SELECT INTO`
  - `SET CONSTRAINTS`
  - Aggiornamenti delle sequenze: `nextval()`, `setval()`
  - `TRUNCATE`

- Comandi di commit in due fasi: `PREPARE TRANSACTION`, `COMMIT PREPARED`, `ROLLBACK PREPARED`
- Funzioni e procedure definite dall'utente.
- `VACUUM`

Valutare l'utilizzo dei seguenti tipi di istruzioni SQL con l'inoltro di scrittura:

- Un'istruzione DML può essere costituita da più parti, ad esempio un'istruzione `INSERT ... SELECT` o un'istruzione `DELETE ... WHERE`. In questo caso, l'intera istruzione viene inoltrata all'istanza database di scrittura ed eseguita lì.
- Istruzioni DML (Data Manipulation Language), ad esempio `INSERT`, `DELETE` e `UPDATE`.
- Istruzioni `EXPLAIN` con le istruzioni in questo elenco.
- Istruzioni `PREPARE` e `EXECUTE`.
- Istruzioni `SELECT FOR { UPDATE | NO KEY UPDATE | SHARE | KEY SHARE }`.

Configurazione di Aurora PostgreSQL per l'inoltro di scrittura locale

Le seguenti sezioni descrivono come abilitare l'inoltro di scrittura locale per il cluster di database Amazon Aurora PostgreSQL, come configurare i livelli di coerenza e come gestire le transazioni con l'inoltro di scrittura locale.

Abilitazione dell'inoltro di scrittura locale

Per impostazione predefinita, l'inoltro di scrittura locale non è abilitato per i cluster di database Aurora PostgreSQL. Abilitare l'inoltro di scrittura locale a livello di cluster, non a livello di istanza.

Console

Usando la Console di gestione AWS, seleziona la casella di controllo Attiva l'inoltro di scrittura locale in Inoltro di scrittura di repliche di lettura quando si crea o si modifica un cluster database.

AWS CLI

Per abilitare l'inoltro di scrittura locale con la AWS CLI, utilizzare l'opzione `--enable-local-write-forwarding`. Questa opzione funziona quando si crea un nuovo cluster database secondario tramite il comando `create-db-cluster`. Inoltre, funziona quando si modifica un cluster

database esistente tramite il comando `modify-db-cluster`. È possibile disabilitare l'inoltro di scrittura locale mediante l'opzione `--no-enable-local-write-forwarding` con questi stessi comandi dell'interfaccia della riga di comando.

Nell'esempio seguente viene creato un cluster di database Aurora PostgreSQL con l'inoltro di scrittura locale abilitato.

```
aws rds create-db-cluster \
--db-cluster-identifier write-forwarding-test-cluster \
--enable-local-write-forwarding \
--engine aurora-postgresql \
--engine-version 16.4 \
--master-username myuser \
--master-user-password mypassword \
--backup-retention 1
```

Vengono quindi create istanze database di scrittura e lettura in modo da poter utilizzare l'inoltro di scrittura. Per ulteriori informazioni, consulta Creazione di un cluster database Amazon Aurora.

API RDS

Per abilitare l'inoltro di scrittura locale con l'API Amazon RDS, impostare il parametro `EnableLocalWriteForwarding` su `true`. Questo parametro funziona quando si crea un nuovo cluster database secondario utilizzando l'operazione `CreateDBCluster`. Funziona anche quando si modifica un cluster database esistente utilizzando l'operazione `ModifyDBCluster`. È possibile disabilitare l'inoltro di scrittura locale impostando il parametro `EnableLocalWriteForwarding` su `false`.

Abilitazione dell'inoltro di scrittura locale per le sessioni di database

Il parametro `apg_write_forward.consistency_mode` è un parametro database e un parametro del cluster database che abilita l'inoltro di scrittura. È possibile specificare `SESSION`, `EVENTUAL`, `GLOBAL` o `OFF` per il livello di coerenza di lettura. Per ulteriori informazioni sui livelli di coerenza, consulta Coerenza e isolamento per l'inoltro di scrittura locale in Aurora PostgreSQL.

A questo parametro si applicano le seguenti regole:

- Il valore predefinito è `SESSION`.

- L'inoltro di scrittura locale è disponibile solo se `apg_write_forward.consistency_mode` è impostato su `EVENTUAL`, `SESSION` o `GLOBAL`. Questo parametro è rilevante solo nelle istanze di lettura di cluster di database con l'inoltro di scrittura locale abilitato.
- L'impostazione del valore su `OFF` disabilita l'inoltro di scrittura locale nella sessione.

Coerenza e isolamento per l'inoltro di scrittura locale in Aurora PostgreSQL

È possibile controllare il grado di coerenza di lettura in una replica di lettura. È possibile regolare il livello di coerenza di lettura per garantire che tutte le operazioni di scrittura inoltrate dalla sessione siano visibili nella replica di lettura prima di qualsiasi query successiva. Inoltre, è possibile utilizzare questa impostazione per garantire che le query sulla replica di lettura visualizzino sempre gli aggiornamenti più recenti dall'istanza database di scrittura. Ciò si verifica anche per quelli inviati da altre sessioni o altri cluster. Per specificare questo tipo di comportamento per l'applicazione, scegli il valore appropriato per il parametro a livello di sessione `apg_write_forward.consistency_mode`. Il parametro `apg_write_forward.consistency_mode` ha effetto solo sulle repliche di lettura in cui è abilitato l'inoltro di scrittura locale.

> **Note**
>
> Per il parametro `apg_write_forward.consistency_mode`, è possibile specificare i valori `SESSION`, `EVENTUAL`, `GLOBAL` o `OFF`. Per impostazione predefinita, il valore è impostato su `SESSION`. L'impostazione del valore su `OFF` disabilita l'inoltro di scrittura.

Aumentando il livello di coerenza, l'applicazione attende più tempo per la propagazione delle modifiche alle repliche di lettura. È possibile scegliere in che modo bilanciare i tempi di risposta rapidi e la piena disponibilità delle modifiche apportate in altre posizioni prima dell'esecuzione delle query.

Ogni impostazione della modalità di coerenza disponibile, produce un effetto come descritto di seguito:

- `SESSION`: una sessione in una replica di lettura che utilizza l'inoltro di scrittura locale vede i risultati di tutte le modifiche apportate a tale sessione. Le modifiche sono visibili indipendentemente dal fatto che la transazione sia stata impegnata. Se necessario, la query attende che i risultati delle operazioni di scrittura inoltrate vengano replicati nell'istanza database di lettura corrente. Non attende i risultati aggiornati delle operazioni di scrittura eseguite in altre sessioni all'interno del cluster di database corrente.

- `EVENTUAL`: una sessione in una replica di lettura che utilizza l'inoltro di scrittura locale potrebbe vedere dati leggermente obsoleti a causa del ritardo di replica. I risultati delle operazioni di scrittura nella stessa sessione non sono visibili finché l'operazione di scrittura non viene eseguita sull'istanza database di scrittura e replicata nella replica di lettura. La query non attende la disponibilità dei risultati aggiornati. Pertanto, potrebbe recuperare i dati meno recenti o i dati aggiornati, a seconda della tempistica delle istruzioni e della quantità di ritardo di replica.
- `GLOBAL`: una sessione in una replica di lettura vede le modifiche apportate da quella sessione. Vede inoltre tutte le modifiche richieste sia dall'istanza database di scrittura che da altre repliche di lettura. Ogni query potrebbe attendere un periodo che varia a seconda della quantità di ritardo della sessione. La query procede quando la replica di lettura viene aggiornata con tutti i dati richiesti dall'istanza database di scrittura, a partire dall'inizio della query.

  > **Note**
  >
  > La modalità di coerenza globale influisce sulla latenza delle query eseguite all'interno di una sessione, comportando un'attesa anche quando la sessione non ha inviato alcuna query di scrittura.

- `OFF`: l'inoltro di scrittura locale è disabilitato.

Nelle sessioni che utilizzano l'inoltro di scrittura, è possibile utilizzare solo i livelli di isolamento `REPEATABLE READ` e `READ COMMITTED`. Tuttavia, il livello di isolamento `SERIALIZABLE` non è supportato.

Per ulteriori informazioni su tutti i parametri coinvolti nell'inoltro di scrittura, consulta Impostazioni predefinite dei parametri per l'inoltro di scrittura.

Modalità di accesso alle transazioni con inoltro di scrittura

Se la modalità di accesso alle transazioni è in sola lettura, l'inoltro di scrittura locale non viene utilizzato. È possibile impostare l'accesso in modalità di sola lettura e scrittura mentre si è connessi a un cluster di database e a una sessione con l'inoltro di scrittura locale abilitato.

Per ulteriori informazioni sulle modalità di accesso alle transazioni, consultare SET TRANSACTION.

Utilizzo dell'inoltro di scrittura locale per Aurora PostgreSQL

Le seguenti sezioni spiegano come verificare se in un cluster di database è abilitato l'inoltro di scrittura locale, come visualizzare le considerazioni sulla compatibilità e come vedere i parametri

configurabili e l'impostazione dell'autenticazione. Queste informazioni forniscono i dettagli per utilizzare in modo efficace la funzionalità di inoltro di scrittura locale in Aurora PostgreSQL.

> **Note**
>
> Quando un'istanza di scrittura in un cluster che utilizza l'inoltro di scrittura locale viene riavviata, le eventuali transazioni e query attive, inoltrate sulle istanze di lettura che utilizzano l'inoltro di scrittura locale vengono chiuse automaticamente. Dopo che l'istanza di scrittura sarà nuovamente disponibile, è possibile riprovare a eseguire queste transazioni.

Verifica dell'abilitazione dell'inoltro di scrittura locale in un cluster di database

Per scoprire se è possibile utilizzare l'inoltro di scrittura locale in un cluster di database, controllare se l'attributo `LocalWriteForwardingStatus` del cluster è impostato su `enabled`.

In Console di gestione AWS, nella scheda Configurazione della pagina dei dettagli del cluster, viene visualizzato lo stato Abilitato per Inoltro di scrittura locale di replica di lettura.

Per visualizzare lo stato dell'impostazione di inoltro di scrittura locale per tutti i cluster, utilizzare il comando seguente dell'AWS CLI.

Example

```
aws rds describe-db-clusters \
--query '*[].
{DBClusterIdentifier:DBClusterIdentifier,LocalWriteForwardingStatus:LocalWriteForwardingStatus}

[
{
"LocalWriteForwardingStatus": "enabled",
"DBClusterIdentifier": "write-forwarding-test-cluster-1"
},
{
"LocalWriteForwardingStatus": "disabled",
"DBClusterIdentifier": "write-forwarding-test-cluster-2"
},
{
"LocalWriteForwardingStatus": "requested",
"DBClusterIdentifier": "test-global-cluster-2"
},
{
```

```
"LocalWriteForwardingStatus": "null",
"DBClusterIdentifier": "aurora-postgresql-v2-cluster"
}
]
```

Un cluster database può avere i seguenti valori per `LocalWriteForwardingStatus`:

- `disabled`: l'inoltro di scrittura locale è disabilitato.
- `disabling`: l'inoltro di scrittura locale è in fase di disabilitazione.
- `enabled`: l'inoltro di scrittura locale è abilitato.
- `enabling`: l'inoltro di scrittura locale è in fase di abilitazione.
- `null`: l'inoltro di scrittura locale non è disponibile per questo cluster di database.
- `requested`: l'inoltro di scrittura locale è stato richiesto, ma non è ancora attivo.

Impostazioni predefinite dei parametri per l'inoltro di scrittura

I gruppi di parametri del cluster Aurora contengono delle impostazioni per la funzionalità di inoltro di scrittura locale. Poiché si tratta di parametri cluster, tutte le istanze database in ogni cluster hanno gli stessi valori per queste variabili. I dettagli su questi parametri sono riepilogati nella tabella seguente, con note di utilizzo dopo la tabella.

Parametro	Ambito	Tipo	Valore predefinito	Valori validi
`apg_write_forward.connect_timeout`	Sessione	secondi	30	0–2147483647
`apg_write_forward.consistency_mode`	Sessione	enum	Sessione	`SESSION`, `EVENTUAL`, `GLOBAL`, e `OFF`
`apg_write_forward.idle_in_transaction_session_timeout`	Sessione	millisecondi	86400000	0–2147483647
`apg_write_forward.idle_session_timeout`	Sessione	millisecondi	300000	0–2147483647

Parametro	Ambito	Tipo	Valore predefinito	Valori validi
`apg_write_forward.max_forwarding_connections_percent`	Globale	int	25	1-100

Il parametro `apg_write_forward.max_forwarding_connections_percent` rappresenta il limite superiore degli slot di connessione al database che possono essere utilizzati per gestire le query inoltrate dalle istanze di lettura. Viene espresso come percentuale dell'impostazione `max_connections` per l'istanza database di scrittura. Ad esempio, se `max_connections` è `800` e `apg_write_forward.max_forwarding_connections_percent` è `10`, allora l'istanza di scrittura consente un massimo di 80 sessioni inoltrate simultanee. Queste connessioni provengono dallo stesso pool di connessioni gestito dall'impostazione `max_connections`. Questa impostazione si applica solo all'istanza database di scrittura quando l'inoltro di scrittura locale è abilitato nel cluster.

Utilizzare le seguenti impostazioni per controllare le richieste di inoltro di scrittura locale:

- `apg_write_forward.consistency_mode`: un parametro a livello di sessione che controlla il grado di coerenza di lettura sulla replica di lettura. I valori validi sono `SESSION`, `EVENTUAL`, `GLOBAL` o `OFF`. Per impostazione predefinita, il valore è impostato su `SESSION`. L'impostazione del valore su `OFF` disabilita l'inoltro di scrittura locale nella sessione. Per ulteriori informazioni sui livelli di coerenza, consulta Coerenza e isolamento per l'inoltro di scrittura locale in Aurora PostgreSQL. Questo parametro è rilevante solo nelle istanze di lettura con l'inoltro di scrittura locale abilitato.
- `apg_write_forward.connect_timeout`: il numero massimo di secondi di attesa della replica di lettura per stabilire una connessione all'istanza database di scrittura prima di rinunciare. Il valore `0` indica un tempo di attesa infinito.
- `apg_write_forward.idle_in_transaction_session_timeout`: numero di millisecondi per cui l'istanza database di scrittura resta in attesa di attività su una connessione inoltrata da una replica di lettura con una transazione aperta prima di chiuderla. Se la sessione continua ad avere una transazione inattiva oltre questo periodo, Aurora la chiude. Il valore `0` disabilita il timeout.
- `apg_write_forward.idle_session_timeout`: numero di millisecondi per cui l'istanza database di scrittura resta in attesa di attività su una connessione inoltrata da una replica di lettura prima di chiuderla. Se la sessione rimane inattiva oltre questo periodo, Aurora la chiude. Il valore `0` disabilita il timeout.

rdswriteforwarduser

`rdswriteforwarduser` è l'utente utilizzato per stabilire una connessione tra la replica di lettura e l'istanza database di scrittura.

> **Note**
>
> `rdswriteforwarduser` eredita i privilegi CONNECT nei database dei clienti tramite il ruolo PUBLIC. Se i privilegi per il ruolo PUBLIC vengono revocati, saranno necessari privilegi GRANT CONNECT per i database a cui inoltrare le scritture.

Monitoraggio dell'inoltro di scrittura locale in Aurora PostgreSQL

Le seguenti sezioni descrivono come monitorare l'inoltro di scrittura locale nei cluster Aurora PostgreSQL, incluse le metriche CloudWatch pertinenti e gli eventi di attesa per monitorare le prestazioni e identificare i potenziali problemi.

Metriche Amazon CloudWatch e variabili di stato Aurora PostgreSQL per l'inoltro di scrittura

Le metriche Amazon CloudWatch seguenti si applicano alle istanze database di scrittura quando si utilizza l'inoltro di scrittura su una o più repliche di lettura.

Parametro CloudWatch	Unità e descrizione
`AuroraLocal ForwardingWriterDML Throughput`	Conteggio (al secondo) Numero di istruzioni DML inoltrate elaborate ogni secondo da questa istanza database writer.
`AuroraLocalForwardingWriter OpenSessions`	Conteggio Numero di sessioni aperte su questa istanza DB di scrittura che elabora le query inoltrate.
`AuroraLocal ForwardingWriterTotalSessions`	Conteggio Numero totale di sessioni inoltrate sull'istanza database di scrittura.

Le metriche CloudWatch seguenti si applicano a ogni replica di lettura. Queste metriche vengono misurate su ogni istanza database di lettura nel cluster di database con l'inoltro di scrittura locale abilitato.

Parametro CloudWatch	Unità e descrizione
`AuroraForwardingReplicaCommitThroughput`	Conteggio (al secondo) Numero di commit in sessioni inoltrate da questa replica ogni secondo.
`AuroraForwardingReplicaDMLLatency`	Millisecondi Tempo medio di risposta in millisecondi di DML inoltrati sulla replica.
`AuroraForwardingReplicaDMLThroughput`	Conteggio (al secondo) Numero di istruzioni DML inoltrate sulla replica elaborate ogni secondo.
`AuroraForwardingReplicaErrorSessionsLimit`	Conteggio Numero di sessioni rifiutate dall'istanza database di scrittura quando viene raggiunto il limite massimo di connessioni o di connessioni create per l'inoltro di scrittura.
`AuroraForwardingReplicaOpenSessions`	Conteggio Il numero di sessioni che utilizzano l'inoltro di scrittura locale su un'istanza di replica.
`AuroraForwardingReplicaReadWaitLatency`	Millisecondi Tempo medio di attesa in millisecondi della replica per garantire la coerenza con l'LSN dell'istanza database di scrittura. Il grado di attesa dell'istanza database in lettura dipende dall'impostazione `apg_write_forward.consistency_mode`. Per ulteriori informazioni su questa impostazione, consulta the section called "Parametri di configurazione per l'inoltro di scrittura in Aurora PostgreSQL".

Eventi di attesa per l’inoltro di scrittura locale in Aurora PostgreSQL

Amazon Aurora genera i seguenti eventi di attesa quando si utilizza l'inoltro di scrittura con Aurora PostgreSQL.

Argomenti

- IPC:Aurora WriteForwardConnect
- IPC:AuroraWriteForwardConsistencyPoint
- IPC:AuroraWriteForwardExecute
- IPC:AuroraWriteForwardGetGlobalConsistencyPoint
- IPC:AuroraWriteForwardXactAbort
- IPC:AuroraWriteForwardXactCommit
- IPC:AuroraWriteForwardXactStart

IPC:Aurora WriteForwardConnect

L’evento `IPC:AuroraWriteForwardConnect` si verifica quando un processo di backend sulla replica di lettura è in attesa dell’apertura di una connessione all’istanza database di scrittura.

Probabili cause di aumento delle attese

Questo evento diventa più frequente all’aumentare del numero dei tentativi di connessione da una replica di lettura al nodo di scrittura.

Azioni

Ridurre il numero di connessioni simultanee da una replica di lettura al nodo di scrittura.

IPC:AuroraWriteForwardConsistencyPoint

L’evento `IPC:AuroraWriteForwardConsistencyPoint` descrive il tempo di attesa di una query generata da un nodo sulla replica di lettura affinché i risultati delle operazioni di scrittura inoltrate vengano replicati nella Regione attuale. Questo evento viene generato solo se il parametro `apg_write_forward.consistency_mode` a livello di sessione è impostato su uno dei seguenti:

- `SESSION`: le query su una replica di lettura attendono i risultati di tutte le modifiche apportate in quella sessione.

- `GLOBAL`: le query su una replica di lettura attendono i risultati delle modifiche apportate da quella sessione, oltre a tutte le modifiche richieste dall'istanza database di scrittura e dalla replica di lettura.

Per ulteriori informazioni sull'impostazione del parametro `apg_write_forward.consistency_mode`, consulta the section called "Parametri di configurazione per l'inoltro di scrittura in Aurora PostgreSQL".

Probabili cause di aumento delle attese

Alcune cause comuni dei tempi di attesa più lunghi sono:

- Aumento del ritardo di replica, misurato dalla metrica `ReplicaLag` di Amazon CloudWatch. Per ulteriori informazioni su questa metrica, consulta Monitoraggio della replica Aurora PostgreSQL..
- Un carico maggiore sull'istanza database di scrittura o sulla replica di lettura.

Azioni

Modifica la modalità di coerenza in base ai requisiti dell'applicazione.

IPC:AuroraWriteForwardExecute

L'evento `IPC:AuroraWriteForwardExecute` si verifica quando un processo di backend su una replica di lettura attende il completamento di una query inoltrata e la ricezione dei risultati dal nodo di scrittura del cluster di database.

Probabili cause di aumento delle attese

Alcune cause comuni dell'aumento dei tempi di attesa sono:

- Il recupero di un numero elevato di righe dal nodo di scrittura.
- Una maggiore latenza di rete tra il nodo di scrittura e la replica di lettura aumenta il tempo necessario alla replica di lettura per ricevere i dati del nodo di scrittura.
- Un carico maggiore sulla replica di lettura può ritardare la trasmissione della richiesta di query dalla replica di lettura al nodo di scrittura.
- Un carico maggiore sul nodo di scrittura può ritardare la trasmissione dei dati dal nodo di scrittura alla replica di lettura.

Azioni

Consigliamo azioni diverse a seconda delle cause dell'evento di attesa.

- Ottimizza le query per recuperare solo i dati necessari.
- Ottimizza le operazioni DML (Data Manipulation Language) per modificare solo i dati necessari.
- Se la replica di lettura o il nodo di scrittura sono limitati dalla CPU o dalla larghezza di banda della rete, valutare la possibilità di passare a un tipo di istanza con maggiore capacità di CPU o maggiore larghezza di banda della rete.

IPC:AuroraWriteForwardGetGlobalConsistencyPoint

L'evento `IPC:AuroraWriteForwardGetGlobalConsistencyPoint` si verifica quando un processo di backend sulla replica di lettura che utilizza la modalità di coerenza GLOBAL è in attesa di ottenere il punto di coerenza globale dal nodo di scrittura prima di eseguire una query.

Probabili cause di aumento delle attese

Alcune cause comuni dell'aumento dei tempi di attesa sono:

- Una maggiore latenza di rete tra la replica di lettura e il nodo di scrittura aumenta il tempo necessario alla replica di lettura per ricevere i dati del nodo di scrittura.
- Un carico maggiore sulla replica di lettura può ritardare la trasmissione della richiesta di query dalla replica di lettura al nodo di scrittura.
- Un carico maggiore sul nodo di scrittura può ritardare la trasmissione dei dati dal nodo di scrittura alla replica di lettura.

Azioni

Consigliamo azioni diverse a seconda delle cause dell'evento di attesa.

- Modifica la modalità di coerenza in base ai requisiti dell'applicazione.
- Se la replica di lettura o il nodo di scrittura sono limitati dalla CPU o dalla larghezza di banda della rete, valutare la possibilità di passare a un tipo di istanza con maggiore capacità di CPU o maggiore larghezza di banda della rete.

IPC:AuroraWriteForwardXactAbort

L'evento `IPC:AuroraWriteForwardXactAbort` si verifica quando un processo di backend sulla replica di lettura è in attesa del risultato di una query di pulizia remota. Le query di pulizia vengono

emesse per riportare il processo allo stato ottimale dopo l'interruzione di una transazione di scrittura inoltrata. Amazon Aurora le esegue perché è stato rilevato un errore o perché un utente ha chiamato un comando `ABORT` esplicito o ha annullato una query in esecuzione.

Probabili cause di aumento delle attese

Alcune cause comuni dell'aumento dei tempi di attesa sono:

- Una maggiore latenza di rete tra la replica di lettura e il nodo di scrittura aumenta il tempo necessario alla replica di lettura per ricevere i dati del nodo di scrittura.
- Un carico maggiore sulla replica di lettura può ritardare la trasmissione della richiesta di query di pulizia dalla replica di lettura al nodo di scrittura.
- Un carico maggiore sul nodo di scrittura può ritardare la trasmissione dei dati dal nodo di scrittura alla replica di lettura.

Azioni

Consigliamo azioni diverse a seconda delle cause dell'evento di attesa.

- Indaga la causa della transazione interrotta.
- Se la replica di lettura o l'istanza database di scrittura sono limitate dalla CPU o dalla larghezza di banda della rete, valutare la possibilità di passare a un tipo di istanza con maggiore capacità di CPU o maggiore larghezza di banda della rete.

IPC:AuroraWriteForwardXactCommit

L'evento `IPC:AuroraWriteForwardXactCommit` si verifica quando un processo di backend sulla replica di lettura è in attesa del risultato di un comando di transazione di commit inoltrato.

Probabili cause di aumento delle attese

Alcune cause comuni dell'aumento dei tempi di attesa sono:

- Una maggiore latenza di rete tra la replica di lettura e il nodo di scrittura aumenta il tempo necessario alla replica di lettura per ricevere i dati del nodo di scrittura.
- Un carico maggiore sulla replica di lettura può ritardare la trasmissione della richiesta di query dalla replica di lettura al nodo di scrittura.
- Un carico maggiore sul nodo di scrittura può ritardare la trasmissione dei dati dal nodo di scrittura alla replica di lettura.

Azioni

Se la replica di lettura o il nodo di scrittura sono limitati dalla CPU o dalla larghezza di banda della rete, valutare la possibilità di passare a un tipo di istanza con maggiore capacità di CPU o maggiore larghezza di banda della rete.

IPC:AuroraWriteForwardXactStart

L'evento `IPC:AuroraWriteForwardXactStart` si verifica quando un processo di backend sulla replica di lettura è in attesa del risultato di un comando di transazione di avvio inoltrato.

Probabili cause di aumento delle attese

Alcune cause comuni dell'aumento dei tempi di attesa sono:

- Una maggiore latenza di rete tra la replica di lettura e il nodo di scrittura aumenta il tempo necessario alla replica di lettura per ricevere i dati del nodo di scrittura.
- Un carico maggiore sulla replica di lettura può ritardare la trasmissione della richiesta di query dalla replica di lettura al nodo di scrittura.
- Un carico maggiore sul nodo di scrittura può ritardare la trasmissione dei dati dal nodo di scrittura alla replica di lettura.

Azioni

Se la replica di lettura o il nodo di scrittura sono limitati dalla CPU o dalla larghezza di banda della rete, valutare la possibilità di passare a un tipo di istanza con maggiore capacità di CPU o maggiore larghezza di banda della rete.

Utilizzo di Aurora PostgreSQL come Knowledge Base per Amazon Bedrock

È possibile utilizzare un cluster di database Aurora PostgreSQL come Knowledge Base per Amazon Bedrock. Per ulteriori informazioni, consulta Creazione di un archivio vettoriale in Amazon Aurora. Una Knowledge Base acquisisce automaticamente dati di testo non strutturati memorizzati in un bucket Amazon S3, li converte in blocchi di testo e vettori, quindi li archivia in un database PostgreSQL. Con le applicazioni di IA generativa, è possibile utilizzare Agent per Amazon Bedrock per eseguire query sui dati memorizzati nella Knowledge Base e utilizzarne i risultati per aumentare

le risposte fornite dai modelli di fondazione. Questo flusso di lavoro si chiama generazione potenziata da recupero dati (RAG). Per ulteriori informazioni sulla RAG, consulta Retrieval Augmented Generation (RAG).

Per informazioni dettagliate sull'utilizzo di Aurora PostgreSQL per creare applicazioni di IA generativa utilizzando RAG, consulta questo post di blog.

Argomenti

- Prerequisiti
- Preparazione di Aurora PostgreSQL per l'utilizzo come Knowledge Base per Amazon Bedrock
- Creazione di una Knowledge Base nella console Bedrock
- Creazione rapida di una Knowledge Base per Amazon Bedrock di Aurora PostgreSQL

Prerequisiti

È necessario acquisire familiarità con i seguenti prerequisiti per utilizzare il cluster Aurora PostgreSQL come Knowledge Base per Amazon Bedrock. A un livello elevato, si devono configurare i seguenti servizi da utilizzare con Bedrock:

- Cluster di database Amazon Aurora PostgreSQL creato in una delle seguenti versioni:
  - 16.1 e tutte le versioni successive
  - 15.4 e versioni successive
  - 14.9 e versioni successive
  - 13.12 e versioni successive
  - 12.16 e versioni successive

> **Note**
>
> È necessario abilitare l'estensione `pgvector` nel database di destinazione e utilizzare la versione 0.5.0 o successiva. Per ulteriori informazioni, consulta pgvector v0.5.0 with HNSW indexing.

- API dati RDS
- Un utente gestito in Gestione dei segreti AWS. Per ulteriori informazioni, consulta Gestione delle password con Amazon Aurora e Gestione dei segreti AWS.

Preparazione di Aurora PostgreSQL per l'utilizzo come Knowledge Base per Amazon Bedrock

I passaggi descritti nelle sezioni seguenti consentono di preparare Aurora PostgreSQL per l'utilizzo come Knowledge Base per Amazon Bedrock.

Creazione e configurazione di Aurora PostgreSQL

Per configurare Amazon Bedrock con un cluster di database Aurora PostgreSQL, è necessario prima creare un cluster di database Aurora PostgreSQL e prendere nota dei campi importanti per configurarlo con Amazon Bedrock. Per ulteriori informazioni sulla creazione di un cluster di database Aurora PostgreSQL, consulta Creazione e connessione di un cluster di database Aurora PostgreSQL.

- Abilitare l'API dati durante la creazione del cluster di database Aurora PostgreSQL. Per ulteriori informazioni sulle versioni supportate, consulta Utilizzo dell'API dati di Amazon RDS.
- Assicurarsi di prendere nota del nome della risorsa Amazon (ARN) del cluster di database Aurora PostgreSQL. Servirà per configurare il cluster di database da utilizzare con Amazon Bedrock. Per ulteriori informazioni, consulta Amazon Resource Names (ARNs).

Connessione a un database e installazione di pgvector

È possibile connettersi ad Aurora PostgreSQL utilizzando un'utilità di connessione. Per informazioni più dettagliate su queste utilità, consulta Connessione a un cluster di database Amazon Aurora PostgreSQL. In alternativa, è possibile utilizzare l'editor di query della console RDS per eseguire le query. Per utilizzare l'editor di query è necessario un cluster di database Aurora con l'API dati RDS abilitata.

1. Accedere al database con l'utente principale e configurare pgvector. Utilizzare il comando seguente se l'estensione non è installata:

   ```
   CREATE EXTENSION IF NOT EXISTS vector;
   ```

   Utilizzare `pgvector` 0.5.0 e le versioni successive che supportano l'indicizzazione HNSW. Per ulteriori informazioni, consulta pgvector v0.5.0 with HNSW indexing.
2. Utilizzare il comando seguente per verificare la versione di `pg_vector` installata:

   ```
   SELECT extversion FROM pg_extension WHERE extname='vector';
   ```

Impostazione degli oggetti e dei privilegi del database

1. Creare uno schema specifico che Bedrock può utilizzare per eseguire query sui dati. Utilizzare il comando seguente per creare uno schema:

```
CREATE SCHEMA bedrock_integration;
```

2. Creare un nuovo ruolo che Bedrock può utilizzare per eseguire query sul database. Utilizzare il comando seguente per creare un nuovo ruolo:

```
CREATE ROLE bedrock_user WITH PASSWORD 'password' LOGIN;
```

> **Note**
>
> Prendere nota di questa password, poiché sarà necessaria in seguito per creare una password di Secrets Manager.

Se si utilizza il client `psql`, utilizzare i comandi seguenti per creare un nuovo ruolo:

```
CREATE ROLE bedrock_user LOGIN;
\PASSWORD password;
```

3. Fornire le autorizzazioni `bedrock_user` per gestire lo schema `bedrock_integration`. Questa operazione consente di creare tabelle o indici all'interno dello schema.

```
GRANT ALL ON SCHEMA bedrock_integration to bedrock_user;
```

4. Effettuare l'accesso come `bedrock_user` e creare una tabella in `bedrock_integration schema`.

```
CREATE TABLE bedrock_integration.bedrock_kb (id uuid PRIMARY KEY, embedding
 vector(n), chunks text, metadata json, custom_metadata jsonb);
```

Questo comando crea la tabella `bedrock_kb` nello schema `bedrock_integration` con gli embedding Titan.

Sostituire n nel tipo di dati `vector(n)` con la dimensione appropriata per il modello di embedding in uso. I consigli riportati di seguito aiutano a selezionare le dimensioni:

- Per il modello Titan v2, utilizzare `vector(1024)`, `vector(512)` o `vector (256)`. Per ulteriori informazioni, consulta Amazon Titan Embeddings Text.
- Per il modello Titan v1.2, utilizzare `vector(1536)`. Per ulteriori informazioni, consulta Amazon Titan Multimodal Embeddings G1.
- Per il modello Cohere Embed, utilizzare `vector(1024)`. Per ulteriori informazioni, consulta Modelli Cohere Embed.
- Per Cohere Embed Multilingual v3, utilizzare `vector(1024)`.

Le prime quattro colonne sono obbligatorie. Per la gestione dei metadati, Bedrock scrive i dati dei file di metadati nella colonna `custom_metadata`. È consigliabile creare questa colonna se si intendono utilizzare metadati e filtri. Se non si crea una colonna `custom_metadata`, aggiungere singole colonne per ogni attributo di metadati nella tabella prima di iniziare l'importazione. Per ulteriori informazioni, consulta Configure and customize queries and response generation.

5. I seguenti passaggi consentono di creare gli indici richiesti che Bedrock utilizza per eseguire query sui dati:

   - Creare un indice con l'operatore coseno che Bedrock può utilizzare per eseguire query sui dati.

   ```
   CREATE INDEX ON bedrock_integration.bedrock_kb USING hnsw (embedding
    vector_cosine_ops);
   ```

   - È consigliabile impostare il valore `ef_construction` su 256 per `pgvector` 0.6.0 e versioni successive che utilizzano la creazione di indici paralleli.

   ```
   CREATE INDEX ON bedrock_integration.bedrock_kb USING hnsw (embedding
    vector_cosine_ops) WITH (ef_construction=256);
   ```

   - Creare un indice che Bedrock può utilizzare per eseguire query sui dati di testo.

   ```
   CREATE INDEX ON bedrock_integration.bedrock_kb USING gin (to_tsvector('simple',
    chunks));
   ```

   - Se è stata creata una colonna per i metadati personalizzati, creare un indice che Bedrock può utilizzare per eseguire query sui metadati.

   ```
   CREATE INDEX ON bedrock_integration.bedrock_kb USING gin (custom_metadata);
   ```

Creazione di un segreto in Secrets Manager

Secrets Manager consente di archiviare le credenziali Aurora per poterle trasmettere in modo sicuro alle applicazioni. Se non è stata scelta l'opzione di gestione dei segreti AWS durante la creazione del cluster di database Aurora PostgreSQL, è possibile creare un segreto ora. Per ulteriori informazioni sulla creazione di un segreto di database Gestione dei segreti AWS, consulta AWS Secrets Manager database secret.

Creazione di una Knowledge Base nella console Bedrock

Durante la preparazione di Aurora PostgreSQL per l'utilizzo come archivio vettoriale per una Knowledge Base, è necessario raccogliere i seguenti dettagli da fornire alla console Amazon Bedrock.

- ARN del cluster di database Amazon Aurora: l'ARN del cluster di database.
- ARN segreto: l'ARN della chiave AWS Secrets Manager per il cluster di database.
- Nome del database: il nome del database. Ad esempio, è possibile utilizzare il database predefinito *`postgres`*.
- Nome della tabella: è consigliabile fornire il nome completo dello schema durante la creazione della tabella utilizzando un comando simile al seguente:

  ```
  CREATE TABLE bedrock_integration.bedrock_kb;
  ```

  Questo comando creerà la tabella `bedrock_kb` nello schema `bedrock_integration`.
- Quando si crea la tabella, è necessario configurarla con le colonne e i tipi di dati specificati. È possibile utilizzare nomi di colonna preferiti anziché quelli elencati nella tabella. Occorre prendere nota dei nomi scelti per utilizzarli come riferimento durante la configurazione della Knowledge Base.

Nome colonna	Tipo di dati	Descrizione
id	UUID (chiave primaria)	Contiene identificatori univoci per ogni record.
blocchi	Testo	Contiene i blocchi di testo non elaborato provenienti dalle origini dati.

Nome colonna	Tipo di dati	Descrizione
incorporamento	Vettore	Contiene gli incorporamenti vettoriali delle origini dati.
metadata	JSON	Contiene i metadati necessari per eseguire l'attribuzione dell'origine e per consentir e l'importazione dei dati e l'esecuzione di query.
metadati_personalizzati	JSONB	(Facoltativo) Definisce la colonna di destinazione in cui Amazon Bedrock scrive i dettagli dei metadati dalle origini dati.

Con questi dettagli, ora è possibile creare una Knowledge Base nella console Bedrock. Per informazioni più dettagliate sulla configurazione di un indice vettoriale e sulla creazione di informazioni della Knowledge Base, consulta Creazione di un archivio vettoriale in Amazon Aurora e Creazione di un archivio vettoriale in Amazon Aurora.

Dopo aver aggiunto Aurora come Knowledge Base, è possibile importare le origini dati per eseguire ricerche e query. Per ulteriori informazioni, consulta Inserisci le tue origini dati nella knowledge base.

Creazione rapida di una Knowledge Base per Amazon Bedrock di Aurora PostgreSQL

Il flusso di lavoro della generazione potenziata da recupero dati (RAG) di Amazon Bedrock si basa su dati vettoriali memorizzati in un database Aurora PostgreSQL per potenziare il recupero dei contenuti. In precedenza, la configurazione di Aurora PostgreSQL come archivio dati vettoriale per le Knowledge Base per Bedrock era un processo in più fasi, che richiedeva numerose azioni manuali su diverse interfacce utente. Ciò ha reso difficile per i data scientist e gli sviluppatori sfruttare Aurora per i progetti Bedrock.

Per migliorare l'esperienza utente, AWS ha creato una nuova opzione di creazione rapida basata su CloudFormation che semplifica il processo di configurazione. Con la creazione rapida di, ora è

possibile effettuare il provisioning di un cluster di database Aurora PostgreSQL preconfigurato come archivio vettoriale per le Knowledge Base per Amazon Bedrock con un solo clic.

Argomenti

- Regioni e versioni supportate di Aurora PostgreSQL
- Informazioni sul processo di creazione rapida
- Vantaggi dell'utilizzo della creazione rapida Aurora
- Limitazioni del processo di creazione rapida Aurora

Regioni e versioni supportate di Aurora PostgreSQL

L'opzione di creazione rapida di Aurora è disponibile in tutte le Regioni AWS che supportano le Knowledge Base per Amazon Bedrock. Per impostazione predefinita, crea un cluster di database Aurora PostgreSQL con versione 15.7. Per ulteriori informazioni sulle Regioni supportate, consulta Supported models and regions for Amazon Bedrock Knowledge Bases.

Informazioni sul processo di creazione rapida

Il processo di creazione rapida effettua automaticamente il provisioning delle seguenti risorse per configurare un database Amazon Aurora PostgreSQL come archivio dati vettoriale per la Knowledge Base per Amazon Bedrock:

Un cluster di database Aurora PostgreSQL nel proprio account, configurato con impostazioni predefinite.

- Le unità di capacità Aurora (ACU, Aurora Capacity Units) sono impostate da 0 a 16. Ciò consente di ridurre verticalmente fino a zero l'archivio vettoriale quando non è in uso, risparmiando sui costi di elaborazione. Le ACU possono essere modificate in un secondo momento nella console Amazon RDS.
- Indice HNSW (Hierarchical Navigable Small World) che utilizza la distanza euclidea come misura di somiglianza per gli embedding vettoriali Bedrock memorizzati in Aurora.
- L'istanza database è un'istanza serverless v2.
- Il cluster è associato al VPC e alle sottoreti predefiniti e ha l'API dati RDS abilitata.
- Le credenziali di amministratore del cluster sono gestite da AWS Secrets Manager.

Oltre alle impostazioni predefinite, vengono configurate automaticamente le seguenti impostazioni. Durante il processo, verranno visualizzate delle schermate che spiegano il flusso di lavoro.

- L'inizializzazione del cluster Aurora con gli oggetti di database necessari:
  - Creare l'estensione pgvector, lo schema, il ruolo e le tabelle necessari per la Knowledge Base per Bedrock.
  - Registra un utente del database con privilegi limitati per consentire a Bedrock di interagire con il cluster.
- Durante tutto il processo di provisioning delle risorse verrà visualizzato un banner di avanzamento, che consente di tenere traccia dello stato dei seguenti eventi secondari:
  - Creazione dei cluster Aurora
  - Inizializzazione del cluster Aurora
  - Creazione della Knowledge Base

  Il banner rimane visibile fino alla creazione completa della knowledge base, anche se si esce dalla pagina e si torna indietro.
- È possibile fare clic su `View details` nel banner di avanzamento per visualizzare lo stato di ogni passaggio. Per ulteriori informazioni sugli eventi durante la creazione della knowledge base, scegliere il link CloudFormation nella schermata di visualizzazione dei dettagli. Una volta completato il processo, la nuova Knowledge Base per Bedrock sarà pronta per l'uso.
- Gli ID dello stack per tutte le risorse di creazione rapida sono disponibili nei tag della Knowledge Base per Bedrock, nel caso in cui fosse necessario farvi riferimento.

Viene creata una Knowledge Base per Bedrock, con la configurazione del cluster Aurora di cui è stato appena effettuato il provisioning come archivio vettoriale.

Vantaggi dell'utilizzo della creazione rapida Aurora

- Il processo di creazione rapida basato su CloudFormation riduce significativamente il tempo e la complessità necessari per utilizzare Aurora come archivio vettoriale.
- Aurora offre prestazioni eccellenti, scalabilità vettoriale e vantaggi in termini di costi con la possibilità di scalare fino a zero i costi di elaborazione quando non viene utilizzato.
- Il processo di creazione rapida semplifica l'esperienza end-to-end, consentendo di creare e configurare facilmente le Knowledge Base per Bedrock utilizzando Aurora.

- I clienti possono basarsi sul modello CloudFormation per personalizzare il provisioning con le proprie configurazioni.

Limitazioni del processo di creazione rapida Aurora

- Con l'opzione di creazione rapida Aurora, viene effettuato il provisioning del cluster di database con le configurazioni predefinite. Tuttavia, queste impostazioni predefinite potrebbero non soddisfare i requisiti specifici o il caso d'uso previsto. La creazione rapida non offre opzioni per modificare le configurazioni durante il processo di provisioning. Le configurazioni vengono impostate automaticamente per semplificare l'esperienza di implementazione. Se è necessario personalizzare la configurazione del cluster di database Aurora, è possibile farlo dopo l'implementazione iniziale tramite la creazione rapida nella console Amazon RDS.
- Sebbene il flusso di creazione rapida semplifica il processo di configurazione, il tempo necessario per creare il cluster di database Aurora è ancora di circa 10 minuti, lo stesso di un'implementazione manuale. Ciò è dovuto al tempo necessario per effettuare il provisioning dell'infrastruttura Aurora.
- L'opzione di creazione rapida è progettata per la sperimentazione e la configurazione rapida. Le risorse create tramite la creazione rapida potrebbero non essere adatte all'uso in produzione e non sarà possibile migrarle direttamente in un ambiente di produzione nel proprio VPC.

Integrazione di Amazon Aurora PostgreSQL con altri servizi AWS

Amazon Aurora si integra con altri servizi AWS per permettere di estendere il cluster database Aurora PostgreSQL allo scopo di utilizzare capacità aggiuntive in AWS Cloud. Il cluster database Aurora PostgreSQL può utilizzare i servizi AWS per gli scopi seguenti:

- Raccolta, visualizzazione e valutazione delle prestazioni in modo rapido per le istanze database Aurora PostgreSQL con Performance Insights di Amazon RDS. Performance Insights si espande sulle caratteristiche di monitoraggio esistenti Amazon RDS per illustrare le prestazioni del database e aiutare ad analizzare eventuali problemi che lo riguardano. Con il pannello di controllo di Performance Insights, è possibile visualizzare il carico del database e filtrare il carico in base alle attese, alle istruzioni SQL, agli host o agli utenti. Per ulteriori informazioni su Performance Insights, consulta Monitoraggio del carico DB con Performance Insights su Amazon Aurora.
- Puoi configurare il cluster database Aurora PostgreSQL per la pubblicazione dei dati di log in Amazon CloudWatch Logs. CloudWatch Logs fornisce spazio di archiviazione estremamente durevole per i dati di log. Con CloudWatch Logs puoi eseguire analisi in tempo reale dei dati di log

e usare CloudWatch per creare allarmi e visualizzare parametri. Per ulteriori informazioni, consulta Pubblicazione di log Aurora PostgreSQL in Amazon CloudWatch Logs.

- Importa dati da un bucket Amazon S3 in un cluster DB Aurora PostgreSQL oppure esporta dati da un cluster DB Aurora PostgreSQL in un bucket Amazon S3. Per ulteriori informazioni, consultare Importazione di dati da Amazon S3 in un cluster database Aurora PostgreSQL e Esportazione di dati da del cluster di database Aurora PostgreSQLRDS per PostgreSQL a Amazon S3.
- Aggiungere previsioni basate sul machine learning alle applicazioni di database utilizzando il linguaggio SQL. Il machine learning di Aurora è un'integrazione altamente ottimizzata tra il database Aurora e i servizi di machine learning (ML) di AWS SageMaker AI e Amazon Comprehend. Per ulteriori informazioni, consulta Utilizzo del machine learning di Amazon Aurora con Aurora PostgreSQL.
- Richiamo di una funzione AWS Lambda da un cluster DB Aurora PostgreSQL. Per fare ciò, utilizzare l'estensione PostgreSQL `aws_lambda` fornita con Aurora PostgreSQL. Per ulteriori informazioni, consulta .
- Integra le query di Amazon Redshift e Aurora PostgreSQL. Per ulteriori informazioni, consulta Nozioni di base sull'utilizzo di query federate su PostgreSQL nella Guida per gli sviluppatori Amazon Redshift Database.

Importazione di dati da Amazon S3 in un cluster database Aurora PostgreSQL

Puoi importare i dati che sono stati archiviati utilizzando Servizio di archiviazione semplice Amazon in una tabella su un'istanza cluster database Aurora PostgreSQL. A questo scopo, installa innanzitutto l'estensione Aurora PostgreSQL `aws_s3`. Questa estensione fornisce le funzioni utilizzate per importare i dati da un bucket Amazon S3. Un bucket è un container Amazon S3 per oggetti e file. I dati possono trovarsi in un file con valori separati da virgole (CSV), un file di testo o un file compresso (gzip). Di seguito, sono fornite informazioni su come installare l'estensione e come importare dati da Amazon S3 in una tabella.

Per eseguire l'importazione da Amazon S3 a , il database deve eseguire PostgreSQL versione 10.7 o successive. Aurora PostgreSQL.

Se Amazon S3 non contiene dati, occorre innanzitutto creare un bucket e archiviare i dati. Per ulteriori informazioni, consulta i seguenti argomenti nella Guida per l'utente di Servizio di archiviazione semplice Amazon.

- Creazione di un bucket
- Aggiunta di un oggetto a un bucket.

È supportata l'importazione multiaccount da Amazon S3. Per ulteriori informazioni, consulta Concessione di autorizzazioni multiaccount nella Guida per l'utente di Amazon Simple Storage Service.

Puoi utilizzare la chiave gestita dal cliente per la crittografia durante l'importazione dei dati da S3. Per ulteriori informazioni, consulta Chiavi KMS archiviate in AWS KMS nella Guida per l'utente di Amazon Simple Storage Service.

> **Note**
>
> L'importazione di dati da Amazon S3 non è supportata per Aurora Serverless v1. È supportata per Aurora Serverless v2.

Argomenti

- Installazione dell'estensione aws_s3
- Panoramica dell'importazione di dati dai dati di Amazon S3
- Configurazione dell'accesso a un bucket Simple Storage Service (Amazon S3)
- Importazione di dati da Amazon S3 nel cluster database Aurora PostgreSQL
- Informazioni di riferimento sulle funzioni

Installazione dell'estensione aws_s3

Prima di poter utilizzare Amazon S3 con il cluster database Aurora PostgreSQL, è necessario installare l'estensione `aws_s3`. Questa estensione fornisce funzioni per l'importazione dei dati da Amazon S3. Inoltre, fornisce funzioni per l'esportazione di dati da un'istanza di un cluster database Aurora PostgreSQL in un bucket Amazon S3. Per ulteriori informazioni, consulta Esportazione di dati da del cluster di database Aurora PostgreSQLRDS per PostgreSQL a Amazon S3. L'estensione `aws_s3` dipende da alcune delle funzioni helper nell'estensione `aws_commons`, che vengono installate automaticamente quando necessario.

Per installare l'estensione **aws_s3**

1. Usa psql (o pgAdmin) per connetterti all'istanza di scrittura del cluster database Aurora PostgreSQL come un utente che dispone di privilegi `rds_superuser`. Se hai mantenuto il nome predefinito durante il processo di configurazione, esegui la connessione come `postgres`.

```
psql --host=111122223333.aws-region.rds.amazonaws.com --port=5432 --
username=postgres --password
```

2. Per installare l'estensione, esegui il comando seguente.

```
postgres=> CREATE EXTENSION aws_s3 CASCADE;
NOTICE: installing required extension "aws_commons"
CREATE EXTENSION
```

3. Per verificare che l'estensione sia installata, puoi usare il metacomando psql `\dx`.

```
postgres=> \dx
       List of installed extensions
    Name     | Version |   Schema   |                 Description
-------------+---------+------------+---------------------------------------------
 aws_commons | 1.2     | public     | Common data types across AWS services
 aws_s3      | 1.1     | public     | AWS S3 extension for importing data from S3
 plpgsql     | 1.0     | pg_catalog | PL/pgSQL procedural language
(3 rows)
```

Le funzioni per importare dati da Amazon S3 ed esportare dati in Amazon S3 sono ora disponibili per l'uso.

Panoramica dell'importazione di dati dai dati di Amazon S3

Per importare i dati S3 in Aurora PostgreSQL

Raccogli innanzitutto i dettagli che devi fornire alla funzione. Questi includono il nome della tabella sull'istanza del cluster di database Aurora PostgreSQL, e il nome del bucket, il percorso del file, il tipo di file eRegione AWSdove sono memorizzati i dati Amazon S3. Per ulteriori informazioni, consulta Visualizzazione di un oggetto nella Guida per l'utente di Servizio di archiviazione semplice Amazon.

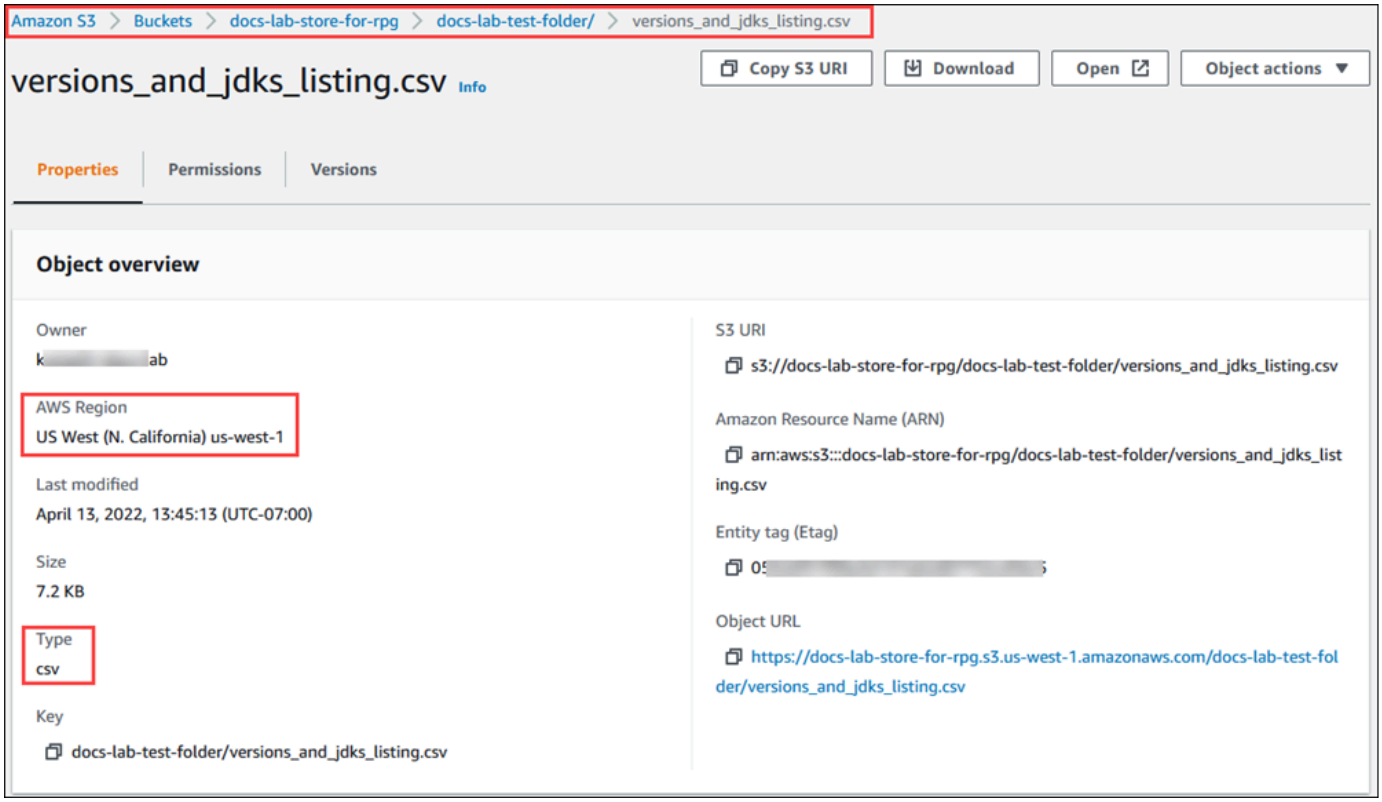
> **Note**
>
> L'importazione in più parti da Amazon S3 non è attualmente supportata.

1. Otteni il nome della tabella in cui la funzione `aws_s3.table_import_from_s3` deve importare dati. Il seguente comando, ad esempio, crea una tabella `t1` che può essere utilizzata in fasi successive.

   ```
   postgres=> CREATE TABLE t1
       (col1 varchar(80),
       col2 varchar(80),
       col3 varchar(80));
   ```

2. Ottieni i dettagli relativi al bucket Amazon S3 e i dati da importare. A tale scopo, apri la console Amazon S3 all'indirizzo https://console.aws.amazon.com/s3/ e scegli Buckets (Bucket). Individua il bucket contenente i dati nell'elenco. Scegli il bucket, apri la pagina Object overview (Panoramica degli oggetti) e quindi scegli Properties (Proprietà).

   Prendi nota del nome del bucket, del percorso, della Regione AWS e del tipo di file. Il nome della risorsa Amazon (ARN) è richiesto in un secondo momento per configurare l'accesso ad Amazon S3 tramite un ruolo IAM. Per ulteriori informazioni, consulta Configurazione dell'accesso a un bucket Simple Storage Service (Amazon S3). Un esempio è illustrato nell'immagine seguente.



3. Puoi verificare il percorso ai dati sul bucket Amazon S3 utilizzando il comando AWS CLI `aws s3 cp`. Se le informazioni sono corrette, questo comando scarica una copia del file Amazon S3.

```
aws s3 cp s3://amzn-s3-demo-bucket/sample_file_path ./
```

4. Configura le autorizzazioni sul cluster database Aurora PostgreSQL per consentire l'accesso al file sul bucket Amazon S3. A questo scopo, utilizza un ruolo AWS Identity and Access Management (IAM) o le credenziali di sicurezza. Per ulteriori informazioni, consulta Configurazione dell'accesso a un bucket Simple Storage Service (Amazon S3).

5. Fornisci alla funzione `create_s3_uri` il percorso e gli altri dettagli dell'oggetto Amazon S3 raccolti (vedi passaggio 2) per costruire un oggetto URI Amazon S3. Per ulteriori informazioni su questa funzione, consulta aws_commons.create_s3_uri. Di seguito è riportato un esempio di costruzione dell'oggetto durante una sessione psql.

```
postgres=> SELECT aws_commons.create_s3_uri(
   'docs-lab-store-for-rpg',
   'versions_and_jdks_listing.csv',
   'us-west-1'
) AS s3_uri \gset
```

Nella fase seguente, si passa questo oggetto (`aws_commons._s3_uri_1`) alla funzione `aws_s3.table_import_from_s3` per importare i dati nella tabella.

6. Invoca la funzione `aws_s3.table_import_from_s3` per importare dati da Amazon S3 nella tabella. Per informazioni di riferimento, consulta aws_s3.table_import_from_s3. Per alcuni esempi, consulta Importazione di dati da Amazon S3 nel cluster database Aurora PostgreSQL.

Configurazione dell'accesso a un bucket Simple Storage Service (Amazon S3)

Per importare i dati da un file Amazon S3, concedere al cluster database Aurora PostgreSQL a l'autorizzazione ad accedere al bucket Amazon S3 che contiene il file. Puoi concedere l'accesso a un bucket Amazon S3 in uno dei due modi descritti negli argomenti seguenti.

Argomenti

- Utilizzo di un ruolo IAM per accedere a un bucket Amazon S3.
- Utilizzo delle credenziali di sicurezza per accedere a un bucket Amazon S3
- Risoluzione dei problemi di accesso a Amazon S3

Utilizzo di un ruolo IAM per accedere a un bucket Amazon S3.

Prima di caricare i dati da un file Amazon S3, è necessario concedere al cluster di database Aurora PostgreSQL l'autorizzazione per accedere al bucket Amazon S3 che contiene il file. In questo modo non dovrai gestire ulteriori informazioni sulle credenziali né fornirle nella chiamata della funzione aws_s3.table_import_from_s3.

Per svolgere questa operazione, creare una policy IAM che fornisca accesso al bucket Amazon S3. Creare un ruolo IAM e collegarvi la policy. Quindi, assegnare il ruolo IAM al cluster .

> **Note**
>
> Non è possibile associare un ruolo IAM a un cluster di database Aurora Serverless v1, quindi i seguenti passaggi non sono attinenti.

Per consentire a un cluster database Aurora PostgreSQL l'accesso ad Amazon S3 tramite un ruolo IAM

1. Creare una policy IAM

Questa policy fornisce le autorizzazioni bucket e di oggetto che consentono al cluster di database Aurora PostgreSQL di accedere a Amazon S3.

Includere nella policy le seguenti operazioni necessarie per consentire il trasferimento dei file da un bucket Amazon S3 a Aurora PostgreSQL:

- `s3:GetObject`
- `s3:ListBucket`

Includere nella policy le seguenti risorse per identificare il bucket Amazon S3 e gli oggetti nel bucket. Questo mostra il formato Amazon Resource Name (ARN) per accedere a Amazon S3.

- arn:aws:s3::: *`amzn-s3-demo-bucket`*
- arn:aws:s3::: /* *`amzn-s3-demo-bucket`*

Per ulteriori informazioni sulla creazione di una policy IAM per Aurora PostgreSQL, consulta Creazione e utilizzo di una policy IAM per l'accesso al database IAM. Consulta anche il Tutorial: Creare e collegare la prima policy gestita dal cliente nella Guida per l'utente di IAM.

Il AWS CLI comando seguente crea una policy IAM denominata con queste opzioni. `rds-s3-import-policy` Concede l'accesso a un bucket denominato *`amzn-s3-demo-bucket`*.

> **Note**
>
> Prendi nota del nome della risorsa Amazon (ARN) della policy restituita mediante questo comando. L'ARN sarà richiesto in una fase successiva quando si associa la policy a un ruolo IAM.

Example

Per Linux, macOS o Unix:

```
aws iam create-policy \
   --policy-name rds-s3-import-policy \
   --policy-document '{
     "Version": "2012-10-17",
     "Statement": [
```

```
      {
        "Sid": "s3import",
        "Action": [
          "s3:GetObject",
          "s3:ListBucket"
        ],
        "Effect": "Allow",
        "Resource": [
          "arn:aws:s3:::amzn-s3-demo-bucket",
          "arn:aws:s3:::amzn-s3-demo-bucket/*"
        ]
      }
    ]
  }'
```

Per Windows:

```
aws iam create-policy ^
   --policy-name rds-s3-import-policy ^
   --policy-document '{
     "Version": "2012-10-17",
     "Statement": [
       {
         "Sid": "s3import",
         "Action": [
           "s3:GetObject",
           "s3:ListBucket"
         ],
         "Effect": "Allow",
         "Resource": [
           "arn:aws:s3:::amzn-s3-demo-bucket",
           "arn:aws:s3:::amzn-s3-demo-bucket/*"
         ]
       }
     ]
   }'
```

2. Crea un ruolo IAM.

   In questo modo, Aurora PostgreSQL può assumere questo ruolo IAM per tuo conto, per accedere ai bucket Amazon S3. Per ulteriori informazioni, consulta la pagina relativa alla creazione di un ruolo per delegare le autorizzazioni a un utente IAM nella Guida per l'utente IAM.

Si consiglia di utilizzare le chiavi di contesto delle condizioni globali `aws:SourceArn` e `aws:SourceAccount` nelle policy basate sulle risorse per limitare le autorizzazioni del servizio a una risorsa specifica. Questo è il modo più efficace per proteggersi dal problema di deputy confused.

Se si utilizzano entrambe le chiavi di contesto delle condizioni globali e il valore `aws:SourceArn` contiene l'ID account, il valore `aws:SourceAccount` e l'account nel valore `aws:SourceArn` devono utilizzare lo stesso ID account quando viene utilizzato nella stessa dichiarazione di policy.

- Utilizzare `aws:SourceArn` se si desidera un accesso cross-service per una singola risorsa.
- Utilizzare `aws:SourceAccount` se si desidera consentire l'associazione di qualsiasi risorsa in tale account all'uso cross-service.

Nella policy, assicurarsi di utilizzare la chiave di contesto della condizione globale `aws:SourceArn` con l'ARN completo della risorsa. L'esempio seguente mostra come eseguire questa operazione utilizzando il AWS CLI comando per creare un ruolo denominato`rds-s3-import-role`.

Example

Per Linux, macOS o Unix:

```
aws iam create-role \
   --role-name rds-s3-import-role \
   --assume-role-policy-document '{
     "Version": "2012-10-17",
     "Statement": [
       {
         "Effect": "Allow",
         "Principal": {
            "Service": "rds.amazonaws.com"
          },
         "Action": "sts:AssumeRole",
         "Condition": {
             "StringEquals": {
                "aws:SourceAccount": "111122223333",
                "aws:SourceArn": "arn:aws:rds:us-
east-1:111122223333:cluster:clustername"
```

```
                }
            }
        }
    ]
}'
```

Per Windows:

```
aws iam create-role ^
   --role-name rds-s3-import-role ^
   --assume-role-policy-document '{
     "Version": "2012-10-17",
     "Statement": [
       {
         "Effect": "Allow",
         "Principal": {
            "Service": "rds.amazonaws.com"
          },
         "Action": "sts:AssumeRole",
         "Condition": {
             "StringEquals": {
                "aws:SourceAccount": "111122223333",
                "aws:SourceArn": "arn:aws:rds:us-
east-1:111122223333:cluster:clustername"
                }
            }
       }
     ]
   }'
```

3. Collegare la policy IAM al ruolo IAM creato.

   Il AWS CLI comando seguente collega la policy creata nel passaggio precedente al ruolo denominato `rds-s3-import-role` Replace *`your-policy-arn`* con l'ARN della policy annotato in un passaggio precedente.

   Example

   Per Linux, macOS o Unix:

```
aws iam attach-role-policy \
   --policy-arn your-policy-arn \
```

```
    --role-name rds-s3-import-role
```

Per Windows:

```
aws iam attach-role-policy ^
   --policy-arn your-policy-arn ^
   --role-name rds-s3-import-role
```

4. Aggiungere il ruolo IAM al cluster.

   A tale scopo, utilizzare Console di gestione AWS o AWS CLI, come descritto di seguito.

Console

Per aggiungere un ruolo IAM al cluster di database PostgreSQL tramite la console

1. Accedi a Console di gestione AWS e apri la console Amazon RDS all'indirizzo https://console.aws.amazon.com/rds/.
2. Scegliere il nome del cluster di database PostgreSQL per visualizzarne i dettagli.
3. Nella scheda Connettività e sicurezza, nella sezione Gestisci ruoli IAM, scegli il ruolo da aggiungere in Aggiungi ruoli IAM a questa istanza del .
4. In Feature (Caratteristica), scegliere s3Import.
5. Scegliere Add role (Aggiungi ruolo).

AWS CLI

Per aggiungere un ruolo IAM a un cluster di database PostgreSQL tramite CLI

- Utilizzare il seguente comando per aggiungere il ruolo al cluster di database PostgreSQL denominato `my-db-cluster`. Sostituire *`your-role-arn`* con l'ARN del ruolo annotato in precedenza. Utilizzare `s3Import` come valore dell'opzione `--feature-name`.

  Example

  Per Linux, macOS o Unix:

```
aws rds add-role-to-db-cluster \
   --db-cluster-identifier my-db-cluster \
   --feature-name s3Import \
```

```
   --role-arn your-role-arn   \
   --region your-region
```

Per Windows:

```
aws rds add-role-to-db-cluster ^
   --db-cluster-identifier my-db-cluster ^
   --feature-name s3Import ^
   --role-arn your-role-arn ^
   --region your-region
```

API RDS

Per aggiungere un ruolo IAM per un' del cluster DB PostgreSQL utilizzando l'API Amazon RDS, chiama l'operazione. AddRoleToDBCluster

Utilizzo delle credenziali di sicurezza per accedere a un bucket Amazon S3

Se preferisci, puoi utilizzare le credenziali di sicurezza per fornire accesso a un bucket Amazon S3 invece di fornire accesso con un ruolo IAM. A tale scopo, specifica il parametro `credentials` nella chiamata di funzione aws_s3.table_import_from_s3.

Il `credentials` parametro è una struttura di tipo `aws_commons._aws_credentials_1` che contiene le credenziali. AWS Utilizzare la funzione aws_commons.create_aws_credentials per impostare la chiave di accesso e la chiave segreta in una struttura `aws_commons._aws_credentials_1`, come illustrato di seguito.

```
postgres=> SELECT aws_commons.create_aws_credentials(
   'sample_access_key', 'sample_secret_key', '')
AS creds \gset
```

Dopo aver creato la struttura `aws_commons._aws_credentials_1` , utilizzare la funzione aws_s3.table_import_from_s3 con il parametro `credentials` per importare i dati, come illustrato di seguito.

```
postgres=> SELECT aws_s3.table_import_from_s3(
   't', '', '(format csv)',
   :'s3_uri',
   :'creds'
);
```

Oppure si può includere la chiamata inline di funzione aws_commons.create_aws_credentials all'interno della chiamata di funzione `aws_s3.table_import_from_s3`.

```
postgres=> SELECT aws_s3.table_import_from_s3(
   't', '', '(format csv)',
   :'s3_uri',
   aws_commons.create_aws_credentials('sample_access_key', 'sample_secret_key', '')
);
```

Risoluzione dei problemi di accesso a Amazon S3

Se riscontri problemi di connessione quanto tenti di importare i dati da Amazon S3, segui questi suggerimenti:

- Risoluzione dei problemi di identità e accesso in Amazon Aurora
- Risoluzione dei problemi di Amazon S3 nella Guida per l'utente di Amazon Simple Storage Service
- Risoluzione dei problemi di Amazon S3 e IAM nella Guida per l'utente IAM.

Importazione di dati da Amazon S3 nel cluster database Aurora PostgreSQL

Importa i dati dal bucket Amazon S3 utilizzando la funzione `table_import_from_s3` dell'estensione aws_s3. Per informazioni di riferimento, consulta aws_s3.table_import_from_s3.

> **Note**
>
> Gli esempi seguenti utilizzano il metodo del ruolo IAM per consentire l'accesso al bucket Amazon S3. Pertanto, le chiamate della funzione `aws_s3.table_import_from_s3` non includono parametri di credenziali.

Di seguito viene illustrato un tipico esempio.

```
postgres=> SELECT aws_s3.table_import_from_s3(
   't1',
   '',
   '(format csv)',
   :'s3_uri'
);
```

I parametri sono i seguenti:

- `t1` – Il nome della tabella nel cluster di database PostgreSQL in cui copiare i dati.
- `''` – Un elenco opzionale di colonne nella tabella di database. Questo parametro può essere utilizzato per indicare quali colonne di dati S3 vanno in quali colonne della tabella. Se non viene specificata alcuna colonna, tutte le colonne vengono copiate nella tabella. Per un esempio di utilizzo di un elenco di colonne, consulta Importazione di un file Amazon S3 che utilizza un delimitatore personalizzato.
- `(format csv)` – Argomenti COPY di PostgreSQL. La procedura di copia utilizza gli argomenti e il formato del comando COPY di PostgreSQL per importare i dati. Le scelte di formato includono valori separati da virgole (CSV), come mostrato in questo esempio, testo e file binario. Il valore predefinito è testo.
- `s3_uri` – Una struttura contenente le informazioni che identificano il file Amazon S3. Per un esempio di utilizzo della funzione aws_commons.create_s3_uri per creare una struttura `s3_uri`, consulta Panoramica dell'importazione di dati dai dati di Amazon S3.

Per ulteriori informazioni su questa funzione, consulta aws_s3.table_import_from_s3.

La funzione restituisce `aws_s3.table_import_from_s3`. Per specificare altri tipi di file da importare da un bucket Amazon S3, consulta uno dei seguenti esempi.

> **Note**
>
> L'importazione di un file da 0 byte genererà un errore.

Argomenti

- Importazione di un file Amazon S3 che utilizza un delimitatore personalizzato
- Importazione di un file compresso (gzip) Amazon S3
- Importazione di un file Amazon S3 codificato

Importazione di un file Amazon S3 che utilizza un delimitatore personalizzato

Il seguente esempio mostra come importare un file che utilizza un delimitatore personalizzato. Mostra anche come controllare dove inserire i dati nella tabella di database utilizzando il parametro `column_list` della funzione aws_s3.table_import_from_s3.

In questo esempio si presuppone che le seguenti informazioni siano organizzate in colonne delimitate da pipe nel file Amazon S3.

```
1|foo1|bar1|elephant1
2|foo2|bar2|elephant2
3|foo3|bar3|elephant3
4|foo4|bar4|elephant4
...
```

Per importare un file che utilizza un delimitatore personalizzato

1. Creare una tabella nel database per i dati importati.

   ```
   postgres=> CREATE TABLE test (a text, b text, c text, d text, e text);
   ```

2. Utilizzare il seguente formato della funzione aws_s3.table_import_from_s3 per importare i dati dal file Amazon S3.

   Si può includere la chiamata inline di funzione aws_commons.create_s3_uri all'interno della chiamata di funzione `aws_s3.table_import_from_s3` per specificare il file.

   ```
   postgres=> SELECT aws_s3.table_import_from_s3(
      'test',
      'a,b,d,e',
      'DELIMITER ''|''',
      aws_commons.create_s3_uri('amzn-s3-demo-bucket', 'pipeDelimitedSampleFile', 'us-
   east-2')
   );
   ```

I dati sono ora nella tabella nelle seguenti colonne.

```
postgres=> SELECT * FROM test;
a | b | c | d | e
---+------+---+---+------+-----------
1 | foo1 | | bar1 | elephant1
2 | foo2 | | bar2 | elephant2
3 | foo3 | | bar3 | elephant3
4 | foo4 | | bar4 | elephant4
```

Importazione di un file compresso (gzip) Amazon S3

Il seguente esempio mostra come importare da Amazon S3 un file compresso con gzip. Il file importato deve avere i seguenti metadati Amazon S3:

- Chiave: `Content-Encoding`
- Valore: `gzip`

Se carichi il file utilizzando la Console di gestione AWS, i metadati vengono in genere applicati dal sistema. Per informazioni sul caricamento di file in Amazon S3 utilizzando la Console di gestione AWS, la AWS CLI o l'API, consulta Caricamento degli oggetti nella Guida per l'utente di Amazon Simple Storage Service.

Per ulteriori informazioni sui metadati di Amazon S3 e i dettagli sui metadati forniti dal sistema, consulta la sezione Modifica dei metadati degli oggetti nella console Amazon S3 nella Guida per l'utente di Amazon Simple Storage Service.

Importare il file gzip nel cluster di database Aurora PostgreSQL, come illustrato di seguito.

```
postgres=> CREATE TABLE test_gzip(id int, a text, b text, c text, d text);
postgres=> SELECT aws_s3.table_import_from_s3(
 'test_gzip', '', '(format csv)',
 'amzn-s3-demo-bucket', 'test-data.gz', 'us-east-2'
);
```

Importazione di un file Amazon S3 codificato

Il seguente esempio mostra come importare da Amazon S3 un file codificato con Windows-1252

```
postgres=> SELECT aws_s3.table_import_from_s3(
 'test_table', '', 'encoding ''WIN1252''',
 aws_commons.create_s3_uri('amzn-s3-demo-bucket', 'SampleFile', 'us-east-2')
);
```

Informazioni di riferimento sulle funzioni

Funzioni

- aws_s3.table_import_from_s3
- aws_commons.create_s3_uri

- aws_commons.create_aws_credentials

aws_s3.table_import_from_s3

Importa dati Amazon S3 in una tabella Aurora PostgreSQL. L'estensione `aws_s3` fornisce la funzione `aws_s3.table_import_from_s3`. Il valore restituito è testo.

Sintassi

I parametri richiesti sono `table_name`, `column_list` e `options`. Identificano la tabella di database e specificano il modo in cui i dati vengono copiati nella tabella

Puoi inoltre utilizzare i seguenti parametri:

- Il parametro `s3_info` specifica il file Amazon S3 da importare. Se si utilizza questo parametro, l'accesso a Amazon S3 è fornito da un ruolo IAM per il cluster di database PostgreSQL.

```
aws_s3.table_import_from_s3 (
   table_name text,
   column_list text,
   options text,
   s3_info aws_commons._s3_uri_1
)
```

- Il parametro `credentials` specifica le credenziali per accedere a Amazon S3. Se si utilizza questo parametro, non si utilizza il ruolo IAM.

```
aws_s3.table_import_from_s3 (
   table_name text,
   column_list text,
   options text,
   s3_info aws_commons._s3_uri_1,
   credentials aws_commons._aws_credentials_1
)
```

Parametri

table_name

Una stringa di testo obbligatoria contenente il nome della tabella di database PostgreSQL in cui importare i dati.

column_list

Una stringa di testo obbligatoria contenente un elenco opzionale delle colonne della tabella di database PostgreSQL nelle quali copiare i dati. Se la stringa è vuota, vengono utilizzate tutte le colonne della tabella. Per un esempio, consulta Importazione di un file Amazon S3 che utilizza un delimitatore personalizzato.

options

Una stringa di testo obbligatorio contenente gli argomenti del comando `COPY` di PostgreSQL. Tali argomenti specificano in che modo i dati vengono copiati nella tabella PostgreSQL. Per maggiori dettagli, consulta la documentazione di COPY PostgreSQL.

s3_info

Un tipo composito `aws_commons._s3_uri_1` contenente le seguenti informazioni sull'oggetto S3:

- `bucket` – Il nome del bucket Amazon S3 contenente il file.
- `file_path` – Il nome file di Amazon S3, incluso il percorso.
- `region`: la regione AWS in cui si trova il file. Per un elenco di nomi di regione AWS e dei valori associati, consulta Regioni e zone di disponibilità.

credenziali

Un tipo composito `aws_commons._aws_credentials_1` contenente le seguenti credenziali da utilizzare per l'operazione di importazione:

- Chiave di accesso
- Chiave segreta
- Token di sessione

Per informazioni sulla creazione di una struttura composita `aws_commons._aws_credentials_1`, consulta aws_commons.create_aws_credentials.

Sintassi alternativa

Per un aiuto nei test, si può utilizzare un set più ampio di parametri al posto dei parametri `s3_info` e `credentials`. Di seguito vengono riportate le variazioni di sintassi aggiuntive per la funzione `aws_s3.table_import_from_s3`.

- Invece di utilizzare il parametro `s3_info` per identificare un file Amazon S3, utilizzare la combinazione dei parametri `bucket`, `file_path` e `region`. Con questo formato della funzione, l'accesso a Amazon S3 viene fornito da un ruolo IAM nell'istanza database PostgreSQL.

```
aws_s3.table_import_from_s3 (
   table_name text,
   column_list text,
   options text,
   bucket text,
   file_path text,
   region text
)
```

- Invece di utilizzare il parametro `credentials` per specificare l'accesso a Amazon S3, utilizzare la combinazione dei parametri `access_key`, `session_key` e `session_token`.

```
aws_s3.table_import_from_s3 (
   table_name text,
   column_list text,
   options text,
   bucket text,
   file_path text,
   region text,
   access_key text,
   secret_key text,
   session_token text
)
```

Parametri alternativi

bucket

Una stringa di testo contenente il nome del bucket Amazon S3 che contiene il file

file_path

Una stringa di testo contenente il nome file di Amazon S3, incluso il percorso.

Regione

Una stringa di testo che identifica la posizione Regione AWS del file. Per un elenco di nomi di Regione AWS e di valori associati, consulta Regioni e zone di disponibilità.

chiave_accesso

Una stringa di testo contenente la chiave di accesso da utilizzare per l'operazione di importazione. Il valore predefinito è NULL.

secret_key

Una stringa di testo contenente la chiave segreta da utilizzare per l'operazione di importazione. Il valore predefinito è NULL.

session_token

(Opzionale) Una stringa di testo contenente la chiave di sessione da utilizzare per l'operazione di importazione. Il valore predefinito è NULL.

aws_commons.create_s3_uri

Crea una struttura `aws_commons._s3_uri_1` per conservare le informazioni relative al file Amazon S3. Si utilizzano i risultati della funzione `aws_commons.create_s3_uri` nel parametro `s3_info` della funzione aws_s3.table_import_from_s3.

Sintassi

```
aws_commons.create_s3_uri(
   bucket text,
   file_path text,
   region text
)
```

Parametri

bucket

Una stringa di testo obbligatoria contenente il nome del bucket Amazon S3 del file.

file_path

Una stringa di testo obbligatoria contenente il nome file di Amazon S3, incluso il percorso.

Regione

Una stringa di testo obbligatoria contenente la Regione AWS in cui si trova il file. Per un elenco di nomi di Regione AWS e di valori associati, consulta Regioni e zone di disponibilità.

aws_commons.create_aws_credentials

Imposta una chiave di accesso e una chiave segreta in una struttura `aws_commons._aws_credentials_1`. Si utilizzano i risultati della funzione `aws_commons.create_aws_credentials` nel parametro `credentials` della funzione aws_s3.table_import_from_s3.

Sintassi

```
aws_commons.create_aws_credentials(
   access_key text,
   secret_key text,
   session_token text
)
```

Parametri

chiave_accesso

Una stringa di testo obbligatoria contenente la chiave di accesso da utilizzare per l'importazione di un file Amazon S3. Il valore predefinito è NULL.

secret_key

Una stringa di testo obbligatoria contenente la chiave segreta da utilizzare per l'importazione di un file Amazon S3. Il valore predefinito è NULL.

session_token

Una stringa di testo opzionale contenente il token di sessione da utilizzare per l'importazione di un file Amazon S3. Il valore predefinito è NULL. Se si fornisce un `session_token` opzionale, è possibile utilizzare credenziali temporanee.

Esportazione di dati da del cluster di database Aurora PostgreSQLRDS per PostgreSQL a Amazon S3

È possibile eseguire query sui dati da del cluster di database Aurora PostgreSQLRDS per PostgreSQL ed esportarli direttamente in file memorizzati in un bucket Amazon S3. A questo scopo, installa innanzitutto l'estensione Aurora PostgreSQL `aws_s3`. Questa estensione fornisce le funzioni utilizzate per esportare i risultati delle quey in Amazon S3. Di seguito, sono disponibili informazioni su come installare l'estensione ed esportare i dati in Amazon S3.

Puoi esportare da un'istanza database con provisioning o Aurora Serverless v2. Questi passaggi non sono supportati per Aurora Serverless v1.

> **Note**
>
> L'esportazione tra account in Amazon S3 non è supportata.

Tutte le versioni attualmente disponibili di Aurora PostgreSQL supportano l'esportazione dei dati in Servizio di archiviazione semplice Amazon. Per informazioni dettagliate sulla versione, consulta gli aggiornamenti di Amazon Aurora PostgreSQL nelle Note di rilascio di Aurora PostgreSQL.

Se non disponi di un bucket configurato per l'esportazione, consulta i seguenti argomenti nella Guida per l'utente di Servizio di archiviazione semplice Amazon.

- Configurazione di Amazon S3
- Creazione di un bucket

Per impostazione predefinita, i dati esportati da Aurora PostgreSQL su Amazon S3 utilizzano la crittografia lato server con una Chiave gestita da AWS. In alternativa, è possibile utilizzare la chiave gestita dal cliente che è già stata creata. Se si utilizza la crittografia del bucket, il bucket Amazon S3 deve essere crittografato con una chiave AWS Key Management Service (AWS KMS) (SSE-KMS). Attualmente, i bucket crittografati con chiavi gestite da Amazon S3 (SSE-S3) non sono supportati.

> **Note**
>
> È possibile salvare i dati dello snapshot del database e del cluster di database in Amazon S3 utilizzando la Console di gestione AWS, AWS CLI o l'API di Amazon RDS. Per ulteriori informazioni, consulta Esportazione dei dati dello snapshot del cluster di database in Amazon S3.

Argomenti

- Installazione dell'estensione aws_s3
- Panoramica dell'esportazione di dati in Amazon S3
- Specifica del percorso del file Amazon S3 in cui eseguire l'esportazione

- Configurazione dell'accesso a un bucket Amazon S3
- Esportazione dei dati della query utilizzando la funzione aws_s3.query_export_to_s3
- Informazioni di riferimento sulle funzioni
- Risoluzione dei problemi di accesso a Amazon S3

Installazione dell'estensione aws_s3

Prima di poter utilizzare Servizio di archiviazione semplice Amazon con il cluster database Aurora PostgreSQL, è necessario installare l'estensione `aws_s3`. Questa estensione fornisce funzioni per l'esportazione di dati dall'istanza di scrittura di un cluster database Aurora PostgreSQL in un bucket Amazon S3. Fornisce inoltre funzioni per l'importazione dei dati da Amazon S3. Per ulteriori informazioni, consulta Importazione di dati da Amazon S3 in un cluster database Aurora PostgreSQL. L'estensione `aws_s3` dipende da alcune delle funzioni helper nell'estensione `aws_commons`, che vengono installate automaticamente quando necessario.

Per installare l'estensione **`aws_s3`**

1. Usa psql (o pgAdmin) per connetterti all'istanza di scrittura del cluster database Aurora PostgreSQL come un utente che dispone di privilegi `rds_superuser`. Se hai mantenuto il nome predefinito durante il processo di configurazione, esegui la connessione come `postgres`.

```
psql --host=111122223333.aws-region.rds.amazonaws.com --port=5432 --
username=postgres --password
```

2. Per installare l'estensione, esegui il comando seguente.

```
postgres=> CREATE EXTENSION aws_s3 CASCADE;
NOTICE: installing required extension "aws_commons"
CREATE EXTENSION
```

3. Per verificare che l'estensione sia installata, puoi usare il metacomando psql `\dx`.

```
postgres=> \dx
       List of installed extensions
    Name     | Version |   Schema   |                Description
-------------+---------+------------+-------------------------------------------
 aws_commons | 1.2     | public     | Common data types across AWS services
 aws_s3      | 1.1     | public     | AWS S3 extension for importing data from S3
 plpgsql     | 1.0     | pg_catalog | PL/pgSQL procedural language
```

```
(3 rows)
```

Le funzioni per importare dati da Amazon S3 ed esportare dati in Amazon S3 sono ora disponibili per l'uso.

Verifica che la versione di Aurora PostgreSQL supporti le esportazioni in Amazon S3

Per verificare che la versione di Aurora PostgreSQL supporti l'esportazione in Amazon S3, puoi utilizzare il comando `describe-db-engine-versions`. L'esempio seguente verifica se la versione 10.14 può eseguire l'esportazione in Amazon S3.

```
aws rds describe-db-engine-versions --region us-east-1 \
--engine aurora-postgresql --engine-version 10.14 | grep s3Export
```

Se l'output include la stringa `"s3Export"`, allora il motore supporta le esportazioni Amazon S3. In caso contrario, il motore non le supporta.

Panoramica dell'esportazione di dati in Amazon S3

Per esportare i dati archiviati in un database Aurora PostgreSQL verso un bucket Amazon S3, attenersi alla procedura descritta di seguito.

Per esportare i dati Aurora PostgreSQL in S3.

1. Identifica un percorso del file Amazon S3 da utilizzare per l'esportazione dei dati. Per informazioni dettagliate su questo processo, consulta Specifica del percorso del file Amazon S3 in cui eseguire l'esportazione.
2. Fornisci l'autorizzazione ad accedere al bucket Amazon S3.

   Per esportare i dati in un file Amazon S3, concedi al cluster di database Aurora PostgreSQL l'autorizzazione per accedere al bucket Amazon S3 che verrà utilizzato dall'esportazione per lo storage. Questa operazione include le seguenti fasi:

   1. Crea una policy IAM che fornisce l'accesso a un bucket Amazon S3 in cui desideri eseguire l'esportazione.
   2. Creare un ruolo IAM.
   3. Collega la policy creata al ruolo creato.
   4. Aggiungi questo ruolo IAM al cluster di database .

Per informazioni dettagliate su questo processo, consulta Configurazione dell'accesso a un bucket Amazon S3.

3. Identifica una query del database per ottenere i dati. Esporta i dati della query chiamando la funzione `aws_s3.query_export_to_s3`.

   Dopo aver completato le attività di preparazione precedenti, utilizza la funzione aws_s3.query_export_to_s3 per esportare i risultati della query in Amazon S3. Per informazioni dettagliate su questo processo, consulta Esportazione dei dati della query utilizzando la funzione aws_s3.query_export_to_s3.

Specifica del percorso del file Amazon S3 in cui eseguire l'esportazione

Specifica le seguenti informazioni per identificare la posizione in Amazon S3 in cui desideri esportare i dati:

- Nome del bucket – Un bucket è un container di oggetti o file Amazon S3.

  Per ulteriori informazioni sull'archiviazione dei dati con Amazon S3, consulta Creazione di un bucket e Utilizzo degli oggetti nella Guida per l'utente di Amazon Simple Storage Service.

- Percorso del file – Il percorso del file identifica la posizione di archiviazione dell'esportazione nel bucket Amazon S3. Il percorso del file comprende:
  - Un prefisso del percorso facoltativo che identifica un percorso di cartella virtuale.
  - Un prefisso del file che identifica uno o più file da archiviare. Esportazioni di dimensioni maggiori vengono archiviate in più file, ciascuno con una dimensione massima di circa 6 GB. I nomi di file aggiuntivi hanno lo stesso prefisso di file ma con l'aggiunta di `_part`*XX*. *XX* rappresenta 2, poi 3 e così via.

  Ad esempio, un percorso del file con una cartella `exports` e un prefisso del file `query-1-export` è `/exports/query-1-export`.

- Regione AWS (facoltativo): la regione AWS in cui si trova il bucket Amazon S3. Se non si specifica un valore per la regione AWS, allora Aurora salva i file in Amazon S3 nella stessa regione AWS del cluster database di esportazione.

> **Note**
>
> Al momento, la regione AWS deve essere la stessa del cluster DB di esportazione.

Per un elenco di nomi di regione AWS e dei valori associati, consulta Regioni e zone di disponibilità.

Per conservare le informazioni sul file Amazon S3 relative alla posizione di archiviazione dell'esportazione, puoi utilizzare la funzione aws_commons.create_s3_uri per creare una struttura composita `aws_commons._s3_uri_1` come descritto di seguito.

```
psql=> SELECT aws_commons.create_s3_uri(
   'amzn-s3-demo-bucket',
   'sample-filepath',
   'us-west-2'
) AS s3_uri_1 \gset
```

In seguito fornisci questo valore `s3_uri_1` come un parametro nella chiamata alla funzione aws_s3.query_export_to_s3. Per alcuni esempi, consulta Esportazione dei dati della query utilizzando la funzione aws_s3.query_export_to_s3.

Configurazione dell'accesso a un bucket Amazon S3

Per esportare i dati in Amazon S3, concedi all' del cluster di database PostgreSQL l'autorizzazione per accedere al bucket Amazon S3 di destinazione dei file.

A tale scopo, procedi come indicato di seguito.

Per concedere a un cluster di database PostgreSQL l'accesso ad Amazon S3 tramite un ruolo IAM

1. Creare una policy IAM

   Questa policy fornisce le autorizzazioni bucket e di oggetto che consentono all' del cluster di database PostgreSQL di accedere a Amazon S3.

   Come parte della creazione di questa policy, attenersi alla seguente procedura:

a. Includere nella policy le seguenti operazioni obbligatorie per consentire il trasferimento dei file dall' del cluster del database PostgreSQL a un bucket Amazon S3:

   - `s3:PutObject`
   - `s3:AbortMultipartUpload`

b. Includere l'Amazon Resource Name (ARN) che identifica il bucket Amazon S3 e gli oggetti nel bucket. Il formato ARN per l'accesso a Amazon S3 è: `arn:aws:s3:::`*`amzn-s3-demo-bucket`*`/*`

Per ulteriori informazioni sulla creazione di una policy IAM per Aurora PostgreSQL, consulta Creazione e utilizzo di una policy IAM per l'accesso al database IAM. Consulta anche il Tutorial: Creare e collegare la prima policy gestita dal cliente nella Guida per l'utente di IAM.

Il AWS CLI comando seguente crea una policy IAM denominata `rds-s3-export-policy` con queste opzioni. Concede l'accesso a un bucket denominato *`amzn-s3-demo-bucket`*.

> **Warning**
>
> Si consiglia di impostare il database all'interno di un VPC privato con policy di endpoint configurate per accedere a bucket specifici. Per ulteriori informazioni, consulta Utilizzo delle policy dell'endpoint per Amazon S3 nella Guida per l'utente di Amazon VPC.
> Si consiglia di non creare una policy con accesso a tutte le risorse. Questo accesso può rappresentare una minaccia per la sicurezza dei dati. Se si crea una policy che consente a `S3:PutObject` di accedere a tutte le risorse utilizzando `"Resource":"*"`, un utente con privilegi di esportazione può esportare i dati in tutti i bucket dell'account. Inoltre, l'utente può esportare i dati in qualsiasi bucket pubblicamente scrivibile all'interno della regione AWS .

Dopo aver creato la policy, annotarne l'Amazon Resource Name (ARN). Per la fase successiva, in cui si associa la policy a un ruolo IAM, è necessario l'ARN.

```
aws iam create-policy  --policy-name rds-s3-export-policy  --policy-document '{
     "Version": "2012-10-17",
     "Statement": [
       {
         "Sid": "s3export",
```

```
        "Action": [
          "s3:PutObject*",
          "s3:ListBucket",
          "s3:GetObject*",
          "s3:DeleteObject*",
          "s3:GetBucketLocation",
          "s3:AbortMultipartUpload"
        ],
        "Effect": "Allow",
        "Resource": [
          "arn:aws:s3:::amzn-s3-demo-bucket/*"
        ]
      }
    ]
}'
```

2. Creare un ruolo IAM.

   In questo modo, Aurora PostgreSQL può assumere questo ruolo IAM per tuo conto, per accedere ai bucket Amazon S3. Per ulteriori informazioni, consulta la pagina relativa alla creazione di un ruolo per delegare le autorizzazioni a un utente IAM nella Guida per l'utente IAM.

   Si consiglia di utilizzare le chiavi di contesto delle condizioni globali `aws:SourceArn` e `aws:SourceAccount` nelle policy basate sulle risorse per limitare le autorizzazioni del servizio a una risorsa specifica. Questo è il modo più efficace per proteggersi dal problema di deputy confused.

   Se si utilizzano entrambe le chiavi di contesto delle condizioni globali e il valore `aws:SourceArn` contiene l'ID account, il valore `aws:SourceAccount` e l'account nel valore `aws:SourceArn` devono utilizzare lo stesso ID account quando viene utilizzato nella stessa dichiarazione di policy.

   - Utilizzare `aws:SourceArn` se si desidera un accesso cross-service per una singola risorsa.
   - Utilizzare `aws:SourceAccount` se si desidera consentire l'associazione di qualsiasi risorsa in tale account all'uso cross-service.

   Nella policy, assicurarsi di utilizzare la chiave di contesto della condizione globale `aws:SourceArn` con l'ARN completo della risorsa. L'esempio seguente mostra come eseguire questa operazione utilizzando il AWS CLI comando per creare un ruolo denominato`rds-s3-export-role`.

Example

Per Linux, macOS o Unix:

```
aws iam create-role  \
    --role-name rds-s3-export-role  \
    --assume-role-policy-document '{
     "Version": "2012-10-17",
     "Statement": [
       {
         "Effect": "Allow",
         "Principal": {
            "Service": "rds.amazonaws.com"
          },
         "Action": "sts:AssumeRole",
         "Condition": {
             "StringEquals": {
                "aws:SourceAccount": "111122223333",
                "aws:SourceArn": "arn:aws:rds:us-
east-1:111122223333:cluster:dbname"
                }
             }
       }
     ]
   }'
```

Per Windows:

```
aws iam create-role  ^
    --role-name rds-s3-export-role  ^
    --assume-role-policy-document '{
     "Version": "2012-10-17",
     "Statement": [
       {
         "Effect": "Allow",
         "Principal": {
            "Service": "rds.amazonaws.com"
          },
         "Action": "sts:AssumeRole",
         "Condition": {
             "StringEquals": {
                "aws:SourceAccount": "111122223333",
```

```
                "aws:SourceArn": "arn:aws:rds:us-
east-1:111122223333:cluster:dbname"
                }
            }
        }
      ]
   }'
```

3. Collegare la policy IAM al ruolo IAM creato.

   Il AWS CLI comando seguente associa la policy creata in precedenza al ruolo denominato `rds-s3-export-role.` Replace *`your-policy-arn`* con l'ARN della policy annotato in un passaggio precedente.

```
aws iam attach-role-policy  --policy-arn your-policy-arn  --role-name rds-s3-
export-role
```

4. Aggiungere il ruolo IAM al cluster. A tale scopo, utilizzare Console di gestione AWS o AWS CLI, come descritto di seguito.

Console

Per aggiungere un ruolo IAM al cluster di database PostgreSQL tramite la console

1. Accedi a Console di gestione AWS e apri la console Amazon RDS all'indirizzo https://console.aws.amazon.com/rds/.
2. Scegliere il nome del cluster di database PostgreSQL per visualizzarne i dettagli.
3. Nella scheda Connectivity & security (Connettività e sicurezza), nella sezione Manage IAM roles (Gestisci ruoli IAM), selezionare il ruolo da aggiungere sotto Add IAM roles to this instance (Aggiungi ruoli IAM a questa istanza).
4. In Feature (Caratteristica), scegliere s3Export.
5. Scegliere Add role (Aggiungi ruolo).

AWS CLI

Per aggiungere un ruolo IAM a un cluster di database PostgreSQL tramite CLI

- Utilizzare il seguente comando per aggiungere il ruolo al cluster di database PostgreSQL denominato `my-db-cluster`. Sostituire *`your-role-arn`* con l'ARN del ruolo annotato in precedenza. Utilizzare `s3Export` come valore dell'opzione `--feature-name`.

  Example

  Per Linux, macOS o Unix:

  ```
  aws rds add-role-to-db-cluster \
     --db-cluster-identifier my-db-cluster \
     --feature-name s3Export \
     --role-arn your-role-arn   \
     --region your-region
  ```

  Per Windows:

  ```
  aws rds add-role-to-db-cluster ^
     --db-cluster-identifier my-db-cluster ^
     --feature-name s3Export ^
     --role-arn your-role-arn ^
     --region your-region
  ```

Esportazione dei dati della query utilizzando la funzione aws_s3.query_export_to_s3

Esporta i dati PostgreSQL in Amazon S3 chiamando la funzione aws_s3.query_export_to_s3.

Argomenti

- Prerequisiti
- Chiamare aws_s3.query_export_to_s3
- Esportazione in un file CSV che utilizza un delimitatore personalizzato
- Esportazione in un file binario con codifica

Prerequisiti

Prima di utilizzare la funzione `aws_s3.query_export_to_s3`, assicurati di completare i seguenti prerequisiti:

- Installa le estensioni PostgreSQL richieste come descritto in Panoramica dell'esportazione di dati in Amazon S3.
- Determina dove esportare i dati in Amazon S3 come descritto in Specifica del percorso del file Amazon S3 in cui eseguire l'esportazione.
- Assicurati che il cluster di database abbia accesso di esportazione a Amazon S3 come descritto in Configurazione dell'accesso a un bucket Amazon S3.

Gli esempi seguenti utilizzano una tabella del database denominata `sample_table`. Questi esempi esportano i dati in un bucket denominato *`amzn-s3-demo-bucket`*. La tabella e i dati di esempio vengono creati con le seguenti istruzioni SQL in psql.

```
psql=> CREATE TABLE sample_table (bid bigint PRIMARY KEY, name varchar(80));
psql=> INSERT INTO sample_table (bid,name) VALUES (1, 'Monday'), (2,'Tuesday'), (3,
 'Wednesday');
```

Chiamare aws_s3.query_export_to_s3

Di seguito vengono illustrati le modalità di base per chiamare la funzione aws_s3.query_export_to_s3.

In questi esempi viene utilizzata la variabile `s3_uri_1` per identificare una struttura contenente le informazioni che identificano il file Amazon S3. Utilizzare la funzione aws_commons.create_s3_uri per creare la struttura.

```
psql=> SELECT aws_commons.create_s3_uri(
   'amzn-s3-demo-bucket',
   'sample-filepath',
   'us-west-2'
) AS s3_uri_1 \gset
```

Anche se i parametri variano per le due chiamate di funzione `aws_s3.query_export_to_s3` seguenti, i risultati sono gli stessi per questi esempi. Tutte le righe della tabella `sample_table` vengono esportate in un bucket denominato *`amzn-s3-demo-bucket`*.

```
psql=> SELECT * FROM aws_s3.query_export_to_s3('SELECT * FROM
 sample_table', :'s3_uri_1');

psql=> SELECT * FROM aws_s3.query_export_to_s3('SELECT * FROM
 sample_table', :'s3_uri_1', options :='format text');
```

I parametri sono descritti come segue:

- `'SELECT * FROM sample_table'` – Il primo parametro è una stringa di testo obbligatoria contenente una query SQL. Il motore PostgreSQL esegue questa query. I risultati della query vengono copiati nel bucket S3 identificato in altri parametri.
- `:'s3_uri_1'` – Questo parametro è una struttura che identifica il file Amazon S3. In questo esempio viene utilizzata una variabile per identificare la struttura creata in precedenza. È invece possibile creare la struttura includendo la chiamata di funzione `aws_commons.create_s3_uri` in linea all'interno della chiamata di funzione `aws_s3.query_export_to_s3` come segue.

  ```
  SELECT * from aws_s3.query_export_to_s3('select * from sample_table',
     aws_commons.create_s3_uri('amzn-s3-demo-bucket', 'sample-filepath', 'us-west-2')
  );
  ```
- `options :='format text'` – Il parametro `options` è una stringa di testo opzionale contenente argomenti `COPY` PostgreSQL. La procedura di copia utilizza gli argomenti e il formato del comando COPY di PostgreSQL.

Se il file specificato non esiste nel bucket Amazon S3, viene creato. Se il file esiste già, viene sovrascritto. La sintassi per accedere ai dati esportati Amazon S3 è la seguente.

```
s3-region://bucket-name[/path-prefix]/file-prefix
```

Esportazioni di dimensioni maggiori vengono archiviate in più file, ciascuno con una dimensione massima di circa 6 GB. I nomi di file aggiuntivi hanno lo stesso prefisso di file ma con l'aggiunta di `_part`*XX*. *XX* rappresenta 2, poi 3 e così via. Ad esempio, supponi di specificare il percorso in cui archivi i file di dati come segue.

```
s3-us-west-2://amzn-s3-demo-bucket/my-prefix
```

Se l'esportazione deve creare tre file di dati, il bucket Amazon S3 contiene i seguenti file di dati.

```
s3-us-west-2://amzn-s3-demo-bucket/my-prefix
s3-us-west-2://amzn-s3-demo-bucket/my-prefix_part2
s3-us-west-2://amzn-s3-demo-bucket/my-prefix_part3
```

Per il riferimento completo per questa funzione e altri modi per chiamarla, consulta aws_s3.query_export_to_s3. Per ulteriori informazioni sull'accesso ai file in Amazon S3, consulta Visualizzazione di un oggetto nella Guida per l'utente di Amazon Simple Storage Service.

Esportazione in un file CSV che utilizza un delimitatore personalizzato

Nell'esempio seguente viene illustrato come chiamare la funzione aws_s3.query_export_to_s3 per esportare i dati in un file che utilizza un delimitatore personalizzato. Nell'esempio vengono utilizzati gli argomenti del comando PostgreSQL COPY per specificare il formato CSV (valori delimitati da virgole) e un delimitatore : (due punti).

```
SELECT * from aws_s3.query_export_to_s3('select * from basic_test', :'s3_uri_1',
 options :='format csv, delimiter $$:$$');
```

Esportazione in un file binario con codifica

Nell'esempio seguente viene illustrato come chiamare la funzione aws_s3.query_export_to_s3 per esportare i dati in un file binario con codifica Windows-1253.

```
SELECT * from aws_s3.query_export_to_s3('select * from basic_test', :'s3_uri_1',
 options :='format binary, encoding WIN1253');
```

Informazioni di riferimento sulle funzioni

Funzioni

- aws_s3.query_export_to_s3
- aws_commons.create_s3_uri

aws_s3.query_export_to_s3

Esporta un risultato della query PostgreSQL in un bucket Amazon S3. L'estensione `aws_s3` fornisce la funzione `aws_s3.query_export_to_s3`.

I due parametri richiesti sono `query` e `s3_info`. Questi definiscono la query da esportare e identificano il bucket Amazon S3 in cui eseguire l'esportazione. Un parametro opzionale chiamato

`options` fornisce la definizione di vari parametri di esportazione. Per esempi di utilizzo della funzione `aws_s3.query_export_to_s3`, consulta Esportazione dei dati della query utilizzando la funzione aws_s3.query_export_to_s3.

Sintassi

```
aws_s3.query_export_to_s3(
    query text,
    s3_info aws_commons._s3_uri_1,
    options text,
    kms_key text
)
```

Parametri di input

query

Una stringa di testo obbligatoria contenente una query SQL eseguita dal motore PostgreSQL. I risultati di questa query vengono copiati in un bucket S3 identificato nel parametro `s3_info`.

s3_info

Un tipo composito `aws_commons._s3_uri_1` contenente le seguenti informazioni sull'oggetto S3:

- `bucket` – Il nome del bucket Amazon S3 per contenere il file.
- `file_path` – Il nome e il percorso del file Amazon S3.
- `region`: la regione AWS in cui si trova il bucket. Per un elenco di nomi di regione AWS e dei valori associati, consulta Regioni e zone di disponibilità.

  Al momento, questo valore deve essere la stessa regione AWS di quella del cluster DB di esportazione. Il valore predefinito è la regione AWS del cluster DB di esportazione.

Per creare una struttura composita `aws_commons._s3_uri_1`, consulta la funzione aws_commons.create_s3_uri.

options

Una stringa di testo opzionale contenente gli argomenti del comando `COPY` di PostgreSQL. Questi argomenti specificano come i dati devono essere copiati quando vengono esportati. Per maggiori dettagli, consulta la documentazione di COPY PostgreSQL.

testo kms_key

Una stringa di testo opzionale contenente la chiave KMS gestita dal cliente del bucket S3 in cui esportare i dati.

Parametri di input alternativi

Per facilitare il testing, puoi utilizzare un set esteso di parametri al posto del parametro `s3_info`. Di seguito vengono riportate le variazioni di sintassi aggiuntive per la funzione `aws_s3.query_export_to_s3`.

Invece di utilizzare il parametro `s3_info` per identificare un file Amazon S3, utilizzare la combinazione dei parametri `bucket`, `file_path` e `region`.

```
aws_s3.query_export_to_s3(
    query text,
    bucket text,
    file_path text,
    region text,
    options text,
    kms_key text
)
```

query

Una stringa di testo obbligatoria contenente una query SQL eseguita dal motore PostgreSQL. I risultati di questa query vengono copiati in un bucket S3 identificato nel parametro `s3_info`.

bucket

Una stringa di testo obbligatoria contenente il nome del bucket Amazon S3 che contiene il file

file_path

Una stringa di testo obbligatoria contenente il nome file di Amazon S3, incluso il percorso.

Regione

Una stringa di testo opzionale contenente la regione AWS in cui si trova il bucket. Per un elenco di nomi di regione AWS e dei valori associati, consulta Regioni e zone di disponibilità.

Al momento, questo valore deve essere la stessa regione AWS di quella del cluster DB di esportazione. Il valore predefinito è la regione AWS del cluster DB di esportazione.

options

Una stringa di testo opzionale contenente gli argomenti del comando COPY di PostgreSQL. Questi argomenti specificano come i dati devono essere copiati quando vengono esportati. Per maggiori dettagli, consulta la documentazione di COPY PostgreSQL.

testo kms_key

Una stringa di testo opzionale contenente la chiave KMS gestita dal cliente del bucket S3 in cui esportare i dati.

Parametri di output

```
aws_s3.query_export_to_s3(
    OUT rows_uploaded bigint,
    OUT files_uploaded bigint,
    OUT bytes_uploaded bigint
)
```

rows_uploaded

Il numero di righe della tabella che sono state caricate correttamente in Amazon S3 per la query specificata.

files_uploaded

Il numero di file caricati in Amazon S3. I file vengono creati in dimensioni di circa 6 GB. A ogni file aggiuntivo creato è stato aggiunto `_part`*XX* al nome. *XX* rappresenta 2, poi 3 e così via, se necessario.

bytes_uploaded

Il numero totale di byte caricati in Amazon S3.

Esempi

```
psql=> SELECT * from aws_s3.query_export_to_s3('select * from sample_table', 'amzn-s3-demo-bucket', 'sample-filepath');
psql=> SELECT * from aws_s3.query_export_to_s3('select * from sample_table', 'amzn-s3-demo-bucket', 'sample-filepath','us-west-2');
psql=> SELECT * from aws_s3.query_export_to_s3('select * from sample_table', 'amzn-s3-demo-bucket', 'sample-filepath','us-west-2','format text');
```

aws_commons.create_s3_uri

Crea una struttura `aws_commons._s3_uri_1` per conservare le informazioni relative al file Amazon S3. I risultati della funzione `aws_commons.create_s3_uri` vengono utilizzati nel parametro `s3_info` della funzione aws_s3.query_export_to_s3. Per un esempio di utilizzo della funzione `aws_commons.create_s3_uri`, consulta Specifica del percorso del file Amazon S3 in cui eseguire l'esportazione.

Sintassi

```
aws_commons.create_s3_uri(
   bucket text,
   file_path text,
   region text
)
```

Parametri di input

bucket

Una stringa di testo obbligatoria contenente il nome del bucket Amazon S3 del file.

file_path

Una stringa di testo obbligatoria contenente il nome file di Amazon S3, incluso il percorso.

Regione

Una stringa di testo obbligatoria contenente la regione AWS in cui si trova il file. Per un elenco di nomi di regione AWS e dei valori associati, consulta Regioni e zone di disponibilità.

Risoluzione dei problemi di accesso a Amazon S3

Se si verificano problemi di connessione durante il tentativo di esportare i dati in Amazon S3, conferma innanzi tutto che le regole di accesso in uscita per il gruppo di sicurezza VPC associato all'istanza DB consentano la connettività di rete. In particolare, il gruppo di sicurezza deve disporre di una regola che consenta all'istanza database di inviare traffico TCP alla porta 443 e a qualsiasi indirizzo IPv4 (0.0.0.0/0). Per ulteriori informazioni, consulta Fornitura dell'accesso al cluster di database nel VPC creando un gruppo di sicurezza.

Consulta anche quanto segue per i suggerimenti:

- Risoluzione dei problemi di identità e accesso in Amazon Aurora
- Risoluzione dei problemi di Amazon S3 nella Guida per l'utente di Amazon Simple Storage Service
- Risoluzione dei problemi di Amazon S3 e IAM nella Guida per l'utente di IAM.

AWS Lambda è un servizio di elaborazione basato sugli eventi che consente di eseguire codice senza fornire o gestire server. Ad esempio, è possibile utilizzare le funzioni Lambda per elaborare le notifiche di eventi da un database o per caricare dati da file ogni volta che un nuovo file viene caricato su Amazon S3. Per ulteriori informazioni su Lambda, vedi Cos'è? AWS Lambda nella Guida per gli AWS Lambda sviluppatori.

> **Note**
>
> Le AWS Lambda funzioni di richiamo sono supportate in Aurora PostgreSQL 11.9 e versioni successive (incluso). Aurora Serverless v2

Di seguito sono riportati i riepiloghi dei passaggi necessari.

Per ulteriori informazioni sulle funzioni Lambda, consulta Nozioni di base su Lambda e Fondamenti di AWS Lambda nella Guida per gli sviluppatori di AWS Lambda .

Argomenti

- Fase 1: configurare l'istanza DB del cluster Aurora PostgreSQL DB per PostgreSQL per le connessioni in uscita a AWS Lambda
- AWS Lambda
- Fase 3: installazione dell'estensione aws_lambda per un cluster database Aurora PostgreSQL
- Fase 4: utilizzo delle funzioni di supporto Lambda con il cluster database Aurora PostgreSQL (facoltativo)
- Fase 5: richiamo di una funzione Lambda dal cluster database Aurora PostgreSQL
- Fase 6: concessione delle autorizzazioni ad altri utenti per richiamare le funzioni Lambda
- Esempi: Richiamo delle funzioni Lambda dal cluster di database Aurora PostgreSQL
- Messaggi di errore della funzione Lambda
- Riferimento alla funzione AWS Lambda e ai parametri

Fase 1: configurare l'istanza DB del cluster Aurora PostgreSQL DB per PostgreSQL per le connessioni in uscita a AWS Lambda

Le funzioni Lambda vengono sempre eseguite all'interno di un Amazon VPC di proprietà del servizio. AWS Lambda Lambda applica le regole di accesso alla rete e di sicurezza a questo VPC e mantiene e monitora il VPC automaticamente. Il cluster database Aurora PostgreSQL deve inviare traffico di rete al VPC del servizio Lambda. La modalità di configurazione dipende dal fatto che l'istanza database principale del cluster database Aurora sia pubblica o privata.

- Cluster database pubblico Aurora PostgreSQL RDS per PostgreSQL è pubblica se si trova in una sottorete pubblica sul VPC e se la proprietà "" dell' è. PubliclyAccessible `true` Per trovare il valore di questa proprietà, puoi usare il comando. describe-db-instances AWS CLI In alternativa, puoi utilizzare la Console di gestione AWS per aprire la scheda Connectivity & security (Connettività e sicurezza) e controllare che l'opzione Publicly accessible (Accessibile pubblicamente) sia impostata su Yes (Sì). Per verificare se l'istanza si trova nella sottorete pubblica del VPC, puoi utilizzare la Console di gestione AWS o la AWS CLI.

  Per configurare l'accesso a Lambda, usi Console di gestione AWS o AWS CLI per creare una regola in uscita sul gruppo di sicurezza del tuo VPC. La regola in uscita specifica che TCP può utilizzare la porta 443 per inviare pacchetti a qualsiasi indirizzo (0.0.0.0/0). IPv4

- Istanza privata del cluster Aurora PostgreSQL DB RDS per PostgreSQL si trova in una sottorete privata. PubliclyAccessible `false` Per consentire all'istanza di funzionare con Lambda, è possibile utilizzare un gateway Network Address Translation (NAT). Per ulteriori informazioni, consulta Gateway NAT. In alternativa, puoi configurare il VPC con un endpoint VPC per Lamda. Per ulteriori informazioni, consultare Endpoint VPC nella Guida per l'utente di Amazon VPC. L'endpoint risponde alle chiamate effettuate dal cluster database Aurora PostgreSQL alle funzioni Lambda.

Il tuo VPC può ora interagire con il AWS Lambda VPC a livello di rete. Configura quindi le autorizzazioni utilizzando IAM.

AWS Lambda

Il richiamo di funzioni Lambda dal cluster database Aurora PostgreSQL richiede determinati privilegi. Per configurare i privilegi necessari, si consiglia di creare una policy IAM che consenta di richiamare le funzioni Lambda, assegnare tale policy a un ruolo e quindi applicare il ruolo al cluster database. Questo approccio fornisce al cluster database privilegi per richiamare la funzione Lambda specificata per tuo conto. La procedura seguente mostra come eseguire questa operazione in AWS CLI.

Configurare le autorizzazioni IAM per l'utilizzo del cluster con Lambda

1. Usa il AWS CLI comando create-policy per creare una policy IAM che consenta all'istanza database Aurora PostgreSQL DB del cluster RDS per di richiamare la funzione Lambda specificata. (L'ID dichiarazione (Sid) è una descrizione facoltativa per la dichiarazione di policy e non ha alcun effetto sull'utilizzo). Questa policy fornisce al cluster database Aurora le autorizzazioni minime necessarie per richiamare la funzione Lambda specificata.

```
aws iam create-policy  --policy-name rds-lambda-policy --policy-document '{
    "Version": "2012-10-17",
    "Statement": [
        {
        "Sid": "AllowAccessToExampleFunction",
        "Effect": "Allow",
        "Action": "lambda:InvokeFunction",
        "Resource": "arn:aws:lambda:aws-region:444455556666:function:my-function"
        }
    ]
}'
```

   In alternativa, puoi utilizzare la policy `AWSLambdaRole` predefinita che ti consente di richiamare una qualsiasi delle tue funzioni Lambda. Per ulteriori informazioni, consulta Policy IAM basate sull'identità per Lambda

2. Utilizza il comando create-role per creare un ruolo IAM che la policy può assumere in fase di esecuzione AWS CLI .

```
aws iam create-role  --role-name rds-lambda-role --assume-role-policy-document '{
    "Version": "2012-10-17",
    "Statement": [
        {
        "Effect": "Allow",
        "Principal": {
            "Service": "rds.amazonaws.com"
        },
        "Action": "sts:AssumeRole"
        }
    ]
}'
```

3. Applica la policy al ruolo utilizzando il attach-role-policy AWS CLI comando.

```
aws iam attach-role-policy \
    --policy-arn arn:aws:iam::444455556666:policy/rds-lambda-policy \
    --role-name rds-lambda-role --region aws-region
```

4. https://awscli.amazonaws.com/v2/documentation/api/latest/reference/rds/add-role-to-db-cluster.html AWS CLI Quest'ultimo passaggio consente agli utenti del cluster database di richiamare le funzioni Lambda.

```
aws rds add-role-to-db-cluster \
       --db-cluster-identifier my-cluster-name \
       --feature-name Lambda \
       --role-arn  arn:aws:iam::444455556666:role/rds-lambda-role   \
       --region aws-region
```

Con le configurazioni di VPC e IAM completate, puoi ora installare l'estensione `aws_lambda`. (Puoi installare l'estensione in qualsiasi momento, ma fino a quando non configuri il supporto VPC e i privilegi IAM corretti, l'estensione `aws_lambda`non aggiunge nulla alle funzionalità del cluster database Aurora PostgreSQL).

Fase 3: installazione dell'estensione **`aws_lambda`** per un cluster database Aurora PostgreSQL

Questa estensione fornisce al cluster database Aurora PostgreSQL la possibilità di chiamare le funzioni Lambda da PostgreSQL.

Installare l'estensione **`aws_lambda`**nel cluster database Aurora PostgreSQL

Utilizza la riga di comando PostgreSQL `psql` o lo strumento pgAdmin per connetterti al cluster database Aurora PostgreSQL.

1. Connettiti al cluster database Aurora PostgreSQL come utente con privilegi `rds_superuser`. L'utente `postgres` predefinito viene visualizzato nell'esempio.

```
psql -h cluster-instance.444455556666.aws-region.rds.amazonaws.com -U postgres -p
 5432
```

2. Installa l'estensione `aws_lambda`. Anche l'estensione `aws_commons` è obbligatoria. Fornisce funzioni di supporto ad `aws_lambda` e molte altre estensioni Aurora per PostgreSQL. Se non si

trova già nel tuo cluster database Aurora PostgreSQL , viene installata con `aws_lambda` come mostrato di seguito.

```
CREATE EXTENSION IF NOT EXISTS aws_lambda CASCADE;
NOTICE:  installing required extension "aws_commons"
CREATE EXTENSION
```

L'estensione `aws_lambda` è installata nell'istanza database principale del cluster database Aurora PostgreSQL. Ora puoi creare strutture utili per richiamare le tue funzioni Lambda.

Fase 4: utilizzo delle funzioni di supporto Lambda con il cluster database Aurora PostgreSQL (facoltativo)

Puoi utilizzare le funzioni di supporto nell'estensione `aws_commons` per preparare entità che è possibile richiamare più facilmente da PostgreSQL. Per farlo, devi disporre delle seguenti informazioni sulle funzioni Lambda:

- Nome funzione: il nome, l'Amazon Resource Name (ARN), la versione o l'alias della funzione Lambda. La policy IAM creata in Fase 2: configurazione di IAM per il cluster e Lambda richiede l'ARN, quindi ti consigliamo di utilizzare l'ARN della tua funzione.
- AWS Regione

Per mantenere le informazioni sul nome della funzione Lambda, puoi utilizzare la funzione aws_commons.create_lambda_function_arn. Questa funzione di supporto crea una struttura composita `aws_commons._lambda_function_arn_1` con i dettagli necessari alla funzione di richiamo. Di seguito, puoi trovare tre approcci alternativi per l'impostazione di questa struttura composita.

```
SELECT aws_commons.create_lambda_function_arn(
   'my-function',
   'aws-region'
) AS aws_lambda_arn_1 \gset
```

```
SELECT aws_commons.create_lambda_function_arn(
   '111122223333:function:my-function',
   'aws-region'
) AS lambda_partial_arn_1 \gset
```

```
SELECT aws_commons.create_lambda_function_arn(
   'arn:aws:lambda:aws-region:111122223333:function:my-function'
) AS lambda_arn_1 \gset
```

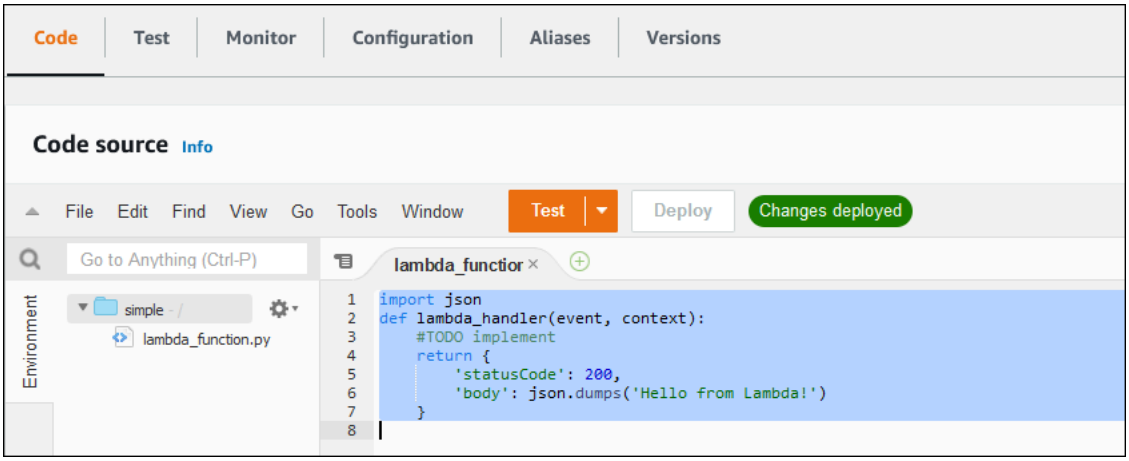
Ognuno di questi valori può essere utilizzato nelle chiamate alla funzione aws_lambda.invoke. Per alcuni esempi, consulta Fase 5: richiamo di una funzione Lambda dal cluster database Aurora PostgreSQL.

Fase 5: richiamo di una funzione Lambda dal cluster database Aurora PostgreSQL

La funzione `aws_lambda.invoke` si comporta in modo sincrono o asincrono, a seconda del `invocation_type`. Le due alternative per questo parametro sono `RequestResponse` (il valore predefinito) e `Event`, come di seguito riportato.

- **`RequestResponse`**: questo tipo di richiamo è sincrono. Questo è il comportamento predefinito quando viene effettuata la chiamata senza specificare un tipo di chiamata. Il payload di risposta include i risultati della funzione `aws_lambda.invoke`. Utilizza questo tipo di chiamata quando il flusso di lavoro richiede la ricezione dei risultati della funzione Lambda prima di procedere.
- **`Event`**: questo tipo di richiamo è asincrono. La risposta non include un payload contenente i risultati. Utilizza questo tipo di richiamo quando il flusso di lavoro non ha bisogno di un risultato della funzione Lambda per continuare l'elaborazione.

Come semplice test della configurazione, puoi connetterti all'istanza database utilizzando `psql` e richiamare una funzione di esempio dalla riga di comando. Supponiamo di avere una delle funzioni di base impostate sul tuo servizio Lambda, come la semplice funzione Python mostrata nello screenshot di seguito.



Per richiamare una funzione di esempio

1. Connettiti all'istanza database principale utilizzando `psql` o pgAdmin.

```
psql -h cluster.444455556666.aws-region.rds.amazonaws.com -U postgres -p 5432
```

2. Richiama la funzione utilizzando il relativo ARN.

```
SELECT * from
 aws_lambda.invoke(aws_commons.create_lambda_function_arn('arn:aws:lambda:aws-
region:444455556666:function:simple', 'us-west-1'), '{"body": "Hello from
 Postgres!"}'::json );
```

   La risposta è la seguente.

```
status_code |                         payload                         |
 executed_version | log_result
-------------+---------------------------------------------------------
+------------------+------------
         200 | {"statusCode": 200, "body": "\"Hello from Lambda!\""} | $LATEST
     |
(1 row)
```

Se il tuo tentativo di richiamo non ha esito positivo, vedi Messaggi di errore della funzione Lambda .

Fase 6: concessione delle autorizzazioni ad altri utenti per richiamare le funzioni Lambda

A questo punto della procedura, solo tu in qualità di `rds_superuser` puoi richiamare le funzioni Lambda. Per consentire ad altri utenti di richiamare le funzioni che crei, è necessario concedere loro l'autorizzazione.

Per concedere ad altri utenti l'autorizzazione per richiamare le funzioni Lambda

1. Connettiti all'istanza database principale utilizzando `psql` o pgAdmin.

```
psql -h cluster.444455556666.aws-region.rds.amazonaws.com -U postgres -p 5432
```

2. Esegui i seguenti comandi SQL:

```
postgres=>  GRANT USAGE ON SCHEMA aws_lambda TO db_username;
GRANT EXECUTE ON ALL FUNCTIONS IN SCHEMA aws_lambda TO db_username;
```

Esempi: Richiamo delle funzioni Lambda dal cluster di database Aurora PostgreSQL

Di seguito, è possibile trovare diversi esempi di chiamate alla funzione aws_lambda.invoke. Nella maggior parte degli esempi viene utilizzata la struttura composita `aws_lambda_arn_1` creata in Fase 4: utilizzo delle funzioni di supporto Lambda con il cluster database Aurora PostgreSQL (facoltativo) per semplificare il passaggio dei dettagli della funzione. Per un esempio di chiamata asincrona, consulta Esempio: richiamo (di eventi) asincroni di funzioni Lambda. Tutti gli altri esempi elencati utilizzano il richiamo sincrono.

Per ulteriori informazioni sui tipi di chiamata Lambda, consulta Richiamo di funzioni Lambda nella Guida per gli sviluppatori di AWS Lambda. Per ulteriori informazioni su `aws_lambda_arn_1`, consulta aws_commons.create_lambda_function_arn.

Elenco di esempi

- Esempio: richiamo sincrono (RequestResponse) di funzioni Lambda
- Esempio: richiamo (di eventi) asincroni di funzioni Lambda
- Esempio: acquisizione del registro di esecuzione Lambda in una risposta di funzione
- Esempio: inclusione del contesto client in una funzione Lambda
- Esempio: richiamo di una versione specifica di una funzione Lambda

Esempio: richiamo sincrono (RequestResponse) di funzioni Lambda

Di seguito sono riportati due esempi di una chiamata di funzione Lambda sincrona. I risultati di queste chiamate di funzione `aws_lambda.invoke` sono gli stessi.

```
SELECT * FROM aws_lambda.invoke('aws_lambda_arn_1', '{"body": "Hello from
 Postgres!"}'::json);
```

```
SELECT * FROM aws_lambda.invoke('aws_lambda_arn_1', '{"body": "Hello from
 Postgres!"}'::json, 'RequestResponse');
```

I parametri sono descritti come segue:

- `:'aws_lambda_arn_1'`: questo parametro identifica la struttura composita creata in Fase 4: utilizzo delle funzioni di supporto Lambda con il cluster database Aurora PostgreSQL (facoltativo) con la funzione di supporto di `aws_commons.create_lambda_function_arn`. Puoi anche creare questa struttura in linea all'interno della chiamata `aws_lambda.invoke` come segue:

```
SELECT * FROM aws_lambda.invoke(aws_commons.create_lambda_function_arn('my-function',
 'aws-region'),
'{"body": "Hello from Postgres!"}'::json
);
```

- `'{"body": "Hello from PostgreSQL!"}'::json`: il payload JSON da passare alla funzione Lambda.
- `'RequestResponse'`: il tipo di richiamo Lambda.

Esempio: richiamo (di eventi) asincroni di funzioni Lambda

Di seguito è riportato un esempio di una chiamata di funzione Lambda asincrona. Il tipo di richiamo `Event` pianifica il richiamo della funzione Lambda con il payload di input specificato e restituisce immediatamente un risultato. Utilizza il tipo di chiamata `Event` in determinati flussi di lavoro che non dipendono dai risultati della funzione Lambda.

```
SELECT * FROM aws_lambda.invoke('aws_lambda_arn_1', '{"body": "Hello from
 Postgres!"}'::json, 'Event');
```

Esempio: acquisizione del registro di esecuzione Lambda in una risposta di funzione

È possibile includere gli ultimi 4 KB del registro di esecuzione nella risposta della funzione utilizzando il parametro `log_type` nella chiamata di funzione `aws_lambda.invoke`. Per impostazione predefinita, questo parametro è impostato su `None`, ma puoi specificare `Tail` per acquisire i risultati del registro di esecuzione Lambda nella risposta, come illustrato di seguito.

```
SELECT *, select convert_from(decode(log_result, 'base64'), 'utf-8') as log FROM
 aws_lambda.invoke(:'aws_lambda_arn_1', '{"body": "Hello from Postgres!"}'::json,
 'RequestResponse', 'Tail');
```

Impostare il parametro aws_lambda.invoke della funzione `log_type` su `Tail` per includere il log di esecuzione nella risposta. Il valore predefinito per il parametro `log_type` è `None`.

Il `log_result` che viene restituito è una stringa `base64` codificata. È possibile decodificare i contenuti utilizzando una combinazione delle funzioni PostgreSQL `decode` e `convert_from`.

Per ulteriori informazioni su `log_type`, consulta aws_lambda.invoke.

Esempio: inclusione del contesto client in una funzione Lambda

La funzione `aws_lambda.invoke` ha un parametro `context` che puoi utilizzare per passare le informazioni separate dal payload, come illustrato di seguito.

```
SELECT *, convert_from(decode(log_result, 'base64'), 'utf-8') as log FROM
 aws_lambda.invoke(:'aws_lambda_arn_1', '{"body": "Hello from Postgres!"}'::json,
 'RequestResponse', 'Tail');
```

Per includere il contesto client, utilizzare un oggetto JSON per il parametro aws_lambda.invoke della funzione `context`.

Per ulteriori informazioni sul parametro `context`, consulta la documentazione di riferimento di aws_lambda.invoke.

Esempio: richiamo di una versione specifica di una funzione Lambda

Puoi specificare una determinata versione di una funzione Lambda includendo il parametro `qualifier` con la chiamata `aws_lambda.invoke`. Di seguito puoi trovare un esempio in cui viene utilizzato '*`custom_version`*' come alias per la versione.

```
SELECT * FROM aws_lambda.invoke('aws_lambda_arn_1', '{"body": "Hello from
 Postgres!"}'::json, 'RequestResponse', 'None', NULL, 'custom_version');
```

Puoi inoltre fornire un qualificatore di funzione Lambda con i dettagli relativi al nome della funzione, come mostrato di seguito.

```
SELECT * FROM aws_lambda.invoke(aws_commons.create_lambda_function_arn('my-
function:custom_version', 'us-west-2'),
'{"body": "Hello from Postgres!"}'::json);
```

Per ulteriori informazioni su `qualifier` e altri parametri, consulta documentazione di riferimento di aws_lambda.invoke.

Messaggi di errore della funzione Lambda

Nell'elenco seguente sono disponibili informazioni sui messaggi di errore, con le possibili cause e soluzioni.

- Problemi di configurazione del VPC

  I problemi di configurazione del VPC possono generare i seguenti messaggi di errore al momento della connessione:

  ```
  ERROR:  invoke API failed
  DETAIL: AWS Lambda client returned 'Unable to connect to endpoint'.
  CONTEXT:  SQL function "invoke" statement 1
  ```

  Una causa comune di questo errore è il gruppo di sicurezza VPC configurato in modo errato. Assicurati di avere una regola in uscita per TCP aperta sulla porta 443 del gruppo di sicurezza VPC in modo che il VPC possa connettersi al VPC Lambda.

- Mancanza delle autorizzazioni necessarie per richiamare le funzioni Lambda

  Se viene visualizzato uno dei seguenti messaggi di errore, l'utente (ruolo) che richiama la funzione non dispone delle autorizzazioni appropriate.

  ```
  ERROR:  permission denied for schema aws_lambda
  ```

  ```
  ERROR:  permission denied for function invoke
  ```

  A un utente (ruolo) devono essere concesse autorizzazioni specifiche per richiamare le funzioni Lambda. Per ulteriori informazioni, consulta Fase 6: concessione delle autorizzazioni ad altri utenti per richiamare le funzioni Lambda.

- Gestione impropria degli errori nelle funzioni Lambda

  Se una funzione Lambda genera un'eccezione durante l'elaborazione della richiesta, `aws_lambda.invoke` non riesce e indica un errore PostgreSQL come quello seguente.

  ```
  SELECT * FROM aws_lambda.invoke('aws_lambda_arn_1', '{"body": "Hello from
   Postgres!"}'::json);
  ERROR:  lambda invocation failed
  ```

```
DETAIL:  "arn:aws:lambda:us-west-2:555555555555:function:my-function" returned error
 "Unhandled", details: "<Error details string>".
```

Assicurati di gestire gli errori nelle funzioni Lambda o nell'applicazione PostgreSQL.

Riferimento alla funzione AWS Lambda e ai parametri

Di seguito è riportato il riferimento per le funzioni e i parametri da utilizzare per invocare Lambda con Aurora PostgreSQL .

Funzioni e parametri

- aws_lambda.invoke
- aws_commons.create_lambda_function_arn
- parametri aws_lambda

aws_lambda.invoke

Esegue una funzione Lambda per un cluster DB Aurora PostgreSQL .

Per ulteriori dettagli sul richiamo delle funzioni Lambda, consulta anche Invoke nella Guida per gli sviluppatori di AWS Lambda.

Sintassi

JSON

```
aws_lambda.invoke(
IN function_name TEXT,
IN payload JSON,
IN region TEXT DEFAULT NULL,
IN invocation_type TEXT DEFAULT 'RequestResponse',
IN log_type TEXT DEFAULT 'None',
IN context JSON DEFAULT NULL,
IN qualifier VARCHAR(128) DEFAULT NULL,
OUT status_code INT,
OUT payload JSON,
OUT executed_version TEXT,
OUT log_result TEXT)
```

```
aws_lambda.invoke(
```

```
IN function_name aws_commons._lambda_function_arn_1,
IN payload JSON,
IN invocation_type TEXT DEFAULT 'RequestResponse',
IN log_type TEXT DEFAULT 'None',
IN context JSON DEFAULT NULL,
IN qualifier VARCHAR(128) DEFAULT NULL,
OUT status_code INT,
OUT payload JSON,
OUT executed_version TEXT,
OUT log_result TEXT)
```

JSONB

```
aws_lambda.invoke(
IN function_name TEXT,
IN payload JSONB,
IN region TEXT DEFAULT NULL,
IN invocation_type TEXT DEFAULT 'RequestResponse',
IN log_type TEXT DEFAULT 'None',
IN context JSONB DEFAULT NULL,
IN qualifier VARCHAR(128) DEFAULT NULL,
OUT status_code INT,
OUT payload JSONB,
OUT executed_version TEXT,
OUT log_result TEXT)
```

```
aws_lambda.invoke(
IN function_name aws_commons._lambda_function_arn_1,
IN payload JSONB,
IN invocation_type TEXT DEFAULT 'RequestResponse',
IN log_type TEXT DEFAULT 'None',
IN context JSONB DEFAULT NULL,
IN qualifier VARCHAR(128) DEFAULT NULL,
OUT status_code INT,
OUT payload JSONB,
OUT executed_version TEXT,
OUT log_result TEXT
)
```

Parametri di input

function_name

Nome identificativo della funzione Lambda. Il valore può essere il nome della funzione, un ARN o un ARN parziale. Per un elenco dei formati possibili, consulta Formati dei nomi delle funzioni Lambda nella Guida per gli sviluppatori di AWS Lambda.

payload

L'input per la funzione Lambda. Il formato può essere JSON o JSONB. Per ulteriori informazioni, consulta Tipi di JSON nella documentazione di PostgreSQL.

region

(Facoltativo) La regione Lambda per la funzione. Per impostazione predefinita, Aurora risolve la regione AWS dall'ARN completo nella `function_name` oppure utilizza la regione dell'istanza database Aurora PostgreSQL . Se il valore di questa Regione è in conflitto con quello fornito nell'ARN `function_name`, viene generato un errore.

invocation_type

Il tipo di chiamata della funzione Lambda. Il valore prevede la distinzione tra lettere maiuscole e minuscole. I valori possibili sono:

- `RequestResponse` – Il valore predefinito. Questo tipo di chiamata per una funzione Lambda è sincrono e restituisce un payload di risposta nel risultato. Utilizzare il tipo di chiamata `RequestResponse` quando il flusso di lavoro dipende dalla ricezione immediata del risultato della funzione Lambda.
- `Event` – Questo tipo di chiamata per una funzione Lambda è asincrono e restituisce immediatamente una risposta senza un payload restituito. Utilizzare il tipo di chiamata `Event` quando non sono necessari i risultati della funzione Lambda prima che il flusso di lavoro proceda.
- `DryRun` – Questo tipo di chiamata verifica l'accesso senza eseguire la funzione Lambda.

log_type

Il tipo di log Lambda da restituire nel parametro di output `log_result`. Il valore prevede la distinzione tra lettere maiuscole e minuscole. I valori possibili sono:

- Tail – Il parametro di output `log_result` restituito includerà gli ultimi 4 KB del registro di esecuzione.
- None – Non viene restituita nessuna informazione di log Lambda.

context

Contesto client in formato JSON o JSONB. I campi da utilizzare includono `custom` e `env`.

qualifier

Un qualificatore che identifica la versione di una funzione Lambda da richiamare. Se questo valore è in conflitto con quello fornito nell' `function_name` ARN, viene generato un errore.

Parametri di output

status_code

Un codice di risposta allo stato HTTP. Per ulteriori informazioni, consulta Elementi di risposta del richiamo di Lambda nella Guida per gli sviluppatori di AWS Lambda.

payload

Le informazioni restituite dalla funzione Lambda eseguita. Il formato è in JSON o JSONB.

executed_version

La versione della funzione Lambda eseguita.

log_result

Le informazioni del registro di esecuzione restituite se il valore `log_type` è `Tail` quando è stata richiamata la funzione Lambda. Il risultato contiene gli ultimi 4 KB del registro di esecuzione codificato in Base64.

aws_commons.create_lambda_function_arn

Crea una struttura `aws_commons._lambda_function_arn_1` per contenere le informazioni sul nome della funzione Lambda. È possibile utilizzare i risultati della `aws_commons.create_lambda_function_arn` funzione nel parametro `function_name` della funzione aws_lambda.invoke aws_lambda.invoke.

Sintassi

```
aws_commons.create_lambda_function_arn(
    function_name TEXT,
    region TEXT DEFAULT NULL
    )
```

```
    RETURNS aws_commons._lambda_function_arn_1
```

Parametri di input

function_name

Stringa di testo obbligatoria contenente il nome della funzione Lambda. Il valore può essere un nome di funzione, un ARN parziale o un ARN completo.

region

Una stringa di testo facoltativa contenente la regione AWS in cui si trova la funzione Lambda. Per un elenco di nomi di regione e dei valori associati, consulta Regioni e zone di disponibilità.

parametri aws_lambda

In questa tabella sono illustrati i parametri associati alla funzione `aws_lambda`.

Parametro	Descrizione
`aws_lambda.connect_timeout_ms`	Si tratta di un parametro dinamico che imposta il tempo di attesa massimo durante la connessione a AWS Lambda. Il valore predefinito è `1000`. I valori consentiti per questo parametro sono compresi tra 1 e 900000.
`aws_lambda.request_timeout_ms`	Si tratta di un parametro dinamico che imposta il tempo di attesa massimo durante l'attesa della risposta da AWS Lambda. Il valore predefinito è `3000`. I valori consentiti per questo parametro sono compresi tra 1 e 900000.
`aws_lambda.endpoint_override`	Specifica l'endpoint che può essere utilizzato per connettersi a AWS Lambda. Una stringa vuota seleziona l'endpoint AWS Lambda predefinito per la Regione. Per rendere effettiva la modifica del parametro statico, è necessario riavviare il database.

Pubblicazione di log Aurora PostgreSQL in Amazon CloudWatch Logs

È possibile configurare il cluster database Aurora PostgreSQL per l'esportazione dei dati di registro in File di log Amazon CloudWatch in modo regolare. In questo caso, gli eventi del registro PostgreSQL del cluster database Aurora PostgreSQL vengono pubblicati automaticamente in Amazon CloudWatch, come File di log Amazon CloudWatch. In CloudWatch, sono disponibili i dati di registro esportati in un Gruppo di log per il cluster database Aurora PostgreSQL. Il gruppo di log contiene uno o più flussi di log che contengono gli eventi del registro PostgreSQL di ogni istanza nel cluster.

La pubblicazione dei registri in File di log CloudWatch consente di mantenere i record di registro PostgreSQL del cluster in uno storage altamente durevole. Con i dati di registro disponibili in File di log CloudWatch, puoi valutare e migliorare le operazioni del cluster. Puoi anche usare CloudWatch per creare allarmi e visualizzare metriche. Per ulteriori informazioni, consulta Monitoraggio degli eventi di log in Amazon CloudWatch.

> **Note**
>
> La pubblicazione dei registri PostgreSQL in File di log CloudWatch consuma spazio di archiviazione e comporta costi di archiviazione. Assicurati di eliminare gli eventuali File di log CloudWatch non più necessari.

La disattivazione dell'opzione Export log (Esporta log) per un cluster database Aurora PostgreSQL esistente non influenza gli eventuali dati già contenuti in File di log CloudWatch. I registri esistenti rimangono disponibili in File di log CloudWatch in base alle impostazioni di conservazione dei registri. Per ulteriori informazioni su File di log CloudWatch, consulta Che cos'è File di log Amazon CloudWatch?

Aurora PostgreSQL supporta la pubblicazione di registri in File di log CloudWatch per le seguenti versioni.

- 16.1 e tutte le versioni successive
- 15.2 o versioni successive alla 15
- 14.3 o versioni successive alla 14
- 13.3 o versioni successive alla 13
- 12.8 e versioni successive alla 12

- 11.9 o versioni successive alla 11

Per informazioni sull'attivazione dell'opzione di pubblicazione dei log su CloudWatch Logs, sul monitoraggio degli eventi di log in CloudWatch Logs e sull'analisi dei log utilizzando CloudWatch Logs Insights, consulta i seguenti argomenti.

Argomenti

- Attivazione dell'opzione per pubblicare i registri in Amazon CloudWatch
- Monitoraggio degli eventi di log in Amazon CloudWatch
- Analisi dei registri Aurora PostgreSQL utilizzando approfondimenti sui file di log Amazon CloudWatch

Attivazione dell'opzione per pubblicare i registri in Amazon CloudWatch

Per pubblicare il registro PostgreSQL del cluster database Aurora PostgreSQL in File di log CloudWatch, seleziona l'opzione Log Export (Esportazione log) per il cluster. Puoi scegliere l'impostazione Log export (Esportazione log) durante la creazione del cluster database Aurora PostgreSQL. In alternativa, puoi modificare il cluster in un secondo momento. Quando si modifica un cluster esistente, i relativi registri PostgreSQL di ciascuna istanza vengono pubblicati nel cluster CloudWatch da quel momento in poi. Per Aurora PostgreSQL, il registro PostgreSQL (`postgresql.log`) è l'unico registro pubblicato in Amazon CloudWatch.

Puoi utilizzare la Console di gestione AWS, la AWS CLI o l'API RDS per attivare la funzionalità Log export (Esportazione log) per il cluster database Aurora PostgreSQL.

Console

Scegliere l'opzione Log exports (Esportazioni log) per avviare la pubblicazione dei registri PostgreSQL dal cluster database Aurora PostgreSQL in File di log Amazon CloudWatch.

Per attivare la funzionalità Log export (Esportazione log) dalla console

1. Apri la console Amazon RDS all'indirizzo https://console.aws.amazon.com/rds/.
2. Nel pannello di navigazione, scegliere Databases (Database).
3. Scegli il cluster database Aurora PostgreSQL di cui desideri pubblicare i dati di registro in File di log CloudWatch.
4. Scegliere Modify (Modifica).

5. Nella sezione Log exports (Esportazioni log) scegli Postgresql log (Log PostgreSQL).
6. Scegliere Continue (Continua) e quindi selezionare Modify cluster (Modifica cluster) nella pagina di riepilogo.

AWS CLI

Puoi attivare l'opzione Log export (Esportazione log) per iniziare a pubblicare i registri di Aurora PostgreSQL in File di log Amazon CloudWatch con la AWS CLI. A questo scopo, esegui il comando AWS CLI modify-db-cluster con le opzioni seguenti:

- `--db-cluster-identifier`: l'identificatore del cluster di database.
- `--cloudwatch-logs-export-configuration`—L'impostazione di configurazione per i tipi di log da abilitare per l'esportazione in CloudWatch Logs per il cluster DB.

Puoi pubblicare i log Aurora PostgreSQL anche emettendo uno dei seguenti comandi della AWS CLI:

- create-db-cluster
- restore-db-cluster-from-s3
- restore-db-cluster-from-snapshot
- restore-db-cluster-to-point-in-time

Esegui uno di questi comandi dell'AWS CLI con le opzioni seguenti:

- `--db-cluster-identifier`: l'identificatore del cluster di database.
- `--engine`— – Il motore di database.
- `--enable-cloudwatch-logs-exports`— – L'impostazione di configurazione per i tipi di log da abilitare per l'esportazione in CloudWatch Logs per il cluster DB.

Altre opzioni potrebbero essere richieste a seconda del comando dell'AWS CLI che esegui.

Example

Il seguente comando crea un cluster DB Aurora PostgreSQL per pubblicare i file di log in CloudWatch Logs.

Per Linux, macOS o Unix:

```
aws rds create-db-cluster \
    --db-cluster-identifier my-db-cluster \
    --engine aurora-postgresql \
    --enable-cloudwatch-logs-exports postgresql
```

Per Windows:

```
aws rds create-db-cluster ^
    --db-cluster-identifier my-db-cluster ^
    --engine aurora-postgresql ^
    --enable-cloudwatch-logs-exports postgresql
```

Example

Il seguente comando modifica un cluster DB Aurora PostgreSQL esistente per pubblicare i file di log in CloudWatch Logs. Il valore `--cloudwatch-logs-export-configuration` è un oggetto JSON. La chiave per questo oggetto è `EnableLogTypes` e il relativo valore è `postgresql` e `instance`.

Per Linux, macOS o Unix:

```
aws rds modify-db-cluster \
    --db-cluster-identifier my-db-cluster \
    --cloudwatch-logs-export-configuration '{"EnableLogTypes":
["postgresql","instance"]}'
```

Per Windows:

```
aws rds modify-db-cluster ^
    --db-cluster-identifier my-db-cluster ^
    --cloudwatch-logs-export-configuration '{\"EnableLogTypes\":[\"postgresql\",
\"instance\"]}'
```

> **Note**
>
> Quando usi il prompt comandi di Windows, non devi inserire le doppie virgolette (") nel codice JSON precedendole con il backslash (\).

Example

Nell'esempio seguente viene modificato un cluster Aurora PostgreSQL DB esistente per disabilitare i file di registro di pubblicazione CloudWatch Logs. Il valore `--cloudwatch-logs-export-configuration` è un oggetto JSON. La chiave per questo oggetto è `DisableLogTypes` e il relativo valore è `postgresql` e `instance`.

Per Linux, macOS o Unix:

```
aws rds modify-db-cluster \
    --db-cluster-identifier mydbinstance \
    --cloudwatch-logs-export-configuration '{"DisableLogTypes":
["postgresql","instance"]}'
```

Per Windows:

```
aws rds modify-db-cluster ^
    --db-cluster-identifier mydbinstance ^
    --cloudwatch-logs-export-configuration "{\"DisableLogTypes\":[\"postgresql\",
\"instance\"]}"
```

> **Note**
>
> Quando usi il prompt comandi di Windows, non devi inserire le doppie virgolette (") nel codice JSON precedendole con il backslash (\).

API RDS

Puoi attivare l'opzione Log export (Esportazione log) per iniziare a pubblicare i registri di Aurora PostgreSQL con l'API RDS. A questo scopo, esegui l'operazione ModifyDBCluster con le seguenti opzioni:

- `DBClusterIdentifier` - L'identificatore del cluster di database.
- `CloudwatchLogsExportConfiguration` – L'impostazione di configurazione per i tipi di log da abilitare per l'esportazione in CloudWatch Logs per il cluster di database.

Puoi anche pubblicare i log Aurora MySQL con l'API RDS eseguendo una delle seguenti operazioni dell'API RDS:

- CreateDBCluster
- RestoreDBClusterFromS3
- RestoreDBClusterFromSnapshot
- RestoreDBClusterToPointInTime

Esegui l'operazione dell'API RDS con i seguenti parametri:

- `DBClusterIdentifier`: l'identificatore del cluster di database.
- `Engine`— – Il motore di database.
- `EnableCloudwatchLogsExports`— – L'impostazione di configurazione per i tipi di log da abilitare per l'esportazione in CloudWatch Logs per il cluster DB.

Altri parametri potrebbero essere richiesti a seconda del comando dell'AWS CLI che esegui.

Monitoraggio degli eventi di log in Amazon CloudWatch

Con il registro eventi di Aurora PostgreSQL pubblicato e disponibile come File di log Amazon CloudWatch, è possibile visualizzare e monitorare gli eventi utilizzando Amazon CloudWatch. Per ulteriori informazioni sul monitoraggio, consulta Visualizzare i dati di log inviati a CloudWatch Logs.

Se si attiva Log exports (Esportazioni log), un nuovo gruppo di log viene creato automaticamente utilizzando il prefisso `/aws/rds/cluster/` con il nome di Aurora PostgreSQL e il tipo di registro, come nel modello seguente.

```
/aws/rds/cluster/your-cluster-name/postgresql
```

Ad esempio, si supponga che un cluster database Aurora PostgreSQL denominato `docs-lab-apg-small` esporti il registro in File di log Amazon CloudWatch. Il relativo nome del relativo gruppo di log in Amazon CloudWatch viene mostrato di seguito.

```
/aws/rds/cluster/docs-lab-apg-small/postgresql
```

Se esiste un gruppo di log con il nome specificato, Aurora utilizza quel gruppo di log per esportare i dati di log per il cluster DB Aurora. Ogni istanza database nel cluster database Aurora PostgreSQL carica il proprio registro PostgreSQL nel gruppo di log come flusso di log distinto. È possibile

esaminare il gruppo di log e i relativi flussi di log utilizzando i vari strumenti grafici e analitici disponibili in Amazon CloudWatch.

Ad esempio, è possibile cercare informazioni negli eventi di registro dal cluster database Aurora PostgreSQL e filtrare gli eventi utilizzando la console di File di log Amazon CloudWatch, la AWS CLI o l'API di File di log Amazon CloudWatch. Per ulteriori informazioni sulla ricerca dei dati dei log, consulta Ricerca e filtro dei dati di registro nella Guida per l'utente di File di log Amazon CloudWatch.

Per impostazione predefinita, i nuovi gruppi di log vengono creati utilizzando Never expire (Non scade mai) per il periodo di conservazione. Puoi usare la console CloudWatch Logs, la AWS CLI oppure l'API di CloudWatch Logs per cambiare il periodo di conservazione dei log. Per ulteriori informazioni, consulta Modifica della conservazione dei dati di registro in File di log Amazon CloudWatch nella Guida per l'utente di File di log Amazon CloudWatch.

> **Tip**
>
> Puoi utilizzare la configurazione automatizzata, ad esempio AWS CloudFormation, per creare i gruppi di log con periodi di conservazione di registro predefiniti, filtri di metriche e autorizzazioni di accesso.

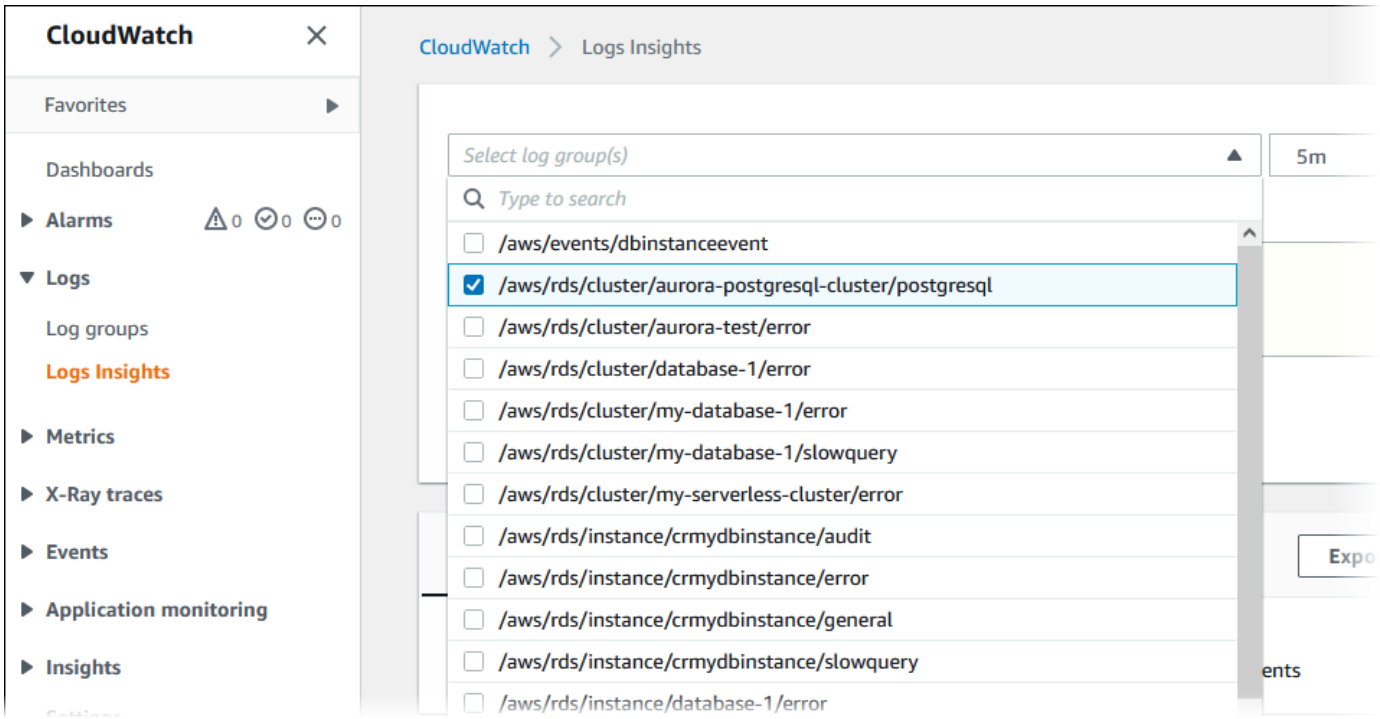
Analisi dei registri Aurora PostgreSQL utilizzando approfondimenti sui file di log Amazon CloudWatch

Con i registri PostgreSQL del cluster database Aurora PostgreSQL pubblicati come File di log Amazon CloudWatch, è possibile usare approfondimenti sui File di log Amazon CloudWatch per cercare e analizzare in modo interattivo i dati di registro in File di log Amazon CloudWatch. Approfondimenti sui file di log Amazon CloudWatch include un linguaggio di query, query di esempio e altri strumenti per l'analisi dei dati di registro in modo da poter identificare potenziali problemi e verificare le correzioni. Per ulteriori informazioni, consulta Analisi dei dati di registro con approfondimenti sui file di log Amazon CloudWatch nella Guida per l'utente di File di log Amazon CloudWatch.

Per analizzare i registri Aurora PostgreSQL con Approfondimenti sui file di log Amazon CloudWatch

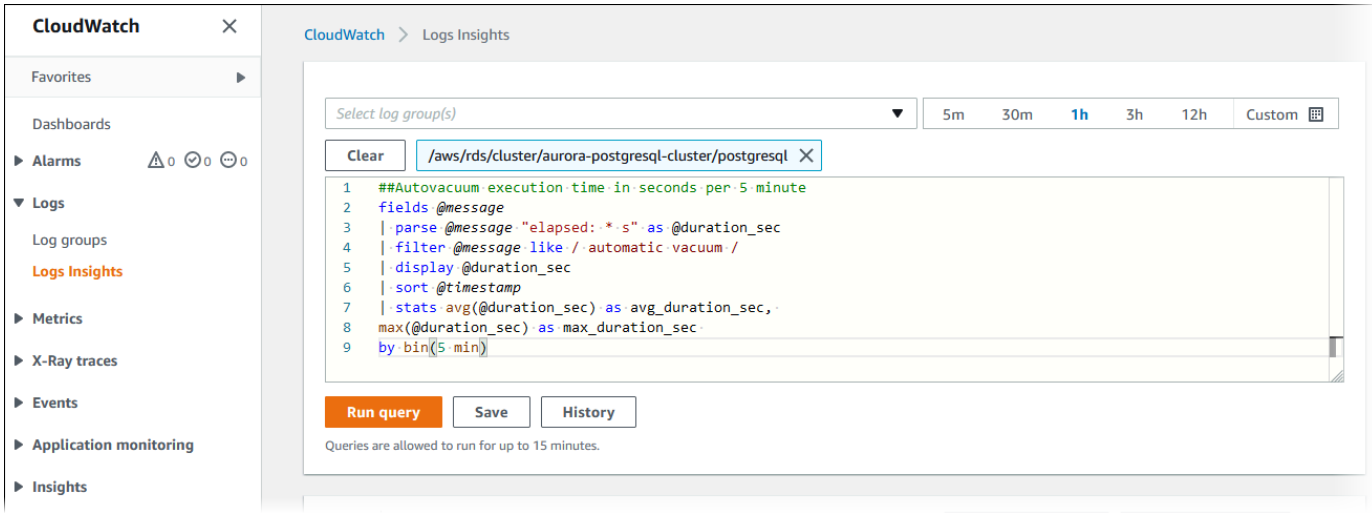
1. Apri la console CloudWatch all'indirizzo https://console.aws.amazon.com/cloudwatch/.
2. Nel pannello di navigazione, aprire Logs (Log), e scegliere Log Insights (Informazioni dettagliate Log.

3. In Select log group(s) (Seleziona uno o più i gruppi di log), seleziona il gruppo di log per il cluster database Aurora PostgreSQL.

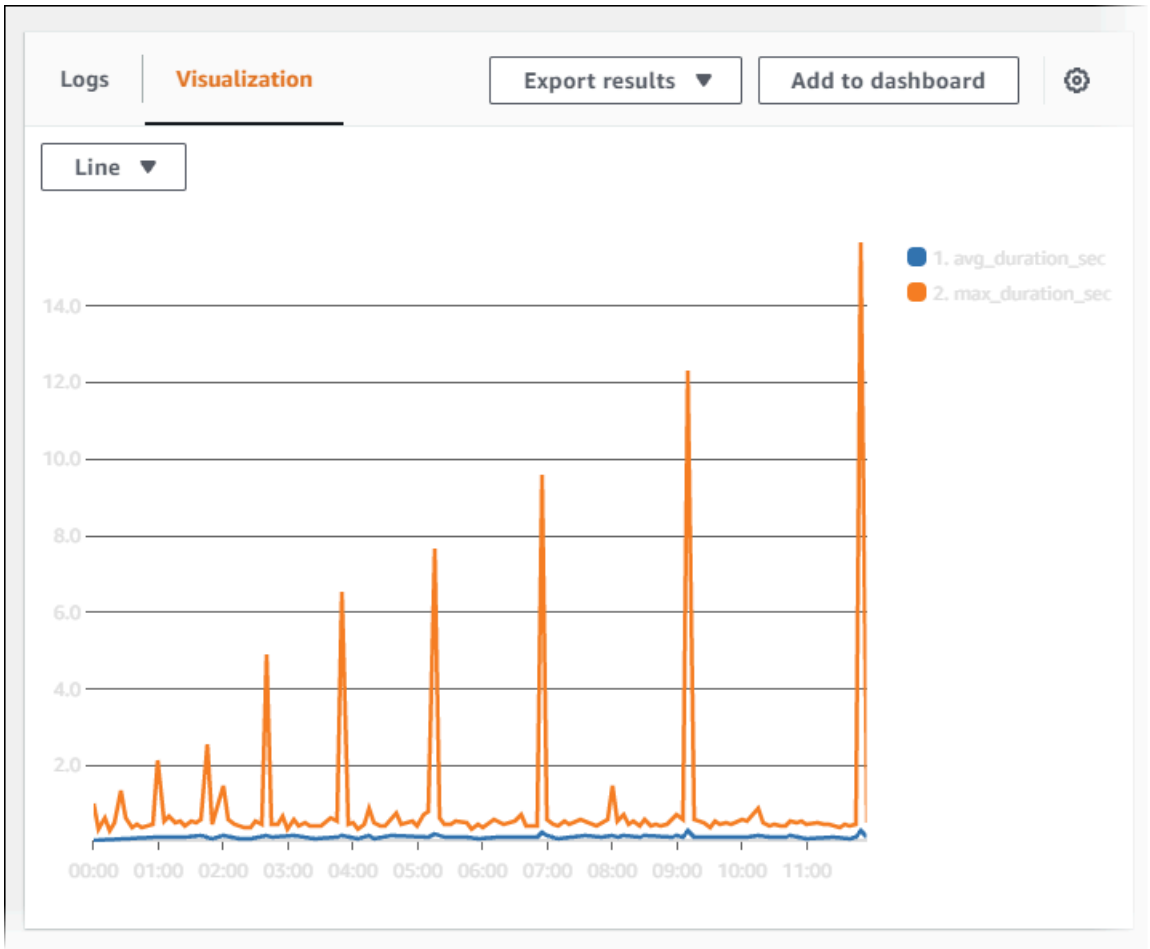


4. Nell'editor di query, elimina la query attualmente visualizzata, quindi immetti quanto segue e scegli Run (Esegui).

```
##Autovacuum execution time in seconds per 5 minute
fields @message
| parse @message "elapsed: * s" as @duration_sec
| filter @message like / automatic vacuum /
| display @duration_sec
| sort @timestamp
| stats avg(@duration_sec) as avg_duration_sec,
max(@duration_sec) as max_duration_sec
by bin(5 min)
```

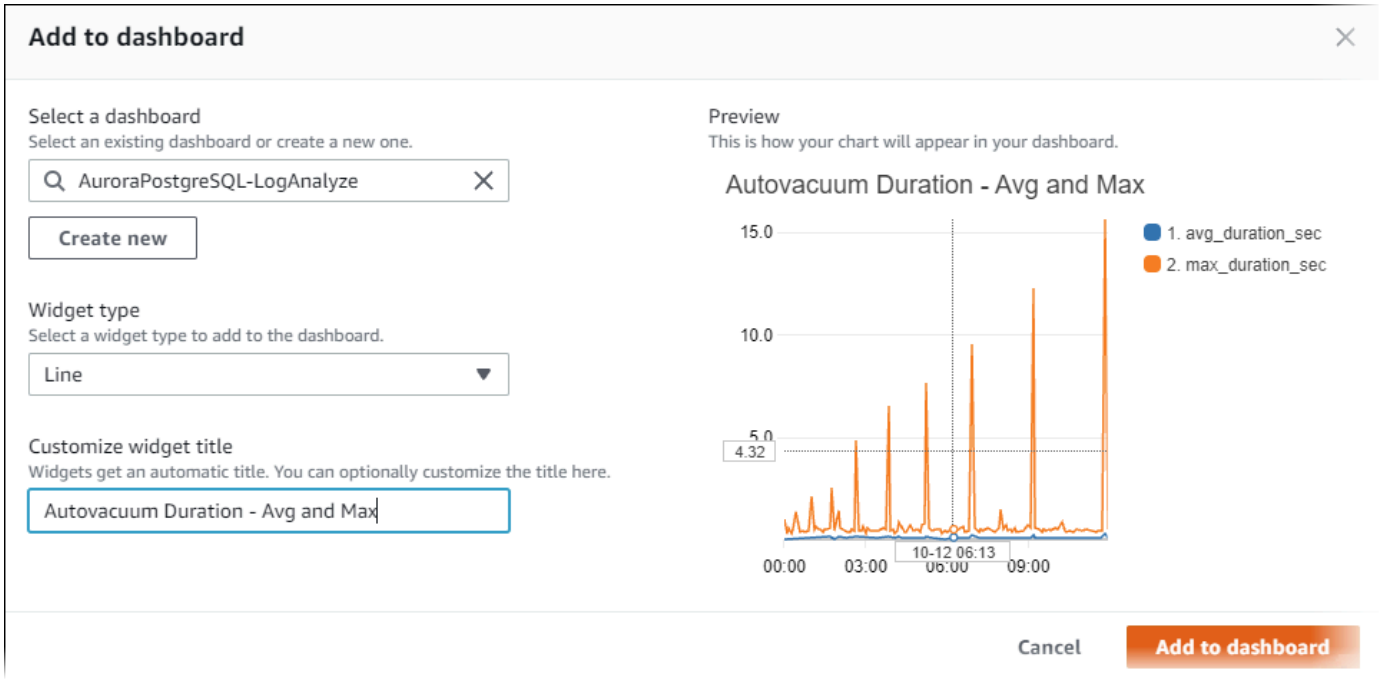


5. Seleziona la scheda Visualization (Visualizzazione).



6. Scegli Add to dashboard (Aggiungi a pannello di controllo).
7. In Seleziona un pannello di controllo, seleziona un pannello di controllo o inserisci un nome per creare un nuovo pannello di controllo.

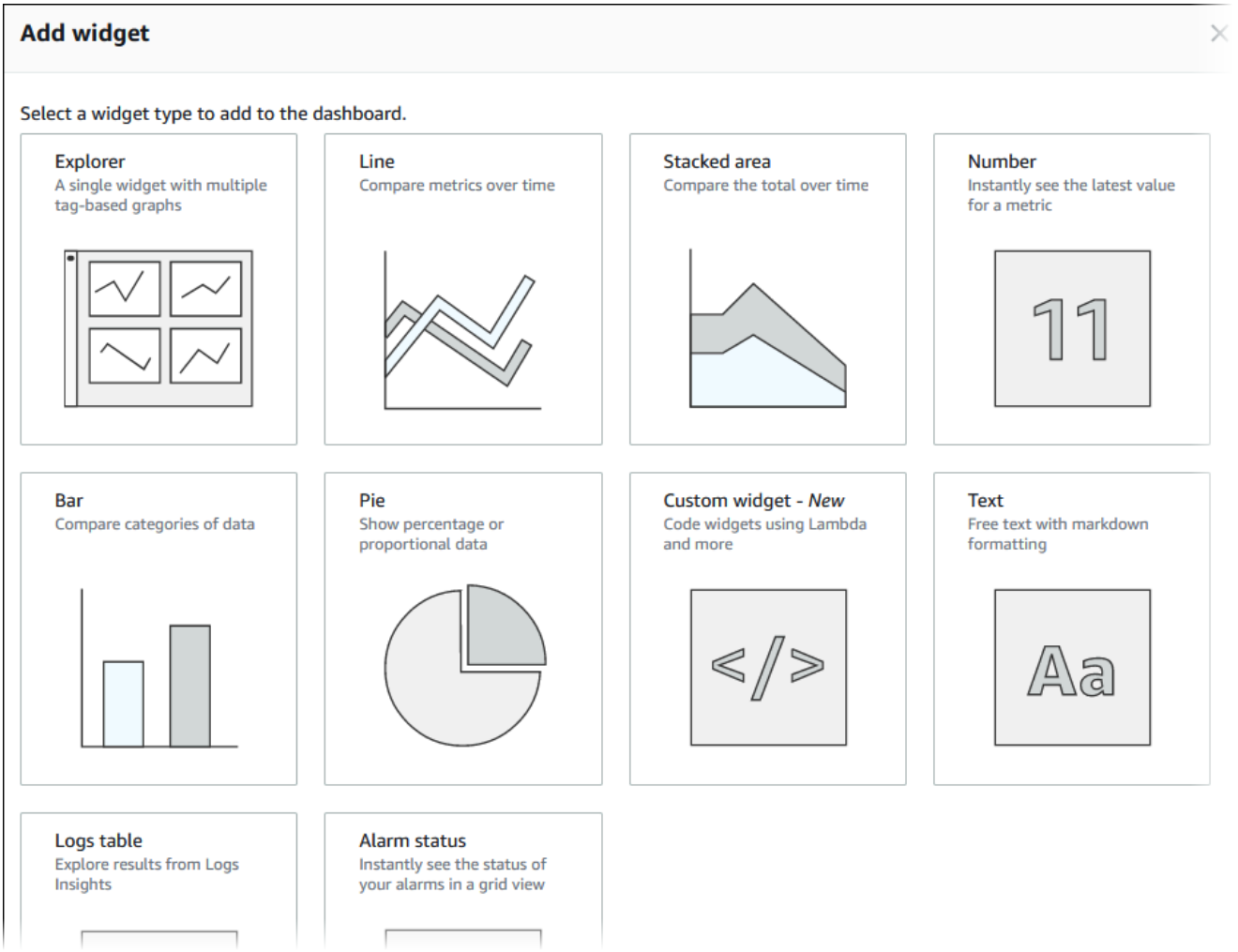
8. In Tipo di widget, scegli un tipo di widget per la tua visualizzazione.



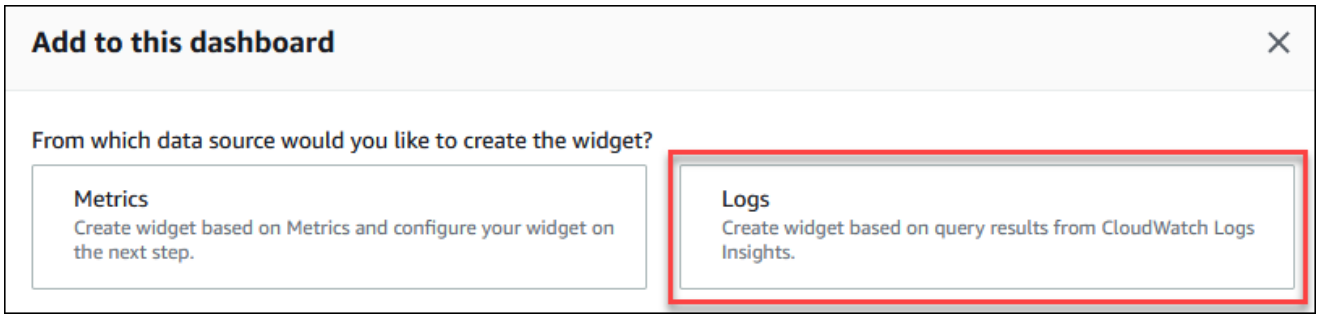
9. (Facoltativo) Aggiungi altri widget in base ai risultati della query di log.

   a. Seleziona Add widget (Aggiungi widget).

   b. Scegli un tipo di widget, ad esempio Line (Linea).



c. Nella finestra Aggiungi a questo pannello di controllo, scegli Log.

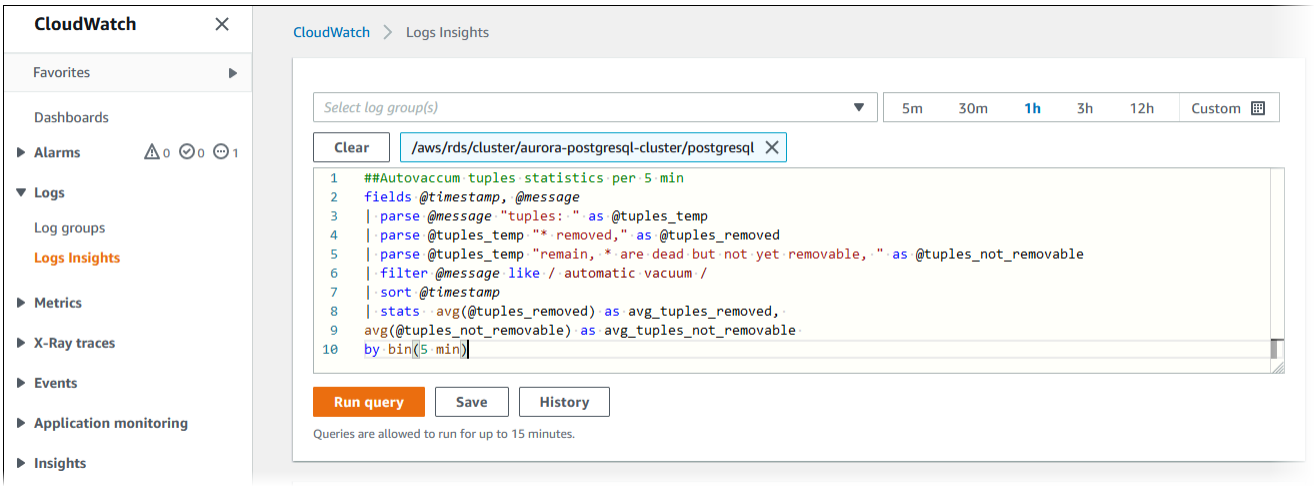


d. Nello stato Seleziona gruppi di log, seleziona il gruppo di log per il cluster di database di.

e. Nell'editor di query, elimina la query attualmente visualizzata, quindi immetti quanto segue e scegli Run (Esegui).

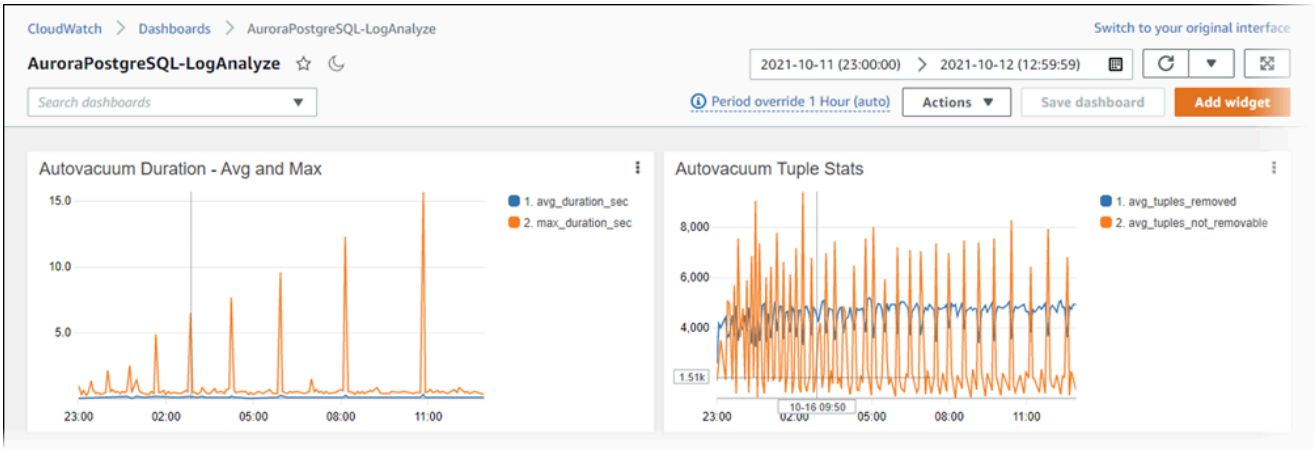
```
##Autovacuum tuples statistics per 5 min
fields @timestamp, @message
| parse @message "tuples: " as @tuples_temp
```

```
| parse @tuples_temp "* removed," as @tuples_removed
| parse @tuples_temp "remain, * are dead but not yet removable, " as
 @tuples_not_removable
| filter @message like / automatic vacuum /
| sort @timestamp
| stats  avg(@tuples_removed) as avg_tuples_removed,
avg(@tuples_not_removable) as avg_tuples_not_removable
by bin(5 min)
```



f. Seleziona Crea widget.

Il pannello di controllo dovrebbe apparire simile alla seguente immagine.



Monitoraggio dei piani di esecuzione delle query e della memoria di picco per Aurora PostgreSQL

È possibile monitorare i piani di esecuzione delle query nell'istanza database Aurora PostgreSQL per rilevare i piani di esecuzione che contribuiscono al carico corrente del database e tenere traccia delle statistiche sulle prestazioni dei piani di esecuzione nel tempo utilizzando il parametro `aurora_compute_plan_id`. Ogni volta che viene eseguita una query, al piano di esecuzione utilizzato dalla query viene assegnato un identificatore e lo stesso identificatore viene utilizzato nelle successive esecuzioni dello stesso piano.

`aurora_compute_plan_id` è impostato su `OFF` per impostazione predefinita nel gruppo di parametri del database delle versioni 14.10, 15.5 e successive di Aurora PostgreSQL. Per assegnare un identificatore del piano, impostare `aurora_compute_plan_id` su `ON` nel gruppo di parametri.

Questo identificatore del piano viene utilizzato in diverse utilità che hanno uno scopo diverso.

È possibile monitorare l'utilizzo di picco della memoria delle query nell'istanza database per rilevare le query che contribuiscono all'elevato utilizzo della memoria del database nelle seguenti versioni:

- 16.3 e tutte le versioni successive
- 15.7 e versioni successive
- 14.12 e versioni successive

Ogni volta che viene eseguita una query, viene tracciata la memoria di picco utilizzata dalla query. Le query vengono in genere eseguite più volte; per ogni query è possibile visualizzare i valori di utilizzo medio, minimo e massimo della memoria in tutte le esecuzioni.

Argomenti

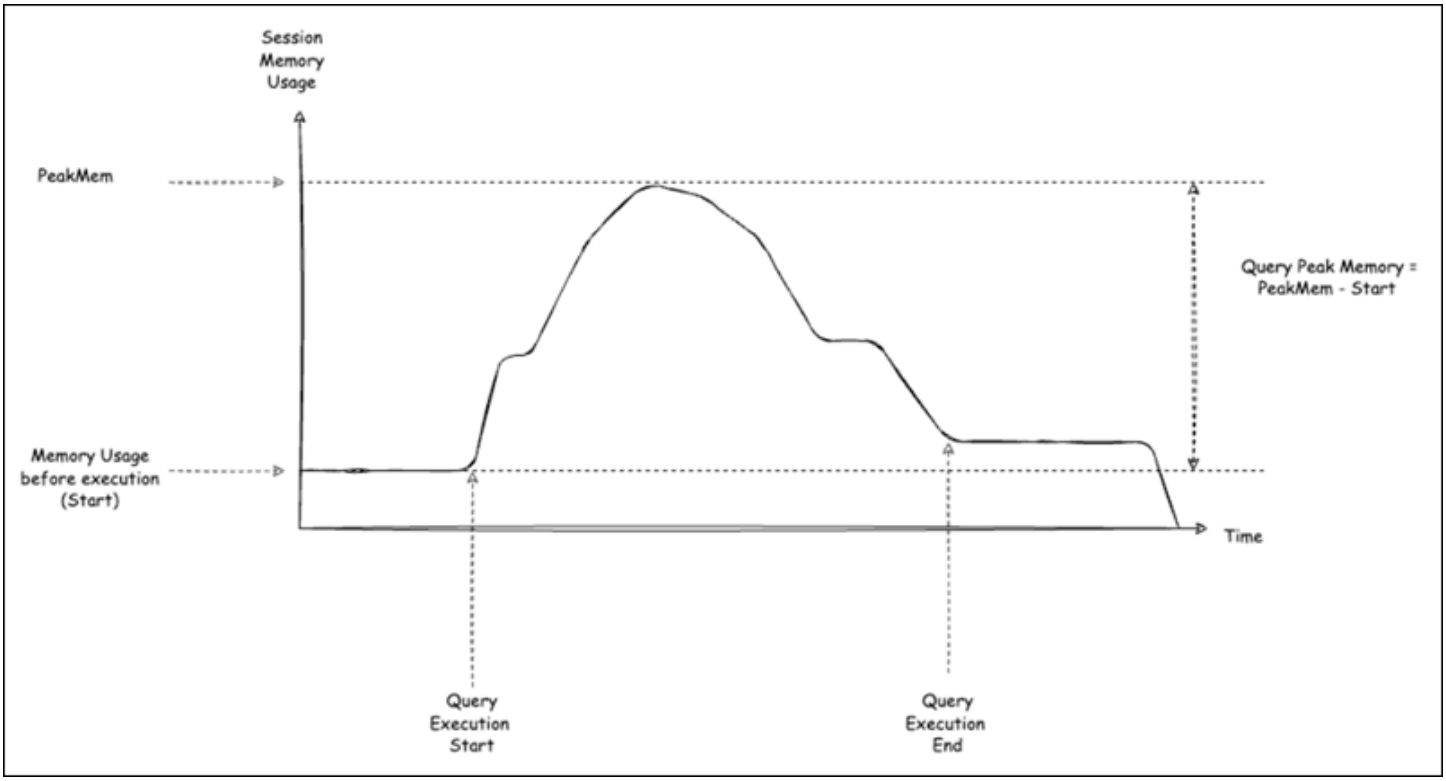
- Accesso ai piani di esecuzione delle query e alla memoria di picco utilizzando le funzioni Aurora
- Documentazione di riferimento dei parametri per i piani di esecuzione delle query Aurora PostgreSQL

Accesso ai piani di esecuzione delle query e alla memoria di picco utilizzando le funzioni Aurora

Con `aurora_compute_plan_id`, è possibile accedere ai piani di esecuzione utilizzando le seguenti funzioni:

- aurora_stat_activity
- aurora_stat_plans

La memoria di picco delle query non include la memoria allocata prima dell'inizio dell'elaborazione delle query. L'utilizzo di picco della memoria viene monitorato e riportato separatamente per le fasi di pianificazione ed esecuzione di ogni query.



È possibile accedere alle statistiche di picco della memoria delle query utilizzando le seguenti funzioni:

- aurora_stat_statements
- aurora_stat_plans

Per ulteriori informazioni su queste funzioni, consulta Riferimenti relativi alle funzioni Aurora PostgreSQL.

Documentazione di riferimento dei parametri per i piani di esecuzione delle query Aurora PostgreSQL

È possibile monitorare i piani di esecuzione delle query utilizzando i parametri seguenti in un gruppo di parametri del database.

Parametri

- aurora_compute_plan_id
- aurora_stat_plans.minutes_until_recapture
- aurora_stat_plans.calls_until_recapture
- aurora_stat_plans.with_costs
- aurora_stat_plans.with_analyze
- aurora_stat_plans.with_timing
- aurora_stat_plans.with_buffers
- aurora_stat_plans.with_wal
- aurora_stat_plans.with_triggers

> **Note**
>
> La configurazione dei parametri `aurora_stat_plans.with_*` ha effetto solo per i piani appena acquisiti.

aurora_compute_plan_id

`aurora_compute_plan_id` è un parametro di configurazione che controlla se viene assegnato un identificatore del piano durante l'esecuzione della query.

Default	Valori consentiti	Descrizione
off	0(off)	Impostare su `off` per impedire l'assegnazione di un identific atore del piano.

Default	Valori consentiti	Descrizione
	1(on)	Impostare su on per assegnare un identificatore del piano.

aurora_stat_plans.minutes_until_recapture

Il numero di minuti che devono trascorrere prima che un piano venga riacquisito. L'impostazione predefinita è 0, che disabiliterà la riacquisizione di un piano. Quando viene superata la soglia `aurora_stat_plans.calls_until_recapture`, il piano verrà riacquisito.

Default	Valori consentiti	Descrizione
0	0-1073741823	Impostare il numero di minuti che devono trascorrere prima che un piano venga riacquisito.

aurora_stat_plans.calls_until_recapture

Il numero di chiamate a un piano prima che venga riacquisito. L'impostazione predefinita è 0, che disabiliterà la riacquisizione di un piano dopo un numero di chiamate. Quando viene superata la soglia `aurora_stat_plans.minutes_until_recapture`, il piano verrà riacquisito.

Default	Valori consentiti	Descrizione
0	0-1073741823	Impostare il numero di chiamate prima che un piano venga riacquisito.

aurora_stat_plans.with_costs

Acquisisce un piano EXPLAIN con costi stimati. I valori consentiti sono on e `off`. Il valore predefinito è on.

Default	Valori consentiti	Descrizione
on	0(off)	Non mostra il costo e le righe stimati per ogni nodo del piano.
	1(on)	Mostra il costo e le righe stimati per ogni nodo del piano.

aurora_stat_plans.with_analyze

Controlla il piano EXPLAIN con ANALYZE. Questa modalità viene utilizzata solo la prima volta che viene acquisito un piano. I valori consentiti sono `on` e `off`. Il valore predefinito è `off`.

Default	Valori consentiti	Descrizione
off	0(off)	Non include le statistiche effettive sulla durata di esecuzione del piano.
	1(on)	Include le statistiche effettive sulla durata di esecuzione del piano.

aurora_stat_plans.with_timing

La tempistica del piano verrà acquisita nell'explain quando viene utilizzato ANALYZE. Il valore predefinito è `on`.

Default	Valori consentiti	Descrizione
on	0(off)	Non include il tempo di avvio effettivo e il tempo impiegato in ciascun nodo del piano.
	1(on)	Include il tempo di avvio effettivo e il tempo impiegato in ciascun nodo del piano.

aurora_stat_plans.with_buffers

Le statistiche sull'utilizzo del buffer del piano verranno acquisite nell'explain quando viene utilizzato ANALYZE. Il valore predefinito è `off`.

Default	Valori consentiti	Descrizione
off	0(off)	Non include informazioni sull'utilizzo del buffer.
	1(on)	Include informazioni sull'utilizzo del buffer.

aurora_stat_plans.with_wal

Le statistiche sull'utilizzo del wal del piano verranno acquisite nell'explain quando viene utilizzato ANALYZE. Il valore predefinito è `off`.

Default	Valori consentiti	Descrizione
off	0(off)	Non include informazioni sulla generazione di record WAL.
	1(on)	Include informazioni sulla generazione di record WAL.

aurora_stat_plans.with_triggers

Le statistiche sull'esecuzione del trigger del piano verranno acquisite nell'explain quando viene utilizzato `ANALYZE`. Il valore predefinito è `off`.

Default	Valori consentiti	Descrizione
off	0(off)	Non include le statistiche di esecuzione dei trigger.
	1(on)	Include le statistiche di esecuzione dei trigger.

Gestione dei piani di esecuzione delle query per Aurora PostgreSQL

La gestione del piano di query per Aurora PostgreSQL è una funzionalità opzionale che è possibile utilizzare con il cluster database Edizione compatibile con PostgreSQL di Amazon Aurora. Questa funzionalità è nel pacchetto come estensione `apg_plan_mgmt` che puoi installare nel cluster database Aurora PostgreSQL. La gestione del piano di query consente di gestire i piani di esecuzione delle query generati dall'ottimizzatore per le applicazioni SQL. L'estensione `apg_plan_mgmt` AWS si basa sulla funzionalità di elaborazione delle query nativa del motore di database PostgreSQL.

Di seguito, sono disponibili informazioni sulle funzionalità di gestione del piano di query per Aurora PostgreSQL, su come configurarlo e su come utilizzarlo con il cluster database Aurora PostgreSQL. Prima di iniziare, ti consigliamo di leggere tutte le note di rilascio per la versione specifica dell'estensione `apg_plan_mgmt` disponibile per la tua versione di Aurora PostgreSQL. Per ulteriori informazioni, consulta Versioni dell'estensione Aurora PostgreSQL apg_plan_mgmt nelle Note di rilascio per Aurora PostgreSQL.

Argomenti

- Panoramica della gestione del piano di query per Aurora PostgreSQL
- Best practice per la gestione del piano di query Aurora PostgreSQL
- Gestione del piano di query di Aurora PostgreSQL
- Acquisizione dei piani di esecuzione Aurora PostgreSQL
- Utilizzo dei piani gestiti per Aurora PostgreSQL
- Esame dei piani di query Aurora PostgreSQL nella vista dba_plans
- Miglioramento dei piani di query di Aurora PostgreSQL
- Eliminazione dei piani di query di Aurora PostgreSQL
- Esportazione e importazione dei piani gestiti per Aurora PostgreSQL
- Documentazione di riferimento dei parametri per la gestione del piano di query Aurora PostgreSQL
- Informazioni di riferimento sulle funzioni per la gestione del piano di query Aurora PostgreSQL
- Riferimento per la visualizzazione apg_plan_mgmt.dba_plans per Aurora edizione compatibile con PostgreSQL
- Funzionalità avanzate della gestione del piano di query

Panoramica della gestione del piano di query per Aurora PostgreSQL

La gestione del piano di query per Aurora PostgreSQL è progettata per garantire la stabilità del piano a prescindere dalle modifiche apportate al database che potrebbero causare la regressione del piano di query. La regressione del piano di query si verifica quando l'ottimizzatore sceglie un piano non ottimale per una determinata istruzione SQL dopo modifiche al sistema o al database. Le modifiche a statistiche, vincoli, impostazioni dell'ambiente, associazioni dei parametri di query e aggiornamenti al motore di database PostgreSQL possono tutte causare la regressione del piano.

La gestione del piano di query per Aurora PostgreSQLconsente di controllare come e quando cambiano i piani di esecuzione delle query. I vantaggi della gestione del piano di query per Aurora PostgreSQL includono i seguenti.

- Migliorare la stabilità del piano forzando l'ottimizzatore a selezionare tra un numero ridotto di piani corretti.
- Ottimizzare i piani centralmente e distribuire i migliori a livello globale.
- Identificare gli indici non utilizzati e valutare l'impatto della creazione o della rimozione di un indice.
- Rilevare automaticamente un nuovo piano a costo minimo individuato dall'ottimizzatore.
- Provare nuove funzionalità dell'ottimizzatore con meno rischi, perché è possibile scegliere di approvare solo le modifiche dei piani che migliorano le performance.

Gli strumenti forniti dalla gestione del piano di query possono essere usati in modo proattivo per specificare il piano migliore per determinate query. In alternativa, è possibile utilizzare la gestione del piano di query per reagire a circostanze mutevoli ed evitare regressioni del piano. Per ulteriori informazioni, consulta Best practice per la gestione del piano di query Aurora PostgreSQL.

Argomenti

- Istruzioni SQL supportate
- Limitazioni della gestione del piano di query
- Terminologia della gestione del piano di query
- Versioni di gestione del piano di query per Aurora PostgreSQL
- Attivazione della gestione del piano di query per Aurora PostgreSQL
- Aggiornamento della gestione del piano di query per Aurora PostgreSQL
- Disattivazione della gestione del piano di query per Aurora PostgreSQL

Istruzioni SQL supportate

La gestione del piano di query supporta i seguenti tipi di istruzioni SQL.

- Qualsiasi istruzione SELECT, INSERT, UPDATE o DELETE, a prescindere dalla complessità.
- Istruzioni preparate. Per ulteriori informazioni, consultare PREPARE nella documentazione di PostgreSQL.
- Istruzioni dinamiche, comprese quelle eseguite in modalità immediata. Per ulteriori informazioni, consultare Dynamic SQL ed EXECUTE IMMEDIATE nella documentazione di PostgreSQL.
- Comandi e istruzioni SQL incorporati. Per ulteriori informazioni, consultare Embedded SQL Commands nella documentazione di PostgreSQL.
- Istruzioni all'interno di funzioni denominate. Per ulteriori informazioni, consultare CREATE FUNCTION nella documentazione di PostgreSQL.
- Istruzioni contenenti tabelle temporanee.
- Istruzioni all'interno di procedure e blocchi DO.

La gestione del piano di query può essere utilizzata con `EXPLAIN` in modalità manuale per acquisire un piano senza eseguirlo effettivamente. Per ulteriori informazioni, consulta Analisi del piano scelto dall'ottimizzatore. Per ulteriori informazioni sulle modalità di gestione del piano di query (manuale, automatica), consultare Acquisizione dei piani di esecuzione Aurora PostgreSQL.

La gestione del piano di query per Aurora PostgreSQL supporta tutte le funzionalità del linguaggio PostgreSQL, incluse le tabelle partizionate, l'ereditarietà, la sicurezza a livello di riga e le espressioni di tabelle comuni ricorsive (CTE). Per ulteriori informazioni su queste caratteristiche del linguaggio PostgreSQL, consultare Table Partitioning, Row Security Policies e WITH Queries (Common Table Expressions) e altri argomenti nella documentazione di PostgreSQL.

Per informazioni sulle diverse versioni della funzionalità di gestione del piano di query di Aurora PostgreSQL, consulta Versioni dell'estensione Aurora PostgreSQL apg_plan_mgmt nelle Note di rilascio per Aurora PostgreSQL.

Limitazioni della gestione del piano di query

La versione corrente della gestione del piano di query di Aurora PostgreSQL include le seguenti limitazioni.

- I piani non vengono acquisiti per le istruzioni che fanno riferimento alle relazioni di sistema: le istruzioni che fanno riferimento alle relazioni di sistema, ad esempio `pg_class`, non vengono

acquisite. Si tratta di un'impostazione predefinita per evitare che venga acquisito un numero elevato di piani generati dal sistema utilizzati internamente. Questo vale anche per le tabelle di sistema all'interno delle viste.

- Il cluster database Aurora PostgreSQL potrebbe richiedere una classe di istanza database più grande: a seconda del carico di lavoro, la gestione del piano di query può richiedere una classe di istanza database che fornisce più di 2 vCPU. Il numero di `max_worker_processes` è limitato dalla dimensione della classe di istanza database. Il numero di `max_worker_processes` forniti da una classe di istanza database con 2 vCPU (ad esempio db.t3.medium) può non essere sufficiente per un determinato carico di lavoro. Se utilizzi la gestione del piano di query, ti consigliamo di scegliere una classe di istanza database con più di 2 vCPU per il cluster database Aurora PostgreSQL.

  Se la classe di istanza database non è in grado di supportare il carico di lavoro, la gestione del piano di query genera un messaggio di errore come quello riportato di seguito.

  ```
  WARNING: could not register plan insert background process
  HINT: You may need to increase max_worker_processes.
  ```

  In questo caso, è necessario aumentare il cluster database Aurora PostgreSQL fino a una dimensione della classe di istanza database con più memoria. Per ulteriori informazioni, consulta Motori di database supportati per classi di istanza database.
- I piani già archiviati nelle sessioni non vengono modificati: la gestione dei piani di query fornisce un modo per influenzare i piani di query senza modificare il codice dell'applicazione. Tuttavia, quando un piano generico è già archiviato in una sessione esistente e se desideri modificarne il piano di query, devi prima impostare `plan_cache_mode` su `force_custom_plan` nel gruppo di parametri del cluster database.
- `queryid` in `apg_plan_mgmt.dba_plans` e `pg_stat_statements` può essere diverso quando:
  - Gli oggetti vengono eliminati e ricreati dopo l'archiviazione in apg_plan_mgmt.dba_plans.
  - La tabella `apg_plan_mgmt.plans` viene importata da un altro cluster.

Per informazioni sulle diverse versioni della funzionalità di gestione del piano di query di Aurora PostgreSQL, consulta Versioni dell'estensione Aurora PostgreSQL apg_plan_mgmt nelle Note di rilascio per Aurora PostgreSQL.

Terminologia della gestione del piano di query

In questo argomento vengono utilizzati i seguenti termini.

istruzione gestita

Un'istruzione SQL acquisita dall'ottimizzatore durante la gestione del piano di query. Un'istruzione gestita ha uno o più piani di esecuzione della query archiviati nella vista `apg_plan_mgmt.dba_plans`.

baseline del piano

L'insieme di piani approvati per un'istruzione gestita specificata. Ovvero, tutti i piani per l'istruzione gestita che contengono "Approvato" per la relativa colonna `status` nella vista `dba_plan`.

cronologia del piano

L'insieme di tutti i piani acquisiti per un'istruzione gestita specificata. La cronologia del piano contiene tutti i piani acquisiti per l'istruzione, a prescindere dallo stato.

regressione del piano di query

Il caso in cui l'ottimizzatore sceglie un piano non ottimale rispetto a prima di una determinata modifica all'ambiente del database, ad esempio una nuova versione di PostgreSQL o modifiche alle statistiche.

Versioni di gestione del piano di query per Aurora PostgreSQL

La gestione del piano di query è supportata da tutte le versioni di Aurora PostgreSQL attualmente disponibili. Per ulteriori informazioni, consultare l'elenco di aggiornamenti di Amazon Aurora PostgreSQL nelle Note di rilascio di Aurora PostgreSQL.

La funzionalità di gestione del piano di query viene aggiunta al cluster database Aurora PostgreSQL quando si installa l'estensione `apg_plan_mgmt`. Versioni diverse di Aurora PostgreSQL supportano versioni differenti dell'estensione `apg_plan_mgmt`. Ti consigliamo di aggiornare l'estensione gestione del piano di query alla versione più recente per la versione di Aurora PostgreSQL.

> **Note**
>
> Per le note di rilascio per ogni versione dell'estensione `apg_plan_mgmt`, consultare Aurora PostgreSQL apg_plan_mgmt extension versions nelle Note di rilascio per Aurora PostgreSQL.

È possibile identificare la versione in esecuzione sul cluster connettendosi a un'istanza mediante `psql` e utilizzando il metacomando \dx per elencare le estensioni come mostrato di seguito.

```
labdb=> \dx
                        List of installed extensions
     Name      | Version |    Schema     |                            Description
---------------+---------+---------------
+-------------------------------------------------------------------
 apg_plan_mgmt | 1.0     | apg_plan_mgmt | Amazon Aurora with PostgreSQL compatibility
 Query Plan Management
 plpgsql       | 1.0     | pg_catalog    | PL/pgSQL procedural language
(2 rows)
```

L'output mostra che questo cluster utilizza la versione 1.0 dell'estensione. Solo alcune versioni di `apg_plan_mgmt` sono disponibili per una determinata versione di Aurora PostgreSQL. In alcuni casi, potrebbe essere necessario aggiornare il cluster database Aurora PostgreSQL a una nuova versione secondaria o applicare una patch in modo da poter eseguire l'aggiornamento alla versione più recente della gestione del piano di query. La `apg_plan_mgmt` versione 1.0 mostrata nell'output proviene da un cluster database Aurora PostgreSQL versione 10.17 che non dispone di una versione più recente di `apg_plan_mgmt` disponibile. In questo caso, il cluster database Aurora PostgreSQL deve essere aggiornato a una versione più recente di PostgreSQL.

Per ulteriori informazioni sull'aggiornamento di un cluster database Aurora PostgreSQL a una nuova versione di PostgreSQL, consultare Aggiornamenti del motore di database Amazon Aurora PostgreSQL per 02/06/2020.

Per informazioni su come aggiornare l'estensione `apg_plan_mgmt`, consultare Aggiornamento della gestione del piano di query per Aurora PostgreSQL.

Attivazione della gestione del piano di query per Aurora PostgreSQL

La configurazione della gestione del piano di query per il cluster database Aurora PostgreSQL implica l'installazione di un'estensione e la modifica di diverse impostazioni dei parametri cluster database. Per installare l'estensione `apg_plan_mgmt` e attivare la funzionalità per il cluster database Aurora PostgreSQL sono necessarie autorizzazioni `rds_superuser`.

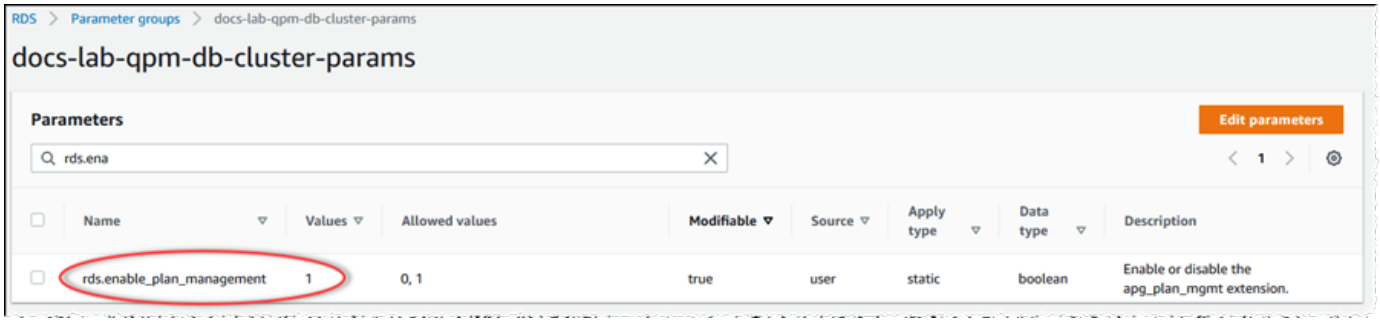
L'installazione dell'estensione crea un nuovo ruolo, `apg_plan_mgmt`, che consente agli utenti del database di visualizzare, gestire e mantenere i piani di query. In qualità di amministratore con privilegi `rds_superuser`, assicurati di concedere il ruolo `apg_plan_mgmt` agli utenti del database in base alle esigenze.

Solo gli utenti con il ruolo `rds_superuser` possono completare la procedura riportata di seguito. Il `rds_superuser` è necessario per creare l'estensione `apg_plan_mgmt` e il relativo ruolo `apg_plan_mgmt`. Il ruolo `apg_plan_mgmt` deve essere concesso agli utenti per gestire l'estensione `apg_plan_mgmt`.

Come attivare la gestione del piano di query per il cluster database Aurora PostgreSQL

I passaggi seguenti attivano la gestione del piano di query per tutte le istruzioni SQL che vengono inviate al cluster database Aurora PostgreSQL. Questa è nota come modalità automatica. Per ulteriori informazioni sulla differenza tra le modalità, consultare Acquisizione dei piani di esecuzione Aurora PostgreSQL.

1. Apri la console di Amazon RDS all'indirizzo https://console.aws.amazon.com/rds/.
2. Scegli un gruppo di parametri cluster di database personalizzato per il cluster database Aurora PostgreSQL. Per attivare la gestione del piano di query e impostare il suo comportamento, è necessario modificare determinati parametri. Per ulteriori informazioni, consulta Creazione di un gruppo di parametri database in Amazon Aurora.
3. Apri il gruppo di parametri cluster di database personalizzato e imposta il parametro `rds.enable_plan_management` su `1`, come mostrato nell'immagine seguente.



   Per ulteriori informazioni, consulta Modifica dei parametri in un gruppo di parametri del cluster di database in Amazon Aurora.
4. Crea un gruppo di parametri database personalizzato che puoi utilizzare per impostare i parametri del piano di query a livello di istanza. Per ulteriori informazioni, consulta Creazione di un gruppo di parametri del cluster di database in Amazon Aurora.
5. Modifica l'istanza di scrittura del cluster database Aurora PostgreSQL per utilizzare il gruppo di parametri database personalizzato. Per ulteriori informazioni, consulta Modifica di un'istanza database in un cluster database.

6. Modifica il cluster database Aurora PostgreSQL per utilizzare il gruppo di parametri database personalizzato. Per ulteriori informazioni, consulta Modifica del cluster di database tramite la console, la CLI e l'API.
7. Riavvia l'istanza database per abilitare le impostazioni del gruppo di parametri personalizzati.
8. Effettua la connessione all'endpoint dell'istanza database del cluster database Aurora PostgreSQL utilizzando `psql` o pgAdmin. L'esempio seguente utilizza l'account `postgres` predefinito per il ruolo `rds_superuser`.

```
psql --host=cluster-instance-1.111122223333.aws-region.rds.amazonaws.com --
port=5432 --username=postgres --password --dbname=my-db
```

9. Crea l'estensione `apg_plan_mgmt` per l'istanza database, come mostrato di seguito.

```
labdb=> CREATE EXTENSION apg_plan_mgmt;
CREATE EXTENSION
```

> **Tip**
>
> Installa l'estensione `apg_plan_mgmt` nel database modello per l'applicazione. Il database modello predefinito è denominato `template1`. Per ulteriori informazioni, consultare Template Databases nella documentazione di PostgreSQL.

10. Modifica il parametro `apg_plan_mgmt.capture_plan_baselines` in `automatic`. Questa impostazione consente all'ottimizzatore di generare i piani per ogni istruzione SQL pianificata o eseguita due o più volte.

> **Note**
>
> La gestione del piano di Query dispone anche di una modalità manuale che è possibile utilizzare per istruzioni SQL specifiche. Per ulteriori informazioni, consulta Acquisizione dei piani di esecuzione Aurora PostgreSQL.

11. Modifica il valore del parametro `apg_plan_mgmt.use_plan_baselines` in "on". Questo parametro consente all'ottimizzatore di scegliere un piano per l'istruzione dalla sua baseline del piano. Per ulteriori informazioni, consulta Utilizzo dei piani gestiti per Aurora PostgreSQL.

> **Note**
>
> È possibile modificare il valore di uno di questi parametri dinamici per la sessione senza dover riavviare l'istanza.

Una volta completata la configurazione della gestione del piano di query, assicurati di concedere il ruolo `apg_plan_mgmt` a tutti gli utenti del database che devono visualizzare, gestire o mantenere piani di query.

Aggiornamento della gestione del piano di query per Aurora PostgreSQL

Ti consigliamo di aggiornare l'estensione gestione del piano di query alla versione più recente per la versione di Aurora PostgreSQL.

1. Esegui la connessione all'istanza di scrittura del cluster database Aurora PostgreSQL come un utente che dispone di privilegi `rds_superuser`. Se hai mantenuto il nome predefinito durante la configurazione dell'istanza, esegui la connessione come `postgres`. In questo esempio viene illustrato come utilizzare `psql`, ma puoi anche utilizzare pgAdmin, se preferisci.

   ```
   psql --host=111122223333.aws-region.rds.amazonaws.com --port=5432 --
   username=postgres --password
   ```

2. Esegui la seguente query per aggiornare l'estensione.

   ```
   ALTER EXTENSION apg_plan_mgmt UPDATE TO '2.1';
   ```

3. Utilizza la funzione apg_plan_mgmt.validate_plans per aggiornare gli hash di tutti i piani. L'ottimizzatore convalida tutti i piani Approvato, Non approvato e Rifiutato per garantire che siano ancora piani validi per la nuova versione dell'estensione.

   ```
   SELECT apg_plan_mgmt.validate_plans('update_plan_hash');
   ```

   Per ulteriori informazioni sull'utilizzo di questa funzione, consultare Convalida dei piani.

4. Utilizza la funzione apg_plan_mgmt.reload per aggiornare eventuali piani nella memoria condivisa con i piani convalidati dalla vista dba_plans.

   ```
   SELECT apg_plan_mgmt.reload();
   ```

Per ulteriori informazioni su tutte le funzioni disponibili per la gestione del piano di query, consultare Informazioni di riferimento sulle funzioni per la gestione del piano di query Aurora PostgreSQL.

Disattivazione della gestione del piano di query per Aurora PostgreSQL

Puoi disabilitare la gestione del piano di query in qualsiasi momento disattivando `apg_plan_mgmt.use_plan_baselines` e `apg_plan_mgmt.capture_plan_baselines`:

```
labdb=> SET apg_plan_mgmt.use_plan_baselines = off;

labdb=> SET apg_plan_mgmt.capture_plan_baselines = off;
```

Best practice per la gestione del piano di query Aurora PostgreSQL

La gestione del piano di query ti permette di controllare come e quando cambiano i piani di esecuzione delle query. In qualità di amministratore di database (DBA), gli obiettivi principali che desideri raggiungere utilizzando QPM sono prevenire le regressioni a seguito di modifiche al database e controllare quando l'ottimizzatore deve utilizzare un nuovo piano. Di seguito sono riportate alcune best practice consigliate su come utilizzare la gestione del piano di query. L'approccio di gestione del piano proattivo differisce da quello reattivo in termini di modalità e tempistiche dell'approvazione dei nuovi piani.

Indice

- Gestione del piano proattiva per prevenire la regressione delle prestazioni
  - Garantire la stabilità del piano dopo un aggiornamento di versione principale
- Gestione del piano reattiva per rilevare e correggere le regressioni delle prestazioni

Gestione del piano proattiva per prevenire la regressione delle prestazioni

Per evitare che si verifichino regressioni delle prestazioni del piano, è possibile far evolvere le baseline del piano eseguendo una procedura che confronta le prestazioni dei piani appena individuati con le prestazioni della baseline esistente dei piani approvati e quindi approva automaticamente il set di piani più rapido come nuova baseline. In questo modo, la baseline dei piani migliora nel tempo man mano che vengono individuati i piani più rapidi.

1. In un ambiente di sviluppo, identificare le istruzioni SQL che hanno il maggiore impatto sulle prestazioni o sulla velocità effettiva del sistema. Quindi acquisire i piani per queste dichiarazioni

come descritto in Acquisizione manuale dei piani per specifiche istruzioni SQL e Acquisizione automatica dei piani.

2. Esportare i piani acquisiti dall'ambiente di sviluppo e importarli nell'ambiente di produzione. Per ulteriori informazioni, consulta Esportazione e importazione dei piani gestiti per Aurora PostgreSQL.
3. In produzione, eseguire l'applicazione e applicare l'utilizzo di piani gestiti approvati. Per ulteriori informazioni, consulta Utilizzo dei piani gestiti per Aurora PostgreSQL. Durante l'esecuzione dell'applicazione, aggiungere nuovi piani quando l'ottimizzatore li rileva. Per ulteriori informazioni, consulta Acquisizione automatica dei piani.
4. Analizzare i piani non approvati e approvare quelli che funzionano bene. Per ulteriori informazioni, consulta Valutazione delle prestazioni del piano.
5. Mentre l'applicazione continua a essere eseguita, l'ottimizzatore inizia a usare i nuovi piani secondo le necessità.

Garantire la stabilità del piano dopo un aggiornamento di versione principale

Ogni versione principale di PostgreSQL include miglioramenti e modifiche all'ottimizzatore di query progettati per potenziare le prestazioni. Tuttavia, i piani di esecuzione delle query generati dal servizio di ottimizzazione nelle versioni precedenti potrebbero causare regressioni delle prestazioni nelle versioni aggiornate più recenti. È possibile utilizzare la gestione del piano di query per risolvere i problemi di prestazione e garantire la stabilità del piano dopo un aggiornamento della versione principale.

L'ottimizzatore utilizza sempre il piano di costo minimo approvato, anche se esistono più piani approvati per la stessa istruzione. Dopo un aggiornamento, l'ottimizzatore potrebbe scoprire nuovi piani, ma questi verranno salvati come piani non approvati. Questi piani vengono eseguiti solo se approvati utilizzando lo stile reattivo di gestione del piano con il parametro unapproved_plan_execution_threshold. È possibile massimizzare la stabilità del piano utilizzando lo stile proattivo di gestione del piano con il parametro evolve_plan_baselines. In questo modo si confrontano le prestazioni dei nuovi piani con quelle dei vecchi piani e si approvano o rifiutano i piani che sono almeno il 10% più veloci rispetto al piano migliore successivo.

Dopo l'aggiornamento, utilizza la funzione `evolve_plan_baselines` con le associazioni dei parametri di query per confrontare le prestazioni del piano prima e dopo l'aggiornamento. I seguenti passaggi presuppongono che tu abbia utilizzato piani gestiti approvati nel tuo ambiente di produzione, come descritto in Utilizzo dei piani gestiti per Aurora PostgreSQL.

1. Prima di procedere con l'aggiornamento, esegui l'applicazione e assicurati che il gestore del piano di query sia in esecuzione. Mentre l'applicazione è in esecuzione, aggiungi i nuovi piani rilevati dall'ottimizzatore. Per ulteriori informazioni, consulta Acquisizione automatica dei piani.
2. Valuta le prestazioni di ciascun piano. Per ulteriori informazioni, consulta Valutazione delle prestazioni del piano.
3. Dopo l'aggiornamento, analizza di nuovo i piani approvati utilizzando la funzione `evolve_plan_baselines`. Confronta le prestazioni prima e dopo l'utilizzo delle associazioni dei parametri di query. Se il nuovo piano è veloce, aggiungilo all'elenco dei piani approvati. Se è più veloce di un altro piano in base alle stesse associazioni di parametri, è possibile contrassegnare il piano più lento come Rejected (Rifiutato).

   Per ulteriori informazioni, consulta Approvazione dei piani migliori. Per ulteriori informazioni su questa funzione, consulta apg_plan_mgmt.evolve_plan_baselines.

Per ulteriori informazioni, consulta Garantire prestazioni coerenti dopo gli aggiornamenti delle versioni principali con la gestione del piano di query di Amazon Aurora edizione compatibile con PostgreSQL.

> **Note**
>
> Quando esegui un aggiornamento di una versione principale utilizzando la replica logica di AWS DMS, assicurati di replicare lo schema `apg_plan_mgmt` per garantire che i piani esistenti vengano copiati nell'istanza aggiornata. Per ulteriori informazioni sulla replica logica, consulta Utilizzo della replica logica per eseguire l'aggiornamento a una versione principale per Aurora PostgreSQL.

Gestione del piano reattiva per rilevare e correggere le regressioni delle prestazioni

Rileva quali piani causano regressioni delle prestazioni monitorando l'applicazione mentre è in esecuzione. Quando rilevi regressioni, rifiuta o correggi manualmente i piani inefficienti seguendo questi passaggi:

1. Durante l'esecuzione dell'applicazione, applica l'uso dei piani gestiti e aggiungi automaticamente i piani appena rilevati come non approvati. Per ulteriori informazioni, consulta Utilizzo dei piani gestiti per Aurora PostgreSQL e Acquisizione automatica dei piani.
2. Monitorare l'applicazione in esecuzione per le regressioni delle prestazioni.

3. Quando viene rilevata una regressione del piano, impostare lo stato del piano su `rejected`. La volta successiva che l'ottimizzatore esegue l'istruzione SQL, ignora automaticamente il piano rifiutato e utilizza invece un piano approvato diverso. Per ulteriori informazioni, consulta Rifiuto o disabilitazione dei piani più lenti.

   In alcuni casi, si potrebbe preferire correggere un piano danneggiato piuttosto che rifiutarlo, disabilitarlo o cancellarlo. Usare l'estensione `pg_hint_plan` per sperimentare il miglioramento di un piano. Con `pg_hint_plan`, si usano commenti speciali per indicare all'ottimizzatore di ignorare il modo in cui normalmente crea un piano. Per ulteriori informazioni, consulta Correzione dei piani mediante pg_hint_plan.

Gestione del piano di query di Aurora PostgreSQL

Con la gestione del piano di query attivata per il cluster database Aurora PostgreSQL, l'ottimizzatore genera e archivia i piani di esecuzione delle query per qualsiasi istruzione SQL che viene elaborata più di una volta. L'ottimizzatore imposta sempre lo stato del primo piano generato dell'istruzione gestita su `dba_plans` e lo archivia nella vista `Approved`.

Il set di piani approvati salvati per un'istruzione gestita è noto come baseline del piano. Mentre l'applicazione è in corso, l'ottimizzatore potrebbe generare piani aggiuntivi per le istruzioni gestite. L'ottimizzatore imposta i piani acquisiti aggiuntivi su uno stato di `Unapproved`.

In seguito, si può decidere se i piani `Unapproved` vengono eseguiti correttamente e cambiarli in `Approved`, `Rejected` o `Preferred`. A tale scopo, si utilizza la funzione `apg_plan_mgmt.evolve_plan_baselines` o la funzione `apg_plan_mgmt.set_plan_status`.

Quando l'ottimizzatore genera un piano per un'istruzione SQL, la gestione del piano di query salva il piano nella tabella `apg_plan_mgmt.plans`. Gli utenti del database cui è stato concesso il ruolo `apg_plan_mgmt` possono visualizzare i dettagli del piano eseguendo una query sulla vista `apg_plan_mgmt.dba_plans`. Ad esempio, nella seguente query vengono elencati i dettagli dei piani attualmente nella vista per un cluster database Aurora PostgreSQL non di produzione.

- `sql_hash`: un identificatore per l'istruzione SQL che è il valore hash per il testo normalizzato dell'istruzione SQL.
- `plan_hash`: un identificatore univoco per il piano che è una combinazione di `sql_hash` e di un hash del piano.
- `status`: lo stato del piano. L'ottimizzatore può eseguire un piano approvato.

- `enabled`: indica se il piano è pronto per l'uso (true) o no (false).
- `plan_outline`: una rappresentazione del piano che viene utilizzata per ricreare il piano di esecuzione effettivo. Gli operatori nella struttura ad albero vengono mappati agli operatori nell'output EXPLAIN.

La vista `apg_plan_mgmt.dba_plans` contiene molte altre colonne contenenti tutti i dettagli del piano, ad esempio la data dell'ultimo utilizzo del piano. Per tutti i dettagli completi, consultare Riferimento per la visualizzazione apg_plan_mgmt.dba_plans per Aurora edizione compatibile con PostgreSQL.

Normalizzazione e l'hash SQL

Nella vista `apg_plan_mgmt.dba_plans`, è possibile identificare un'istruzione gestita in base al suo valore hash SQL. L'hash SQL viene calcolato su una rappresentazione normalizzata dell'istruzione SQL che rimuove alcune differenze, ad esempio i valori letterali.

Il processo di normalizzazione per ogni istruzione SQL consente di conservare spazio e maiuscole/minuscole, in modo da poter leggere e comprendere l'essenza dell'istruzione SQL. La normalizzazione rimuove o sostituisce i seguenti elementi.

- Commenti di blocco iniziali
- La parola chiave EXPLAIN e le opzioni EXPLAIN ed EXPLAIN ANALYZE
- Spazi finali
- Tutti i letterali

Ad esempio, considerare la seguente istruzione.

```
/*Leading comment*/ EXPLAIN SELECT /* Query 1 */ * FROM t WHERE x > 7 AND y = 1;
```

La gestione del piano di query normalizza questa istruzione come mostrato:

```
SELECT /* Query 1 */ * FROM t WHERE x > CONST AND y = CONST;
```

La normalizzazione consente di utilizzare lo stesso hash SQL per istruzioni SQL simili che potrebbero differire solo nei loro valori letterali o dei parametri. In altre parole, possono esistere più piani per lo stesso hash SQL, con un piano diverso ottimale in condizioni diverse.

> **Note**
>
> Una singola istruzione SQL utilizzata con schemi diversi dispone di piani diversi perché è associata allo schema specifico in fase di runtime. Il planner utilizza le statistiche per l'associazione dello schema per scegliere il piano ottimale.

Per ulteriori informazioni su come l'ottimizzatore sceglie un piano, consultare Utilizzo dei piani gestiti per Aurora PostgreSQL. In questa sezione vengono fornite informazioni su come utilizzare `EXPLAIN` e `EXPLAIN ANALYZE` per visualizzare l'anteprima di un piano prima che venga effettivamente utilizzato. Per informazioni dettagliate, consultare Analisi del piano scelto dall'ottimizzatore. Per un'immagine che illustra il processo di scelta di un piano, consultare In che modo l'ottimizzatore sceglie quale piano eseguire..

Acquisizione dei piani di esecuzione Aurora PostgreSQL

La gestione del piano di query di Aurora PostgreSQL offre due diverse modalità di acquisizione dei piani di esecuzione delle query: automatica o manuale. La modalità viene scelta impostando il valore di `apg_plan_mgmt.capture_plans_baselines` su `automatic` o su `manual`. Puoi acquisire i piani di esecuzione per specifiche istruzioni SQL utilizzando l'acquisizione manuale del piano. In alternativa, puoi acquisire tutti i piani (o quelli più lenti) che vengono eseguiti due o più volte durante l'esecuzione dell'applicazione utilizzando l'acquisizione automatica del piano.

Durante l'acquisizione dei piani, l'ottimizzatore imposta lo stato del primo piano acquisito dell'istruzione gestita su `approved`. L'ottimizzatore imposta lo stato di qualsiasi altro piano acquisito per un'istruzione gestita su `unapproved`. Tuttavia, a volte, più di un piano potrebbe essere salvato con lo stato `approved`. Ciò può accadere quando vengono creati più piani per un'istruzione in parallelo e prima che venga eseguito il commit del primo piano per l'istruzione.

Per controllare il numero massimo di piani che possono essere acquisiti e archiviati nella visualizzazione `dba_plans`, imposta il parametro `apg_plan_mgmt.max_plans` nel gruppo di parametri a livello di istanza database. La modifica del parametro `apg_plan_mgmt.max_plans` richiede il riavvio dell'istanza database per rendere effettivo il nuovo valore. Per ulteriori informazioni, vedi il parametro apg_plan_mgmt.max_plans.

Acquisizione manuale dei piani per specifiche istruzioni SQL

Se hai un set di istruzioni SQL da gestire, inserisci le istruzioni in un file script SQL e quindi acquisisci manualmente i piani. Di seguito è riportato un esempio di psql per acquisire manualmente i piani di query per un set di istruzioni SQL.

```
psql> SET apg_plan_mgmt.capture_plan_baselines = manual;
psql> \i my-statements.sql
psql> SET apg_plan_mgmt.capture_plan_baselines = off;
```

Dopo aver acquisito un piano per ogni istruzione SQL, l'ottimizzatore aggiunge una nuova riga alla visualizzazione `apg_plan_mgmt.dba_plans`.

Ti consigliamo di utilizzare le istruzioni EXPLAIN o EXPLUTE EXPLAIN nel file di script SQL. Assicurati di includere abbastanza variazioni nei valori dei parametri per acquisire tutti i piani di interesse.

Se conosci un piano migliore del piano a costo minimo dell'ottimizzatore, puoi forzare l'ottimizzatore a utilizzare il piano migliore. Per farlo, specifica uno o più hint dell'ottimizzatore. Per ulteriori informazioni, consulta Correzione dei piani mediante pg_hint_plan. Per confrontare le prestazioni dei piani `unapproved` e `approved` e approvarli, rifiutarli o eliminarli, consulta Valutazione delle prestazioni del piano.

Acquisizione automatica dei piani

Utilizza l'acquisizione automatica dei piani per situazioni come le seguenti:

- Non conosci le istruzioni SQL specifiche da gestire.
- Hai centinaia o migliaia di istruzioni SQL da gestire.
- L'applicazione utilizza un'API client. Ad esempio, JDBC utilizza le istruzioni preparate senza nome o le istruzioni in modalità bulk che non possono essere espresse in psql.

Per acquisire i piani automaticamente

1. Attivare l'acquisizione automatica dei piani impostando `apg_plan_mgmt.capture_plan_baselines` su `automatic` nel gruppo di parametri a livello di istanza database. Per ulteriori informazioni, consulta Modifica dei parametri in un gruppo di parametri database in Amazon Aurora.

2. Durante l'esecuzione dell'applicazione, l'ottimizzatore acquisisce i piani per ogni istruzione SQL eseguita almeno due volte.

   Durante l'esecuzione dell'applicazione con le impostazioni predefinite dei parametri della gestione del piano di query, l'ottimizzatore acquisisce i piani per ogni istruzione SQL eseguita almeno due volte. L'acquisizione di tutti i piani mentre si utilizzano i valori predefiniti ha un overhead di runtime molto ridotto e può essere abilitato in produzione.

Per disattivare l'acquisizione automatica dei piani

- Impostare il parametro `apg_plan_mgmt.capture_plan_baselines` su `off` nel gruppo di parametri a livello di istanza database.

Per misurare le prestazioni dei piani non approvati e approvarli, rifiutarli o eliminarli, consulta Valutazione delle prestazioni del piano.

Utilizzo dei piani gestiti per Aurora PostgreSQL

Per fare in modo che l'ottimizzatore utilizzi i piani acquisiti per le istruzioni gestite, imposta il parametro `apg_plan_mgmt.use_plan_baselines` su `true`. Di seguito è riportato un esempio di istanza locale.

```
SET apg_plan_mgmt.use_plan_baselines = true;
```

Durante l'esecuzione dell'applicazione, questa impostazione fa in modo che l'ottimizzatore utilizzi il piano a costo minimo, preferito o approvato, valido e abilitato per ciascuna istruzione gestita.

Analisi del piano scelto dall'ottimizzatore

Quando il parametro `apg_plan_mgmt.use_plan_baselines` è impostato su `true`, puoi utilizzare le istruzioni SQL EXPLAIN ANALYZE per far sì che l'ottimizzatore mostri il piano che userebbe se dovesse eseguire l'istruzione . Di seguito è riportato un esempio.

```
EXPLAIN ANALYZE EXECUTE rangeQuery (1,10000);
```

```
                                                        QUERY PLAN
-----------------------------------------------------------------------------------
 Aggregate  (cost=393.29..393.30 rows=1 width=8) (actual time=7.251..7.251 rows=1
 loops=1)
   ->  Index Only Scan using t1_pkey on t1 t  (cost=0.29..368.29 rows=10000 width=0)
 (actual time=0.061..4.859 rows=10000 loops=1)
Index Cond: ((id >= 1) AND (id <= 10000))
         Heap Fetches: 10000
 Planning time: 1.408 ms
 Execution time: 7.291 ms
 Note: An Approved plan was used instead of the minimum cost plan.
 SQL Hash: 1984047223, Plan Hash: 512153379
```

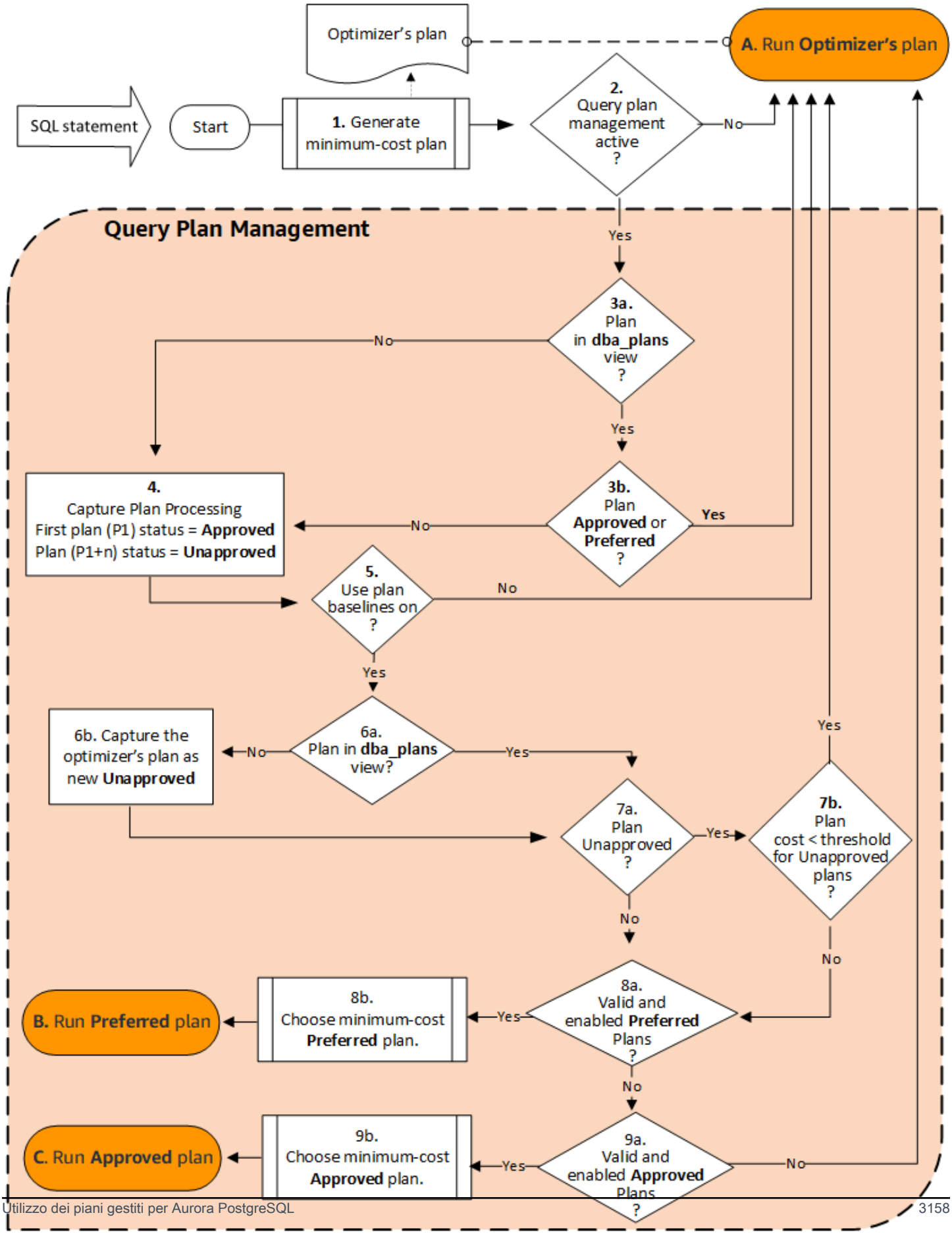
L'output mostra il piano Approvato dalla baseline che viene eseguita. Tuttavia, l'output mostra anche che è stato trovato un piano a costo inferiore. In questo caso, acquisisci questo nuovo piano a costo minimo attivando l'acquisizione automatica del piano come descritto in Acquisizione automatica dei piani.

I nuovi piani vengono sempre acquisiti dall'ottimizzatore come `Unapproved`. Utilizza la funzione `apg_plan_mgmt.evolve_plan_baselines` per confrontare i piani e cambiarli in approvati, rifiutati o disabilitati. Per ulteriori informazioni, consulta Valutazione delle prestazioni del piano.

In che modo l'ottimizzatore sceglie quale piano eseguire.

Il costo di un piano di esecuzione è una stima valutata dall'ottimizzatore per confrontare diversi piani. Durante il calcolo del costo di un piano, l'ottimizzatore include fattori quali le operazioni di CPU e I/O richieste da tale piano. Per ulteriori informazioni sui costi del pianificatore query PostgreSQL, consulta Query Planning nella documentazione di PostgreSQL.

Nella seguente immagine viene illustrata la modalità di scelta di un piano per una determinata istruzione SQL, a seconda che la gestione del piano di query sia attiva o non attiva.


Optimizer's plan
A. Run Optimizer's plan
SQL statement
Start
1. Generate minimum-cost plan
2. Query plan management active ?
No
Query Plan Management
Yes
3a. Plan in dba_plans view ?
No
Yes
4. Capture Plan Processing
First plan (P1) status = Approved
Plan (P1+n) status = Unapproved
3b. Plan Approved or Preferred ?
No
Yes
5. Use plan baselines on ?
No
Yes
6a. Plan in dba_plans view?
No
6b. Capture the optimizer's plan as new Unapproved
Yes
7a. Plan Unapproved ?
Yes
7b. Plan cost < threshold for Unapproved plans ?
Yes
No
No
8a. Valid and enabled Preferred Plans ?
Yes
8b. Choose minimum-cost Preferred plan.
B. Run Preferred plan
No
9a. Valid and enabled Approved Plans ?
Yes
9b. Choose minimum-cost Approved plan.
C. Run Approved plan
No

Di seguito è riportato il flusso:

1. L'ottimizzatore genera un piano a costo minimo per l'istruzione SQL.
2. Se la gestione del piano di query non è attiva, il piano dell'ottimizzatore viene eseguito immediatamente (A. Run Optimizer's plan). La gestione del piano di query è inattiva quando si utilizzano le impostazioni predefinite dei parametri `apg_plan_mgmt.capture_plan_baselines` e `apg_plan_mgmt.use_plan_baselines` (rispettivamente "off" e "false").

   In caso contrario, la gestione del piano di query è attiva. In questo caso, l'istruzione SQL e il relativo piano dell'ottimizzatore vengono ulteriormente valutati prima che venga scelto un piano.

   > **Tip**
   >
   > Utenti del database con il ruolo `apg_plan_mgmt` possono confrontare in maniera proattiva i piani, modificare lo stato dei piani e imporre l'uso di piani specifici, in base alle esigenze. Per ulteriori informazioni, consulta Miglioramento dei piani di query di Aurora PostgreSQL.

3. È possibile che l'istruzione SQL disponga già di piani archiviati da gestione dei piani di query in passato. I piani sono archiviati in `apg_plan_mgmt.dba_plans`, insieme alle informazioni relative alle istruzioni SQL utilizzate per crearli. Le informazioni relative a un piano includono il suo stato. Lo stato di un piano può determinare se è utilizzato o meno, come descritto di seguito.
   a. Se il piano non è incluso tra i piani archiviati per l'istruzione SQL, significa che è la prima volta che questo particolare piano è stato generato dall'ottimizzatore per l'istruzione SQL specificata. Il piano viene inviato all'elaborazione del piano di acquisizione (4).
   b. Se il piano è incluso tra i piani archiviati e il suo stato è Approvato o Preferito, il piano viene eseguito (A. Run Optimizer's plan).

      Se il piano è incluso tra i piani archiviati ma non è né Approvato né Preferito, viene inviato all'elaborazione del piano di acquisizione (4).
4. Quando un piano viene acquisito per la prima volta per una determinata istruzione SQL, il suo stato è sempre impostato su Approvato (P1). Se l'ottimizzatore genera successivamente lo stesso piano per la stessa istruzione SQL, lo stato viene modificato in Non approvato (p1+n).

   Con il piano di acquisito e il relativo stato aggiornato, la valutazione continua nel passaggio successivo (5).

5. La linea di base di un piano è costituita dalla cronologia dell'istruzione SQL e dei relativi piani in vari stati. La gestione del piano di query può tenere conto della linea di base durante la scelta di un piano, a seconda che l'opzione Use plan baselines (Usa linee di base del piano) sia attivata o meno, come illustrato di seguito.
   - L'opzione Use plan baselines (Usa linee di base del piano) è "disattivata" quando il parametro `apg_plan_mgmt.use_plan_baselines` è impostato sul suo valore predefinito (`false`). Il piano non viene confrontato con la linea di base prima di essere eseguito (A. Run Optimizer's plan).
   - L'opzione Use plan baselines (Usa linee di base del piano) è "attivata" quando il parametro `apg_plan_mgmt.use_plan_baselines` è impostato su `true`. Il piano viene ulteriormente valutato utilizzando la linea di base (6).
6. Il piano viene confrontato ad altri piani per l'istruzione nella linea di base.
   a. Se il piano dell'ottimizzatore è incluso tra i piani della linea di base, il suo stato viene controllato (7a).
   b. Se il piano dell'ottimizzatore non è incluso tra i piani della linea di base, viene aggiunto ai piani per l'istruzione come un nuovo piano `Unapproved`.
7. Lo stato del piano viene controllato per determinare solo se è Non è approvato.
   a. Se lo stato del piano è Non approvato, il costo stimato del piano viene confrontato alla stima dei costi specificata per la soglia del piano di esecuzione non approvato.
      - Se il costo stimato del piano è inferiore alla soglia, l'ottimizzatore lo utilizza anche se è Non approvato (A. Run Optimizer's plan). In genere, l'ottimizzatore non esegue un piano Non approvato. Tuttavia, quando il parametro `apg_plan_mgmt.unapproved_plan_execution_threshold` specifica un valore di soglia di costo, l'ottimizzatore confronta il costo del piano Non approvato alla soglia. Se il costo stimato è inferiore alla soglia, l'ottimizzatore esegue il piano. Per ulteriori informazioni, consulta apg_plan_mgmt.unapproved_plan_execution_threshold.
      - Se il costo stimato del piano non è inferiore alla soglia, vengono controllati gli altri attributi del piano (8a).
   b. Se lo stato del piano è diverso da Non approvato, vengono controllati i suoi altri attributi (8a).
8. L'ottimizzatore non utilizzerà un piano che è disabilitato. Ovvero, il piano il cui attributo `enable` è impostato su 'f' (falso). Inoltre, l'ottimizzatore non utilizzerà un piano il cui stato è Rifiutato.

   L'ottimizzatore non può utilizzare piani non validi. I piani possono diventare non validi nel tempo quando gli oggetti da cui dipendono, ad esempio indici e partizioni di tabella, vengono rimossi o eliminati.

a. Se l'istruzione dispone di piani preferiti abilitati e validi, l'ottimizzatore sceglie il piano a costo minimo tra i piani preferiti archiviati per l'istruzione SQL. L'ottimizzatore quindi esegue il piano preferito a costo minimo.
b. Se l'istruzione non dispone di piani preferiti abilitati e validi, viene valutata nel passaggio successivo (9).

9. Se l'istruzione dispone di piani approvati abilitati e validi, l'ottimizzatore sceglie il piano a costo minimo tra i piani approvati archiviati per l'istruzione SQL. L'ottimizzatore quindi esegue il piano approvato a costo minimo.

   Se l'istruzione non dispone di piani approvati validi e abilitati, l'ottimizzatore utilizza il piano a costo minimo (A. Run Optimizer's plan).

Esame dei piani di query Aurora PostgreSQL nella vista dba_plans

Gli utenti e gli amministratori del database cui è stato concesso il ruolo `apg_plan_mgmt` possono visualizzare e gestire i piani archiviati in `apg_plan_mgmt.dba_plans`. Un amministratore di un cluster database Aurora PostgreSQL (un utente con autorizzazioni `rds_superuser`) deve concedere esplicitamente questo ruolo agli utenti del database che devono utilizzare la gestione del piano di query.

La vista `apg_plan_mgmt` contiene la cronologia del piano per tutte le istruzioni SQL gestite per ogni database sull'istanza di scrittura del cluster database Aurora PostgreSQL. Questa vista consente di esaminare i piani, il relativo stato, la data di ultimo utilizzo e tutti gli altri dettagli pertinenti.

Come discusso in Normalizzazione e l'hash SQL, ogni piano gestito è identificato dalla combinazione di un valore hash SQL e un valore hash del piano. Con questi identificatori, puoi usare strumenti come Amazon RDS Performance Insights per tenere traccia delle prestazioni del singolo piano. Per ulteriori informazioni su Performance Insights, consulta Utilizzo di Amazon RDS Performance Insights.

Elenco dei piani gestiti.

Per elencare i piani gestiti, utilizza un'istruzione SELECT sulla visualizzazione `apg_plan_mgmt.dba_plans`. L'esempio seguente mostra alcune colonne nella visualizzazione `dba_plans`, come la colonna `status` che identifica i piani approvati e non approvati.

```
SELECT sql_hash, plan_hash, status, enabled, stmt_name
FROM apg_plan_mgmt.dba_plans;
```

```
 sql_hash   | plan_hash |   status   | enabled | stmt_name
------------+-----------+------------+---------+------------
 1984047223 | 512153379 | Approved   | t       | rangequery
 1984047223 | 512284451 | Unapproved | t       | rangequery
 (2 rows)
```

Per una migliore leggibilità, la query e l'output mostrati elencano solo alcune delle colonne della vista `dba_plans`. Per informazioni complete, consultare Riferimento per la visualizzazione apg_plan_mgmt.dba_plans per Aurora edizione compatibile con PostgreSQL.

Miglioramento dei piani di query di Aurora PostgreSQL

Migliora la gestione del piano di interrogazione valutando le prestazioni del piano e correggendo i piani. Per ulteriori informazioni sul miglioramento dei piani di query, consulta gli argomenti seguenti.

Argomenti

- Valutazione delle prestazioni del piano
- Correzione dei piani mediante pg_hint_plan

Valutazione delle prestazioni del piano

Dopo che l'ottimizzatore acquisisce i piani come non approvati, utilizza la funzione `apg_plan_mgmt.evolve_plan_baselines` per confrontare i piani in base alle prestazioni effettive. A seconda dell'esito degli esperimenti sulle prestazioni, puoi modificare lo stato di un piano da non approvato ad approvato o rifiutato. Puoi invece decidere di utilizzare la funzione `apg_plan_mgmt.evolve_plan_baselines` per disabilitare temporaneamente un piano se non soddisfa i requisiti.

Approvazione dei piani migliori

Nell'esempio seguente viene illustrato come modificare lo stato dei piani gestiti per l'approvazione utilizzando la funzione `apg_plan_mgmt.evolve_plan_baselines`.

```
SELECT apg_plan_mgmt.evolve_plan_baselines (
   sql_hash,
   plan_hash,
   min_speedup_factor := 1.0,
   action := 'approve'
)
```

```
FROM apg_plan_mgmt.dba_plans WHERE status = 'Unapproved';
```

```
NOTICE:     rangequery (1,10000)
NOTICE:     Baseline   [ Planning time 0.761 ms, Execution time 13.261 ms]
NOTICE:     Baseline+1 [ Planning time 0.204 ms, Execution time 8.956 ms]
NOTICE:     Total time benefit: 4.862 ms, Execution time benefit: 4.305 ms
NOTICE:     Unapproved -> Approved
evolve_plan_baselines
----------------------
0
(1 row)
```

L'output mostra un rapporto delle prestazioni per l'istruzione `rangequery` con i binding di parametri 1 e 10.000. Il nuovo piano non approvato (`Baseline+1`) è migliore del piano migliore approvato in precedenza (`Baseline`). Per confermare che il nuovo piano è ora `Approved`, controlla la visualizzazione `apg_plan_mgmt.dba_plans`.

```
SELECT sql_hash, plan_hash, status, enabled, stmt_name
FROM apg_plan_mgmt.dba_plans;
```

```
sql_hash  | plan_hash |  status  | enabled | stmt_name
------------+-----------+----------+---------+------------
1984047223 | 512153379 | Approved | t       | rangequery
1984047223 | 512284451 | Approved | t       | rangequery
(2 rows)
```

Il piano gestito ora include due piani approvati che rappresentano la baseline del piano dell'istruzione. Puoi anche chiamare la funzione `apg_plan_mgmt.set_plan_status` per impostare direttamente il campo di stato di un piano su `'Approved'`, `'Rejected'`, `'Unapproved'` o `'Preferred'`.

Rifiuto o disabilitazione dei piani più lenti

Per rifiutare o disabilitare i piani, passa `'reject'` o `'disable'` come parametro di operazione alla funzione `apg_plan_mgmt.evolve_plan_baselines`. Questo esempio disabilita qualsiasi piano `Unapproved` acquisito che è più lento per almeno il 10 percento del miglior piano `Approved` per l'istruzione.

```
SELECT apg_plan_mgmt.evolve_plan_baselines(
sql_hash,  -- The managed statement ID
plan_hash, -- The plan ID
```

```
1.1,       -- number of times faster the plan must be
'disable'  -- The action to take. This sets the enabled field to false.
)
FROM apg_plan_mgmt.dba_plans
WHERE status = 'Unapproved' AND   -- plan is Unapproved
origin = 'Automatic';       -- plan was auto-captured
```

Puoi anche impostare direttamente un piano su rifiutato o disabilitato. Per impostare direttamente il campo abilitato di un piano su `true` o `false`, chiama la funzione `apg_plan_mgmt.set_plan_enabled`. Per impostare direttamente il campo di stato di un piano su `'Approved'`, `'Rejected'`, `'Unapproved'` o `'Preferred'`, chiama la funzione `apg_plan_mgmt.set_plan_status`.

Per eliminare i piani non validi e che si prevede rimangano non validi, utilizza la funzione `apg_plan_mgmt.validate_plans`. Questa funzione consente di eliminare o disabilitare i piani non validi. Per ulteriori informazioni, consulta Convalida dei piani.

Correzione dei piani mediante pg_hint_plan

L'ottimizzatore di query è progettato per trovare un piano ottimale per tutte le istruzioni e nella maggior parte dei casi trova un buon piano. Tuttavia, occasionalmente potresti realizzare che esiste un piano molto migliore di quello generato dall'ottimizzatore. Due modi consigliati per ottenere che l'ottimizzatore generi un piano desiderato sono l'uso dell'estensione `pg_hint_plan` o l'impostazione delle variabili GUC (Grand Unified Configuration) in PostgreSQL:

- Estensione `pg_hint_plan` – Specifica un "hint" per modificare il funzionamento del pianificatore utilizzando l'estensione `pg_hint_plan` di PostgreSQL. Per installare e ottenere ulteriori informazioni su come utilizzare l'estensione `pg_hint_plan`, consulta la documentazione di pg_hint_plan.
- Variabili GUC – Sostituisci uno o più parametri del modello di costo o altri parametri dell'ottimizzatore, come ad esempio `from_collapse_limit` o `GEQO_threshold`.

Quando utilizzi una di queste tecniche per imporre all'ottimizzatore di query di utilizzare un piano, puoi utilizzare anche la gestione del piano di query per acquisire e imporre l'utilizzo del nuovo piano.

È possibile utilizzare l'estensione `pg_hint_plan` per modificare l'ordine di join, i metodi di join oi percorsi di accesso per un'istruzione SQL. Utilizza un commento SQL con la speciale sintassi `pg_hint_plan` per modificare il modo in cui l'ottimizzatore crea un piano. Ad esempio, supponiamo che l'istruzione SQL del problema abbia un join bidirezionale.

```
SELECT *
FROM t1, t2
WHERE t1.id = t2.id;
```

Quindi, supponiamo che l'ottimizzatore scelga l'ordine di join (t1, t2), ma si sa che l'ordine di join (t2, t1) è più veloce. Il seguente hint forza l'ottimizzatore a utilizzare l'ordine di join più veloce, (t2, t1). Includi EXPLAIN in modo che l'ottimizzatore generi un piano per l'istruzione SQL ma senza eseguire l'istruzione (output non mostrato).

```
/*+ Leading ((t2 t1)) */ EXPLAIN SELECT *
FROM t1, t2
WHERE t1.id = t2.id;
```

I passaggi seguenti illustrano come utilizzare `pg_hint_plan`.

Per modificare il piano generato dall'ottimizzatore e acquisire il piano utilizzando pg_hint_plan

1. Attivare la modalità di acquisizione manuale.

   ```
   SET apg_plan_mgmt.capture_plan_baselines = manual;
   ```

2. Specificare un hint per l'istruzione SQL desiderata.

   ```
   /*+ Leading ((t2 t1)) */ EXPLAIN SELECT *
   FROM t1, t2
   WHERE t1.id = t2.id;
   ```

   Al termine dell'esecuzione, l'ottimizzatore acquisisce il piano nella visualizzazione `apg_plan_mgmt.dba_plans`. Il piano acquisito non include la speciale sintassi dei commenti `pg_hint_plan` perché la gestione del piano di query normalizza l'istruzione rimuovendo i commenti iniziali.

3. Visualizzare i piani gestiti utilizzando la visualizzazione `apg_plan_mgmt.dba_plans`.

   ```
   SELECT sql_hash, plan_hash, status, sql_text, plan_outline
   FROM apg_plan_mgmt.dba_plans;
   ```

4. Impostare lo stato del piano su `Preferred`. In questo modo si assicura che l'ottimizzatore scelga di eseguirlo invece di selezionare un piano dall'insieme dei piani approvati quando il piano a costo minimo non è già `Approved` o `Preferred`.

```
SELECT apg_plan_mgmt.set_plan_status(sql-hash, plan-hash, 'preferred' );
```

5. Disattivare l'acquisizione del piano manuale e applicare l'utilizzo dei piani gestiti.

```
SET apg_plan_mgmt.capture_plan_baselines = false;
SET apg_plan_mgmt.use_plan_baselines = true;
```

Ora, quando viene eseguita l'istruzione SQL originale, l'ottimizzatore sceglierà un piano `Approved` o `Preferred`. Se il piano a costo minimo non è `Approved` o `Preferred`, l'ottimizzatore sceglierà il piano `Preferred`.

Eliminazione dei piani di query di Aurora PostgreSQL

Eliminare i piani di esecuzione non utilizzati o i piani non validi. Per ulteriori informazioni sull'eliminazione dei piani, consultare le sezioni successive.

Argomenti

- Eliminazione dei piani
- Convalida dei piani

Eliminazione dei piani

I piani vengono eliminati automaticamente se non vengono utilizzati da più di un mese, specificatamente, 32 giorni. Questa è l'impostazione di default per il parametro `apg_plan_mgmt.plan_retention_period`. È possibile modificare il periodo di conservazione del piano in un periodo di tempo più lungo o più breve a partire dal valore di 1. La determinazione del numero di giorni dall'ultimo utilizzo di un piano viene calcolata sottraendo la data `last_used` dalla data corrente. La data `last_used` si riferisce alla data più recente in cui l'ottimizzatore ha scelto un piano come il piano di costo minimo o alla data di esecuzione del piano. La data viene archiviata per il piano nella vista `apg_plan_mgmt.dba_plans`.

Ti consigliamo di eliminare i piani che non sono stati utilizzati per molto tempo o che non sono utili. Ad ogni piano è assegnata una data `last_used` che viene aggiornata dall'ottimizzatore ogni volta che esegue il piano o che sceglie il piano come il piano a costo minimo per un'istruzione. Controlla le date `last_used` più recenti per identificare i piani che è possibile eliminare in modo sicuro.

La seguente query restituisce una tabella a tre colonne con il numero totale di piani, i piani che non sono stati eliminati e i piani eliminati con successo. Include una query annidata che è un esempio di utilizzo della funzione `apg_plan_mgmt.delete_plan` per eliminare tutti i piani che non sono stati scelti come il piano a costo minimo negli ultimi 31 giorni e il cui stato non è `Rejected`.

```
SELECT (SELECT COUNT(*) from apg_plan_mgmt.dba_plans) total_plans,
       COUNT(*) FILTER (WHERE result = -1) failed_to_delete,
       COUNT(*) FILTER (WHERE result = 0) successfully_deleted
       FROM (
            SELECT apg_plan_mgmt.delete_plan(sql_hash, plan_hash) as result
            FROM apg_plan_mgmt.dba_plans
            WHERE last_used < (current_date - interval '31 days')
            AND status <> 'Rejected'
            ) as dba_plans ;
```

```
 total_plans | failed_to_delete | successfully_deleted
-------------+------------------+----------------------
           3 |                0 |                    2
```

Per ulteriori informazioni, consulta apg_plan_mgmt.delete_plan.

Per eliminare i piani non validi e che si prevede rimangano non validi, utilizza la funzione `apg_plan_mgmt.validate_plans`. Questa funzione consente di eliminare o disabilitare i piani non validi. Per ulteriori informazioni, consulta Convalida dei piani.

> **Important**
>
> Se i piani estranei non vengono eliminati, esiste il rischio di esaurimento della memoria condivisa dedicata alla gestione del piano di query. Per controllare la quantità di memoria disponibile per i piani gestiti, utilizzare il parametro `apg_plan_mgmt.max_plans`. Impostare questo parametro nel gruppo di parametri database personalizzati e riavviare l'istanza database per rendere effettive le modifiche. Per ulteriori informazioni, vedi il parametro apg_plan_mgmt.max_plans.

Convalida dei piani

Usa la funzione `apg_plan_mgmt.validate_plans` per eliminare o disabilitare i piani non validi.

I piani possono diventare non validi (obsoleti) quando gli oggetti da cui dipendono vengono rimossi, ad esempio un indice o una tabella. Tuttavia, un piano potrebbe diventare non valido solo temporaneamente se l'oggetto rimosso viene poi ricreato. Se un piano non valido può diventare successivamente valido, potresti preferire di disabilitare un piano non valido o non fare nulla anziché eliminarlo.

Per trovare ed eliminare tutti i piani che non sono validi e non sono stati utilizzati nell'ultima settimana, utilizza la funzione `apg_plan_mgmt.validate_plans` come segue.

```
SELECT apg_plan_mgmt.validate_plans(sql_hash, plan_hash, 'delete')
FROM apg_plan_mgmt.dba_plans
WHERE last_used < (current_date - interval '7 days');
```

Per abilitare o disabilitare un piano direttamente, utilizza la funzione `apg_plan_mgmt.set_plan_enabled`.

Esportazione e importazione dei piani gestiti per Aurora PostgreSQL

Puoi esportare i piani gestiti e importarli in un'altra istanza database.

Per esportare i piani gestiti

Un utente autorizzato può copiare qualsiasi sottoinsieme della tabella `apg_plan_mgmt.plans` in un'altra tabella e quindi salvarlo utilizzando il comando pg_dump. Di seguito è riportato un esempio.

```
CREATE TABLE plans_copy AS SELECT *
FROM apg_plan_mgmt.plans [ WHERE predicates ] ;
```

```
% pg_dump --table apg_plan_mgmt.plans_copy -Ft mysourcedatabase > plans_copy.tar
```

```
DROP TABLE apg_plan_mgmt.plans_copy;
```

Per importare i piani gestiti

1. Copiare il file .tar dei piani gestiti esportati nel percorso di sistema in cui devono essere ripristinati i piani.
2. Usare il comando `pg_restore` per copiare il file tar in una nuova tabella.

   ```
   % pg_restore --dbname mytargetdatabase -Ft plans_copy.tar
   ```

3. Unire la tabella `plans_copy` alla tabella `apg_plan_mgmt.plans`, come mostrato nell'esempio seguente.

> **Note**
>
> In alcuni casi, potresti dover eseguire il dump da una versione dell'estensione `apg_plan_mgmt` e ripristinarlo in una versione diversa. In questi casi, le colonne nella tabella dei piani potrebbero essere diverse. In tal caso, denominare le colonne esplicitamente anziché utilizzare SELECT *.

```
INSERT INTO apg_plan_mgmt.plans SELECT * FROM plans_copy
 ON CONFLICT ON CONSTRAINT plans_pkey
 DO UPDATE SET
 status = EXCLUDED.status,
 enabled = EXCLUDED.enabled,
 -- Save the most recent last_used date
 --
 last_used = CASE WHEN EXCLUDED.last_used > plans.last_used
 THEN EXCLUDED.last_used ELSE plans.last_used END,
 -- Save statistics gathered by evolve_plan_baselines, if it ran:
 --
 estimated_startup_cost = EXCLUDED.estimated_startup_cost,
 estimated_total_cost = EXCLUDED.estimated_total_cost,
 planning_time_ms = EXCLUDED.planning_time_ms,
 execution_time_ms = EXCLUDED.execution_time_ms,
 total_time_benefit_ms = EXCLUDED.total_time_benefit_ms,
 execution_time_benefit_ms = EXCLUDED.execution_time_benefit_ms;
```

4. Ricaricare i piani gestiti nella memoria condivisa e rimuovere la tabella dei piani temporanei.

```
SELECT apg_plan_mgmt.reload(); -- refresh shared memory
DROP TABLE plans_copy;
```

Documentazione di riferimento dei parametri per la gestione del piano di query Aurora PostgreSQL

È possibile impostare le preferenze per l'estensione `apg_plan_mgmt` utilizzando i parametri elencati in questa sezione. Questi sono disponibili nel parametro del cluster database personalizzato e

nel gruppo di parametri DB associato al cluster database Aurora PostgreSQL. Questi parametri controllano il comportamento della funzionalità di gestione del piano di query e in che modo influenza l'ottimizzatore. Per informazioni su come impostare la gestione del piano di query, consulta Attivazione della gestione del piano di query per Aurora PostgreSQL. La modifica dei seguenti parametri non ha effetto se l'estensione `apg_plan_mgmt` non è impostata come descritto in tale sezione. Per informazioni sulla modifica dei parametri, consulta Modifica dei parametri in un gruppo di parametri del cluster di database in Amazon Aurora e Gruppi di parametri database per istanze database Amazon Aurora.

Parametri

- apg_plan_mgmt.capture_plan_baselines
- apg_plan_mgmt.plan_capture_threshold
- apg_plan_mgmt.explain_hashes
- apg_plan_mgmt.log_plan_enforcement_result
- apg_plan_mgmt.max_databases
- apg_plan_mgmt.max_plans
- apg_plan_mgmt.plan_hash_version
- apg_plan_mgmt.plan_retention_period
- apg_plan_mgmt.unapproved_plan_execution_threshold
- apg_plan_mgmt.use_plan_baselines
- auto_explain.hashes

apg_plan_mgmt.capture_plan_baselines

Acquisisce i piani di esecuzione delle query generati dall'ottimizzatore per ogni istruzione SQL e li memorizza nella vista `dba_plans`. Per impostazione predefinita, il numero massimo di piani che è possibile memorizzare è 10.000, come specificato dal parametro `apg_plan_mgmt.max_plans`. Per informazioni di riferimento, consulta apg_plan_mgmt.max_plans.

Questo parametro può essere impostato nel gruppo di parametri del cluster database personalizzato o nel gruppo di parametri DB personalizzato. La modifica del valore di questo parametro non richiede un riavvio.

Predefinita	Valori consentiti	Descrizione
off	automatic	Applica questa impostazione a livello di sessione o in un gruppo di parametri per acquisire i piani che vengono utilizzati due o più volte.
	manual	Applica questa impostazione a livello di sessione o in un gruppo di parametri per acquisire i piani che vengono utilizzati una o più volte.
	off	Disattiva l'acquisizione del piano.

Per ulteriori informazioni, consulta Acquisizione dei piani di esecuzione Aurora PostgreSQL.

apg_plan_mgmt.plan_capture_threshold

Specifica una soglia in modo che se il costo totale del piano di esecuzione delle query è inferiore alla soglia, il piano non venga acquisito nella vista `apg_plan_mgmt.dba_plans`.

La modifica del valore di questo parametro non richiede un riavvio.

Predefinita	Valori consentiti	Descrizione
0	0 - 1.79769e+308	Imposta la soglia del costo totale di esecuzione del piano di query `apg_plan_mgmt` per l'acquisizione dei piani.

Per ulteriori informazioni, consulta Esame dei piani di query Aurora PostgreSQL nella vista dba_plans.

apg_plan_mgmt.explain_hashes

Specifica se `EXPLAIN [ANALYZE]` mostra `sql_hash` e `plan_hash` alla fine del relativo output. La modifica del valore di questo parametro non richiede un riavvio.

Predefinita	Valori consentiti	Descrizione
0	0 (off)	EXPLAIN non mostra sql_hash e plan_hash senza l'opzione true degli hash.
	1 (on)	EXPLAIN mostra l'opzione sql_hash e plan_hash senza l'opzione true degli hash.

apg_plan_mgmt.log_plan_enforcement_result

Specifica se i risultati devono essere registrati per verificare se i piani gestiti dalla funzionalità di gestione dei piani di query vengono utilizzati correttamente. Quando viene utilizzato un piano generico archiviato, non verrà scritto alcun record nei file di log. La modifica del valore di questo parametro non richiede un riavvio.

Predefinita	Valori consentiti	Descrizione
nessuno	nessuno	Non mostra alcun risultato dell'applicazione del piano nei file di log.
	on_error	Mostra i risultati dell'applicazione del piano nei file di log solo quando la funzionalità di gestione dei piani di query non utilizza i piani gestiti.
	tutto	Mostra tutti i risultati dell'applicazione del piano nei file di log, incluse le operazioni riuscite e non riuscite.

apg_plan_mgmt.max_databases

Specifica il numero massimo di database sull'istanza di scrittura del cluster database Aurora PostgreSQL che possono utilizzare la gestione del piano di query. L'impostazione predefinita è 10 database. Se sull'istanza sono presenti più di 10 database, il valore dell'impostazione può essere modificato. Per scoprire quanti database sono presenti in una determinata istanza, connettersi all'istanza utilizzando `psql`. Quindi, utilizzare il metacomando psql, `\l`, per elencare i database.

La modifica del valore di questo parametro richiede il riavvio dell'istanza affinché l'impostazione diventi effettiva.

Predefinita	Valori consentiti	Descrizione
10	10-2147483647	Numero massimo di database che possono utilizzare la gestione del piano di query sull'istanza.

Questo parametro può essere impostato nel gruppo di parametri del cluster database personalizzato o nel gruppo di parametri DB personalizzato.

apg_plan_mgmt.max_plans

Imposta il numero massimo di istruzioni SQL che il gestore del piano di query può mantenere nella visualizzazione `apg_plan_mgmt.dba_plans`. Si consiglia di impostare questo parametro su `10000` o superiore per tutte le versioni di Aurora PostgreSQL.

Questo parametro può essere impostato nel gruppo di parametri del cluster database personalizzato o nel gruppo di parametri DB personalizzato. La modifica del valore di questo parametro richiede il riavvio dell'istanza affinché l'impostazione diventi effettiva.

Predefinita	Valori consentiti	Descrizione
10000	10-2147483647	Numero massimo di piani che possono essere archiviati nella vista `apg_plan_mgmt.dba_plans` . Il valore predefinito per Aurora PostgreSQL versione 10 e versioni precedenti è 1000.

Per ulteriori informazioni, consulta Esame dei piani di query Aurora PostgreSQL nella vista dba_plans.

apg_plan_mgmt.plan_hash_version

Specifica i casi d'uso che il calcolo plan_hash è progettato per coprire. Una versione superiore di `apg_plan_mgmt.plan_hash_version` copre tutte le funzionalità della versione inferiore. Ad esempio, la versione 3 copre i casi d'uso supportati dalla versione 2.

La modifica del valore di questo parametro deve essere seguita da una chiamata a `apg_plan_mgmt.validate_plans('update_plan_hash')`. Aggiorna i valori plan_hash in ogni database con apg_plan_mgmt installato e le voci nella tabella dei piani. Per ulteriori informazioni, consulta Convalida dei piani

Predefinita	Valori consentiti	Descrizione
1	1	Calcolo di plan_hash predefinito.
	2	Calcolo di plan_hash modificato per il supporto di più schemi.
	3	Calcolo di plan_hash modificato per il supporto di più schemi e delle tabelle partizionate.
	4	plan_hash calculation modificato per gli operatori paralleli e per supportare i nodi di materializzazione.

apg_plan_mgmt.plan_retention_period

Specifica il numero di giorni per cui si desidera mantenere i piani nella vista `apg_plan_mgmt.dba_plans` prima che vengano eliminati automaticamente. Per impostazione predefinita, un piano viene eliminato quando sono trascorsi 32 giorni dall'ultimo utilizzo del piano (la colonna `last_used` nella vista `apg_plan_mgmt.dba_plans`). Questa impostazione può essere modificata in un numero qualsiasi, 1 e valori superiori.

La modifica del valore di questo parametro richiede il riavvio dell'istanza affinché l'impostazione diventi effettiva.

Predefinita	Valori consentiti	Descrizione
32	1-2147483647	Numero massimo di giorni dall'ultimo utilizzo di un piano prima che venga eliminato.

Per ulteriori informazioni, consulta Esame dei piani di query Aurora PostgreSQL nella vista dba_plans.

apg_plan_mgmt.unapproved_plan_execution_threshold

Specifica una soglia di costo al di sotto della quale un piano non approvato può essere utilizzato dall'ottimizzatore. La soglia è 0 per impostazione predefinita, in modo che l'ottimizzatore non esegua piani non approvati. L'impostazione di questo parametro su una soglia di costo estremamente bassa, ad esempio 100, evita il sovraccarico di applicazione del piano nei piani banali. È inoltre possibile impostare questo parametro su un valore estremamente elevato, ad esempio 10000000, utilizzando lo stile reattivo di gestione del piano. Ciò consente all'ottimizzatore di utilizzare tutti i piani scelti senza generare sovraccarichi dovuti all'applicazione del piano. Tuttavia, quando viene trovato un piano errato, è possibile contrassegnarlo manualmente come "rejected" in modo che non venga utilizzato la volta successiva.

Il valore di questo parametro rappresenta una stima dei costi per l'esecuzione di un determinato piano. Se il costo di un piano non approvato è inferiore a tale costo stimato, verrà utilizzato dall'ottimizzatore per l'istruzione SQL. I piani acquisiti e il relativo stato (Approvato, Non approvato) sono visibili nella vista `dba_plans`. Per ulteriori informazioni, consulta Esame dei piani di query Aurora PostgreSQL nella vista dba_plans.

La modifica del valore di questo parametro non richiede un riavvio.

Predefinita	Valori consentiti	Descrizione
0	0-2147483647	Costo del piano stimato al di sotto del quale viene utilizzato un piano non approvato.

Per ulteriori informazioni, consulta Utilizzo dei piani gestiti per Aurora PostgreSQL.

apg_plan_mgmt.use_plan_baselines

Specifica che l'ottimizzatore deve utilizzare uno dei piani approvati acquisiti e archiviati nella vista `apg_plan_mgmt.dba_plans`. Per impostazione predefinita, questo parametro è disattivato (false). Di conseguenza, l'ottimizzatore utilizzerà il piano a costo minimo che genera senza un'ulteriore valutazione. L'attivazione di questo parametro (impostandolo su true) obbliga l'ottimizzatore a scegliere un piano di esecuzione della query per l'istruzione dalla sua baseline del piano. Per ulteriori informazioni, consulta Utilizzo dei piani gestiti per Aurora PostgreSQL. Per trovare un'immagine che descrive questo processo, consulta In che modo l'ottimizzatore sceglie quale piano eseguire..

Questo parametro può essere impostato nel gruppo di parametri del cluster database personalizzato o nel gruppo di parametri DB personalizzato. La modifica del valore di questo parametro non richiede un riavvio.

Predefinita	Valori consentiti	Descrizione
false	true	Utilizzare un piano Approvato, Preferito o Non approvato del parametro `apg_plan_mgmt.dba_plans` . Se nessuno di questi soddisfa tutti i criteri di valutazione dell'ottimizzatore, è possibile utilizzare il piano a costo minimo generato. Per ulteriori informazioni, consulta In che modo l'ottimizzatore sceglie quale piano eseguire..
	false	Utilizzare il piano a costo minimo generato dall'ottimizzatore.

I tempi di risposta dei diversi piani acquisiti possono essere valutati e lo stato del piano modificato, in base alle esigenze. Per ulteriori informazioni, consulta Miglioramento dei piani di query di Aurora PostgreSQL.

auto_explain.hashes

Specifica se l'output di auto_explain mostra sql_hash e plan_hash. La modifica del valore di questo parametro non richiede un riavvio.

Predefinita	Valori consentiti	Descrizione
0(off)	0(off)	Il risultato `auto_explain` non mostra `sql_hash` e `plan_hash` .
	1(on)	Il risultato `auto_explain` mostra `sql_hash` e `plan_hash` .

Informazioni di riferimento sulle funzioni per la gestione del piano di query Aurora PostgreSQL

L'estensione `apg_plan_mgmt` fornisce le seguenti funzioni.

Funzioni

- apg_plan_mgmt.copy_outline
- apg_plan_mgmt.delete_plan
- apg_plan_mgmt.evolve_plan_baselines
- apg_plan_mgmt.get_explain_plan
- apg_plan_mgmt.plan_last_used
- apg_plan_mgmt.reload
- apg_plan_mgmt.set_plan_enabled
- apg_plan_mgmt.set_plan_status
- apg_plan_mgmt.update_plans_last_used
- apg_plan_mgmt.validate_plans

apg_plan_mgmt.copy_outline

Copia un determinato hash e una struttura del piano SQL in un hash e una struttura del piano SQL di destinazione, sovrascrivendo così l'hash e la struttura del piano di destinazione. Questa funzione è disponibile in `apg_plan_mgmt` 2.3 e versioni successive.

Sintassi

```
apg_plan_mgmt.copy_outline(
```

```
    source_sql_hash,
    source_plan_hash,
    target_sql_hash,
    target_plan_hash,
    force_update_target_plan_hash
)
```

Valore restituito

Restituisce 0 quando la copia ha esito positivo. Genera eccezioni per gli input non validi.

Parametri

Parametro	Descrizione
`source_sql_hash`	L'ID `sql_hash` associato al `plan_hash` da copiare nella query di destinazione.
`source_plan_hash`	L'ID `plan_hash` da copiare nella query di destinazione.
`target_sql_hash`	L'ID `sql_hash` della query da aggiornare con l'hash e la struttura del piano di origine.
`target_plan_hash`	L'ID `plan_hash` della query da aggiornare con l'hash e la struttura del piano di origine.
`force_update_target_plan_hash`	(Facoltativo) L'ID `target_plan_hash` della query viene aggiornato anche se il piano di origine non è riproducibile per `target_sql_hash`. Se impostata su true, la funzione può essere utilizzata per copiare i piani tra schemi in cui le colonne e i nomi delle relazioni sono coerenti.

Note per l'utilizzo

Questa funzione consente di copiare un hash e una struttura del piano che utilizza i suggerimenti per altre dichiarazioni simili e quindi evita la necessità di usare istruzioni di suggerimento in linea ad ogni

occorrenza nelle istruzioni di destinazione. Se la query di destinazione aggiornata genera un piano non valido, questa funzione restituisce un errore ed esegue il rollback del tentativo di aggiornamento.

apg_plan_mgmt.delete_plan

Elimina un piano gestito.

Sintassi

```
apg_plan_mgmt.delete_plan(
    sql_hash,
    plan_hash
)
```

Valore restituito

Restituisce 0 se l'eliminazione ha esito positivo o -1 se l'eliminazione non riesce.

Parametri

Parametro	Descrizione
`sql_hash`	L'ID `sql_hash` dell'istruzione SQL gestita del piano.
`plan_hash`	L'ID `plan_hash` del piano gestito.

apg_plan_mgmt.evolve_plan_baselines

Verifica se un piano già approvato è più veloce o se un piano identificato dall'ottimizzatore di query come piano a costo minimo è più veloce.

Sintassi

```
apg_plan_mgmt.evolve_plan_baselines(
    sql_hash,
    plan_hash,
    min_speedup_factor,
    action
)
```

Valore restituito

Il numero di piani che non sono più veloci del miglior piano approvato.

Parametri

Parametro	Descrizione
`sql_hash`	L'ID `sql_hash` dell'istruzione SQL gestita del piano.
`plan_hash`	L'ID `plan_hash` del piano gestito. Utilizza NULL per indicare tutti i piani che hanno lo stesso valore di ID `sql_hash`.
`min_speedup_factor`	Il fattore di velocità minima indica quante volte un piano deve essere più veloce per essere approvato rispetto al migliore dei piani già approvati In alternativa, questo fattore può corrispondere quante volte deve essere più lento per essere rifiutato o disabilitato. Questo è un valore float positivo.
`action`	L'azione che deve essere eseguita dalla funzione. I valori validi includono i seguenti. Non c'è distinzione tra lettere maiuscole e minuscole. • `'disable'` – Disabilita ogni piano corrispondente che non soddisfa il fattore di accelerazione minimo. • `'approve'` – Abilita ogni piano corrispondente che soddisfa il fattore di accelerazione minimo e imposta lo stato su `approved`. • `'reject'` – Per ogni piano corrispondente che non soddisfa il fattore di accelerazione minimo, imposta lo stato su `rejected`. • NULL – La funzione restituisce semplicemente il numero di piani che non apportano alcun beneficio in termini di prestazioni poiché non soddisfano il fattore di accelerazione minimo.

Note per l'utilizzo

Imposta i piani specificati su approvato, rifiutato o disabilitato in base al fatto che la pianificazione più il tempo di esecuzione sia più veloce del migliore piano approvato da un fattore che puoi impostare. Il parametro action può essere impostato su `'approve'` o `'reject'` per approvare o rifiutare

automaticamente un piano che soddisfa i criteri di prestazione. In alternativa, potrebbe essere impostato su '' (stringa vuota) per eseguire l'esperimento sulle prestazioni e produrre un report, senza intraprendere alcuna azione.

Puoi evitare di rieseguire inutilmente la funzione `apg_plan_mgmt.evolve_plan_baselines` per un piano su cui è stata eseguita di recente. Per farlo, limita i piani solo a quelli creati di recente e non ancora approvati. In alternativa, puoi evitare di eseguire la funzione `apg_plan_mgmt.evolve_plan_baselines` su qualsiasi piano approvato che abbia un timestamp recente `last_verified`.

Conduci un esperimento delle prestazioni per confrontare la pianificazione più il tempo di esecuzione di ciascun piano rispetto agli altri piani nella baseline. In alcuni casi, esiste un solo piano per un'istruzione e il piano è approvato. In tal caso, confronta la pianificazione più il tempo di esecuzione del piano con la pianificazione più il tempo di esecuzione dell'utilizzo di nessun piano.

Il vantaggio (o lo svantaggio) incrementale di ciascun piano è registrato nella visualizzazione `apg_plan_mgmt.dba_plans` nella colonna `total_time_benefit_ms`. Quando questo valore è positivo, c'è un vantaggio misurabile in termini di prestazioni per includere questo piano nella baseline.

Oltre a raccogliere i tempi di pianificazione ed esecuzione di ogni piano candidato, la colonna `last_verified` della visualizzazione `apg_plan_mgmt.dba_plans` viene aggiornata con il `current_timestamp`. È possibile utilizzare il time stamp `last_verified` per evitare di eseguire nuovamente questa funzione su un piano le cui prestazioni sono state verificate di recente.

apg_plan_mgmt.get_explain_plan

Genera il testo di una istruzione `EXPLAIN` per l'istruzione SQL specificata.

Sintassi

```
apg_plan_mgmt.get_explain_plan(
    sql_hash,
    plan_hash,
    [explainOptionList]
)
```

Valore restituito

Restituisce le statistiche di runtime per le istruzioni SQL specificate. Utilizzare senza `explainOptionList` per restituire un piano `EXPLAIN` semplice.

Parametri

Parametro	Descrizione
`sql_hash`	L'ID `sql_hash` dell'istruzione SQL gestita del piano.
`plan_hash`	L'ID `plan_hash` del piano gestito.
`explainOptionList`	Un elenco separato da virgole di opzioni di spiegazione. I valori validi includono `'analyze'` , `'verbose'` , `'buffers'` , `'hashes'` e `'format json'`. Se l'elenco di `explainOptionList` è NULL o una stringa vuota (''), questa funzione genera un'istruzione `EXPLAIN`, senza alcuna statistica.

Note per l'utilizzo

Per il `explainOptionList`, è possibile utilizzare una delle stesse opzioni che si utilizzerebbe con una istruzione `EXPLAIN`. L'ottimizzatore Aurora PostgreSQL concatena l'elenco delle opzioni fornite all'istruzione `EXPLAIN`.

apg_plan_mgmt.plan_last_used

Restituisce la data `last_used` del piano specificato dalla memoria condivisa.

> **Note**
>
> Il valore nella memoria condivisa è sempre aggiornato sull'istanza database primaria nel cluster database. Il valore viene scaricato solo periodicamente nella colonna `last_used` della visualizzazione `apg_plan_mgmt.dba_plans`.

Sintassi

```
apg_plan_mgmt.plan_last_used(
    sql_hash,
```

```
    plan_hash
)
```

Valore restituito

Restituisce la data `last_used`.

Parametri

Parametro	Descrizione
`sql_hash`	L'ID `sql_hash` dell'istruzione SQL gestita del piano.
`plan_hash`	L'ID `plan_hash` del piano gestito.

apg_plan_mgmt.reload

Ricarica i piani nella memoria condivisa dalla visualizzazione `apg_plan_mgmt.dba_plans`.

Sintassi

```
apg_plan_mgmt.reload()
```

Valore restituito

Nessuna.

Parametri

Nessuna.

Note per l'utilizzo

Chiama `reload` per le seguenti situazioni:

- Usalo per aggiornare immediatamente la memoria condivisa di una replica di sola lettura, invece di aspettare che i nuovi piani si propaghino alla replica.
- Usalo dopo l'importazione dei piani gestiti.

apg_plan_mgmt.set_plan_enabled

Abilita o disabilita un piano gestito.

Sintassi

```
apg_plan_mgmt.set_plan_enabled(
    sql_hash,
    plan_hash,
    [true | false]
)
```

Valore restituito

Restituisce 0 se l'impostazione ha esito positivo o -1 se l'impostazione non riesce.

Parametri

Parametro	Descrizione
`sql_hash`	L'ID `sql_hash` dell'istruzione SQL gestita del piano.
`plan_hash`	L'ID `plan_hash` del piano gestito.
`enabled`	Valore booleano true o false: • Il valore `true` abilita il piano. • Il valore `false` disabilita il piano.

apg_plan_mgmt.set_plan_status

Impostare lo stato di un piano gestito su `Approved`, `Unapproved`, `Rejected` o `Preferred`.

Sintassi

```
apg_plan_mgmt.set_plan_status(
    sql_hash,
```

```
    plan_hash,
    status
)
```

Valore restituito

Restituisce 0 se l'impostazione ha esito positivo o -1 se l'impostazione non riesce.

Parametri

Parametro	Descrizione
`sql_hash`	L'ID `sql_hash` dell'istruzione SQL gestita del piano.
`plan_hash`	L'ID `plan_hash` del piano gestito.
`status`	Stringa con uno dei seguenti valori: • `'Approved'` • `'Unapproved'` • `'Rejected'` • `'Preferred'` Il caso utilizzato non ha importanza, tuttavia il valore di stato è impostato su maiuscole iniziali nella vista `apg_plan_mgmt.dba_plans`. Per ulteriori informazioni su questi valori, consulta `status` in Riferimento per la visualizzazione apg_plan_mgmt.dba_plans per Aurora edizione compatibile con PostgreSQL.

apg_plan_mgmt.update_plans_last_used

Aggiorna immediatamente la tabella dei piani con la data `last_used` memorizzata nella memoria condivisa.

Sintassi

```
apg_plan_mgmt.update_plans_last_used()
```

Valore restituito

Nessuna.

Parametri

Nessuna.

Note per l'utilizzo

Eseguire una chiamata a `update_plans_last_used` per assicurarsi che le query contro la colonna `dba_plans.last_used` utilizzino le informazioni più aggiornate. Se il la data `last_used` non viene aggiornata immediatamente, un processo in background aggiorna la tabella dei piani con la data `last_used` una volta ogni ora (per impostazione predefinita).

Ad esempio, se un'istruzione con una certa `sql_hash` inizia a funzionare lentamente, è possibile determinare quali piani per quell'istruzione sono stati eseguiti dall'inizio della regressione delle prestazioni. Per fare ciò, innanzitutto svuotare i dati nella memoria condivisa su disco in modo che le date `last_used` siano correnti e quindi interroghino tutti i piani dell'istruzione `sql_hash` con la regressione delle prestazioni. Nella query, assicurati che la data `last_used` sia maggiore o uguale alla data in cui è iniziata la regressione delle prestazioni. La query identifica il piano o l'insieme di piani che potrebbero essere responsabili della regressione delle prestazioni. È possibile utilizzare `apg_plan_mgmt.get_explain_plan` con `explainOptionList` impostato su `verbose, hashes`. È possibile utilizzare anche `apg_plan_mgmt.evolve_plan_baselines` per analizzare il piano e qualsiasi piano alternativo che potrebbe funzionare meglio.

La funzione `update_plans_last_used` ha un effetto solo sull'istanza database primaria del cluster database.

apg_plan_mgmt.validate_plans

Convalida che l'ottimizzatore può ancora ricreare piani. L'ottimizzatore convalida i piani `Approved`, `Unapproved` e `Preferred`, se il piano è abilitato o disabilitato. I piani `Rejected` non sono convalidati. Facoltativamente, puoi utilizzare la funzione `apg_plan_mgmt.validate_plans` per eliminare o disabilitare i piani non validi.

Sintassi

```
apg_plan_mgmt.validate_plans(
    sql_hash,
```

```
    plan_hash,
    action)

apg_plan_mgmt.validate_plans(
    action)
```

Valore restituito

Numero di piani non validi.

Parametri

Parametro	Descrizione
`sql_hash`	L'ID `sql_hash` dell'istruzione SQL gestita del piano.
`plan_hash`	L'ID `plan_hash` del piano gestito. Utilizza NULL per indicare tutti i piani con lo stesso valore di ID `sql_hash`.
`action`	L'azione che deve essere eseguita dalla funzione per i piani non validi. I valori di stringa validi includono i seguenti: Non c'è distinzione tra lettere maiuscole e minuscole. • `'disable'` – Ogni piano non valido è disabilitato. • `'delete'` – Ogni piano non valido è eliminato. • `'update_plan_hash'` – Aggiorna l'ID `plan_hash` per i piani che non possono essere riprodotti esattamente. Consente inoltre di correggere un piano riscrivendo l'SQL. Puoi registrare il piano buono come piano `Approved` per l'SQL di origine. • NULL – La funzione restituisce semplicemente il numero di piani non validi. Non vengono eseguite altre operazioni. • '' – Una stringa vuota genera un messaggio che indica il numero di piani validi e non validi. Qualsiasi altro valore viene considerato come una stringa vuota.

Note per l'utilizzo

Usa il modulo `validate_plans(action)` per convalidare tutti i piani gestiti per tutte le istruzioni gestite nell'intera visualizzazione `apg_plan_mgmt.dba_plans`.

Usa il modulo `validate_plans(sql_hash, plan_hash, action)` per convalidare un piano gestito specificato con `plan_hash`, per un'istruzione gestita specificata con `sql_hash`.

Usa il modulo `validate_plans(sql_hash, NULL, action)` per convalidare tutti i piani gestiti per l'istruzione gestita specificata con `sql_hash`.

Riferimento per la visualizzazione apg_plan_mgmt.dba_plans per Aurora edizione compatibile con PostgreSQL

Le colonne del piano nella visualizzazione `apg_plan_mgmt.dba_plans` includono le seguenti.

Colonna dba_plans	Descrizione
`cardinality_error`	Misura dell'errore tra la cardinalità stimata e alla cardinalità effettiva. Cardinality è il numero di righe della tabella che saranno elaborate dal piano. Se l'errore di cardinalità è ampio, aumenta la probabilità che il piano non sia ottimale. Questa colonna è popolata dalla funzione apg_plan_mgmt.evolve_plan_baselines.
`compatibility_level`	Questo parametro mostra la data dell'ultima convalida di un piano di query. In Aurora PostgreSQL versioni 12.19, 13.15, 14.12, 15.7, 16.3 e successive, visualizza il numero di versione di Aurora. Per le versioni precedenti, visualizza un numero di versione specifico della funzionalità. **Note** Mantenere questo parametro sul valore predefinito. Aurora PostgreSQL imposta e aggiorna automaticamente questo valore.
`created_by`	L'utente autenticato (`session_user`) che ha creato il piano.

Colonna dba_plans	Descrizione
`enabled`	Indicatore che mostra se il piano è abilitato o disabilitato. Tutte i piani sono abilitati per impostazione predefinita. Puoi disabilit are i piani per impedire che vengano utilizzati dall'ottimizzatore . Per modificare questo valore, utilizza la funzione apg_plan_ mgmt.set_plan_enabled.
`environment_variab les`	Parametri e valori PostgreSQL Grand Unified Configuration (GUC) che l'ottimizzatore ha ignorato nel momento in cui è stato acquisito il piano.
`estimated_startup_ cost`	Il costo di installazione stimato dell'ottimizzatore prima che fornisca le righe di una tabella.
`estimated_total_cost`	Il costo dell'ottimizzatore stimato per la consegna della riga finale della tabella.
`execution_time_ben efit_ms`	Il tempo di esecuzione beneficia in termini di millisecondi dell'abil itazione del piano. Questa colonna è popolata dalla funzione apg_plan_mgmt.evolve_plan_baselines.
`execution_time_ms`	Il tempo stimato di esecuzione del piano in millisecondi. Questa colonna è popolata dalla funzione apg_plan_mgmt.evol ve_plan_baselines.
`has_side_effects`	Valore che indica che l'istruzione SQL è un'istruzione DML (Data Manipulation Language) o un'istruzione SELECT che contiene una funzione VOLATILE.

Colonna dba_plans	Descrizione
`last_used`	Questo valore viene aggiornato alla data corrente ogni volta che il piano viene eseguito o quando il piano è il piano a costo minimo dell'ottimizzatore di query. Questo valore è archiviat o nella memoria condivisa e periodicamente viene riportato su disco. Per ottenere il valore più aggiornato, leggi la data dalla memoria condivisa chiamando la funzione `apg_plan_mgmt.plan_last_used(sql_hash, plan_hash)` anziché il valore `last_used` . Per ulteriori informazioni, vedi il parametro apg_plan_mgmt.plan_retention_period.
`last_validated`	La data e l'ora della verifica più recente del piano che potrebbe essere ricreato dalla funzione apg_plan_mgmt.validate_plans o apg_plan_mgmt.evolve_plan_baselines.
`last_verified`	La data e l'ora della verifica più recente di un piano come il piano più performante per i parametri specificati dalla funzione apg_plan_mgmt.evolve_plan_baselines.
`origin`	Come il piano è stato acquisito con il parametro apg_plan_mgmt.capture_plan_baselines. I valori validi includono i seguenti: `M` – Il piano è stato acquisito con l'acquisizione manuale del piano. `A` – Il piano è stato acquisito con l'acquisizione automatica del piano.
`param_list`	I valori dei parametri che sono stati passati all'istruzione se è un'istruzione preparata.
`plan_created`	La data e l'ora di creazione del piano.
`plan_hash`	Identificatore del piano. La combinazione di `plan_hash` e `sql_hash` identifica in modo univoco un piano specifico.

Colonna dba_plans	Descrizione
`plan_outline`	Rappresentazione del piano che viene utilizzata per ricreare il piano di esecuzione effettivo e che è indipendente dal database. Gli operatori nella struttura ad albero corrispondono agli operatori presenti nell'output di EXPLAIN.
`planning_time_ms`	Il tempo effettivo per eseguire il pianificatore, in millisecondi. Questa colonna è popolata dalla funzione apg_plan_mgmt.evolve_plan_baselines.
`queryId`	Un hash dell'istruzione, come calcolato dall'estensione `pg_stat_statements` . Non è un identificatore stabile o indipendente dal database in quanto dipende dagli identificatori di oggetto (OID). Il valore sarà `0` se `compute_query_id` è `off` durante l'acquisizione del piano di query.
`sql_hash`	Valore hash del testo dell'istruzione SQL, normalizzato con valori letterali rimossi.
`sql_text`	Il testo completo dell'istruzione SQL.

Colonna dba_plans	Descrizione
`status`	Lo stato di un piano che determina il modo in cui l'ottimizzatore utilizza un piano. I valori validi includono i seguenti. • `Approved`: un piano utilizzabile che l'ottimizzatore può scegliere di eseguire. L'ottimizzatore esegue il piano meno costoso da una serie di piani approvati dell'istruzione gestita (baseline). Per reimpostare un piano su approvato, utilizza la funzione apg_plan_mgmt.evolve_plan_baselines. • `Unapproved` : un piano acquisito che non hai verificato per l'uso. Per ulteriori informazioni, consulta Valutazione delle prestazioni del piano. • `Rejected`: un piano che non viene utilizzato dall'ottimizzatore . Per ulteriori informazioni, consulta Rifiuto o disabilitazione dei piani più lenti. • `Preferred` : un piano che hai indicato come piano preferito da utilizzare per un'istruzione gestita. Se il piano a costo minimo dell'ottimizzatore non è un piano approvato o preferito, puoi ridurre l'overhead derivante dall'applicazione del piano. Per farlo, indica come sottoinsi eme dei piani approvat `Preferred` . Quando il piano a costo minimo dell'ottimizzatore non è un piano `Approved`, viene scelto un piano `Preferred` prima di un piano `Approved`. Per reimpostare un piano su `Preferred` , utilizza la funzione apg_plan_mgmt.set_plan_status.
`stmt_name`	Il nome dell'istruzione SQL all'interno di un'istruzione PREPARE. Questo valore è una stringa vuota per un'istruzione preparata senza nome. Questo valore è NULL per un'istruzione non preparata.

Colonna dba_plans	Descrizione
`total_time_benefit_ms`	Il vantaggio in termini di tempo totale in millisecondi derivato dall'abilitazione di questo piano. Questo valore considera sia il tempo di pianificazione che il tempo di esecuzione. Se il valore è negativo, non è vantaggioso abilitare questo piano. Questa colonna è popolata dalla funzione apg_plan_mgmt.evolve_plan_baselines.

Funzionalità avanzate della gestione del piano di query

Di seguito sono disponibili informazioni sulle funzionalità avanzate della gestione dei piani di query (QPM) di Aurora PostgreSQL:

Argomenti

- Acquisizione dei piani di esecuzione Aurora PostgreSQL nelle repliche
- Supporto della partizione di tabelle

Acquisizione dei piani di esecuzione Aurora PostgreSQL nelle repliche

QPM (Query Plan Management) consente di acquisire i piani di query generati dalle repliche di Aurora e di archiviarli sull'istanza database principale del cluster di database Aurora. È possibile raccogliere i piani di query da tutte le repliche Aurora e mantenere un insieme di piani ottimali in una tabella centrale persistente sull'istanza primaria. Potrai applicare tali piani su altre repliche secondo necessità. Ciò consente di mantenere la stabilità dei piani di esecuzione e di migliorare le prestazioni delle query tra i cluster database e le versioni del motore.

Argomenti

- Prerequisiti
- Gestione dell'acquisizione del piano per le repliche di Aurora
- Risoluzione dei problemi

Prerequisiti

Attiva **`capture_plan_baselines parameter`** in una replica di Aurora: imposta il parametro `capture_plan_baselines` su automatico o manuale per acquisire i piani nelle repliche di Aurora. Per ulteriori informazioni, consulta apg_plan_mgmt.capture_plan_baselines.

Installa l'estensione postgres_fdw: è necessario installare l'estensione foreign data wrapper `postgres_fdw` per acquisire i piani nelle repliche di Aurora. Per installare l'estensione, esegui il comando seguente in ogni database.

```
postgres=> CREATE EXTENSION IF NOT EXISTS postgres_fdw;
```

Gestione dell'acquisizione del piano per le repliche di Aurora

Attiva l'acquisizione del piano per le repliche di Aurora

Per creare o rimuovere l'acquisizione del piano nelle repliche Aurora devi disporre dei privilegi di `rds_superuser`. Per ulteriori informazioni sui ruoli e le autorizzazioni degli utenti, consulta Informazioni su ruoli e autorizzazioni di PostgreSQL.

Per acquisire i piani, chiama la funzione apg_plan_mgmt.create_replica_plan_capture nell'istanza database di scrittura, come illustrato di seguito:

```
postgres=> CALL apg_plan_mgmt.create_replica_plan_capture('endpoint', 'password');
```

- endpoint: l'endpoint di scrittura del database globale Aurora o cluster_endpoint fornisce il supporto del failover per l'acquisizione del piano nelle repliche Aurora.

  Per ulteriori informazioni sull'endpoint di scrittura del database globale Aurora, consulta Visualizzazione degli endpoint di un database globale Amazon Aurora.

  Per ulteriori informazioni sugli endpoint del cluster, consulta Endpoint del cluster per Amazon Aurora.
- password: per migliorare la sicurezza, ti consigliamo di seguire queste linee guida durante la creazione della password:
  - Deve contenere almeno 8 caratteri.
  - Deve contenere almeno una lettera maiuscola, una lettera minuscola e un numero.
  - Deve contenere almeno un carattere speciale (?, !, #, <, >, *, eccetera).

> **Note**
>
> Se modifichi l'endpoint, la password o il numero di porta, è necessario eseguire nuovamente l'operazione `apg_plan_mgmt.create_replica_plan_capture()` con l'endpoint e la password per reinizializzare l'acquisizione del piano. In caso contrario, l'acquisizione dei piani dalle repliche Aurora genererà un errore.

Disattiva l'acquisizione del piano per le repliche Aurora

È possibile disattivare il parametro `capture_plan_baselines` nelle repliche Aurora impostandone il valore su `off` nel gruppo Parametri.

Rimuovi l'acquisizione del piano per le repliche Aurora

È possibile rimuovere completamente l'acquisizione dei piani per le repliche Aurora, ma prima di farlo ti consigliamo di rifletterci bene. Per rimuovere l'acquisizione dei piani, chiama `apg_plan_mgmt.remove_replica_plan_capture` come mostrato:

```
postgres=> CALL apg_plan_mgmt.remove_replica_plan_capture();
```

Per attivare l'acquisizione del piano nelle repliche Aurora con l'endpoint e la password, è necessario chiamare nuovamente apg_plan_mgmt.create_replica_plan_capture().

Risoluzione dei problemi

Di seguito, è possibile trovare idee per la risoluzione dei problemi e soluzioni alternative se il piano non viene acquisito come previsto nelle repliche Aurora.

- Impostazioni dei parametri: controlla se il parametro `capture_plan_baselines` è impostato sul valore corretto per attivare l'acquisizione del piano.
- L'estensione **`postgres_fdw`** è installata: utilizza la seguente query per verificare se `postgres_fdw` è installata.

  ```
  postgres=> SELECT * FROM pg_extension WHERE extname = 'postgres_fdw'
  ```

- create_replica_plan_capture() è stata chiamata: utilizza il seguente comando per verificare se la mappatura dell'utente è presente. Altrimenti, chiama `create_replica_plan_capture()` per inizializzare la funzionalità.

```
postgres=> SELECT * FROM pg_foreign_server WHERE srvname =
 'apg_plan_mgmt_writer_foreign_server';
```

- Endpoint e numero di porta: controllare se l'endpoint e il numero di porta sono corretti. Quando questi valori sono errati, non viene visualizzato alcun messaggio di errore.

  Utilizza il seguente comando per verificare se l'endpoint è utilizzato in create() e per controllare in quale database risiede:

```
postgres=> SELECT srvoptions FROM pg_foreign_server WHERE srvname =
 'apg_plan_mgmt_writer_foreign_server';
```

- reload(): è necessario chiamare apg_plan_mgmt.reload() dopo aver chiamato apg_plan_mgmt.delete_plan() nelle repliche Aurora per rendere effettiva la funzione di eliminazione. Ciò garantisce che la modifica sia stata implementata con successo.
- Password: è necessario inserire la password in create_replica_plan_capture() seguendo le linee guida menzionate. In caso contrario, verrà restituito un errore. Per ulteriori informazioni, consulta Gestione dell'acquisizione del piano per le repliche di Aurora. Utilizza un'altra password che soddisfi i requisiti.
- Connessione tra più regioni: l'acquisizione dei piani nelle repliche di Aurora è supportata anche nel database globale Aurora, dove l'istanza di scrittura e le repliche Aurora possono trovarsi in regioni diverse. Assicurarsi di utilizzare l'endpoint di scrittura del database globale Aurora per mantenere la connettività dopo eventi di failover o switchover. Per ulteriori informazioni sugli endpoint del database globale Aurora, consulta Visualizzazione degli endpoint di un database globale Amazon Aurora. L'istanza di scrittura e la replica che si trovano in regioni diverse devono essere in grado di comunicare utilizzando il Peering VPC. Per ulteriori informazioni, consulta Peering VPC. Se si verifica un failover che coinvolge più regioni, è necessario riconfigurare l'endpoint su un nuovo endpoint primario del cluster di database.

> **Note**
>
> Quando si utilizza un endpoint del cluster anziché un endpoint di scrittura del database globale Aurora, è necessario aggiornare l'endpoint del cluster dopo aver eseguito un'operazione di failover globale o di switchover.

Supporto della partizione di tabelle

La funzionalità di gestione dei piani di query (QPM) di Aurora PostgreSQL supporta il partizionamento dichiarativo delle tabelle nelle seguenti versioni:

- 15.3 o versioni successive alla 15
- 14.8 o versioni successive alla 14
- 13.11 o versioni successive alla 13

Per ulteriori informazioni, consulta la pagina relativa al partizionamento delle tabelle.

Argomenti

- Configurazione della partizione delle tabelle
- Acquisizione dei piani per la partizione delle tabelle
- Applicazione di un piano di partizione delle tabelle
- Convenzione di denominazione

Configurazione della partizione delle tabelle

Per configurare la partizione delle tabelle nella funzionalità di gestione dei piani di query di Aurora PostgreSQL, procedi come segue:

1. Imposta `apg_plan_mgmt.plan_hash_version` su 3 o un valore maggiore nel gruppo di parametri del cluster database.
2. Accedi a un database che utilizza la funzionalità di gestione dei piani di query e che contiene voci nella vista `apg_plan_mgmt.dba_plans`.
3. Chiama `apg_plan_mgmt.validate_plans('update_plan_hash')` per aggiornare il valore `plan_hash` nella tabella dei piani.
4. Ripeti i passaggi 2-3 per tutti i database con la funzionalità di gestione dei piani di query abilitata contenenti voci nella vista `apg_plan_mgmt.dba_plans`.

Per ulteriori informazioni su questi parametri, consulta Documentazione di riferimento dei parametri per la gestione del piano di query Aurora PostgreSQL.

Acquisizione dei piani per la partizione delle tabelle

Nella funzionalità di gestione dei piani di query, i vari piani si differenziano per il rispettivo valore `plan_hash`. Per capire come il valore `plan_hash` cambia, devi prima familiarizzare con un tipo simile di piani.

La combinazione di metodi di accesso, nomi di indice con rimozione di cifre e nomi di partizione con rimozione di cifre, accumulati a livello di nodo Append deve essere costante affinché i piani siano considerati uguali. Le partizioni specifiche a cui si accede nei piani non sono significative. Nell'esempio seguente, viene creata una tabella `tbl_a` con 4 partizioni.

```
postgres=>create table tbl_a(i int, j int, k int, l int, m int) partition by range(i);
CREATE TABLE
postgres=>create table tbl_a1 partition of tbl_a for values from (0) to (1000);
CREATE TABLE
postgres=>create table tbl_a2 partition of tbl_a for values from (1001) to (2000);
CREATE TABLE
postgres=>create table tbl_a3 partition of tbl_a for values from (2001) to (3000);
CREATE TABLE
postgres=>create table tbl_a4 partition of tbl_a for values from (3001) to (4000);
CREATE TABLE
postgres=>create index t_i on tbl_a using btree (i);
CREATE INDEX
postgres=>create index t_j on tbl_a using btree (j);
CREATE INDEX
postgres=>create index t_k on tbl_a using btree (k);
CREATE INDEX
```

I piani seguenti sono considerati uguali perché viene utilizzato un unico metodo di scansione di `tbl_a` indipendentemente dal numero di partizioni cercate dalla query.

```
postgres=>explain (hashes true, costs false) select j, k from tbl_a where i between 990
 and 999 and j < 9910 and k > 50;

                        QUERY PLAN
-----------------------------------------------------------------------
Seq Scan on tbl_a1 tbl_a
    Filter: ((i >= 990) AND (i <= 999) AND (j < 9910) AND (k > 50))
SQL Hash: 1553185667, Plan Hash: -694232056
(3 rows)
```

```
postgres=>explain (hashes true, costs false) select j, k from tbl_a where i between 990
 and 1100 and j < 9910 and k > 50;

                        QUERY PLAN
------------------------------------------------------------------
Append
    ->  Seq Scan on tbl_a1 tbl_a_1
            Filter: ((i >= 990) AND (i <= 1100) AND (j < 9910) AND (k > 50))
    ->  Seq Scan on tbl_a2 tbl_a_2
            Filter: ((i >= 990) AND (i <= 1100) AND (j < 9910) AND (k > 50))
    SQL Hash: 1553185667, Plan Hash: -694232056
    (6 rows)
```

```
postgres=>explain (hashes true, costs false) select j, k from tbl_a where i between 990
 and 2100 and j < 9910 and k > 50;

                QUERY PLAN
---------------------------------------------------------------------------
 Append
   ->  Seq Scan on tbl_a1 tbl_a_1
         Filter: ((i >= 990) AND (i <= 2100) AND (j < 9910) AND (k > 50))
   ->  Seq Scan on tbl_a2 tbl_a_2
         Filter: ((i >= 990) AND (i <= 2100) AND (j < 9910) AND (k > 50))
   ->  Seq Scan on tbl_a3 tbl_a_3
         Filter: ((i >= 990) AND (i <= 2100) AND (j < 9910) AND (k > 50))
 SQL Hash: 1553185667, Plan Hash: -694232056
(8 rows)
```

Anche i seguenti 3 piani sono considerati uguali perché, a livello padre, i metodi di accesso, i nomi degli indici con rimozione di cifre e i nomi delle partizioni con rimozione di cifre sono `SeqScan tbl_a`, `IndexScan (i_idx) tbl_a`.

```
postgres=>explain (hashes true, costs false) select j, k from tbl_a where i between 990
 and 1100 and j < 9910 and k > 50;

                                QUERY PLAN
---------------------------------------------------------------------------
 Append
   ->  Seq Scan on tbl_a1 tbl_a_1
         Filter: ((i >= 990) AND (i <= 1100) AND (j < 9910) AND (k > 50))
   ->  Index Scan using tbl_a2_i_idx on tbl_a2 tbl_a_2
         Index Cond: ((i >= 990) AND (i <= 1100))
```

```
         Filter: ((j < 9910) AND (k > 50))
 SQL Hash: 1553185667, Plan Hash: -993736942
(7 rows)
```

```
postgres=>explain (hashes true, costs false) select j, k from tbl_a where i between 990
 and 2100 and j < 9910 and k > 50;

                                QUERY PLAN
--------------------------------------------------------------------------
 Append
   ->  Index Scan using tbl_a1_i_idx on tbl_a1 tbl_a_1
         Index Cond: ((i >= 990) AND (i <= 2100))
         Filter: ((j < 9910) AND (k > 50))
   ->  Seq Scan on tbl_a2 tbl_a_2
         Filter: ((i >= 990) AND (i <= 2100) AND (j < 9910) AND (k > 50))
   ->  Index Scan using tbl_a3_i_idx on tbl_a3 tbl_a_3
         Index Cond: ((i >= 990) AND (i <= 2100))
         Filter: ((j < 9910) AND (k > 50))
 SQL Hash: 1553185667, Plan Hash: -993736942
(10 rows)
```

```
postgres=>explain (hashes true, costs false) select j, k from tbl_a where i between 990
 and 3100 and j < 9910 and k > 50;

                                QUERY PLAN
--------------------------------------------------------------------------
 Append
   ->  Seq Scan on tbl_a1 tbl_a_1
         Filter: ((i >= 990) AND (i <= 3100) AND (j < 9910) AND (k > 50))
   ->  Seq Scan on tbl_a2 tbl_a_2
         Filter: ((i >= 990) AND (i <= 3100) AND (j < 9910) AND (k > 50))
   ->  Seq Scan on tbl_a3 tbl_a_3
         Filter: ((i >= 990) AND (i <= 3100) AND (j < 9910) AND (k > 50))
   ->  Index Scan using tbl_a4_i_idx on tbl_a4 tbl_a_4
         Index Cond: ((i >= 990) AND (i <= 3100))
         Filter: ((j < 9910) AND (k > 50))
 SQL Hash: 1553185667, Plan Hash: -993736942
(11 rows)
```

Indipendentemente dalla diversità a livello di ordine e numero di occorrenze nelle partizioni secondarie, i metodi di accesso, i nomi degli indici con rimozione di cifre e i nomi delle partizioni con rimozione di cifre sono costanti a livello padre per ciascuno dei piani precedenti.

Tuttavia, i piani sarebbero considerati diversi se fosse soddisfatta una delle seguenti condizioni:

- Nel piano viene utilizzato qualsiasi metodo di accesso aggiuntivo.

```
postgres=>explain (hashes true, costs false) select j, k from tbl_a where i between
 990 and 2100 and j < 9910 and k > 50;

                                QUERY PLAN
-------------------------------------------------------------------------
 Append
   ->  Seq Scan on tbl_a1 tbl_a_1
         Filter: ((i >= 990) AND (i <= 2100) AND (j < 9910) AND (k > 50))
   ->  Seq Scan on tbl_a2 tbl_a_2
         Filter: ((i >= 990) AND (i <= 2100) AND (j < 9910) AND (k > 50))
   ->  Bitmap Heap Scan on tbl_a3 tbl_a_3
         Recheck Cond: ((i >= 990) AND (i <= 2100))
         Filter: ((j < 9910) AND (k > 50))
         ->  Bitmap Index Scan on tbl_a3_i_idx
               Index Cond: ((i >= 990) AND (i <= 2100))
 SQL Hash: 1553185667, Plan Hash: 1134525070
(11 rows)
```

- Nessuno dei metodi di accesso del piano non viene più utilizzato.

```
postgres=>explain (hashes true, costs false) select j, k from tbl_a where i between
 990 and 1100 and j < 9910 and k > 50;

                               QUERY PLAN
-------------------------------------------------------------------------
 Append
   ->  Seq Scan on tbl_a1 tbl_a_1
         Filter: ((i >= 990) AND (i <= 1100) AND (j < 9910) AND (k > 50))
   ->  Seq Scan on tbl_a2 tbl_a_2
         Filter: ((i >= 990) AND (i <= 1100) AND (j < 9910) AND (k > 50))
 SQL Hash: 1553185667, Plan Hash: -694232056
(6 rows)
```

- L'indice associato a un metodo di indice viene modificato.

```
postgres=>explain (hashes true, costs false) select j, k from tbl_a where i between
 990 and 1100 and j < 9910 and k > 50;

                               QUERY PLAN
```

```
------------------------------------------------------------------------
 Append
   ->  Seq Scan on tbl_a1 tbl_a_1
         Filter: ((i >= 990) AND (i <= 1100) AND (j < 9910) AND (k > 50))
   ->  Index Scan using tbl_a2_j_idx on tbl_a2 tbl_a_2
         Index Cond: (j < 9910)
         Filter: ((i >= 990) AND (i <= 1100) AND (k > 50))
 SQL Hash: 1553185667, Plan Hash: -993343726
(7 rows)
```

Applicazione di un piano di partizione delle tabelle

I piani approvati per le tabelle partizionate vengono applicati con la corrispondenza a livello di posizione. I piani non sono specifici delle partizioni e possono essere applicati a partizioni diverse dai piani a cui si fa riferimento nella query originale. I piani possono essere applicati anche per le richieste che accedono a un numero di partizioni diverso rispetto alla struttura originale approvata.

Ad esempio, se la struttura approvata si riferisce al seguente piano:

```
postgres=>explain (hashes true, costs false) select j, k from tbl_a where i between 990
 and 2100 and j < 9910 and k > 50;

                              QUERY PLAN
------------------------------------------------------------------------
 Append
   ->  Index Scan using tbl_a1_i_idx on tbl_a1 tbl_a_1
         Index Cond: ((i >= 990) AND (i <= 2100))
         Filter: ((j < 9910) AND (k > 50))
   ->  Seq Scan on tbl_a2 tbl_a_2
         Filter: ((i >= 990) AND (i <= 2100) AND (j < 9910) AND (k > 50))
   ->  Index Scan using tbl_a3_i_idx on tbl_a3 tbl_a_3
         Index Cond: ((i >= 990) AND (i <= 2100))
         Filter: ((j < 9910) AND (k > 50))
 SQL Hash: 1553185667, Plan Hash: -993736942
(10 rows)
```

Questo piano può pertanto essere applicato anche alle query SQL che fanno riferimento a 2, 4 o più partizioni. I possibili piani che potrebbero derivare da questi scenari per l'accesso a 2 e 4 partizioni sono:

```
postgres=>explain (hashes true, costs false) select j, k from tbl_a where i between 990
 and 1100 and j < 9910 and k > 50;

                                QUERY PLAN
---------------------------------------------------------------------------------
 Append
   ->  Index Scan using tbl_a1_i_idx on tbl_a1 tbl_a_1
         Index Cond: ((i >= 990) AND (i <= 1100))
         Filter: ((j < 9910) AND (k > 50))
   ->  Seq Scan on tbl_a2 tbl_a_2
         Filter: ((i >= 990) AND (i <= 1100) AND (j < 9910) AND (k > 50))
 Note: An Approved plan was used instead of the minimum cost plan.
 SQL Hash: 1553185667, Plan Hash: -993736942, Minimum Cost Plan Hash: -1873216041
(8 rows)
```

```
postgres=>explain (hashes true, costs false) select j, k from tbl_a where i between 990
 and 3100 and j < 9910 and k > 50;

                                QUERY PLAN
---------------------------------------------------------------------------
 Append
   ->  Index Scan using tbl_a1_i_idx on tbl_a1 tbl_a_1
         Index Cond: ((i >= 990) AND (i <= 3100))
         Filter: ((j < 9910) AND (k > 50))
   ->  Seq Scan on tbl_a2 tbl_a_2
         Filter: ((i >= 990) AND (i <= 3100) AND (j < 9910) AND (k > 50))
   ->  Index Scan using tbl_a3_i_idx on tbl_a3 tbl_a_3
         Index Cond: ((i >= 990) AND (i <= 3100))
         Filter: ((j < 9910) AND (k > 50))
   ->  Seq Scan on tbl_a4 tbl_a_4
         Filter: ((i >= 990) AND (i <= 3100) AND (j < 9910) AND (k > 50))
 Note: An Approved plan was used instead of the minimum cost plan.
 SQL Hash: 1553185667, Plan Hash: -993736942, Minimum Cost Plan Hash: -1873216041
(12 rows)
```

```
postgres=>explain (hashes true, costs false) select j, k from tbl_a where i between 990
 and 3100 and j < 9910 and k > 50;

                                QUERY PLAN
---------------------------------------------------------------------------------
 Append
   ->  Index Scan using tbl_a1_i_idx on tbl_a1 tbl_a_1
```

```
         Index Cond: ((i >= 990) AND (i <= 3100))
         Filter: ((j < 9910) AND (k > 50))
   ->  Seq Scan on tbl_a2 tbl_a_2
         Filter: ((i >= 990) AND (i <= 3100) AND (j < 9910) AND (k > 50))
   ->  Index Scan using tbl_a3_i_idx on tbl_a3 tbl_a_3
         Index Cond: ((i >= 990) AND (i <= 3100))
         Filter: ((j < 9910) AND (k > 50))
   ->  Index Scan using tbl_a4_i_idx on tbl_a4 tbl_a_4
         Index Cond: ((i >= 990) AND (i <= 3100))
         Filter: ((j < 9910) AND (k > 50))
 Note: An Approved plan was used instead of the minimum cost plan.
 SQL Hash: 1553185667, Plan Hash: -993736942, Minimum Cost Plan Hash: -1873216041
(14 rows)
```

Prendi in considerazione un altro piano approvato con metodi di accesso diversi per ogni partizione:

```
postgres=>explain (hashes true, costs false) select j, k from tbl_a where i between 990
 and 2100 and j < 9910 and k > 50;

                                QUERY PLAN
-------------------------------------------------------------------------
 Append
   ->  Index Scan using tbl_a1_i_idx on tbl_a1 tbl_a_1
         Index Cond: ((i >= 990) AND (i <= 2100))
         Filter: ((j < 9910) AND (k > 50))
   ->  Seq Scan on tbl_a2 tbl_a_2
         Filter: ((i >= 990) AND (i <= 2100) AND (j < 9910) AND (k > 50))
   ->  Bitmap Heap Scan on tbl_a3 tbl_a_3
         Recheck Cond: ((i >= 990) AND (i <= 2100))
         Filter: ((j < 9910) AND (k > 50))
         ->  Bitmap Index Scan on tbl_a3_i_idx
               Index Cond: ((i >= 990) AND (i <= 2100))
 SQL Hash: 1553185667, Plan Hash: 2032136998
(12 rows)
```

In questo caso, qualsiasi piano che esegue letture da due partizioni non viene applicato. A meno che tutte le combinazioni (metodo di accesso, nome di indice) del piano approvato non siano utilizzabili, il piano non può essere applicato. Ad esempio, i seguenti piani hanno hash diversi e il piano approvato non può essere applicato in questi casi:

```
postgres=>explain (hashes true, costs false) select j, k from tbl_a where i between 990
 and 1900 and j < 9910 and k > 50;
```

```
                              QUERY PLAN
-----------------------------------------------------------------------
 Append
   ->  Bitmap Heap Scan on tbl_a1 tbl_a_1
         Recheck Cond: ((i >= 990) AND (i <= 1900))
         Filter: ((j < 9910) AND (k > 50))
         ->  Bitmap Index Scan on tbl_a1_i_idx
               Index Cond: ((i >= 990) AND (i <= 1900))
   ->  Bitmap Heap Scan on tbl_a2 tbl_a_2
         Recheck Cond: ((i >= 990) AND (i <= 1900))
         Filter: ((j < 9910) AND (k > 50))
         ->  Bitmap Index Scan on tbl_a2_i_idx
               Index Cond: ((i >= 990) AND (i <= 1900))
  Note: This is not an Approved plan.  No usable Approved plan was found.
  SQL Hash: 1553185667, Plan Hash: -568647260
(13 rows)
```

```
postgres=>explain (hashes true, costs false) select j, k from tbl_a where i between 990
 and 1900 and j < 9910 and k > 50;

                              QUERY PLAN
-----------------------------------------------------------------------
 Append
   ->  Index Scan using tbl_a1_i_idx on tbl_a1 tbl_a_1
         Index Cond: ((i >= 990) AND (i <= 1900))
         Filter: ((j < 9910) AND (k > 50))
   ->  Seq Scan on tbl_a2 tbl_a_2
         Filter: ((i >= 990) AND (i <= 1900) AND (j < 9910) AND (k > 50))
 Note: This is not an Approved plan.  No usable Approved plan was found.
 SQL Hash: 1553185667, Plan Hash: -496793743
(8 rows)
```

Convenzione di denominazione

Affinché QPM applichi un piano con tabelle partizionate dichiarative, è necessario seguire regole di denominazione specifiche per le tabelle principale, le partizioni di tabella e gli indici:

- Nomi delle tabelle principale: questi nomi devono differire per lettere o caratteri speciali e non solo per cifre. Ad esempio, tA, tB e tC sono nomi accettabili per tabelle padre distinte, mentre t1, t2 e t3 non lo sono.

- Nomi delle singole tabelle delle partizioni: le partizioni della stessa tabella principale devono differire l'una dall'altra solo per le cifre. Ad esempio, i nomi di partizione accettabili per tA potrebbero essere tA1, tA2 o T1a, T2a o anche un numero maggiore di cifre.

  Eventuali altre differenze (lettere, caratteri speciali) non garantiranno l'applicazione del piano.
- Nomi degli indici: nella gerarchia delle tabelle delle partizioni, assicurarsi che tutti gli indici abbiano nomi univoci. Ciò significa che le parti non numeriche dei nomi devono essere diverse. Ad esempio, se si dispone di una tabella partizionata denominata `tA` con un indice denominato `tA_col1_idx1`, non è possibile avere un altro indice denominato `tA_col1_idx2`. Tuttavia, è possibile avere un indice denominato `tA_a_col1_idx2` perché la parte non numerica del nome è univoca. Questa regola si applica agli indici creati sia nella tabella principale che nelle singole tabelle delle partizioni.

La mancata conformità alle convenzioni di denominazione sopra citate può comportare la mancata applicazione dei piani approvati. L'esempio seguente illustra tale mancata applicazione:

```
postgres=>create table t1(i int, j int, k int, l int, m int) partition by range(i);
CREATE TABLE
postgres=>create table t1a partition of t1 for values from (0) to (1000);
CREATE TABLE
postgres=>create table t1b partition of t1 for values from (1001) to (2000);
CREATE TABLE
postgres=>SET apg_plan_mgmt.capture_plan_baselines TO 'manual';
SET
postgres=>explain (hashes true, costs false) select count(*) from t1 where i > 0;

                           QUERY PLAN
-----------------------------------------------------------------------------
 Aggregate
   ->  Append
         ->  Seq Scan on t1a t1_1
               Filter: (i > 0)
         ->  Seq Scan on t1b t1_2
               Filter: (i > 0)
 SQL Hash: -1720232281, Plan Hash: -1010664377
(7 rows)
```

```
postgres=>SET apg_plan_mgmt.use_plan_baselines TO 'on';
SET
postgres=>explain (hashes true, costs false) select count(*) from t1 where i > 1000;
```

```
                            QUERY PLAN
------------------------------------------------------------------------
 Aggregate
   ->  Seq Scan on t1b t1
         Filter: (i > 1000)
 Note: This is not an Approved plan. No usable Approved plan was found.
 SQL Hash: -1720232281, Plan Hash: 335531806
(5 rows)
```

Anche se i due piani possono apparire identici, i relativi valori `Plan Hash` sono diversi a causa dei nomi delle tabelle secondarie. I nomi delle tabelle variano in base ai caratteri alfabetici anziché alle sole cifre, il che comporta un errore di applicazione.

Utilizzo di estensioni e wrapper di dati esterni

Per estendere la funzionalità al cluster database Edizione compatibile con Aurora PostgreSQL, è possibile installare e utilizzare diverse estensioni PostgreSQL. Ad esempio, se il caso d'uso richiede l'immissione intensiva di dati su tabelle molto grandi, è possibile installare l'estensione `pg_partman` per partizionare i dati e quindi distribuire il carico di lavoro.

> **Note**
>
> La funzionalità Trusted Language Extensions per PostgreSQL è supportata da Aurora PostgreSQL a partire dalla versione 14.5. Viene implementata come estensione `pg_tle` che puoi aggiungere ad Aurora PostgreSQL e che può essere utilizzata dagli sviluppatori per creare le proprie estensioni di PostgreSQL in un ambiente sicuro che riduce i requisiti di impostazione e configurazione, nonché gran parte dei test preliminari delle nuove estensioni. Per ulteriori informazioni, consulta Utilizzo di Trusted Language Extensions per PostgreSQL.

In alcuni casi, anziché installare un'estensione, è possibile aggiungere un modulo specifico all'elenco di `shared_preload_libraries` nel gruppo di parametri cluster database personalizzato del cluster database Aurora PostgreSQL. In genere, il gruppo di parametri cluster di database predefinito carica solo `pg_stat_statements`, ma sono disponibili diversi altri moduli da aggiungere all'elenco. Ad esempio, è possibile aggiungere funzionalità di pianificazione aggiungendo il modulo `pg_cron`, come descritto in Pianificazione della manutenzione con l'estensione PostgreSQL pg_cron. Come altro esempio, è possibile registrare i piani di esecuzione delle query caricando il modulo `auto_explain`. Per ulteriori informazioni, consultare Logging execution plans of queries (Registrazione dei piani di esecuzione delle query) nel Knowledge Center di AWS.

Un'estensione che fornisce accesso a dati esterni è più specificatamente conosciuta come un wrapper di dati esterno (FDW). Ad esempio, l'estensione `oracle_fdw` consente al cluster database Aurora PostgreSQL di utilizzare i database Oracle.

È inoltre possibile specificare con precisione le estensioni che possono essere installate sull'istanza database Aurora PostgreSQL, elencandole nel parametro `rds.allowed_extensions`. Per ulteriori informazioni, consultaLimitazione dell'installazione delle estensioni PostgreSQL.

Di seguito sono disponibili informazioni sulla configurazione e l'utilizzo di alcune delle estensioni, dei moduli e degli FDW disponibili per Aurora PostgreSQL. Per semplicità, queste sono tutte denominate "estensioni". Per gli elenchi delle estensioni che è possibile utilizzare con le versioni di Aurora

PostgreSQL attualmente disponibili, consultare la pagina relativa alle versioni delle estensioni per Amazon Aurora PostgreSQL nelle Note di rilascio di Aurora PostgreSQL.

- Gestione di oggetti di grandi dimensioni con il modulo lo
- Gestione dei dati spaziali con estensione PostGIS
- Gestione delle partizioni PostgreSQL con l'estensione pg_partman
- Pianificazione della manutenzione con l'estensione PostgreSQL pg_cron
- Utilizzo di pgAudit per registrare l'attività del database
- Utilizzo di pglogical per sincronizzare i dati tra le istanze
- Interazione con un database Oracle utilizzando l'estensione oracle_fdw
- Interazione con i database MySQL utilizzando l'estensione mysql_fdw

Utilizzo del supporto delle estensioni delegate di Amazon Aurora per PostgreSQL

Utilizzando il supporto delle estensioni delegate di Amazon Aurora per PostgreSQL, è possibile delegare la gestione delle estensioni a un utente che non deve essere un `rds_superuser`. Con questo supporto delle estensioni delegate, viene creato un nuovo ruolo denominato `rds_extension` che è necessario assegnare a un utente per gestire altre estensioni. Questo ruolo può creare, aggiornare ed eliminare estensioni.

È possibile specificare le estensioni che possono essere installate sull'istanza database Aurora PostgreSQL, elencandole nel parametro `rds.allowed_extensions`. Per ulteriori informazioni, consulta Utilizzo delle estensioni PostgreSQL con Amazon RDS per PostgreSQL.

È possibile limitare l'elenco delle estensioni disponibili che possono essere gestite dall'utente con il ruolo `rds_extension` utilizzando il parametro `rds.allowed_delegated_extensions`.

Il supporto delle estensioni delegate è disponibile nelle seguenti versioni:

- Tutte le versioni successive
- 15.5 e versioni successive alla 15
- 14.10 e versioni successive alla 14
- 13.13 e versioni successive alla 13
- 12.17 e versioni successive 12

Argomenti

- Attivazione del supporto delle estensioni delegate per un utente
- Configurazione utilizzata nel supporto delle estensioni delegate Aurora per PostgreSQL
- Disattivazione del supporto per l'estensione delegata
- Vantaggi dell'utilizzo del supporto delle estensioni delegate di Amazon Aurora
- Limitazione del supporto delle estensioni delegate di Aurora per PostgreSQL
- Autorizzazioni necessarie per determinate estensioni
- Considerazioni sulla sicurezza
- Disabilitazione dell'eliminazione delle estensioni con l'opzione cascade
- Estensioni di esempio che possono essere aggiunte utilizzando il supporto delle estensioni delegate

Attivazione del supporto delle estensioni delegate per un utente

È necessario eseguire le seguenti operazioni per abilitare il supporto delle estensioni delegate per un utente:

1. Concedere il ruolo **`rds_extension`** a un utente: connettersi al database come `rds_superuser` ed esegui il seguente comando:

   ```
   Postgres => grant rds_extension to user_name;
   ```

2. Impostare l'elenco delle estensioni disponibili per la gestione degli utenti delegati: `rds.allowed_delegated_extensions` consente di specificare un sottoinsieme delle estensioni disponibili utilizzando `rds.allowed_extensions` nel parametro del cluster di database. È possibile eseguire questa operazione a uno dei seguenti livelli:

   - Nel cluster o nel gruppo di parametri dell'istanza, tramite l'API Console di gestione AWS o. Per ulteriori informazioni, consulta Gruppi di parametri per Amazon Aurora.
   - Utilizzare il seguente comando a livello di database:

     ```
     alter database database_name set rds.allowed_delegated_extensions =
      'extension_name_1,
                        extension_name_2,...extension_name_n';
     ```

   - Utilizzare il seguente comando a livello di utente:

```
alter user user_name set rds.allowed_delegated_extensions = 'extension_name_1,
                    extension_name_2,...extension_name_n';
```

> **Note**
>
> Non è necessario riavviare il database dopo aver modificato il parametro dinamico `rds.allowed_delegated_extensions`.

3. Consentire l'accesso all'utente delegato agli oggetti creati durante il processo di creazione dell'estensione: alcune estensioni creano oggetti che richiedono la concessione di autorizzazioni aggiuntive prima che l'utente con il ruolo `rds_extension` possa accedervi. `rds_superuser` deve concedere all'utente delegato l'accesso a tali oggetti. Una delle opzioni consiste nell'utilizzare un trigger di evento per concedere automaticamente l'autorizzazione all'utente delegato.

   Esempio di attivazione di un evento

   Se si desidera consentire a un utente delegato con `rds_extension` di utilizzare estensioni che richiedono l'impostazione delle autorizzazioni sugli oggetti creati durante la creazione dell'estensione, è possibile personalizzare l'esempio seguente di trigger di evento e aggiungere solo le estensioni per le quali si desidera che gli utenti delegati abbiano accesso alla funzionalità completa. Questo trigger di evento può essere creato su template1 (il modello predefinito), pertanto tutti i database creati da template1 avranno tale trigger di evento. Quando un utente delegato installa l'estensione, questo trigger garantirà automaticamente la proprietà degli oggetti creati dall'estensione.

```
CREATE OR REPLACE FUNCTION create_ext()

  RETURNS event_trigger AS $$

DECLARE

  schemaname TEXT;
  databaseowner TEXT;

  r RECORD;
```

```
BEGIN

  IF tg_tag = 'CREATE EXTENSION' and current_user != 'rds_superuser' THEN
    RAISE NOTICE 'SECURITY INVOKER';
    RAISE NOTICE 'user: %', current_user;
    FOR r IN SELECT * FROM pg_catalog.pg_event_trigger_ddl_commands()
    LOOP
        CONTINUE WHEN r.command_tag != 'CREATE EXTENSION' OR r.object_type !=
 'extension';

        schemaname = (
            SELECT n.nspname
            FROM pg_catalog.pg_extension AS e
            INNER JOIN pg_catalog.pg_namespace AS n
            ON e.extnamespace = n.oid
            WHERE e.oid = r.objid
        );

        databaseowner = (
            SELECT pg_catalog.pg_get_userbyid(d.datdba)
            FROM pg_catalog.pg_database d
            WHERE d.datname = current_database()
        );
        RAISE NOTICE 'Record for event trigger %, objid: %,tag: %, current_user: %,
 schema: %, database_owenr: %', r.object_identity, r.objid, tg_tag, current_user,
 schemaname, databaseowner;
        IF r.object_identity = 'address_standardizer_data_us' THEN
            EXECUTE pg_catalog.format('GRANT SELECT, UPDATE, INSERT, DELETE ON
 TABLE %I.us_gaz TO %I WITH GRANT OPTION;', schemaname, databaseowner);
            EXECUTE pg_catalog.format('GRANT SELECT, UPDATE, INSERT, DELETE ON
 TABLE %I.us_lex TO %I WITH GRANT OPTION;', schemaname, databaseowner);
            EXECUTE pg_catalog.format('GRANT SELECT, UPDATE, INSERT, DELETE ON
 TABLE %I.us_rules TO %I WITH GRANT OPTION;', schemaname, databaseowner);
        ELSIF r.object_identity = 'dict_int' THEN
            EXECUTE pg_catalog.format('ALTER TEXT SEARCH DICTIONARY %I.intdict
 OWNER TO %I;', schemaname, databaseowner);
        ELSIF r.object_identity = 'pg_partman' THEN
            EXECUTE pg_catalog.format('GRANT SELECT, UPDATE, INSERT, DELETE ON
 TABLE %I.part_config TO %I WITH GRANT OPTION;', schemaname, databaseowner);
            EXECUTE pg_catalog.format('GRANT SELECT, UPDATE, INSERT, DELETE ON
 TABLE %I.part_config_sub TO %I WITH GRANT OPTION;', schemaname, databaseowner);
            EXECUTE pg_catalog.format('GRANT SELECT, UPDATE, INSERT, DELETE
 ON TABLE %I.custom_time_partitions TO %I WITH GRANT OPTION;', schemaname,
 databaseowner);
```

```
        ELSIF r.object_identity = 'postgis_topology' THEN
            EXECUTE pg_catalog.format('GRANT SELECT, UPDATE, INSERT, DELETE ON ALL
 TABLES IN SCHEMA topology TO %I WITH GRANT OPTION;', databaseowner);
            EXECUTE pg_catalog.format('GRANT USAGE, SELECT ON ALL SEQUENCES IN
 SCHEMA topology TO %I WITH GRANT OPTION;', databaseowner);
            EXECUTE pg_catalog.format('GRANT EXECUTE ON ALL FUNCTIONS IN SCHEMA
 topology TO %I WITH GRANT OPTION;', databaseowner);
            EXECUTE pg_catalog.format('GRANT USAGE ON SCHEMA topology TO %I WITH
 GRANT OPTION;', databaseowner);
        END IF;
    END LOOP;
  END IF;
END;
$$ LANGUAGE plpgsql SECURITY DEFINER;

CREATE EVENT TRIGGER log_create_ext ON ddl_command_end EXECUTE PROCEDURE
 create_ext();
```

Configurazione utilizzata nel supporto delle estensioni delegate Aurora per PostgreSQL

Nome configurazione	Description	Valore predefinito	Note	Chi può modificare o concedere l'autorizzazione
`rds.allowed_delegated_extensions`	Questo parametro limita le estensioni che un ruolo rds_extension può gestire in un database. Deve essere un sottoinsieme di rds.allowed_extensions.	Stringa vuota	• Per impostazione predefinita, questo parametro è una stringa vuota, il che significa che nessuna estensione è stata delegata agli utenti con `rds_extension` .	rds_superuser

Nome configurazione	Description	Valore predefinito	Note	Chi può modificare o concedere l'autorizzazione
			• È possibile aggiungere qualsiasi estensione supportata se l'utente dispone dell'autorizzazione per farlo. A tale scopo, impostare il parametro `rds.allowed_delegated_extensions` su una stringa di nomi di estensione separati da virgole. L'aggiunta di un elenco di estensioni a questo parametro consente di identificare esplicitamente le estensioni che l'utente con il ruolo `rds_extension` può installare. • Se impostato su `*`, significa che tutte le estensioni elencate in	

Nome configurazione	Description	Valore predefinito	Note	Chi può modificare o concedere l'autorizzazione
			`rds_allowed_extensions` sono delegate a utenti con il ruolo `rds_extension`. Per ulteriori informazioni sulla configurazione di questo parametro, consulta Attivazione del supporto delle estensioni delegate per un utente.	

Nome configurazione	Description	Valore predefinito	Note	Chi può modificare o concedere l'autorizzazione
`rds.allowed_extensions`	Questo parametro consente all'utente di limitare le estensioni che possono essere installate nell'istanza database Aurora PostgreSQL. Per ulteriori informazioni, consulta Limitazione dell'installazione delle estensioni PostgreSQL.	"*"	Per impostazione predefinita, questo parametro è impostato su "*", il che significa che tutte le estensioni supportate su RDS per PostgreSQL e Aurora PostgreSQL possono essere create da utenti con i privilegi necessari. Vuoto significa che non è possibile installare estensioni nell'istanza database Aurora PostgreSQL.	administrator

Nome configurazione	Description	Valore predefinito	Note	Chi può modificare o concedere l'autorizzazione
`rds-delegated_extension_allow_drascad`	Questo parametro controlla la possibilità per l'utente con `rds_extension` di eliminare l'estensione utilizzando un'opzione cascade.	off	Per impostazione predefinita, `rds-delegated_extension_allow_drop_cascade` è impostato su `off`. Ciò significa che gli utenti con `rds_extension` non sono autorizzati a eliminare un'estensione utilizzando l'opzione cascade. Per consentire tale funzionalità, il parametro `rds.delegated_extension_allow_drop_cascade` deve essere impostato su on.	rds_superuser

Disattivazione del supporto per l'estensione delegata

Disattivazione parziale

Gli utenti delegati non possono creare nuove estensioni ma possono comunque aggiornare le estensioni esistenti.

- Ripristinare `rds.allowed_delegated_extensions` sul valore predefinito nel gruppo di parametri del cluster di database.
- Utilizzare il seguente comando a livello di database:

```
alter database database_name reset rds.allowed_delegated_extensions;
```

- Utilizzare il seguente comando a livello di utente:

```
alter user user_name reset rds.allowed_delegated_extensions;
```

Disattivazione completa

La revoca del ruolo `rds_extension` a un utente ripristinerà le autorizzazioni standard per l'utente. L'utente non può più creare, aggiornare o eliminare le estensioni.

```
postgres => revoke rds_extension from user_name;
```

Vantaggi dell'utilizzo del supporto delle estensioni delegate di Amazon Aurora

Utilizzando il supporto delle estensioni delegate di Amazon Aurora per PostgreSQL, è possibile delegare in modo sicuro la gestione delle estensioni agli utenti che non dispongono del ruolo `rds_superuser`. Questa funzionalità fornisce i seguenti vantaggi:

- È possibile delegare facilmente la gestione delle estensioni agli utenti desiderati.
- Non è richiesto il ruolo `rds_superuser`.
- Offre la possibilità di supportare diversi set di estensioni per diversi database nello stesso cluster di database.

Limitazione del supporto delle estensioni delegate di Aurora per PostgreSQL

- Gli oggetti creati durante il processo di creazione dell'estensione possono richiedere privilegi aggiuntivi per il corretto funzionamento dell'estensione.

Autorizzazioni necessarie per determinate estensioni

Per creare, utilizzare o aggiornare le seguenti estensioni, l'utente delegato deve disporre dei privilegi necessari sulle seguenti funzioni, tabelle e schemi.

Ester i che richie o propr o autor zioni	Funzione	Tabelle	Schema	Dizionario di ricerca testuale	Comment
addre tanda er_da s		us_gaz, us_lex, us_lex, l.us_rules			
amch	bt_index_check, bt_index_ parent_check				
dict_i				intdict	
pg_p n		custom_ti me_partitions, part_config, part_config_sub			
pg_s tatem					
Post(	st_tileenvelope	spatial_ref_sys			
postg aster					
postg opolo		topology, layer	topology		L'utente delegato deve essere il

Ester i che richie o propr o autor zioni	Funzione	Tabelle	Schema	Dizionario di ricerca testuale	Comment
					proprietario del database
log_f	create_fo reign_tab le_for_log_file				
rds_t	role_pass word_encr yption_type				
postg iger_ oder		geocode_s ettings_default, geocode_s ettings	tiger		
pg_fr acem	pg_freespace				
pg_vi lity	pg_visibility				

Considerazioni sulla sicurezza

Tenere presente che un utente con il ruolo `rds_extension` sarà in grado di gestire le estensioni su tutti i database su cui ha il privilegio di connessione. Se l'intenzione è fare in modo che un utente delegato gestisca l'estensione su un singolo database, è buona norma revocare tutti i privilegi al

pubblico su ciascun database, quindi concedere esplicitamente il privilegio di connessione per tale database specifico all'utente delegato.

Esistono diverse estensioni che possono consentire a un utente di accedere alle informazioni da più database. Prima di aggiungere queste estensioni a `rds.allowed_delegated_extensions`, assicurarsi che gli utenti a cui si concede `rds_extension` dispongano di funzionalità su più database. Ad esempio, `postgres_fdw` e `dblink` forniscono funzionalità per eseguire query su più database sulla stessa istanza o su istanze remote. `log_fdw` legge i file di log del motore postgres, che riguardano tutti i database dell'istanza, potenzialmente contenenti query lente o messaggi di errore provenienti da più database. `pg_cron` consente l'esecuzione di processi pianificati in background sull'istanza database e può configurare i processi per l'esecuzione in un database diverso.

Disabilitazione dell'eliminazione delle estensioni con l'opzione cascade

La possibilità di eliminare l'estensione con l'opzione cascade da parte di un utente con il ruolo `rds_extension` è controllata dal parametro `rds.delegated_extension_allow_drop_cascade`. Per impostazione predefinita, `rds-delegated_extension_allow_drop_cascade` è impostato su `off`. Ciò significa che agli utenti con il ruolo `rds_extension` non è consentito eliminare un'estensione utilizzando l'opzione cascade, come mostrato nella query seguente.

```
DROP EXTENSION CASCADE;
```

In questo modo verranno eliminati automaticamente gli oggetti che dipendono dall'estensione e, di conseguenza, tutti gli oggetti che dipendono da tali oggetti. Il tentativo di utilizzare l'opzione cascade genererà un errore.

Per consentire tale funzionalità, il parametro `rds.delegated_extension_allow_drop_cascade` deve essere impostato su `on`.

La modifica del parametro dinamico `rds.delegated_extension_allow_drop_cascade` non richiede il riavvio del database. È possibile eseguire questa operazione a uno dei seguenti livelli:

- Nel cluster o nel gruppo di parametri dell'istanza, tramite l'API Console di gestione AWS o.
- Utilizzando il seguente comando a livello di database:

```
alter database database_name set rds.delegated_extension_allow_drop_cascade = 'on';
```

- Utilizzando il seguente comando a livello di utente:

```
alter role tenant_user set rds.delegated_extension_allow_drop_cascade = 'on';
```

Estensioni di esempio che possono essere aggiunte utilizzando il supporto delle estensioni delegate

- `rds_tools`

```
extension_test_db=> create extension rds_tools;
CREATE EXTENSION
extension_test_db=> SELECT * from rds_tools.role_password_encryption_type() where
 rolname = 'pg_read_server_files';
ERROR: permission denied for function role_password_encryption_type
```

- `amcheck`

```
extension_test_db=> CREATE TABLE amcheck_test (id int);
CREATE TABLE
extension_test_db=> INSERT INTO amcheck_test VALUES (generate_series(1,100000));
INSERT 0 100000
extension_test_db=> CREATE INDEX amcheck_test_btree_idx ON amcheck_test USING btree
 (id);
CREATE INDEX
extension_test_db=> create extension amcheck;
CREATE EXTENSION
extension_test_db=> SELECT bt_index_check('amcheck_test_btree_idx'::regclass);
ERROR: permission denied for function bt_index_check
extension_test_db=> SELECT bt_index_parent_check('amcheck_test_btree_idx'::regclass);
ERROR: permission denied for function bt_index_parent_check
```

- `pg_freespacemap`

```
extension_test_db=> create extension pg_freespacemap;
CREATE EXTENSION
extension_test_db=> SELECT * FROM pg_freespace('pg_authid');
ERROR: permission denied for function pg_freespace
extension_test_db=> SELECT * FROM pg_freespace('pg_authid',0);
ERROR: permission denied for function pg_freespace
```

- `pg_visibility`

```
extension_test_db=> create extension pg_visibility;
CREATE EXTENSION
extension_test_db=> select * from pg_visibility('pg_database'::regclass);
ERROR: permission denied for function pg_visibility
```

- `postgres_fdw`

```
extension_test_db=> create extension postgres_fdw;
CREATE EXTENSION
extension_test_db=> create server myserver foreign data wrapper postgres_fdw options
 (host 'foo', dbname 'foodb', port '5432');
ERROR: permission denied for foreign-data wrapper postgres_fdw
```

Gestione di oggetti di grandi dimensioni con il modulo lo

Il modulo lo (estensione) è per utenti di database e sviluppatori che utilizzano database PostgreSQL tramite driver JDBC o ODBC. JDBC e ODBC presumono entrambi che il database gestisca l'eliminazione di oggetti di grandi dimensioni quando i riferimenti ad essi cambiano. Tuttavia, PostgreSQL non funziona in questo modo. PostgreSQL non ipotizza che un oggetto venga eliminato quando il suo riferimento cambia. Il risultato è che gli oggetti rimangono su disco, senza riferimenti. L'estensione lo include una funzione utilizzata per attivare le modifiche dei riferimenti per eliminare gli oggetti se necessario.

> **Tip**
>
> Per determinare se il database può sfruttare l'estensione lo, utilizzare l'utilità `vacuumlo` per verificare la presenza di oggetti orfani di grandi dimensioni. Per ottenere il conteggio di oggetti orfani di grandi dimensioni senza eseguire alcuna azione, esegui l'utilità con l'opzione `-n`(no-op). Per scoprire come, consulta vacuumlo utility seguente.

Il modulo lo è disponibile per Aurora PostgreSQL 13.7, 12.11, 11.16, 10.21 e versioni secondarie successive.

Per installare il modulo (estensione), sono necessari privilegi `rds_superuser`. L'installazione dell'estensione lo aggiunge quanto segue al database:

- `lo` – Questo è un tipo di dati (lo) oggetto di grandi dimensioni che puoi utilizzare per oggetti binari di grandi dimensioni (BLOB) e altri oggetti di grandi dimensioni. Il tipo di dati `lo` è un dominio del tipo di dati `oid`. In altre parole, è un identificatore di oggetti con vincoli opzionali. Per ulteriori informazioni, consulta Identificatori di oggetti nella documentazione di PostgreSQL. In poche parole, puoi utilizzare il tipo di dati `lo` per distinguere le colonne del database che contengono riferimenti di oggetti di grandi dimensioni da altri identificatori di oggetti (OID).
- `lo_manage` – Questa è una funzione che puoi utilizzare nei trigger sulle colonne di tabella contenenti riferimenti di oggetti di grandi dimensioni. Ogni volta che elimini o modifichi un valore che fa riferimento a un oggetto di grandi dimensioni, il trigger scollega l'oggetto (`lo_unlink`) dal suo riferimento. Utilizza il trigger su una colonna solo se la colonna è l'unico riferimento del database all'oggetto di grandi dimensioni.

Per ulteriori informazioni sul modulo di oggetti di grandi dimensioni, consulta lo nella documentazione di PostgreSQL.

Installazione dell'estensione lo

Prima di installare l'estensione lo, assicurati di disporre dei privilegi `rds_superuser`.

Installare l'estensione

1. Utilizza `psql` per connetterti all'istanza database principale del cluster database Aurora PostgreSQL.

   ```
   psql --host=your-cluster-instance-1.666666666666.aws-region.rds.amazonaws.com --
   port=5432 --username=postgres --password
   ```

   Specifica la password, quando richiesto. Il client `psql` si connette e visualizza il database di connessione amministrativa predefinito `postgres=>` come prompt.
2. Installa l'estensione come segue:

   ```
   postgres=> CREATE EXTENSION lo;
   CREATE EXTENSION
   ```

Ora puoi utilizzare il tipo di dati `lo` per definire le colonne nelle tabelle. Ad esempio, puoi creare una tabella (`images`) contenente dati immagine raster. Puoi utilizzare il tipo di dati `lo` per una colonna `raster`, come mostrato nell'esempio seguente, che crea una tabella.

```
postgres=> CREATE TABLE images (image_name text, raster lo);
```

Utilizzo della funzione di trigger lo_manage per eliminare gli oggetti

Puoi utilizzare la funzione `lo_manage` in un trigger su un `lo` o altre colonne di oggetti di grandi dimensioni per rimuovere (e prevenire oggetti orfani) quando `lo` viene aggiornato o eliminato.

Impostare trigger su colonne che fanno riferimento a oggetti di grandi dimensioni

- Esegui una di queste operazioni:
  - Crea un trigger BEFORE UPDATE OR DELETE su ogni colonna per contenere riferimenti univoci a oggetti di grandi dimensioni, utilizzando il nome della colonna come argomento.

    ```
    postgres=> CREATE TRIGGER t_raster BEFORE UPDATE OR DELETE ON images
        FOR EACH ROW EXECUTE FUNCTION lo_manage(raster);
    ```
  - Applica un trigger solo quando la colonna viene aggiornata.

    ```
    postgres=> CREATE TRIGGER t_raster BEFORE UPDATE OF images
        FOR EACH ROW EXECUTE FUNCTION lo_manage(raster);
    ```

La funzione di trigger `lo_manage` funziona solo nel contesto dell'inserimento o dell'eliminazione dei dati di colonna, a seconda di come definisci il trigger. Non ha alcun effetto quando esegui un'operazione `DROP` o `TRUNCATE` su un database. Ciò significa che devi eliminare le colonne di oggetti da qualsiasi tabella prima del rilascio, per evitare la creazione di oggetti orfani.

Ad esempio, supponi di voler rilasciare il database contenente la tabella `images`. Per eliminare la colonna procedi come segue.

```
postgres=> DELETE FROM images COLUMN raster
```

Ipotizzando che la funzione `lo_manage` sia definita su tale colonna per gestire le eliminazioni, ora puoi rilasciare la tabella in modo sicuro.

Rimozione di oggetti orfani di grandi dimensioni orfani con **`vacuumlo`**

L'utilità `vacuumlo` identifica e rimuove oggetti orfani di grandi dimensioni dai database. Questa utilità è disponibile a partire da PostgreSQL 9.1.24. Se gli utenti del database utilizzano regolarmente

oggetti di grandi dimensioni, si consiglia di eseguire `vacuumlo` occasionalmente per rimuovere gli oggetti orfani di grandi dimensioni.

Prima di installare l'estensione lo, è possibile utilizzare `vacuumlo` per valutare se il cluster di database Aurora PostgreSQL può trarne vantaggio. A questo scopo, esegui `vacuumlo` con l'opzione `-n` (no-op) per mostrare cosa viene rimosso, come illustrato di seguito:

```
$ vacuumlo -v -n -h your-cluster-instance-1.666666666666.aws-region.rds.amazonaws.com -
p 5433 -U postgres docs-lab-spatial-db
Password:*****
Connected to database "docs-lab-spatial-db"
Test run: no large objects will be removed!
Would remove 0 large objects from database "docs-lab-spatial-db".
```

Come mostra l'output, gli oggetti orfani di grandi dimensioni non sono un problema per questo particolare database.

Per ulteriori informazioni su questa utilità, consulta `vacuumlo` nella documentazione di PostgreSQL.

Introduzione al funzionamento di **`vacuumlo`**

Il comando `vacuumlo` rimuove gli oggetti orfani di grandi dimensioni (LO) dal database PostgreSQL senza influire o entrare in conflitto con le tabelle utente.

Il comando funziona come segue:

1. `vacuumlo` inizia creando una tabella temporanea contenente tutti gli ID oggetto (OID) degli oggetti di grandi dimensioni presenti nel database.
2. `vacuumlo` analizza quindi ogni colonna del database che utilizza i tipi di dati `oid` o `lo`. Se `vacuumlo` trova un OID corrispondente in queste colonne, rimuove l'OID dalla tabella temporanea. `vacuumlo` controlla solo le colonne denominate specificatamente `oid` o `lo`, non i domini basati su questi tipi.
3. Le voci rimanenti nella tabella temporanea rappresentano gli oggetti orfani di grandi dimensioni, che vengono quindi rimossi in modo sicuro da `vacuumlo`.

Miglioramento delle prestazioni di **`vacuumlo`**

È possibile migliorare le prestazioni di `vacuumlo` aumentando la dimensione del batch con l'opzione `-l`. Ciò consente a `vacuumlo` di elaborare più oggetti di grandi dimensioni contemporaneamente.

Se il sistema dispone di memoria sufficiente ed è possibile inserire completamente la tabella temporanea in memoria, l'aumento dell'impostazione `temp_buffers` a livello di database può migliorare le prestazioni. Ciò consente alla tabella di risiedere interamente in memoria, il che può migliorare le prestazioni complessive.

La seguente query stima la dimensione della tabella temporanea:

```
SELECT
    pg_size_pretty(SUM(pg_column_size(oid))) estimated_lo_temp_table_size
FROM
    pg_largeobject_metadata;
```

Considerazioni per oggetti di grandi dimensioni

Di seguito sono riportate alcune considerazioni importanti da tenere presenti quando si utilizza oggetti di grandi dimensioni:

- `Vacuumlo` è l'unica soluzione in quanto attualmente non esiste un altro metodo per rimuovere gli oggetti orfani di grandi dimensioni.
- Strumenti come pglogical, replica logica nativa, e AWS DMS che utilizzano tecnologie di replica non supportano la replica di oggetti di grandi dimensioni.
- Durante la progettazione dello schema del database, evitare di utilizzare oggetti di grandi dimensioni quando possibile e prendere in considerazione l'utilizzo di tipi di dati alternativi come `bytea`.
- Eseguire `vacuumlo` regolarmente, almeno settimanalmente, per evitare problemi con gli oggetti orfani di grandi dimensioni.
- Utilizzare un trigger con la funzione `lo_manage` sulle tabelle che archiviano oggetti di grandi dimensioni per impedire la creazione di oggetti orfani di grandi dimensioni.

Gestione dei dati spaziali con estensione PostGIS

PostGIS è un'estensione di PostgreSQL per l'archiviazione e la gestione delle informazioni spaziali. Per ulteriori informazioni su PostGIS, consulta PostGIS.net.

A partire dalla versione 10.5, PostgreSQL supporta la libreria libprotobuf 1.3.0 utilizzata da PostGIS per lavorare con i dati delle tile vettoriali delle mappe.

La configurazione dell'estensione PostGIS richiede i privilegi `rds_superuser`. Ti consigliamo di creare un utente (ruolo) per gestire l'estensione PostGIS e i dati spaziali. L'estensione PostGIS e i relativi componenti aggiungono migliaia di funzioni a PostgreSQL. Considera la possibilità di creare l'estensione PostGIS nel proprio schema se ciò ha senso per il tuo caso d'uso. Nell'esempio seguente viene illustrato come installare l'estensione nel proprio database, ma questa operazione non è necessaria.

Argomenti

- Passaggio 1: Creazione di un utente (ruolo) per gestire l'estensione PostGIS
- Passaggio 2: Caricamento delle estensioni di PostGIS
- Fase 3: trasferimento della proprietà degli schemi di estensione
- Fase 4: Trasferimento della proprietà delle tabelle PostGIS
- Passaggio 5: Verificare le estensioni
- Passaggio 6: Aggiornamento dell'estensione PostGIS
- Versioni dell'estensione PostGIS
- Aggiornamento di PostGIS 2 a PostGIS 3

Passaggio 1: Creazione di un utente (ruolo) per gestire l'estensione PostGIS

Per prima cosa, esegui la connessione a un'istanza database RDS per PostgreSQL come utente con i privilegi `rds_superuser`. Se hai mantenuto il nome di default durante la configurazione dell'istanza, esegui la connessione come `postgres`.

```
psql --host=111122223333.aws-region.rds.amazonaws.com --port=5432 --username=postgres
 --password
```

Crea un ruolo separato (utente) per amministrare l'estensione PostGIS.

```
postgres=>  CREATE ROLE gis_admin LOGIN PASSWORD 'change_me';
CREATE ROLE
```

Concedi a questo ruolo i privilegi `rds_superuser` per consentire l'installazione dell'estensione.

```
postgres=> GRANT rds_superuser TO gis_admin;
GRANT
```

Crea un database da utilizzare per gli artefatti PostGIS. Questa fase è facoltativa. In alternativa, puoi creare uno schema nel database utente per le estensioni PostGIS, ma anche questa operazione non è necessaria.

```
postgres=> CREATE DATABASE lab_gis;
CREATE DATABASE
```

Concedi a `gis_admin` tutti i privilegi per il database `lab_gis`.

```
postgres=> GRANT ALL PRIVILEGES ON DATABASE lab_gis TO gis_admin;
GRANT
```

Esci dalla sessione ed esegui nuovamente la connessione all'istanza database RDS per PostgreSQL come `gis_admin`.

```
postgres=> psql --host=111122223333.aws-region.rds.amazonaws.com --port=5432 --
username=gis_admin --password --dbname=lab_gis
Password for user gis_admin:...
lab_gis=>
```

Continua a configurare l'estensione come descritto nei passaggi successivi.

Passaggio 2: Caricamento delle estensioni di PostGIS

L'estensione PostGIS include diverse estensioni correlate che interagiscono per fornire funzionalità geospaziali. A seconda del caso d'uso, è possibile che le estensioni create in questo passaggio non siano tutte necessarie.

Utilizzare `CREATE EXTENSION` le istruzioni per caricare le estensioni PostGIS.

```
CREATE EXTENSION postgis;
CREATE EXTENSION
CREATE EXTENSION postgis_raster;
CREATE EXTENSION
CREATE EXTENSION fuzzystrmatch;
CREATE EXTENSION
CREATE EXTENSION postgis_tiger_geocoder;
CREATE EXTENSION
CREATE EXTENSION postgis_topology;
CREATE EXTENSION
CREATE EXTENSION address_standardizer_data_us;
```

```
CREATE EXTENSION
```

È possibile verificare i risultati eseguendo la query SQL mostrata nel seguente esempio, che elenca le estensioni e i relativi proprietari.

```
SELECT n.nspname AS "Name",
  pg_catalog.pg_get_userbyid(n.nspowner) AS "Owner"
  FROM pg_catalog.pg_namespace n
  WHERE n.nspname !~ '^pg_' AND n.nspname <> 'information_schema'
  ORDER BY 1;
List of schemas
     Name     |   Owner
--------------+-----------
 public       | postgres
 tiger        | rdsadmin
 tiger_data   | rdsadmin
 topology     | rdsadmin
(4 rows)
```

Fase 3: trasferimento della proprietà degli schemi di estensione

Usare le istruzioni ALTER SCHEMA per trasferire la proprietà degli schemi al ruolo `gis_admin`.

```
ALTER SCHEMA tiger OWNER TO gis_admin;
ALTER SCHEMA
ALTER SCHEMA tiger_data OWNER TO gis_admin;
ALTER SCHEMA
ALTER SCHEMA topology OWNER TO gis_admin;
ALTER SCHEMA
```

È possibile confermare la modifica della proprietà eseguendo la seguente query SQL. Oppure è possibile utilizzare il meta-comando \dn dalla riga di comando psql.

```
SELECT n.nspname AS "Name",
  pg_catalog.pg_get_userbyid(n.nspowner) AS "Owner"
  FROM pg_catalog.pg_namespace n
  WHERE n.nspname !~ '^pg_' AND n.nspname <> 'information_schema'
  ORDER BY 1;

      List of schemas
```

```
     Name     |     Owner
--------------+---------------
 public       | postgres
 tiger        | gis_admin
 tiger_data   | gis_admin
 topology     | gis_admin
(4 rows)
```

Fase 4: Trasferimento della proprietà delle tabelle PostGIS

> **Note**
>
> Non modificare la proprietà delle funzioni PostGIS. Il corretto funzionamento e i futuri aggiornamenti di PostGIS richiedono che queste funzioni mantengano la proprietà originale. Per ulteriori informazioni sulle autorizzazioni PostGIS, consulta PostgreSQL Security.

Utilizza la funzione seguente per trasferire la proprietà delle tabelle PostGIS al ruolo `gis_admin`. Eseguire la seguente istruzione dal prompt di psql per creare la funzione.

```
CREATE FUNCTION exec(text) returns text language plpgsql volatile AS $f$ BEGIN EXECUTE
 $1; RETURN $1; END; $f$;
CREATE FUNCTION
```

Successivamente, eseguire la seguente query per eseguire la funzione `exec` che a sua volta esegue le istruzioni e altera le autorizzazioni.

```
SELECT exec('ALTER TABLE ' || quote_ident(s.nspname) || '.' || quote_ident(s.relname)
 || ' OWNER TO gis_admin;')
  FROM (
    SELECT nspname, relname
    FROM pg_class c JOIN pg_namespace n ON (c.relnamespace = n.oid)
    WHERE nspname in ('tiger','topology') AND
    relkind IN ('r','S','v') ORDER BY relkind = 'S')
s;
```

Passaggio 5: Verificare le estensioni

Per evitare di dover specificare il nome dello schema, aggiungi lo schema `tiger` al percorso di ricerca usando il seguente comando.

```
SET search_path=public,tiger;
SET
```

Verifica lo schema `tiger` usando la seguente istruzione SELECT.

```
SELECT address, streetname, streettypeabbrev, zip
 FROM normalize_address('1 Devonshire Place, Boston, MA 02109') AS na;
address | streetname | streettypeabbrev |  zip
---------+------------+------------------+-------
       1 | Devonshire | Pl               | 02109
(1 row)
```

Per ulteriori informazioni su questa estensione, consulta Tiger Geocoder nella documentazione di PostGIS.

Verifica lo schema `topology` usando la seguente istruzione `SELECT`. Questa richiama la funzione `createtopology` per registrare un nuovo oggetto topologia (my_new_topo) con l'identificatore di riferimento spaziale specificato (26986) e la tolleranza predefinita (0,5). Per ulteriori informazioni, consulta CreateTopology nella documentazione di PostGIS.

```
SELECT topology.createtopology('my_new_topo',26986,0.5);
 createtopology
----------------
              1
(1 row)
```

Passaggio 6: Aggiornamento dell'estensione PostGIS

Ogni nuova versione di PostgreSQL supporta una o più versioni dell'estensione PostGIS compatibili con tale versione. L'aggiornamento del motore PostgreSQL a una nuova versione non aggiorna automaticamente l'estensione PostGIS. Prima di aggiornare il motore PostgreSQL, in genere si aggiorna PostGIS alla versione più recente disponibile per la versione di PostgreSQL corrente. Per informazioni dettagliate, consultare Versioni dell'estensione PostGIS.

Dopo l'aggiornamento del motore PostgreSQL, si aggiorna nuovamente l'estensione PostGIS alla versione supportata per la versione del motore PostgreSQL aggiornata. Per ulteriori informazioni sull'aggiornamento del motore PostgreSQL, consulta Test di un aggiornamento del cluster database di produzione a una nuova versione principale.

Puoi verificare la disponibilità di aggiornamenti della versione dell'estensione PostGIS sul cluster database Aurora PostgreSQL in qualsiasi momento. Per farlo, esegui il comando seguente. Questa funzione è disponibile con PostGIS 2.5.0 e versioni successive.

```
SELECT postGIS_extensions_upgrade();
```

Se l'applicazione non supporta la versione più recente di PostGIS, puoi installare una versione precedente di PostGIS disponibile nella versione principale, come indicato di seguito.

```
CREATE EXTENSION postgis VERSION "2.5.5";
```

Se desideri eseguire l'aggiornamento a una versione PostGIS specifica da una versione precedente, puoi anche utilizzare il seguente comando.

```
ALTER EXTENSION postgis UPDATE TO "2.5.5";
```

A seconda della versione da cui si esegue l'aggiornamento, potrebbe essere necessario utilizzare nuovamente questa funzione. Il risultato della prima esecuzione della funzione determina se è necessaria una funzione di aggiornamento aggiuntiva. Ad esempio, questo si verifica per l'aggiornamento da PostGIS 2 a PostGIS 3. Per ulteriori informazioni, consulta Aggiornamento di PostGIS 2 a PostGIS 3.

Se l'estensione è stata aggiornata in preparazione a un aggiornamento della versione principale del motore PostgreSQL, puoi continuare con altre attività preliminari. Per ulteriori informazioni, consulta Test di un aggiornamento del cluster database di produzione a una nuova versione principale.

Versioni dell’estensione PostGIS

Ti consigliamo di installare le versioni di tutte le estensioni, ad esempio PostGIS, come elencato in Versioni delle estensioni per Aurora edizione compatibile con PostgreSQL nelle Note di rilascio di Aurora PostgreSQL. Per ottenere un elenco delle versioni disponibili nella versione, utilizza il comando seguente.

```
SELECT * FROM pg_available_extension_versions WHERE name='postgis';
```

Puoi trovare le informazioni sulle versioni nelle sezioni seguenti delle Note di rilascio di Aurora PostgreSQL:

- Versioni delle estensioni per Aurora PostgreSQL 14

- Versioni delle estensioni per Aurora edizione compatibile con PostgreSQL 13
- Versioni delle estensioni per Aurora edizione compatibile con PostgreSQL 12
- Versioni delle estensioni per Aurora edizione compatibile con PostgreSQL 11
- Versioni delle estensioni per Aurora edizione compatibile con PostgreSQL 10
- Versioni delle estensioni per Aurora edizione compatibile con PostgreSQL 9.6

Aggiornamento di PostGIS 2 a PostGIS 3

A partire dalla versione 3.0, la funzionalità raster di PostGIS è ora un'estensione separata, `postgis_raster`. Questa estensione dispone di un proprio percorso di installazione e aggiornamento. Ciò rimuove dall'estensione `postgis` core dozzine di funzioni, tipi di dati e altri artefatti necessari per l'elaborazione di immagini raster. Ciò significa che se il caso d'uso non richiede l'elaborazione raster, non è necessario installare l'estensione `postgis_raster`.

Nel seguente esempio di aggiornamento, il primo comando di aggiornamento estrae la funzionalità raster nell'estensione `postgis_raster`. È quindi necessario un secondo comando di aggiornamento per eseguire l'aggiornamento di `postgis_raster` alla nuova versione.

Per eseguire l'aggiornamento da PostGIS 2 a PostGIS 3

1. Identifica la versione predefinita di PostGIS disponibile per la versione PostgreSQL sul cluster database Aurora PostgreSQL. A questo scopo, esegui la query seguente.

```
SELECT * FROM pg_available_extensions
    WHERE default_version > installed_version;
  name   | default_version | installed_version |                          comment
---------+-----------------+-------------------
+-----------------------------------------------------------
 postgis | 3.1.4           | 2.3.7             | PostGIS geometry and geography
 spatial types and functions
(1 row)
```

2. Identifica le versioni di PostGIS installate in ogni database sull'istanza di scrittura del cluster database Aurora PostgreSQL. In altre parole, esegui la query su ogni database utente come riportato di seguito.

```
SELECT
    e.extname AS "Name",
    e.extversion AS "Version",
```

```
    n.nspname AS "Schema",
    c.description AS "Description"
FROM
    pg_catalog.pg_extension e
    LEFT JOIN pg_catalog.pg_namespace n ON n.oid = e.extnamespace
    LEFT JOIN pg_catalog.pg_description c ON c.objoid = e.oid
    AND c.classoid = 'pg_catalog.pg_extension'::pg_catalog.regclass
WHERE
    e.extname LIKE '%postgis%'
ORDER BY
    1;
  Name   | Version | Schema |                            Description
---------+---------+--------
+------------------------------------------------------------------------
 postgis | 2.3.7   | public | PostGIS geometry, geography, and raster spatial types
 and functions
(1 row)
```

Questa discrepanza tra la versione predefinita (PostGIS 3.1.4) e la versione installata (PostGIS 2.3.7) indica che è necessario aggiornare l'estensione PostGIS.

```
ALTER EXTENSION postgis UPDATE;
ALTER EXTENSION
WARNING: unpackaging raster
WARNING: PostGIS Raster functionality has been unpackaged
```

3. Esegui la seguente query per verificare che la funzionalità raster sia ora contenuta nel proprio pacchetto.

```
SELECT
    probin,
    count(*)
FROM
    pg_proc
WHERE
    probin LIKE '%postgis%'
GROUP BY
    probin;
          probin          | count
--------------------------+-------
 $libdir/rtpostgis-2.3    | 107
 $libdir/postgis-3        | 487
```

```
(2 rows)
```

L'output mostra che c'è ancora una differenza tra le versioni. Le funzioni PostGIS sono versione 3 (postgis-3), mentre le funzioni raster (rtpostgis) sono versione 2 (rtpostgis-2.3). Per completare l'aggiornamento, esegui nuovamente il comando di aggiornamento, come riportato di seguito.

```
postgres=> SELECT postgis_extensions_upgrade();
```

Puoi ignorare i messaggi di avviso in sicurezza. Esegui nuovamente la seguente query per verificare che l'aggiornamento sia stato completato. L'aggiornamento è completato quando PostGIS e tutte le estensioni correlate non sono contrassegnate come necessarie di aggiornamento.

```
SELECT postgis_full_version();
```

4. Utilizza la seguente query per visualizzare il processo di aggiornamento completato e le estensioni impacchettate separatamente e verifica che le relative versioni corrispondano.

```
SELECT
    e.extname AS "Name",
    e.extversion AS "Version",
    n.nspname AS "Schema",
    c.description AS "Description"
FROM
    pg_catalog.pg_extension e
    LEFT JOIN pg_catalog.pg_namespace n ON n.oid = e.extnamespace
    LEFT JOIN pg_catalog.pg_description c ON c.objoid = e.oid
        AND c.classoid = 'pg_catalog.pg_extension'::pg_catalog.regclass
WHERE
    e.extname LIKE '%postgis%'
ORDER BY
    1;
      Name      | Version | Schema |                            Description
----------------+---------+--------
+------------------------------------------------------------------------
 postgis        | 3.1.5   | public | PostGIS geometry, geography, and raster
 spatial types and functions
 postgis_raster | 3.1.5   | public | PostGIS raster types and functions
(2 rows)
```

L'output mostra che l'estensione PostGIS 2 è stata aggiornata a PostGIS 3 e che `postgis` e l'estensione `postgis_raster` ora separata sono entrambe versione 3.1.5.

Al termine dell'aggiornamento, se non prevedi di utilizzare la funzionalità raster, puoi rimuovere l'estensione come segue.

```
DROP EXTENSION postgis_raster;
```

Gestione delle partizioni PostgreSQL con l'estensione pg_partman

Il partizionamento delle tabelle PostgreSQL fornisce un framework per la gestione ad alte prestazioni di input e reporting dei dati. Utilizzare il partizionamento per database che richiedono un input molto veloce di grandi quantità di dati. Il partizionamento fornisce anche query di tabelle di grandi dimensioni più veloci. Il partizionamento consente di conservare i dati senza influire sull'istanza del database perché richiede meno risorse I/O.

Utilizzando il partizionamento, è possibile suddividere i dati in blocchi di dimensioni personalizzate per l'elaborazione. Ad esempio, è possibile partizionare i dati delle serie temporali per intervalli quali orario, giornaliero, settimanale, mensile, trimestrale, annuale, personalizzato o qualsiasi combinazione di questi. Per un esempio di dati di serie temporali, se la tabella è stata partizionata per ora, ogni partizione conterrà un'ora di dati. Se si partiziona la tabella delle serie temporali per giorno, le partizioni conterranno i dati di un giorno e così via. La chiave di partizione controlla le dimensioni di una partizione.

Quando si utilizza un comando SQL `INSERT` o `UPDATE` in una tabella partizionata, il motore database indirizza i dati alla partizione appropriata. Le partizioni di tabella PostgreSQL che memorizzano i dati sono tabelle figlio della tabella principale.

Durante le letture delle query di database, l'ottimizzatore PostgreSQL esamina la clausola `WHERE` della query e, se possibile, indirizza la scansione del database solo alle partizioni pertinenti.

A partire dalla versione 10, PostgreSQL utilizza il partizionamento dichiarativo per implementare il partizionamento delle tabelle. Questo è noto anche come partizionamento PostgreSQL nativo. Prima di PostgreSQL versione 10, per implementare le partizioni venivano utilizzati i trigger.

Il partizionamento delle tabelle PostgreSQL fornisce le seguenti funzionalità:

- Creazione di nuove partizioni in qualsiasi momento.

- Intervalli di partizione variabili.
- Partizioni scollegabili e ricollegabili utilizzando istruzioni DDL (Data Definition Language).

  Ad esempio, le partizioni scollegabili sono utili per rimuovere i dati storici dalla partizione principale, conservando i dati storici per l'analisi.
- Le nuove partizioni ereditano le proprietà della tabella di database padre, tra cui:
  - Indici
  - Chiavi primarie, che devono includere la colonna delle chiavi di partizione
  - Chiavi esterne
  - Vincoli check
  - Riferimenti
- Creazione di indici per la tabella completa o per ogni partizione specifica.

Non è possibile modificare lo schema per una singola partizione. Tuttavia, è possibile modificare la tabella padre (ad esempio, aggiungendo una nuova colonna), che si propaga alle partizioni.

Argomenti

- Panoramica dell'estensione PostgreSQL pg_partman
- Abilitazione dell'estensione pg_partman
- Configurazione delle partizioni utilizzando la funzione create_parent
- Configurazione della manutenzione delle partizioni utilizzando la funzione run_maintenance ance_proc

Panoramica dell'estensione PostgreSQL pg_partman

È possibile utilizzare l'estensione `pg_partman` PostgreSQL per automatizzare la creazione e la manutenzione delle partizioni di tabella. Per informazioni più generali, consulta PG Partition Manager nella documentazione di `pg_partman`.

> **Note**
>
> L'estensione `pg_partman` è supportata su Aurora PostgreSQL versioni 12.6 e successive.

Invece di dover creare manualmente ogni partizione, è possibile configurare pg_partman con le seguenti impostazioni:

- Tabella da partizionare
- Tipo di partizione
- Chiave di partizione
- Granularità delle partizioni
- Opzioni di pre-creazione e gestione delle partizioni

Dopo aver creato una tabella con partizioni PostgreSQL, la si registra con pg_partman chiamando la funzione create_parent. In questo modo vengono create le partizioni necessarie in base ai parametri passati alla funzione.

L'estensione pg_partman fornisce anche la funzione run_maintenance_proc, che è possibile chiamare su base pianificata per gestire automaticamente le partizioni. Per pianificare la creazione delle partizioni appropriate in base alle esigenze, puoi pianificare questa funzione in modo che venga eseguita periodicamente (ad esempio, ogni ora). È inoltre possibile assicurarsi che le partizioni vengano eliminate automaticamente.

Abilitazione dell'estensione pg_partman

Se disponi di più database all'interno della stessa istanza database per cui desideri gestire le partizioni, è necessario abilitare l'estensione pg_partman separatamente per ogni database. Per abilitare l'estensione pg_partman per un database specifico, crea lo schema di manutenzione delle partizioni, quindi crea l'estensione pg_partman nel modo seguente:

```
CREATE SCHEMA partman;
CREATE EXTENSION pg_partman WITH SCHEMA partman;
```

> **Note**
>
> Per creare l'estensione pg_partman, assicurati di disporre dei privilegi rds_superuser.

Se viene restituito un errore come il seguente, concedi i privilegi rds_superuser all'account o utilizza l'account utente avanzato.

```
ERROR: permission denied to create extension "pg_partman"
HINT: Must be superuser to create this extension.
```

Per concedere i privilegi `rds_superuser`, collegati con l'account utente avanzato ed emetti il seguente comando:

```
GRANT rds_superuser TO user-or-role;
```

Per gli esempi che mostrano l'uso dell'estensione pg_partman, si utilizza la tabella di database e la partizione di esempio seguenti. Questo database utilizza una tabella partizionata basata su un timestamp. Uno schema `data_mart` contiene una tabella denominata `events` con una colonna denominata `created_at`. Nella tabella `events` sono incluse le seguenti impostazioni:

- Chiavi primarie `event_id` e `created_at`, che devono avere la colonna utilizzata per guidare la partizione.
- Un vincolo di controllo `ck_valid_operation` per applicare i valori per una colonna della tabella `operation` .
- Due chiavi esterne, dove una (`fk_orga_membership`)) punta alla tabella esterna `organization` e l'altra (`fk_parent_event_id`) è una chiave esterna autoreferenziata.
- Due indici, dove uno (`idx_org_id`) è per la chiave esterna e l'altro (`idx_event_type`) è per il tipo di evento.

Le seguenti istruzioni DDL creano questi oggetti, che verranno inclusi automaticamente in ogni partizione.

```
CREATE SCHEMA data_mart;
CREATE TABLE data_mart.organization ( org_id BIGSERIAL,
        org_name TEXT,
        CONSTRAINT pk_organization PRIMARY KEY (org_id)
    );

CREATE TABLE data_mart.events(
        event_id        BIGSERIAL,
        operation       CHAR(1),
        value           FLOAT(24),
        parent_event_id BIGINT,
        event_type      VARCHAR(25),
        org_id          BIGSERIAL,
```

```
        created_at      timestamp,
        CONSTRAINT pk_data_mart_event PRIMARY KEY (event_id, created_at),
        CONSTRAINT ck_valid_operation CHECK (operation = 'C' OR operation = 'D'),
        CONSTRAINT fk_orga_membership
            FOREIGN KEY(org_id)
            REFERENCES data_mart.organization (org_id),
        CONSTRAINT fk_parent_event_id
            FOREIGN KEY(parent_event_id, created_at)
            REFERENCES data_mart.events (event_id,created_at)
    ) PARTITION BY RANGE (created_at);

CREATE INDEX idx_org_id     ON  data_mart.events(org_id);
CREATE INDEX idx_event_type ON  data_mart.events(event_type);
```

Configurazione delle partizioni utilizzando la funzione create_parent

Dopo aver abilitato l'estensione `pg_partman`, utilizza la funzione `create_parent` per configurare le partizioni all'interno dello schema di manutenzione delle partizioni. In questo esempio viene utilizzato l'esempio della tabella `events` creato in Abilitazione dell'estensione pg_partman. Richiama la funzione `create_parent` come segue:

```
SELECT partman.create_parent(
 p_parent_table => 'data_mart.events',
 p_control      => 'created_at',
 p_type         => 'range',
 p_interval     => '1 day',
 p_premake      => 30);
```

I parametri sono i seguenti:

- `p_parent_table` – La tabella partizionata padre. Questa tabella deve già esistere ed essere completa, deve ovvero includere lo schema.
- `p_control` – Colonna su cui basare il partizionamento. Il tipo di dati deve essere intero o basato sul tempo.
- `p_type`: il tipo è `'range'` o `'list'`.
- `p_interval` – Intervallo di tempo o intervallo intero per ogni partizione. I valori di esempio includono `1 day`, `1 hour` e così via.
- `p_premake` – Il numero di partizioni da creare in anticipo per supportare nuovi inserimenti.

Per una descrizione completa della funzione `create_parent`, consulta Funzioni di creazione nella documentazione `pg_partman`.

Configurazione della manutenzione delle partizioni utilizzando la funzione run_maintenance ance_proc

È possibile eseguire operazioni di manutenzione delle partizioni per creare automaticamente nuove partizioni, scollegare partizioni o rimuovere partizioni obsolete. La manutenzione delle partizioni si basa sulla funzione `run_maintenance_proc` dell'estensione `pg_partman` e dell'estensione `pg_cron`, che avvia un pianificatore interno. Lo scheduler `pg_cron` esegue automaticamente le istruzioni SQL, le funzioni e le procedure definite nei database.

Nell'esempio seguente viene utilizzato l'esempio della tabella `events` creata in Abilitazione dell'estensione pg_partman per impostare l'esecuzione automatica delle operazioni di manutenzione delle partizioni. Come prerequisito, aggiungere `pg_cron` al parametro `shared_preload_libraries` nel gruppo di parametri dell'istanza database.

```
CREATE EXTENSION pg_cron;

UPDATE partman.part_config
SET infinite_time_partitions = true,
    retention = '3 months',
    retention_keep_table=true
WHERE parent_table = 'data_mart.events';
SELECT cron.schedule('@hourly', $$CALL partman.run_maintenance_proc()$$);
```

Di seguito, è riportata una spiegazione dettagliata dell'esempio precedente:

1. Modifica il gruppo di parametri associato all'istanza database e aggiungi `pg_cron` al valore del parametro `shared_preload_libraries`. Perché questa modifica abbia effetto, è necessario riavviare l'istanza database. Per ulteriori informazioni, consulta Modifica dei parametri in un gruppo di parametri database in Amazon Aurora.
2. Emettere il comando `CREATE EXTENSION pg_cron;` utilizzando un account con le autorizzazioni `rds_superuser`. In questo modo, viene abilitata l'estensione `pg_cron`. Per ulteriori informazioni, consulta Pianificazione della manutenzione con l'estensione PostgreSQL pg_cron.
3. Emettere il comando `UPDATE partman.part_config` per regolare le impostazioni `pg_partman` per la tabella `data_mart.events`.

4. Eseguire il comando `SET . . .` per configurare la tabella `data_mart.events`, con le seguenti clausole:
   a. `infinite_time_partitions = true,` – Configura la tabella in modo da poter creare automaticamente nuove partizioni senza limiti.
   b. `retention = '3 months',` – Configura la tabella in modo che venga conservata per un massimo di tre mesi.
   c. `retention_keep_table=true` – Configura la tabella in modo che quando il periodo di conservazione è scaduto, la tabella non venga eliminata automaticamente. Le partizioni precedenti al periodo di conservazione vengono invece scollegate dalla tabella padre.
5. Eseguire il comando `SELECT cron.schedule . . .` per creare una chiamata di funzione `pg_cron`. Questa chiamata definisce la frequenza con cui lo scheduler esegue la procedura di manutenzione `pg_partman`, `partman.run_maintenance_proc`. Per questo esempio, la procedura viene eseguita ogni ora.

Per una descrizione completa della funzione `run_maintenance_proc`, consulta Funzioni di manutenzione nella documentazione di `pg_partman`.

Pianificazione della manutenzione con l'estensione PostgreSQL pg_cron

Puoi utilizzare l'estensione PostgreSQL `pg_cron` per pianificare i comandi di manutenzione all'interno di un database PostgreSQL. Per ulteriori informazioni sull'estensione, consulta Che cos'è pg_cron? nella documentazione di pg_cron.

L'estensione `pg_cron` è supportata sul motore Aurora PostgreSQL versioni 12.6 e successive

Per ulteriori informazioni sull'uso di `pg_cron`, consulta Schedule jobs with pg_cron on your RDS for PostgreSQL or your Aurora PostgreSQL-Compatible Edition databases (Pianificazione dei processi con pg_cron sui database RDS per PostgreSQL o Aurora edizione compatibile con PostgreSQL).

> **Note**
>
> La versione dell'estensione `pg_cron` viene visualizzata come versione a due cifre, ad esempio 1.6, nella vista pg_available_extensions. Sebbene in alcuni contesti possano essere elencate versioni a tre cifre, ad esempio 1.6.4 o 1.6.5, è necessario specificare la versione a due cifre quando si esegue un aggiornamento dell'estensione.

Argomenti

- Configurazione dell'estensione pg_cron
- Concessione delle autorizzazioni per utilizzare pg_cron agli utenti del database
- Programmazione di processi pg_cron
- Riferimento per l'estensione pg_cron

Configurazione dell'estensione pg_cron

Configura l'estensione `pg_cron` come riportato di seguito:

1. Modifica il gruppo di parametri personalizzati associato all'istanza database PostgreSQL aggiungendo `pg_cron` al valore del parametro `shared_preload_libraries`.

   Per rendere effettive le modifiche apportate al gruppo di parametri, riavvia l'istanza database PostgreSQL. Per ulteriori informazioni sull'utilizzo dei gruppi di parametri, consulta Amazon Aurora PostgreSQL parametri.
2. Dopo il riavvio dell'istanza database PostgreSQL, esegui il comando riportato di seguito utilizzando un account che dispone delle autorizzazioni `rds_superuser`. Ad esempio, se hai utilizzato le impostazioni di default quando hai creato il cluster di database Aurora PostgreSQL, connettiti come utente `postgres` e crea l'estensione.

   ```
   CREATE EXTENSION pg_cron;
   ```

   Lo scheduler `pg_cron` è impostato nel database PostgreSQL predefinito denominato `postgres`. Gli oggetti `pg_cron` vengono creati in questo database `postgres` e tutte le azioni di pianificazione vengono eseguite in questo database.
3. Puoi usare le impostazioni predefinite oppure puoi pianificare i processi da eseguire in altri database all'interno dell'istanza database di PostgreSQL. Per pianificare i processi per altri database all'interno dell'istanza database di PostgreSQL, consulta l'esempio in Pianificazione di un processo cron per un database diverso da quello predefinito.

Concessione delle autorizzazioni per utilizzare pg_cron agli utenti del database

L'installazione dell'estensione `pg_cron` richiede privilegi `rds_superuser`. Tuttavia, le autorizzazioni per utilizzare `pg_cron` possono essere concesse (da un membro del gruppo/ruolo `rds_superuser`) ad altri utenti del database, in modo che possano pianificare i propri lavori. Ti consigliamo di

concedere le autorizzazioni allo schema `cron` solo se in base alle esigenze se migliora le operazioni nell'ambiente di produzione.

Per concedere a un database l'autorizzazione utente nello schema `cron`, esegui il seguente comando:

```
postgres=> GRANT USAGE ON SCHEMA cron TO db-user;
```

Questo dà *`db-user`* il permesso di accedere allo `cron` schema per pianificare i cron job per gli oggetti a cui hanno i permessi di accesso. Se l'utente del database non dispone di autorizzazioni, il processo non va a buon fine dopo aver pubblicato il messaggio di errore nel file `postgresql.log`, come mostrato di seguito:

```
2020-12-08 16:41:00 UTC::@:[30647]:ERROR: permission denied for table table-name
2020-12-08 16:41:00 UTC::@:[27071]:LOG: background worker "pg_cron" (PID 30647) exited
 with exit code 1
```

In altre parole, assicurati che gli utenti del database a cui sono concesse autorizzazioni per lo schema `cron`, dispongano anche di autorizzazioni sugli oggetti (tabelle, schemi e così via) che intendono pianificare.

I dettagli del processo cron e l'esito positivo o negativo sono anche acquisiti nella tabella `cron.job_run_details`. Per ulteriori informazioni, consulta Tabelle per la pianificazione dei processi e l'acquisizione dello stato .

Programmazione di processi pg_cron

Nelle sezioni seguenti viene illustrato come è possibile pianificare varie attività di gestione utilizzando le attività `pg_cron`.

> **Note**
>
> Quando crei processi `pg_cron`, controlla che l'impostazione `max_worker_processes` sia maggiore del numero di `cron.max_running_jobs`. Un processo `pg_cron` non riesce se i processi worker in background vengono esauriti. Il numero predefinito di processi `pg_cron` è 5. Per ulteriori informazioni, consulta Parametri per la gestione dell'estensione pg_cron.

Argomenti

- Vacuum di una tabella
- Eliminazione della tabella della cronologia di pg_cron
- Registrazione degli errori solo nel file postgresql.log
- Pianificazione di un processo cron per un database diverso da quello predefinito

Vacuum di una tabella

Autovacuum gestisce la manutenzione del vacuum per la maggior parte dei casi. Tuttavia, potresti voler programmare un vacuum di una tabella specifica in un momento di tua scelta.

Di seguito è riportato un esempio di utilizzo della funzione `cron.schedule` per impostare un processo in modo da utilizzare `VACUUM FREEZE` su una tabella specifica ogni giorno alle 22:00 (GMT).

```
SELECT cron.schedule('manual vacuum', '0 22 * * *', 'VACUUM FREEZE pgbench_accounts');
 schedule
----------
1
(1 row)
```

Dopo l'esecuzione dell'esempio precedente, è possibile controllare la cronologia nella tabella `cron.job_run_details` come riportato di seguito.

```
postgres=> SELECT * FROM cron.job_run_details;
jobid  | runid | job_pid | database | username | command                        |
 status    | return_message | start_time                    | end_time
-------+-------+---------+----------+----------+--------------------------------
+-----------+----------------+-------------------------------
+-------------------------------
 1     | 1     | 3395    | postgres | adminuser| vacuum freeze pgbench_accounts
 | succeeded | VACUUM         | 2020-12-04 21:10:00.050386+00 | 2020-12-04
 21:10:00.072028+00
(1 row)
```

Di seguito è riportata una query della tabella `cron.job_run_details` per visualizzare i processi non riusciti.

```
postgres=> SELECT * FROM cron.job_run_details WHERE status = 'failed';
```

```
jobid | runid | job_pid | database | username | command                          | status
 | return_message                                | start_time                    |
 end_time
------+-------+---------+----------+----------+----------------------------------+--------
+-----------------------------------------------+-------------------------------
+-------------------------------
 5    | 4     | 30339   | postgres | adminuser| vacuum freeze pgbench_account | failed
 | ERROR: relation "pgbench_account" does not exist | 2020-12-04 21:48:00.015145+00 |
 2020-12-04 21:48:00.029567+00
(1 row)
```

Per ulteriori informazioni, consulta Tabelle per la pianificazione dei processi e l'acquisizione dello stato .

Eliminazione della tabella della cronologia di pg_cron

La tabella `cron.job_run_details` contiene una cronologia di processi cron che può crescere a dismisura nel tempo. Si consiglia di pianificare un processo che elimini questa tabella. Ad esempio, conservare una settimana di voci può essere sufficiente per la risoluzione dei problemi.

Nell'esempio seguente viene utilizzata la funzione cron.schedule per pianificare un processo che viene eseguito ogni giorno a mezzanotte per rimuovere la tabella `cron.job_run_details`. Il processo conserva solo gli ultimi sette giorni. Utilizza l'account `rds_superuser` per pianificare il processo come riportato di seguito.

```
SELECT cron.schedule('0 0 * * *', $$DELETE
    FROM cron.job_run_details
    WHERE end_time < now() - interval '7 days'$$);
```

Per ulteriori informazioni, consulta Tabelle per la pianificazione dei processi e l'acquisizione dello stato .

Registrazione degli errori solo nel file postgresql.log

Per disattivare la scrittura nella tabella `cron.job_run_details`, modifica il gruppo di parametri associato all'istanza database PostgreSQL e disattiva il parametro `cron.log_run`. L'estensione `pg_cron` non scrive più nella tabella e acquisisce gli errori solo nel file `postgresql.log`. Per ulteriori informazioni, consulta Modifica dei parametri in un gruppo di parametri database in Amazon Aurora.

Utilizza il seguente comando per controllare il valore del parametro `cron.log_run`.

```
postgres=> SHOW cron.log_run;
```

Per ulteriori informazioni, consulta Parametri per la gestione dell'estensione pg_cron.

Pianificazione di un processo cron per un database diverso da quello predefinito

I metadati per `pg_cron` sono tutti contenuti nel database predefinito PostgreSQL denominato `postgres`. Poiché i worker in background vengono utilizzati per l'esecuzione dei processi cron di manutenzione, puoi pianificare un processo in uno qualsiasi dei database all'interno dell'istanza database PostgreSQL.

> **Note**
>
> Solo gli utenti con il ruolo `rds_superuser` o i privilegi `rds_superuser` possono elencare tutti i processi cron nel database. Gli altri utenti possono visualizzare solo i propri processi nella tabella `cron.job`.

1. Nel database cron, pianifica il processo come si farebbe normalmente utilizzando cron.schedule.

   ```
   postgres=> SELECT cron.schedule('database1 manual vacuum', '29 03 * * *', 'vacuum
    freeze test_table');
   ```

2. Con il ruolo `rds_superuser`, aggiorna la colonna del database per il processo appena creato in modo che venga eseguito in un altro database all'interno dell'istanza database PostgreSQL.

   ```
   postgres=> UPDATE cron.job SET database = 'database1' WHERE jobid = 106;
   ```

3. Verifica eseguendo una query sulla tabella `cron.job`.

   ```
   postgres=> SELECT * FROM cron.job;
   jobid | schedule    | command                       | nodename  | nodeport |
    database | username  | active | jobname
   ------+-------------+-------------------------------+-----------+----------
   +----------+-----------+--------+-------------------------
   106   | 29 03 * * * | vacuum freeze test_table      | localhost | 8192     |
    database1| adminuser | t      | database1 manual vacuum
     1   | 59 23 * * * | vacuum freeze pgbench_accounts | localhost | 8192     |
    postgres | adminuser | t      | manual vacuum
   (2 rows)
   ```

> **Note**
>
> In alcune situazioni, è possibile aggiungere un processo cron che si intende eseguire in un database diverso. In questi casi, il processo potrebbe essere eseguito nel database predefinito (`postgres`) prima di aggiornare la colonna del database corretta. Se il nome utente dispone delle autorizzazioni, il processo viene eseguito correttamente nel database predefinito.

Riferimento per l'estensione pg_cron

È possibile utilizzare i seguenti parametri, funzioni e tabelle con l'estensione `pg_cron`. Per ulteriori informazioni, consultare Che cos'è pg_cron? nella documentazione di pg_cron.

Argomenti

- Parametri per la gestione dell'estensione pg_cron
- Riferimento alla funzione: cron.schedule
- Riferimento alla funzione: cron.unschedule
- Tabelle per la pianificazione dei processi e l'acquisizione dello stato

Parametri per la gestione dell'estensione pg_cron

Di seguito è riportato l'elenco dei parametri che consentono di controllare il comportamento dell'estensione `pg_cron`.

Parametro	Description
cron.database_name	Il database in cui vengono conservati i metadati `pg_cron`.
`cron.host`	Il nome host per connettersi a PostgressQL. Non è possibile modificare questo valore.
`cron.log_run`	Registra tutti i processi eseguiti nella tabella `job_run_details` . I valori sono `on` o `off`. Per ulteriori informazioni, consulta Tabelle per

Parametro	Description
	la pianificazione dei processi e l'acquisizione dello stato .
`cron.log_statement`	Registra tutte le istruzioni cron prima di eseguirle. I valori sono `on` o `off`.
`cron.max_running_jobs`	Numero massimo di processi che possono essere eseguiti contemporaneamente.
`cron.use_background_workers`	Utilizza i worker in background anziché le sessioni client. Non è possibile modificare questo valore.

Puoi utilizzare il comando SQL seguente per visualizzare questi parametri e i relativi valori.

```
postgres=> SELECT name, setting, short_desc FROM pg_settings WHERE name LIKE 'cron.%'
 ORDER BY name;
```

Riferimento alla funzione: cron.schedule

Questa funzione pianifica un processo cron. Il processo viene inizialmente pianificato nel database `postgres` predefinito. La funzione restituisce un valore `bigint` che rappresenta l'identificativo del processo. Per pianificare l'esecuzione di processi in altri database all'interno dell'istanza database di PostgreSQL, consulta l'esempio in Pianificazione di un processo cron per un database diverso da quello predefinito.

La funzione ha due diverse sintassi.

Sintassi

```
cron.schedule (job_name,
    schedule,
    command
);

cron.schedule (schedule,
    command
);
```

Parametri

Parametro	Descrizione
`job_name`	Il nome del processo cron.
`schedule`	Il testo che indica la pianificazione per il processo cron. Il formato è il formato cron standard.
`command`	Testo del comando da eseguire.

Esempi

```
postgres=> SELECT cron.schedule ('test','0 10 * * *', 'VACUUM pgbench_history');
 schedule
----------
      145
(1 row)

postgres=> SELECT cron.schedule ('0 15 * * *', 'VACUUM pgbench_accounts');
 schedule
----------
      146
(1 row)
```

Riferimento alla funzione: cron.unschedule

Questa funzione elimina un processo cron. Puoi specificare `job_name` o `job_id`. Una policy assicura che l'utente sia il proprietario e possa rimuovere la pianificazione per il processo. La funzione restituisce un valore Booleano che indica la riuscita o l'errore.

La funzione ha la seguente sintassi.

Sintassi

```
cron.unschedule (job_id);

cron.unschedule (job_name);
```

Parametri

Parametro	Descrizione
`job_id`	Un identificativo di processo restituito dalla funzione `cron.schedule` quando è stato pianificato il processo cron.
`job_name`	Il nome di un processo cron pianificato con la funzione `cron.schedule` .

Esempi

```
postgres=> SELECT cron.unschedule(108);
 unschedule
------------
 t
(1 row)

postgres=> SELECT cron.unschedule('test');
 unschedule
------------
 t
(1 row)
```

Tabelle per la pianificazione dei processi e l'acquisizione dello stato

Le seguenti tabelle vengono create e utilizzate per pianificare i processi cron e registrare il modo in cui i processi vengono completati.

Tabella	Descrizione
`cron.job`	Contiene i metadati relativi a ciascun processo pianificato. La maggior parte delle interazioni con questa tabella dovrebbe essere eseguita tramite le funzioni `cron.schedule` e `cron.unschedule` .

Tabella	Descrizione
	Important Si sconsiglia di concedere privilegi di aggiornamento/inserimento direttamente a questa tabella. In questo modo, si consente all'utente di aggiornare la colonna `username` da eseguire come `rds-superuser` .
`cron.job_run_details`	Contiene informazioni cronologiche sulle esecuzioni precedenti dei processi pianificati. Ciò è utile per analizzare lo stato, i messaggi restituiti e l'ora di inizio e di fine dall'esecuzione del processo. **Note** Per evitare che questa tabella cresca in maniera indefinita, è necessario eliminarla su base regolare. Per un esempio, consulta Eliminazione della tabella della cronologia di pg_cron.

Utilizzo di pgAudit per registrare l'attività del database

Gli istituti finanziari, gli enti governativi e molti settori devono conservare i log di audit per soddisfare i requisiti normativi. L'utilizzo dell'estensione PostgreSQL Audit (pgAudit) con il cluster database Aurora PostgreSQL consente di acquisire i record dettagliati richiesti in genere dai revisori o di soddisfare requisiti normativi. Ad esempio, è possibile impostare l'estensione pgAudit per tenere traccia delle modifiche apportate a database e tabelle specifici, per registrare l'utente che ha apportato la modifica e molti altri dettagli.

L'estensione pgAudit si basa sulla funzionalità dell'infrastruttura di registrazione PostgreSQL nativa estendendo i messaggi di registro con maggiori dettagli. In altre parole, l'approccio utilizzato per visualizzare il registro di audit è lo stesso utilizzato per visualizzare i messaggi di registro. Per ulteriori informazioni sulla registrazione PostgreSQL, consulta File di log del database Aurora PostgreSQL.

L'estensione pgAudit consente di oscurare i dati sensibili, come le password in chiaro, dai registri. Se il cluster database Aurora PostgreSQL è configurato per registrare le istruzioni DML (Data Manipulation Language) come descritto in Attivazione della registrazione delle query per il cluster database Aurora PostgreSQL, il problema delle password in chiaro può essere evitato utilizzando l'estensione PostgreSQL Audit.

È possibile configurare l'audit sulle istanze database con un elevato grado di specificità. Puoi eseguire l'audit di tutti i database e di tutti gli utenti. In alternativa, è possibile scegliere di eseguire l'audit solo di determinati database, utenti e altri oggetti. È inoltre possibile escludere esplicitamente determinati utenti e database dall'audit. Per ulteriori informazioni, consulta Esclusione di utenti o database dalla registrazione di audit.

Data la quantità di dettagli che è possibile acquisire, ti consigliamo di monitorare il consumo di archiviazione se utilizzi pgAudit.

L'estensione pgAudit è supportata su tutte le versioni di Aurora PostgreSQL disponibili. Per un elenco delle versioni di pgAudit supportate dalla versione di Aurora PostgreSQL, consulta Extension versions for Amazon Aurora PostgreSQL (Versioni delle estensioni per Amazon Aurora PostgreSQL) in Release Notes for Aurora PostgreSQL (Note di rilascio di Aurora PostgreSQL).

Argomenti

- Configurazione dell'estensione pgAudit
- Audit di oggetti di database
- Esclusione di utenti o database dalla registrazione di audit
- Riferimento per l'estensione pgAudit

Configurazione dell'estensione pgAudit

Per configurare l'estensione pgAudit sul cluster database Aurora PostgreSQL, aggiungi innanzitutto pgAudit alle librerie condivise nel gruppo di parametri cluster database personalizzato per il cluster database Aurora PostgreSQL. Per informazioni sulla creazione di un gruppo di parametri del·cluster database, consulta Gruppi di parametri per Amazon Aurora. Quindi, installa l'estensione pgAudit. Infine, specifica i database e gli oggetti di cui eseguire l'audit. Le procedure in questa sezione mostrano come fare. Puoi utilizzare la Console di gestione AWS o l'AWS CLI.

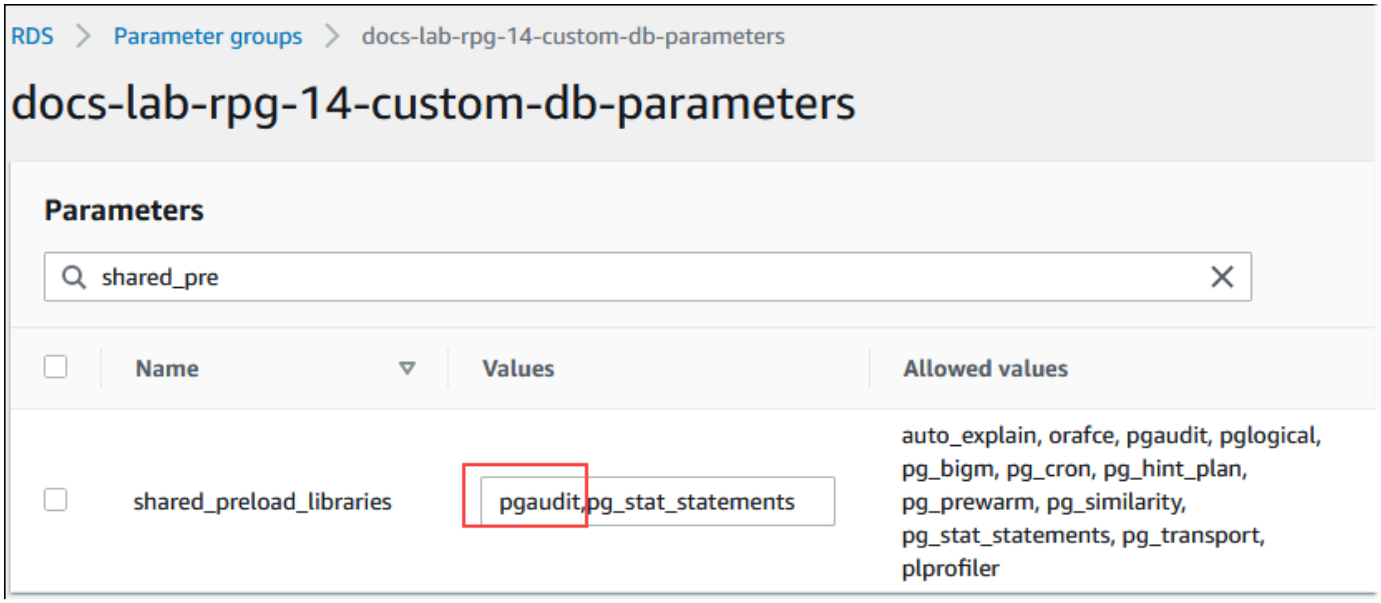
Per eseguire tutte queste attività, sono richieste autorizzazioni come il ruolo `rds_superuser`.

Le fasi seguenti si basano sull'ipotesi che il cluster database Aurora PostgreSQL sia associato a un gruppo di parametri cluster di database personalizzato.

Console

Per impostare l'estensione pgAudit

1. Accedi alla Console di gestione AWS e apri la console Amazon RDS all'indirizzo https://console.aws.amazon.com/rds/.
2. Nel riquadro di navigazione, scegli l'istanza di scrittura del cluster database Aurora PostgreSQL .
3. Apri la scheda Configurazione per l'istanza di scrittura del cluster database Aurora PostgreSQL. Tra i dettagli dell'istanza, individua il collegamento Parameter group (Gruppo di parametri).
4. Scegli il collegamento per aprire i parametri personalizzati associati al cluster database Aurora PostgreSQL.
5. Nel campo di ricerca Parametri, digita `shared_pre` per trovare il parametro `shared_preload_libraries`.
6. Scegli Edit parameters (Modifica parametri) per accedere ai valori delle proprietà.
7. Aggiungi `pgaudit` all'elenco nel campo Values (Valori). Utilizza una virgola per separare gli elementi nell'elenco di valori.



8. Riavvia l'istanza di scrittura del cluster database Aurora PostgreSQL in modo che la modifica al parametro `shared_preload_libraries` diventi effettiva.

9. Quando l'istanza è disponibile, verifica che pgAudit sia stato inizializzato. Utilizza `psql` per connetterti all'istanza di scrittura del cluster database Aurora PostgreSQL, quindi esegui il comando seguente.

```
SHOW shared_preload_libraries;
shared_preload_libraries
--------------------------
rdsutils,pgaudit
(1 row)
```

10. Con pgAudit inizializzato, puoi ora creare l'estensione. L'estensione deve essere creata dopo aver inizializzato la libreria perché l'estensione `pgaudit` installa i trigger evento per l'audit delle istruzioni DDL (Data Definition Language).

```
CREATE EXTENSION pgaudit;
```

11. Chiudi la sessione `psql`.

```
labdb=> \q
```

12. Accedi alla Console di gestione AWS e apri la console Amazon RDS all'indirizzo https://console.aws.amazon.com/rds/.

13. Trova il parametro `pgaudit.log` nell'elenco e impostalo sul valore appropriato per il caso d'uso. Ad esempio, se si imposta il parametro `pgaudit.log` su `write` come mostrato nell'immagine seguente, gli inserimenti, gli aggiornamenti, le eliminazioni e alcuni altri tipi di modifiche vengono acquisiti nel registro.

Puoi anche scegliere uno dei seguenti valori per il parametro `pgaudit.log`.

- none: valore predefinito. Non viene registrata alcuna modifica al database.
- all: registra tutto (read, write, function, role, ddl, misc).
- ddl: registra tutte le istruzioni DDL (Data Definition Language) non incluse nella classe `ROLE`.
- function: registra le chiamate di funzione e blocchi `DO`.
- misc: registra vari comandi come `DISCARD`, `FETCH`, `CHECKPOINT`, `VACUUM` e `SET`.
- read: registra `SELECT` e `COPY` quando l'origine è una relazione (ad esempio una tabella) o una query.
- role: registra le istruzioni correlate a ruoli e privilegi, ad esempio `GRANT`, `REVOKE`, `CREATE ROLE`, `ALTER ROLE` e `DROP ROLE`.
- write: registra `INSERT`, `UPDATE`, `DELETE`, `TRUNCATE` e `COPY` quando la destinazione è una relazione (tabella).

14. Scegli Save changes (Salva modifiche).
15. Apri la console di Amazon RDS all'indirizzo https://console.aws.amazon.com/rds/.
16. Nell'elenco Database, scegli l'istanza di scrittura del cluster di database Aurora PostgreSQL.

AWS CLI

Per configurare pgAudit

Per configurare pgAudit utilizzando AWS CLI, chiama l'operazione modify-db-parameter-group per modificare i parametri del registro di audit nel gruppo di parametri personalizzati, come illustrato nella procedura seguente.

1. Utilizza il seguente comando AWS CLI per aggiungere `pgaudit` al parametro `shared_preload_libraries`.

```
aws rds modify-db-parameter-group \
   --db-parameter-group-name custom-param-group-name \
   --parameters
 "ParameterName=shared_preload_libraries,ParameterValue=pgaudit,ApplyMethod=pending-
reboot" \
   --region aws-region
```

2. Utilizza il comando AWS CLI seguente per riavviare l'istanza di scrittura del cluster database Aurora PostgreSQL in modo che la libreria pgaudit venga inizializzata.

```
aws rds reboot-db-instance \
    --db-instance-identifier writer-instance \
    --region aws-region
```

3. Quando l'istanza è disponibile, verifica che `pgaudit` sia stato inizializzato. Utilizza `psql` per connetterti all'istanza di scrittura del cluster database Aurora PostgreSQL, quindi esegui il comando seguente.

```
SHOW shared_preload_libraries;
shared_preload_libraries
--------------------------
rdsutils,pgaudit
(1 row)
```

   Con pgAudit inizializzato, puoi ora creare l'estensione.

```
CREATE EXTENSION pgaudit;
```

4. Chiudi la sessione `psql` in modo da poter utilizzare AWS CLI.

```
labdb=> \q
```

5. Utilizza il comando AWS CLI seguente per specificare le classi di istruzioni che desideri registrare mediante la registrazione di audit della sessione. L'esempio imposta il parametro `pgaudit.log` su `write`, che acquisisce inserimenti, aggiornamenti ed eliminazioni nel registro.

```
aws rds modify-db-parameter-group \
   --db-parameter-group-name custom-param-group-name \
   --parameters
 "ParameterName=pgaudit.log,ParameterValue=write,ApplyMethod=pending-reboot" \
   --region aws-region
```

   Puoi anche scegliere uno dei seguenti valori per il parametro `pgaudit.log`.

   - none: valore predefinito. Non viene registrata alcuna modifica al database.
   - all: registra tutto (read, write, function, role, ddl, misc).
   - ddl: registra tutte le istruzioni DDL (Data Definition Language) non incluse nella classe `ROLE`.

- function: registra le chiamate di funzione e blocchi `DO`.
- misc: registra vari comandi come `DISCARD`, `FETCH`, `CHECKPOINT`, `VACUUM` e `SET`.
- read: registra `SELECT` e `COPY` quando l'origine è una relazione (ad esempio una tabella) o una query.
- role: registra le istruzioni correlate a ruoli e privilegi, ad esempio `GRANT`, `REVOKE`, `CREATE ROLE`, `ALTER ROLE` e `DROP ROLE`.
- write: registra `INSERT`, `UPDATE`, `DELETE`, `TRUNCATE` e `COPY` quando la destinazione è una relazione (tabella).

Riavvia l'istanza di scrittura del cluster database Aurora PostgreSQL utilizzando il comando AWS CLI seguente.

```
aws rds reboot-db-instance \
    --db-instance-identifier writer-instance \
    --region aws-region
```

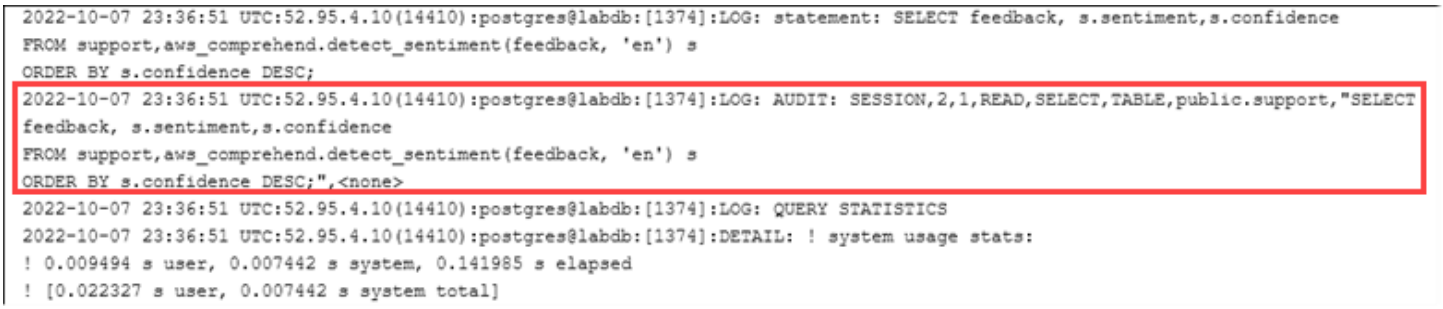
Audit di oggetti di database

Con pgAudit impostata sul cluster database Aurora PostgreSQL e configurata per i requisiti, informazioni più dettagliate vengono acquisite nel registro PostgreSQL. Ad esempio, sebbene la configurazione della registrazione PostgreSQL predefinita consenta di identificare la data e l'ora della modifica apportata a una tabella di database, con l'estensione pgAudit la voce di registro può includere lo schema, l'utente che ha apportato la modifica e altri dettagli a seconda della configurazione dei parametri dell'estensione. È possibile configurare l'audit per tenere traccia delle modifiche nei modi seguenti.

- Per ogni sessione, per utente. Per il livello di sessione, è possibile acquisire il testo del comando completo.
- Per ogni oggetto, per utente e per database.

La funzionalità di audit degli oggetti viene attivata quando il ruolo `rds_pgaudit` viene creato nel sistema e quindi aggiunto al parametro `pgaudit.role` nel gruppo di parametri personalizzati. Per impostazione predefinita, l'impostazione del parametro `pgaudit.role` viene annullata e l'unico valore consentito è `rds_pgaudit`. Nelle fasi seguenti si presume che `pgaudit` sia stato inizializzato

e che l'estensione `pgaudit` sia stata creata seguendo la procedura in Configurazione dell'estensione pgAudit.



Come mostrato in questo esempio, la riga "LOG: AUDIT: SESSION" fornisce informazioni sulla tabella e il relativo schema, insieme ad altri dettagli.

Per configurare l'audit degli oggetti

1. Utilizza `psql` per connetterti all'istanza di scrittura del cluster database Aurora PostgreSQL.

   ```
   psql --host=your-instance-name.aws-region.rds.amazonaws.com --port=5432 --
   username=postgrespostgres --password --dbname=labdb
   ```

2. Crea un ruolo del database denominato `rds_pgaudit` utilizzando il comando seguente.

   ```
   labdb=> CREATE ROLE rds_pgaudit;
   CREATE ROLE
   labdb=>
   ```

3. Chiudi la sessione `psql`.

   ```
   labdb=> \q
   ```

   Nei passaggi successivi, utilizza AWS CLI per modificare i parametri del registro di audit nel gruppo di parametri personalizzati.

4. Utilizza il seguente comando AWS CLI per impostare il parametro `pgaudit.role` su `rds_pgaudit`. Per impostazione predefinita, questo parametro è vuoto e `rds_pgaudit` è l'unico valore consentito.

   ```
   aws rds modify-db-parameter-group \
      --db-parameter-group-name custom-param-group-name \
   ```

```
  --parameters
 "ParameterName=pgaudit.role,ParameterValue=rds_pgaudit,ApplyMethod=pending-reboot"
 \
  --region aws-region
```

5. Utilizza il seguente comando AWS CLI per riavviare l'istanza di scrittura del cluster database Aurora PostgreSQL in modo da rendere effettive le modifiche apportate ai parametri.

```
aws rds reboot-db-instance \
    --db-instance-identifier writer-instance \
    --region aws-region
```

6. Esegui il comando seguente per verificare che `pgaudit.role` sia impostato su `rds_pgaudit`.

```
SHOW pgaudit.role;
pgaudit.role
------------------
rds_pgaudit
```

Per testare la registrazione di pgAudit, è possibile eseguire diversi comandi di esempio da controllare. Ad esempio, si potrebbero eseguire i comandi seguenti.

```
CREATE TABLE t1 (id int);
GRANT SELECT ON t1 TO rds_pgaudit;
SELECT * FROM t1;
id
----
(0 rows)
```

I registri del database devono contenere una voce simile alla seguente:

```
...
2017-06-12 19:09:49 UTC:...:rds_test@postgres:[11701]:LOG: AUDIT:
OBJECT,1,1,READ,SELECT,TABLE,public.t1,select * from t1;
...
```

Per informazioni sulla visualizzazione dei registri, consulta Monitoraggio dei file di log di Amazon Aurora.

Per ulteriori informazioni sull'estensione pgAudit, consulta pgAudit su GitHub.

Esclusione di utenti o database dalla registrazione di audit

Come illustrato in File di log del database Aurora PostgreSQL, i registri di PostgreSQL consumano spazio di archiviazione. L'uso dell'estensione pgAudit consente di aumentare il volume di dati raccolti nei registri in misura diversa, a seconda delle modifiche che vengono monitorate. Potrebbe non essere necessario eseguire l'audit di ogni utente o database nel cluster database Aurora PostgreSQL.

Per ridurre al minimo l'impatto sull'archiviazione ed evitare di acquisire inutilmente i record di audit, è possibile escludere utenti e database dall'audit. È anche possibile modificare la registrazione all'interno di una determinata sessione. Negli esempi seguenti viene mostrato come fare.

> **Note**
>
> Le impostazioni dei parametri a livello di sessione hanno la precedenza sulle impostazioni nel gruppo di parametri cluster database personalizzato per l'istanza di scrittura del cluster database Aurora PostgreSQL. Per evitare che gli utenti del database ignorino le impostazioni di configurazione della registrazione di audit, accertati di modificare le loro autorizzazioni.

Supponi che il cluster database Aurora PostgreSQL sia configurato per eseguire l'audit dello stesso livello di attività per tutti gli utenti e i database. Decidi quindi di non voler eseguire l'audit dell'utente `myuser`. Puoi disattivare l'audit per `myuser` con il comando SQL seguente.

```
ALTER USER myuser SET pgaudit.log TO 'NONE';
```

Quindi, puoi utilizzare la seguente query per controllare la colonna `user_specific_settings` per `pgaudit.log` per verificare che sia impostata su `NONE`.

```
SELECT
    usename AS user_name,
    useconfig AS user_specific_settings
FROM
    pg_user
WHERE
    usename = 'myuser';
```

Viene visualizzato l'output riportato di seguito.

```
 user_name | user_specific_settings
-----------+-----------------------
 myuser    | {pgaudit.log=NONE}
(1 row)
```

Puoi disattivare la registrazione per un determinato utente durante la sessione con il database con il seguente comando.

```
ALTER USER myuser IN DATABASE mydatabase SET pgaudit.log TO 'none';
```

Utilizza la seguente query per controllare la colonna delle impostazioni per pgaudit.log per una combinazione specifica di utente e database.

```
SELECT
    usename AS "user_name",
    datname AS "database_name",
    pg_catalog.array_to_string(setconfig, E'\n') AS "settings"
FROM
    pg_catalog.pg_db_role_setting s
    LEFT JOIN pg_catalog.pg_database d ON d.oid = setdatabase
    LEFT JOIN pg_catalog.pg_user r ON r.usesysid = setrole
WHERE
    usename = 'myuser'
    AND datname = 'mydatabase'
ORDER BY
    1,
    2;
```

L'output visualizzato è simile al seguente.

```
  user_name | database_name |     settings
-----------+---------------+------------------
 myuser    | mydatabase    | pgaudit.log=none
(1 row)
```

Dopo aver disattivato l'audit per `myuser`, decidi di non voler tenere traccia delle modifiche apportate a `mydatabase`. Puoi disattivare l'audit per il database specifico utilizzando il comando seguente.

```
ALTER DATABASE mydatabase SET pgaudit.log to 'NONE';
```

Quindi, utilizza la seguente query per controllare la colonna database_specific_settings per verificare che pgaudit.log sia impostato su NONE.

```
SELECT
a.datname AS database_name,
b.setconfig AS database_specific_settings
FROM
pg_database a
FULL JOIN pg_db_role_setting b ON a.oid = b.setdatabase
WHERE
a.datname = 'mydatabase';
```

Viene visualizzato l'output riportato di seguito.

```
 database_name | database_specific_settings
---------------+----------------------------
 mydatabase    | {pgaudit.log=NONE}
(1 row)
```

Per ripristinare le impostazioni predefinite di myuser, utilizza il comando seguente:

```
ALTER USER myuser RESET pgaudit.log;
```

Per ripristinare i valori predefiniti delle impostazioni di un database, utilizza il comando seguente.

```
ALTER DATABASE mydatabase RESET pgaudit.log;
```

Per ripristinare l'impostazione predefinita di utente e database, utilizza il comando seguente.

```
ALTER USER myuser IN DATABASE mydatabase RESET pgaudit.log;
```

Puoi anche acquisire eventi specifici nel registro impostando `pgaudit.log` su uno degli altri valori consentiti per il parametro `pgaudit.log`. Per ulteriori informazioni, consulta Elenco delle impostazioni consentite per il parametro `pgaudit.log`.

```
ALTER USER myuser SET pgaudit.log TO 'read';
ALTER DATABASE mydatabase SET pgaudit.log TO 'function';
ALTER USER myuser IN DATABASE mydatabase SET pgaudit.log TO 'read,function'
```

Riferimento per l'estensione pgAudit

Puoi specificare il livello di dettaglio desiderato per il registro di audit modificando uno o più dei parametri elencati in questa sezione.

Controllo del comportamento di pgAudit

Puoi controllare la registrazione di audit modificando uno o più dei parametri elencati nella tabella seguente.

Parametro	Descrizione
`pgaudit.log`	Specifica quali classi di istruzioni verranno registrate per registrazione di audit della sessione. I valori consentiti includono ddl, function, misc, read, role, write, none, all. Per ulteriori informazioni, consulta Elenco delle impostazioni consentite per il parametro `pgaudit.log` .
`pgaudit.log_catalog`	Quando è attivato (impostato su 1), aggiunge istruzioni all'audit trail se tutte le relazioni in un'istruzione si trovano in pg_catalog.
`pgaudit.log_level`	Specifica il livello di registro che verrà utilizzato per le voci di registro. Valori consentiti: debug5, debug4, debug3, debug2, debug1, info, notice, warning, log
`pgaudit.log_parameter`	Quando è attivato (impostato su 1), i parametri passati con l'istruzione vengono acquisiti nel registro di audit.
`pgaudit.log_relation`	Quando è attivato (impostato su 1), il registro di audit della sessione crea una voce di registro separata per ogni relazione (TABLE, VIEW e così via) a cui si fa riferimento in un'istruzione SELECT o DML.
`pgaudit.log_statement_once`	Specifica se la registrazione includerà il testo dell'istruzione e i parametri con la prima voce di registro per una combinazione istruzione/istruzione secondaria o con ogni voce.
`pgaudit.role`	Specifica il ruolo principale da utilizzare per la registrazione di verifica degli oggetti. L'unica voce consentita è `rds_pgaudit` .

Elenco delle impostazioni consentite per il parametro **`pgaudit.log`**

Valore	Descrizione
nessuno	Questa è l'impostazione predefinita. Non viene registrata alcuna modifica al database.
tutto	Registra tutto (read, write, function, role, ddl, misc).
ddl	Registra tutte le istruzioni DDL (Data Definition Language) non incluse nella classe `ROLE`.
funzione	Registra le chiamate di funzione e i blocchi `DO`.
misc	Registra vari comandi, ad esempio `DISCARD`, `FETCH`, `CHECKPOINT` , `VACUUM` e `SET`.
read	Registra `SELECT` e `COPY` quando l'origine è una relazione (ad esempio una tabella) o una query.
role	Registra le istruzioni relative a ruoli e privilegi, ad esempio `GRANT`, `REVOKE`, `CREATE ROLE`, `ALTER ROLE` e `DROP ROLE`.
write	Registra `INSERT`, `UPDATE`, `DELETE`, `TRUNCATE` e `COPY` quando la destinazione è una relazione (tabella).

Per registrare più tipi di eventi con controllo di sessioni, utilizzare un elenco separato da virgole. Per registrare tutti i tipi di eventi, impostare `pgaudit.log` su `ALL`. Riavviare l'istanza database per applicare le modifiche.

Con il controllo dell'oggetto, è possibile perfezionare la registrazione di controllo per lavorare con relazioni specifiche. Ad esempio, è possibile richiedere la registrazione di audit per le operazioni `READ` su una o più tabelle.

Utilizzo di pglogical per sincronizzare i dati tra le istanze

Tutte le versioni di Aurora PostgreSQL attualmente disponibili supportano l'estensione `pglogical`, che precede la funzionalità di replica logica funzionalmente simile introdotta nella versione 10 di

PostgreSQL. Per ulteriori informazioni, consulta Panoramica della replica logica di PostgreSQL con Aurora.

L'estensione `pglogical` supporta la replica logica tra due o più cluster database Aurora PostgreSQL. Supporta anche la replica tra diverse versioni di PostgreSQL e tra database in esecuzione in istanze database RDS per PostgreSQL e cluster database Aurora PostgreSQL. L'estensione `pglogical` utilizza un modello publish-subscribe per replicare le modifiche apportate alle tabelle e ad altri oggetti, come le sequenze, da un publisher in un subscriber. Si basa su uno slot di replica per garantire la sincronizzazione delle modifiche da un nodo publisher a un nodo subscriber, definiti come indicato di seguito.

- Il nodo publisher è il cluster database Aurora PostgreSQL che costituisce l'origine dei dati da replicare in altri nodi. Il nodo publisher definisce le tabelle da replicare in un set di pubblicazione.
- Il nodo subscriber è il cluster database Aurora PostgreSQL che riceve gli aggiornamenti WAL dal publisher. Il subscriber crea una sottoscrizione per connettersi al publisher e ottenere i dati WAL decodificati e contemporaneamente nel nodo publisher viene creato lo slot di replica.

Di seguito sono riportati gli argomenti sull'impostazione dell'estensione `pglogical`.

Argomenti

- Requisiti e limitazioni dell'estensione pglogical
- Impostazione dell'estensione pglogical
- Impostazione della replica logica per il cluster database Aurora PostgreSQL
- Riconnessione della replica logica dopo un aggiornamento principale
- Gestione degli slot di replica logica per Aurora PostgreSQL
- Riferimento sui parametri dell'estensione pglogical

Requisiti e limitazioni dell'estensione pglogical

Tutte le versioni attualmente disponibili di Aurora PostgreSQL supportano l'estensione `pglogical`.

Sia il nodo publisher che il nodo subscriber devono essere impostati per la replica logica.

Le tabelle da replicare da un nodo publisher a un nodo subscriber devono avere gli stessi nomi e lo stesso schema. Inoltre devono contenere le stesse colonne e le colonne devono utilizzare gli stessi tipi di dati. Le tabelle del publisher e del subscriber devono avere le stesse chiavi primarie. Si consiglia di utilizzare solo la CHIAVE PRIMARIA come vincolo univoco.

Le tabelle del nodo subscriber possono avere vincoli più permissivi rispetto ai vincoli CHECK e NOT NULL delle tabelle del nodo publisher.

L'estensione `pglogical` fornisce funzionalità, come la replica bidirezionale, che non sono supportate dalla funzionalità di replica logica integrata in PostgreSQL 10 e versioni successive. Per ulteriori informazioni, consulta PostgreSQL bi-directional replication using pglogical (Replica bidirezionale di PostgreSQL utilizzando pglogical).

Impostazione dell'estensione pglogical

Per impostare l'estensione `pglogical` per il cluster database Aurora PostgreSQL, aggiungi `pglogical` alle librerie condivise nel gruppo di parametri del·cluster database personalizzato per il cluster database Aurora PostgreSQL. È inoltre necessario impostare il valore del parametro `rds.logical_replication` su `1` per attivare la decodifica logica. Infine, crei l'estensione nel database. Per queste attività puoi utilizzare la Console di gestione AWS o AWS CLI.

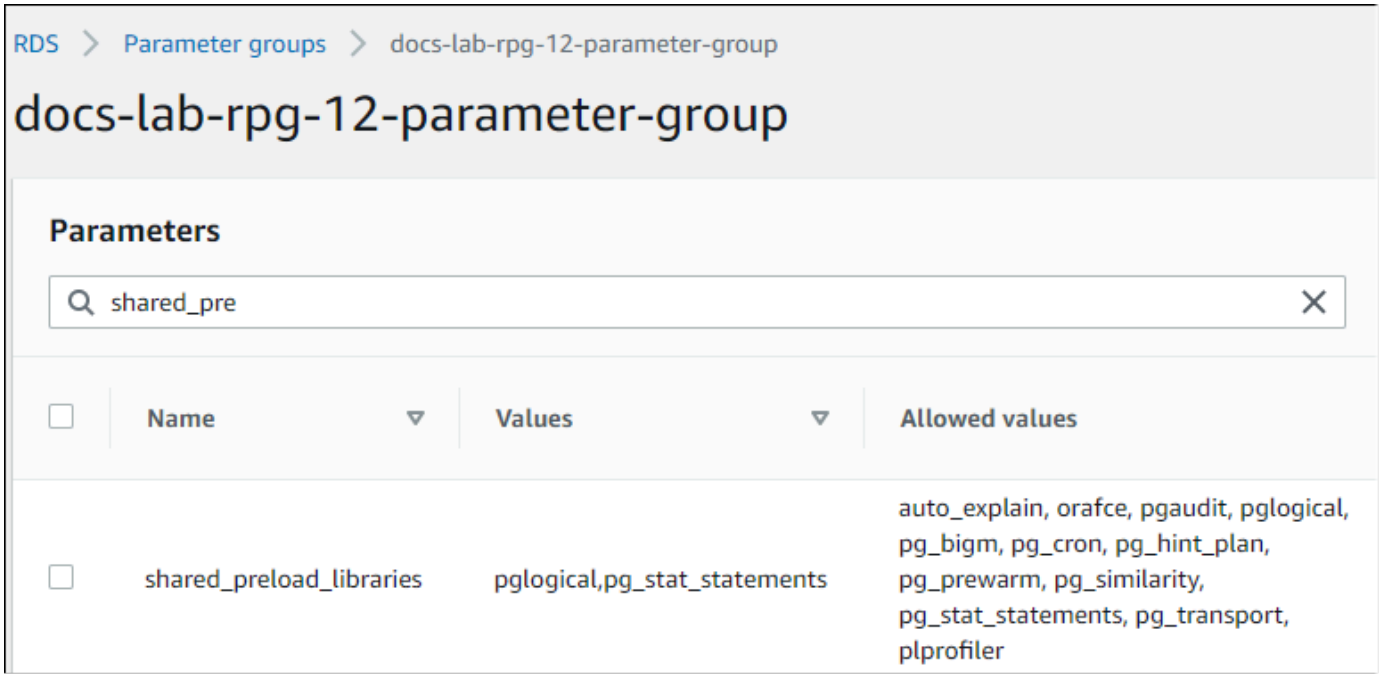
Per eseguire queste attività sono richieste le autorizzazioni del ruolo `rds_superuser`.

Le fasi seguenti si basano sull'ipotesi che il cluster database Aurora PostgreSQL sia associato a un gruppo di parametri cluster di database personalizzato. Per informazioni sulla creazione di un gruppo di parametri del·cluster database, consulta Gruppi di parametri per Amazon Aurora.

Console

Per impostare l'estensione pglogical

1. Accedi alla Console di gestione AWS e apri la console Amazon RDS all'indirizzo https://console.aws.amazon.com/rds/.
2. Nel riquadro di navigazione, scegli l'istanza di scrittura del cluster database Aurora PostgreSQL .
3. Apri la scheda Configurazione per l'istanza di scrittura del cluster database Aurora PostgreSQL. Tra i dettagli dell'istanza, individua il collegamento Parameter group (Gruppo di parametri).
4. Scegli il collegamento per aprire i parametri personalizzati associati al cluster database Aurora PostgreSQL.
5. Nel campo di ricerca Parametri, digita `shared_pre` per trovare il parametro `shared_preload_libraries`.
6. Scegli Edit parameters (Modifica parametri) per accedere ai valori delle proprietà.
7. Aggiungi `pglogical` all'elenco nel campo Values (Valori). Utilizza una virgola per separare gli elementi nell'elenco di valori.



8. Individua il parametro `rds.logical_replication` e impostalo su 1 per attivare la replica logica.
9. Riavvia l'istanza di scrittura del cluster database Aurora PostgreSQL per rendere effettive le modifiche.
10. Quando l'istanza è disponibile, puoi utilizzare `psql` (o pgAdmin) per connetterti all'istanza di scrittura del cluster database Aurora PostgreSQL.

```
psql --host=111122223333.aws-region.rds.amazonaws.com --port=5432 --
username=postgres --password --dbname=labdb
```

11. Per verificare che l'estensione pglogical sia inizializzata, esegui il seguente comando.

```
SHOW shared_preload_libraries;
shared_preload_libraries
--------------------------
rdsutils,pglogical
(1 row)
```

12. Verifica l'impostazione che abilita la decodifica logica, come indicato di seguito.

```
SHOW wal_level;
wal_level
-----------
 logical
```

```
(1 row)
```

13. Crea l'estensione, come indicato di seguito.

```
CREATE EXTENSION pglogical;
EXTENSION CREATED
```

14. Scegli Save changes (Salva modifiche).
15. Apri la console di Amazon RDS all'indirizzo https://console.aws.amazon.com/rds/.
16. Scegli l'istanza di scrittura del cluster database Aurora PostgreSQL dall'elenco di database per selezionarla, quindi scegli Reboot (Riavvia) dal menu Actions (Operazioni).

AWS CLI

Per impostare l'estensione pglogical

Per impostare l'estensione pglogical utilizzando AWS CLI, chiami l'operazione modify-db-parameter-group per modificare i parametri nel gruppo di parametri personalizzato, come illustrato nella procedura seguente.

1. Utilizza il seguente comando AWS CLI per aggiungere `pglogical` al parametro `shared_preload_libraries`.

```
aws rds modify-db-parameter-group \
   --db-parameter-group-name custom-param-group-name \
   --parameters
 "ParameterName=shared_preload_libraries,ParameterValue=pglogical,ApplyMethod=pending-
reboot" \
   --region aws-region
```

2. Utilizza il seguente comando AWS CLI per impostare `rds.logical_replication` su 1 per attivare la funzionalità di decodifica logica per l'istanza di scrittura del cluster database Aurora PostgreSQL.

```
aws rds modify-db-parameter-group \
   --db-parameter-group-name custom-param-group-name \
   --parameters
 "ParameterName=rds.logical_replication,ParameterValue=1,ApplyMethod=pending-
reboot" \
   --region aws-region
```

3. Utilizza il seguente comando AWS CLI per riavviare l'istanza di scrittura del cluster database Aurora PostgreSQL in modo che la libreria pglogical venga inizializzata.

```
aws rds reboot-db-instance \
    --db-instance-identifier writer-instance \
    --region aws-region
```

4. Quando l'istanza è disponibile, utilizza `psql` per connetterti all'istanza di scrittura del cluster database Aurora PostgreSQL.

```
psql --host=111122223333.aws-region.rds.amazonaws.com --port=5432 --
username=postgres --password --dbname=labdb
```

5. Crea l'estensione, come indicato di seguito.

```
CREATE EXTENSION pglogical;
EXTENSION CREATED
```

6. Riavvia l'istanza di scrittura del cluster database Aurora PostgreSQL utilizzando il comando AWS CLI seguente.

```
aws rds reboot-db-instance \
    --db-instance-identifier writer-instance \
    --region aws-region
```

Impostazione della replica logica per il cluster database Aurora PostgreSQL

La seguente procedura mostra come avviare la replica logica tra due cluster database Aurora PostgreSQL. I passaggi presuppongono che sia l'origine (publisher) che la destinazione (subscriber) abbiano l'estensione `pglogical` impostata come descritto dettagliatamente in Impostazione dell'estensione pglogical.

> **Note**
>
> `node_name` di un nodo subscriber non può iniziare con `rds`.

Per creare il nodo publisher e definire le tabelle da replicare

Questi passaggi presuppongono che il cluster database Aurora PostgreSQL abbia un'istanza di scrittura con un database contenente una o più tabelle che desideri replicare in un altro nodo. È necessario ricreare la struttura delle tabelle dal publisher nel subscriber, quindi prima, se occorre, recupera la struttura delle tabelle. Puoi farlo utilizzando il metacomando `psql \d` *`tablename`* e quindi creando la stessa tabella nell'istanza subscriber. Nella procedura seguente viene illustrato come creare una tabella di esempio nel publisher (origine) a scopo dimostrativo.

1. Utilizza `psql` per connetterti all'istanza che include la tabella da usare come origine per i subscriber.

   ```
   psql --host=source-instance.aws-region.rds.amazonaws.com --port=5432 --
   username=postgres --password --dbname=labdb
   ```

   Se non hai una tabella esistente da replicare, puoi creare una tabella di esempio come indicato di seguito.

   a. Crea una tabella di esempio utilizzando la seguente istruzione SQL.

      ```
      CREATE TABLE docs_lab_table (a int PRIMARY KEY);
      ```

   b. Popola la tabella con i dati generati utilizzando la seguente istruzione SQL.

      ```
      INSERT INTO docs_lab_table VALUES (generate_series(1,5000));
      INSERT 0 5000
      ```

   c. Verifica che i dati siano presenti nella tabella utilizzando la seguente istruzione SQL.

      ```
      SELECT count(*) FROM docs_lab_table;
      ```

2. Identifica il cluster database Aurora PostgreSQL come nodo publisher, come indicato di seguito.

   ```
   SELECT pglogical.create_node(
       node_name := 'docs_lab_provider',
       dsn := 'host=source-instance.aws-region.rds.amazonaws.com port=5432
    dbname=labdb');
    create_node
   -------------
      3410995529
   ```

```
(1 row)
```

3. Aggiungi la tabella da replicare al set di replica predefinito. Per ulteriori informazioni sui set di replica, consulta Replication sets (Set di replica) nella documentazione di pglogical.

```
SELECT pglogical.replication_set_add_table('default', 'docs_lab_table', 'true',
 NULL, NULL);
 replication_set_add_table
  ---------------------------
  t
  (1 row)
```

L'impostazione del nodo publisher è completata. Ora puoi impostare il nodo subscriber per ricevere gli aggiornamenti dal publisher.

Per impostare il nodo subscriber e creare una sottoscrizione per ricevere gli aggiornamenti

Questi passaggi presuppongono che sia stata eseguita l'impostazione del cluster database Aurora PostgreSQL con l'estensione `pglogical`. Per ulteriori informazioni, consulta Impostazione dell'estensione pglogical.

1. Utilizza `psql` per connetterti all'istanza per cui vuoi ricevere gli aggiornamenti dal publisher.

```
psql --host=target-instance.aws-region.rds.amazonaws.com --port=5432 --
username=postgres --password --dbname=labdb
```

2. Nel cluster database Aurora PostgreSQL del subscriber crea la stessa tabella presente nel publisher. In questo esempio, la tabella è `docs_lab_table`. È possibile creare la tabella come indicato di seguito.

```
CREATE TABLE docs_lab_table (a int PRIMARY KEY);
```

3. Verifica che questa tabella sia vuota.

```
SELECT count(*) FROM docs_lab_table;
 count
-------
  0
(1 row)
```

4. Identifica il cluster database Aurora PostgreSQL come nodo subscriber, come indicato di seguito.

```
SELECT pglogical.create_node(
    node_name := 'docs_lab_target',
    dsn := 'host=target-instance.aws-region.rds.amazonaws.com port=5432
 sslmode=require dbname=labdb user=postgres password=********');
 create_node
-------------
   2182738256
(1 row)
```

5. Crea la sottoscrizione.

```
SELECT pglogical.create_subscription(
   subscription_name := 'docs_lab_subscription',
   provider_dsn := 'host=source-instance.aws-region.rds.amazonaws.com port=5432
 sslmode=require dbname=labdb user=postgres password=*******',
   replication_sets := ARRAY['default'],
   synchronize_data := true,
   forward_origins := '{}' );
 create_subscription
---------------------
1038357190
(1 row)
```

Una volta completato questo passaggio, i dati della tabella del publisher vengono creati nella tabella del subscriber. È possibile verificare questa operazione utilizzando la seguente query SQL.

```
SELECT count(*) FROM docs_lab_table;
 count
-------
  5000
(1 row)
```

Da questo momento in poi, le modifiche apportate alla tabella del publisher vengono replicate nella tabella del subscriber.

Riconnessione della replica logica dopo un aggiornamento principale

Prima di poter eseguire un aggiornamento della versione principale di un cluster database Aurora PostgreSQL impostata come nodo publisher per la replica logica, è necessario rimuovere tutti gli slot

di replica, anche quelli non attivi. Si consiglia di deviare temporaneamente le transazioni del database dal nodo publisher, rimuovere gli slot di replica, aggiornare il cluster database Aurora PostgreSQL e quindi riconnettere e riavviare la replica.

Gli slot di replica sono ospitati solo nel nodo publisher. Il nodo subscriber Aurora PostgreSQL in uno scenario di replica logica non ha slot da rimuovere. Il processo di aggiornamento alla versione principale di Aurora PostgreSQL supporta l'aggiornamento del subscriber a una nuova versione principale di PostgreSQL indipendente dal nodo publisher. Tuttavia, il processo di aggiornamento interrompe il processo di replica e interferisce con la sincronizzazione dei dati WAL tra il nodo publisher e il nodo subscriber. È necessario riconnettere la replica logica tra publisher e subscriber dopo aver aggiornato il publisher, il subscriber o entrambi. Nella procedura seguente viene illustrato come determinare se la replica è stata interrotta e come risolvere il problema.

Determinazione della replica logica interrotta

È possibile determinare che il processo di replica è stato interrotto eseguendo una query sul nodo publisher o sul nodo subscriber, come indicato di seguito.

Per controllare il nodo publisher

- Utilizza `psql` per connetterti al nodo publisher e quindi esegui la query sulla funzione `pg_replication_slots`. Osserva il valore nella colonna attiva. Normalmente, la query restituisce `t` (true) per indicare che la replica è attiva. Se restituisce `f` (false), indica che la replica nel subscriber è stata interrotta.

```
SELECT slot_name,plugin,slot_type,active FROM pg_replication_slots;
                    slot_name              |      plugin      | slot_type | active
-------------------------------------------+------------------+-----------+--------
 pgl_labdb_docs_labcb4fa94_docs_lab3de412c | pglogical_output | logical   | f
(1 row)
```

Per controllare il nodo subscriber

Nel nodo subscriber è possibile verificare lo stato della replica in tre modi diversi.

- Esamina i log di PostgreSQL sul nodo subscriber per trovare i messaggi di errore. Il log identifica gli errori nei messaggi che includono il codice di uscita 1, come illustrato di seguito.

```
2022-07-06 16:17:03 UTC::@:[7361]:LOG: background worker "pglogical apply
 16404:2880255011" (PID 14610) exited with exit code 1
```

```
2022-07-06 16:19:44 UTC::@:[7361]:LOG: background worker "pglogical apply
 16404:2880255011" (PID 21783) exited with exit code 1
```

- Esegui la query sulla funzione pg_replication_origin. Connettiti al database sul nodo subscriber utilizzando psql ed esegui la query sulla funzione pg_replication_origin, come indicato di seguito.

```
SELECT * FROM pg_replication_origin;
 roident | roname
---------+--------
(0 rows)
```

  Se il set di risultati è vuoto, la replica è stata interrotta. Normalmente, viene restituito l'output riportato di seguito.

```
   roident |                       roname
  ---------+-------------------------------------------------
         1 | pgl_labdb_docs_labcb4fa94_docs_lab3de412c
  (1 row)
```

- Esegui la query sulla funzione pglogical.show_subscription_status come illustrato nell'esempio seguente.

```
SELECT subscription_name,status,slot_name FROM pglogical.show_subscription_status();
     subscription_name | status |              slot_name
---====----------------+--------+-----------------------------------
 docs_lab_subscription | down   | pgl_labdb_docs_labcb4fa94_docs_lab3de412c
(1 row)
```

  Questo output mostra che la replica è stata interrotta. Lo stato è down e normalmente l'output mostra lo stato replicating.

Se il processo di replica logica è stato interrotto, è possibile riconnettere la replica seguendo questi passaggi.

Per riconnettere la replica logica tra i nodi publisher e subscriber

Per riconnettere la replica, devi prima disconnettere il subscriber dal nodo publisher e quindi riconnettere la sottoscrizione, come descritto in questi passaggi.

1. Connettiti al nodo subscriber utilizzando `psql`, come indicato di seguito.

```
psql --host=222222222222.aws-region.rds.amazonaws.com --port=5432 --
username=postgres --password --dbname=labdb
```

2. Disattiva la sottoscrizione utilizzando la funzione `pglogical.alter_subscription_disable`.

```
SELECT pglogical.alter_subscription_disable('docs_lab_subscription',true);
 alter_subscription_disable
----------------------------
 t
(1 row)
```

3. Ottieni l'identificatore del nodo publisher eseguendo la query su `pg_replication_origin`, come indicato di seguito.

```
SELECT * FROM pg_replication_origin;
 roident |               roname
---------+-------------------------------------
       1 | pgl_labdb_docs_labcb4fa94_docs_lab3de412c
(1 row)
```

4. Utilizza la risposta restituita dal passaggio precedente con il comando `pg_replication_origin_create` per assegnare l'identificatore che può essere usato dalla sottoscrizione una volta riconnessa.

```
SELECT pg_replication_origin_create('pgl_labdb_docs_labcb4fa94_docs_lab3de412c');
  pg_replication_origin_create
------------------------------
                             1
(1 row)
```

5. Attiva la sottoscrizione specificando il nome con lo stato `true`, come illustrato nell'esempio seguente.

```
SELECT pglogical.alter_subscription_enable('docs_lab_subscription',true);
  alter_subscription_enable
---------------------------
 t
(1 row)
```

Controlla lo stato del nodo. Lo stato deve essere `replicating` come mostrato nell'esempio.

```
SELECT subscription_name,status,slot_name
  FROM pglogical.show_subscription_status();
            subscription_name |   status    |              slot_name
------------------------------+-------------+-----------------------------------
 docs_lab_subscription         | replicating |
 pgl_labdb_docs_lab98f517b_docs_lab3de412c
(1 row)
```

Verifica lo stato dello slot di replica del subscriber sul nodo publisher. La colonna `active` dello slot deve restituire `t` (true) per indicare che la replica è stata riconnessa.

```
SELECT slot_name,plugin,slot_type,active
  FROM pg_replication_slots;
                    slot_name              |      plugin      | slot_type | active
-------------------------------------------+------------------+-----------+--------
 pgl_labdb_docs_lab98f517b_docs_lab3de412c | pglogical_output | logical   | t
(1 row)
```

Gestione degli slot di replica logica per Aurora PostgreSQL

Prima di poter eseguire un aggiornamento alla versione principale su un'istanza di scrittura del cluster database Aurora PostgreSQL che funge da nodo publisher in uno scenario di replica logica, è necessario rimuovere gli slot di replica nell'istanza. Il processo di verifica preliminare dell'aggiornamento alla versione principale avvisa che l'aggiornamento non può procedere fino a quando gli slot non vengono rimossi.

Per identificare gli slot di replica creati utilizzando l'estensione `pglogical`, accedi a ciascun database e recupera il nome dei nodi. Quando esegui la query sul nodo subscriber, nell'output viene restituito sia il nodo publisher che il nodo subscriber, come mostrato nell'esempio seguente.

```
SELECT * FROM pglogical.node;
node_id   |     node_name
------------+-------------------
 2182738256 | docs_lab_target
 3410995529 | docs_lab_provider
(2 rows)
```

Puoi ottenere i dettagli sulla sottoscrizione con la seguente query.

```
SELECT sub_name,sub_slot_name,sub_target
  FROM pglogical.subscription;
 sub_name |         sub_slot_name          | sub_target
----------+--------------------------------+------------
  docs_lab_subscription     | pgl_labdb_docs_labcb4fa94_docs_lab3de412c | 2182738256
(1 row)
```

A questo punto puoi rimuovere la sottoscrizione, come indicato di seguito.

```
SELECT pglogical.drop_subscription(subscription_name := 'docs_lab_subscription');
 drop_subscription
-------------------
                 1
(1 row)
```

Dopo aver rimosso la sottoscrizione, puoi eliminare il nodo.

```
SELECT pglogical.drop_node(node_name := 'docs-lab-subscriber');
 drop_node
-----------
 t
(1 row)
```

Puoi verificare che il nodo sia stato eliminato, come indicato di seguito.

```
SELECT * FROM pglogical.node;
 node_id | node_name
---------+-----------
(0 rows)
```

Riferimento sui parametri dell'estensione pglogical

Nella tabella sono illustrati i parametri associati all'estensione `pglogical`. Parametri come `pglogical.conflict_log_level` e `pglogical.conflict_resolution` vengono utilizzati per gestire i conflitti di aggiornamento. I conflitti possono emergere quando vengono apportate modifiche localmente alle stesse tabelle che hanno una sottoscrizione con il publisher. I conflitti possono verificarsi anche in altri scenari, ad esempio la replica bidirezionale o quando più subscriber eseguono la replica dallo stesso publisher. Per ulteriori informazioni, consulta PostgreSQL bi-directional replication using pglogical (Replica bidirezionale di PostgreSQL utilizzando pglogical).

Parametro	Descrizione
pglogical.batch_inserts	Esegue inserimenti batch, se possibile. Non impostato per impostazione predefinita. Imposta "1" per attivarlo, "0" per disattivarlo.
pglogical.conflict_log_level	Imposta il livello di log da utilizzare per la registrazione dei conflitti risolti. I valori di stringa supportati sono debug5, debug4, debug3, debug2, debug1, info, notice, warning, error, log, fatal, panic.
pglogical.conflict_resolution	Imposta il metodo da utilizzare per risolvere i conflitti quando sono risolvibili. I valori di stringa supportati sono error, apply_remote, keep_local, last_update_wins, first_update_wins.
pglogical.extra_connection_options	Specifica le opzioni di connessione da aggiungere a tutte le connessioni dei nodi peer.
pglogical.synchronous_commit	Valore di commit sincrono specifico pglogical
pglogical.use_spi	Utilizza la SPI (Server Programming Interface) invece dell'API di basso livello per applicare le modifiche. Imposta "1" per attivarlo , "0" per disattivarlo. Per ulteriori informazioni sulla SPI, consulta Server Programming Interface nella documentazione PostgreSQL.

Utilizzo dei wrapper di dati esterni supportati per Amazon Aurora PostgreSQL

Un wrapper di dati esterni (FDW) è uno specifico tipo di estensione che consente l'accesso a dati esterni. Ad esempio, l'estensione `oracle_fdw` consente all'istanza database Aurora PostgreSQL di interagire con i database Oracle.

Di seguito sono disponibili le informazioni sui diversi wrapper di dati esterni di PostgreSQL supportati.

Argomenti

- Utilizzo dell'estensione log_fdw per accedere al registro di database utilizzando SQL

- Utilizzo dell'estensione postgres_fw per accedere a dati esterni
- Interazione con i database MySQL utilizzando l'estensione mysql_fdw
- Interazione con un database Oracle utilizzando l'estensione oracle_fdw
- Interazione con i database MySQL utilizzando l'estensione mysql_fdw

Utilizzo dell'estensione log_fdw per accedere al registro di database utilizzando SQL

Il cluster di database Aurora PostegreSQL supporta l'estensione `log_fdw` che consente di accedere al log del motore del database utilizzando un'interfaccia SQL. L'estensione `log_fdw` offre due nuove funzioni che semplificano la creazione di tabelle esterne per i registri di database:

- `list_postgres_log_files` – Elenca i file nella directory dei registri di database e le dimensioni dei file in byte.
- `create_foreign_table_for_log_file(table_name text, server_name text, log_file_name text)` – Crea una tabella esterna per il file specificato nel database corrente.

Tutte le funzioni create da `log_fdw` sono di proprietà di `rds_superuser`. I membri del ruolo `rds_superuser` possono concedere l'accesso a queste funzioni ad altri utenti del database.

Per impostazione predefinita, i file di log vengono generati da Amazon Aurora nel formato `stderr` (errore standard), come specificato nel parametro `log_destination`. Esistono solo due opzioni per questo parametro, `stderr` e `csvlog` (valori separati da virgola, CSV). Se aggiungi l'opzione `csvlog` al parametro, Amazon Aurora genera entrambi i log `stderr` e `csvlog`. Ciò può influire sulla capacità di archiviazione del cluster di database, quindi è necessario tenere conto degli altri parametri che influiscono sulla gestione dei log. Per ulteriori informazioni, consulta Impostazione della destinazione del registro (`stderr`, `csvlog`).

Un vantaggio della generazione dei registri `csvlog` è che l'estensione `log_fdw` consente di costruire le tabelle esterne con i dati suddivisi in diverse colonne. Per eseguire questa operazione, l'istanza deve essere associata a un gruppo parametri del database personalizzato in modo da poter modificare l'impostazione per `log_destination`. Per ulteriori informazioni su come fare, consulta Gruppi di parametri per Amazon Aurora.

L'esempio seguente presuppone che il parametro `log_destination` includa `cvslog`.

Per utilizzare l'estensione log_fdw

1. Installa l'estensione `log_fdw`.

```
postgres=> CREATE EXTENSION log_fdw;
CREATE EXTENSION
```

2. Creare un server log come wrapper di dati esterno.

```
postgres=> CREATE SERVER log_server FOREIGN DATA WRAPPER log_fdw;
CREATE SERVER
```

3. Selezionare tutti gli elementi da un elenco di file di registro.

```
postgres=> SELECT * FROM list_postgres_log_files() ORDER BY 1;
```

Di seguito è riportata una risposta di esempio.

```
          file_name           | file_size_bytes
------------------------------+-----------------
 postgresql.log.2023-08-09-22.csv |            1111
 postgresql.log.2023-08-09-23.csv |            1172
 postgresql.log.2023-08-10-00.csv |            1744
 postgresql.log.2023-08-10-01.csv |            1102
(4 rows)
```

4. Creare una tabella con una singola colonna "log_entry" per i file selezionato.

```
postgres=> SELECT create_foreign_table_for_log_file('my_postgres_error_log',
     'log_server', 'postgresql.log.2023-08-09-22.csv');
```

La risposta non fornisce dettagli diversi da quello che la tabella ora esiste.

```
----------------------------------
(1 row)
```

5. Selezionare un campione del file di registro. Il seguente codice recupera l'ora del log e la descrizione del messaggio di errore.

```
postgres=> SELECT log_time, message FROM my_postgres_error_log ORDER BY 1;
```

Di seguito è riportata una risposta di esempio.

```
             log_time             |                                 message
----------------------------------
+-------------------------------------------------------------------------------
Tue Aug 09 15:45:18.172 2023 PDT | ending log output to stderr
Tue Aug 09 15:45:18.175 2023 PDT | database system was interrupted; last known up
 at 2023-08-09 22:43:34 UTC
Tue Aug 09 15:45:18.223 2023 PDT | checkpoint record is at 0/90002E0
Tue Aug 09 15:45:18.223 2023 PDT | redo record is at 0/90002A8; shutdown FALSE
Tue Aug 09 15:45:18.223 2023 PDT | next transaction ID: 0/1879; next OID: 24578
Tue Aug 09 15:45:18.223 2023 PDT | next MultiXactId: 1; next MultiXactOffset: 0
Tue Aug 09 15:45:18.223 2023 PDT | oldest unfrozen transaction ID: 1822, in
 database 1
(7 rows)
```

Utilizzo dell'estensione postgres_fdw per accedere a dati esterni

È possibile accedere ai dati in una tabella su un server di database remoto con l'estensione postgres_fdw. Se si imposta una connessione remota dall'istanza database di PostgreSQL, l'accesso è disponibile anche alla replica di lettura.

Usare postgres_fdw per accedere al server remoto del database

1. Installare l'estensione postgres_fdw.

```
CREATE EXTENSION postgres_fdw;
```

2. Creare un server di dati esterni utilizzando CREATE SERVER.

```
CREATE SERVER foreign_server
FOREIGN DATA WRAPPER postgres_fdw
OPTIONS (host 'xxx.xx.xxx.xx', port '5432', dbname 'foreign_db');
```

3. Creare una mappatura dell'utente per identificare il ruolo da utilizzare sul server remoto.

> **Important**
>
> Per oscurare la password in modo che non compaia nei log, impostare `log_statement=none` a livello di sessione. L'impostazione a livello di parametro non comporta l'oscuramento della password.

```
CREATE USER MAPPING FOR local_user
SERVER foreign_server
OPTIONS (user 'foreign_user', password 'password');
```

4. Creare una tabella che esegua la mappatura della tabella sul server remoto.

```
CREATE FOREIGN TABLE foreign_table (
        id integer NOT NULL,
        data text)
SERVER foreign_server
OPTIONS (schema_name 'some_schema', table_name 'some_table');
```

Interazione con i database MySQL utilizzando l'estensione mysql_fdw

Per accedere a un database compatibile con MySQL dal cluster di database Aurora PostgreSQL è possibile installare e utilizzare l'estensione `mysql_fdw`. Questo wrapper di dati esterni consente di interagire con RDS per MySQL, Aurora MySQL, MariaDB e altri database compatibili con MySQL. La connessione dal cluster di database Aurora PostgreSQL al database MySQL è crittografata in base al miglior tentativo a seconda delle configurazioni di client e server. Tuttavia, se lo si desidera, è possibile imporre l'utilizzo della crittografia. Per ulteriori informazioni, consulta Utilizzo della crittografia in transito con l'estensione.

L'estensione `mysql_fdw` è supportata su Amazon Aurora PostgreSQL 15.4, 14.9, 13.12, 12.16, e versioni successive. Supporta le operazioni di select, insert, update e delete da un database RDS for PostgreSQL su tabelle contenuto in un'istanza database compatibile con MySQL.

Argomenti

- Configurazione del database Aurora PostgreSQL per l'utilizzo dell'estensione mysql_fdw
- Esempio: utilizzo di un database Aurora MySQL da Aurora PostgreSQL
- Utilizzo della crittografia in transito con l'estensione

Configurazione del database Aurora PostgreSQL per l'utilizzo dell'estensione mysql_fdw

La configurazione dell'estensione `mysql_fdw` sul cluster di database Aurora PostgreSQL comporta il caricamento dell'estensione nel cluster di database e quindi la creazione del punto di connessione

all'istanza database MySQL. Per tale attività, è necessario disporre delle seguenti informazioni sull'istanza database MySQL:

- Nome host o endpoint. Per trovare l'endpoint di un cluster di database Aurora MySQL è possibile utilizzare la console. Scegliere la scheda Connectivity & security (Connettività e sicurezza) e cercare nella sezione Endpoint and port (Endpoint e porta).
- Numero della porta. La porta di default per MySQL è 3306.
- Nome del database. L'identificatore del database.

È inoltre necessario fornire l'accesso al gruppo di sicurezza o alla lista di controllo degli accessi (ACL) per la porta MySQL 3306. Il cluster di database Aurora PostegreSQL e il cluster di database Aurora MySQL necessitano dell'accesso alla porta 3306. Se l'accesso non è configurato correttamente, quando si cerca di connettersi alla tabella compatibile con MySQL comparirà un messaggio di errore simile al seguente:

```
ERROR: failed to connect to MySQL: Can't connect to MySQL server on 'hostname.aws-
region.rds.amazonaws.com:3306' (110)
```

Nella seguente procedura, l'utente (utilizzando l'account `rds_superuser`) crea il server esterno. Quindi concede l'accesso al server esterno a specifici utenti. Questi utenti creano quindi i propri mapping agli account utente MySQL appropriati per interagire con l'istanza database MySQL.

Per utilizzare mysql_fdw per accedere al server database MySQL

1. Effettuare la connessione all'istanza database PostgreSQL utilizzando un account che dispone del ruolo `rds_superuser`. Se al momento della creazione del cluster di database Aurora PostgreSQL sono stati accettati i valori predefiniti, il nome utente è `postgres` e lo strumento a riga di comando `psql` può essere usato per collegarsi come segue:

   ```
   psql --host=your-DB-instance.aws-region.rds.amazonaws.com --port=5432 --
   username=postgres --password
   ```

2. Installare l'estensione `mysql_fdw` come segue:

   ```
   postgres=> CREATE EXTENSION mysql_fdw;
   CREATE EXTENSION
   ```

Dopo aver installato l'estensione sul cluster di database Aurora PostgreSQL imposta il server esterno che fornisce la connessione a un database MySQL.

Per creare il server esterno

Esegui queste attività sul cluster di database Aurora PostgreSQL. La procedura presuppone che l'utente sia connesso come utente con i privilegi di `rds_superuser`, come `postgres`.

1. Creazione di un server esterno nel cluster di database Aurora PostgreSQL:

```
postgres=> CREATE SERVER mysql-db FOREIGN DATA WRAPPER mysql_fdw OPTIONS (host 'db-name.111122223333.aws-region.rds.amazonaws.com', port '3306');
CREATE SERVER
```

2. Concedere agli utenti appropriati l'accesso al server esterno. Questi dovrebbero essere utenti non amministratori, cioè utenti senza il ruolo `rds_superuser`.

```
postgres=> GRANT USAGE ON FOREIGN SERVER mysql-db to user1;
GRANT
```

Gli utenti PostgreSQL creano e gestiscono le proprie connessioni al database MySQL tramite il server esterno.

Esempio: utilizzo di un database Aurora MySQL da Aurora PostgreSQL

Supponi di disporre di una semplice tabella su un'istanza database Aurora PostgreSQL. Gli utenti di Aurora PostgreSQL desiderano eseguire query sugli elementi (`SELECT`), `INSERT`, `UPDATE` e `DELETE` contenute in tale tabella. Supponiamo che l'estensione `mysql_fdw` sia stata creata nell'istanza database RDS for PostgreSQL, come descritto nella procedura precedente. Dopo aver effettuato la connessione all'istanza database RDS for PostgreSQL come utente con i privilegi `rds_superuser`, è possibile procedere con i seguenti passaggi.

1. Nel cluster di database Aurora PostgreSQL crea un server esterno:

```
test=> CREATE SERVER mysqldb FOREIGN DATA WRAPPER mysql_fdw OPTIONS (host 'your-DB.aws-region.rds.amazonaws.com', port '3306');
CREATE SERVER
```

2. Concedere l'utilizzo a un utente che non dispone delle autorizzazioni `rds_superuser`, ad esempio `user1`:

```
test=> GRANT USAGE ON FOREIGN SERVER mysqldb TO user1;
GRANT
```

3. Connettersi come *user1* e quindi creare una mappatura per l'utente MySQL:

```
test=> CREATE USER MAPPING FOR user1 SERVER mysqldb OPTIONS (username 'myuser',
 password 'mypassword');
CREATE USER MAPPING
```

4. Creare di una tabella esterna collegata a una tabella MySQL:

```
test=> CREATE FOREIGN TABLE mytab (a int, b text) SERVER mysqldb OPTIONS (dbname
 'test', table_name '');
CREATE FOREIGN TABLE
```

5. Eseguire una semplice query sulla tabella esterna:

```
test=> SELECT * FROM mytab;
a |   b
---+-------
1 | apple
(1 row)
```

6. È possibile aggiungere, modificare e rimuovere i dati dalla tabella MySQL. Ad esempio:

```
test=> INSERT INTO mytab values (2, 'mango');
INSERT 0 1
```

Eseguire nuovamente la query SELECT per visualizzare i risultati:

```
test=> SELECT * FROM mytab ORDER BY 1;
 a |   b
---+-------
1 | apple
2 | mango
(2 rows)
```

Utilizzo della crittografia in transito con l'estensione

La connessione a MySQL da Aurora PostgreSQL utilizza la crittografia in transito (TLS/SSL) per impostazione predefinita. Tuttavia, la connessione torna a essere non crittografata quando la configurazione di client e server differiscono. È possibile applicare la crittografia a tutte le connessioni in uscita specificando l'opzione `REQUIRE SSL` sugli account utente RDS for MySQL. Lo stesso approccio funziona anche per gli account utente MariaDB e Aurora MySQL.

Per gli account utente MySQL configurati su `REQUIRE SSL`, il tentativo di connessione non riesce se non è possibile stabilire una connessione sicura.

Per applicare la crittografia agli account utente esistenti del database MySQL è possibile utilizzare il comando `ALTER USER`. La sintassi varia a seconda della versione MySQL, come mostrato nella tabella seguente. Per ulteriori informazioni, consultare la voce ALTER USER nel Manuale di riferimento di MySQL.

MySQL 5.7, MySQL 8.0	MySQL 5.6
`ALTER USER 'user'@'%' REQUIRE SSL;`	`GRANT USAGE ON *.* to 'user'@'%' REQUIRE SSL;`

Per ulteriori informazioni sull'estensione `mysql_fdw`, consultare la documentazione di mysql_fdw.

Interazione con un database Oracle utilizzando l'estensione oracle_fdw

Per accedere a un database Oracle dal cluster di database Aurora PostgreSQL è possibile installare e utilizzare l'estensione `oracle_fdw`. Questa estensione è un wrapper di dati esterni per database Oracle. Per ulteriori informazioni sull'estensione, consultare la documentazione di oracle_fdw.

L'estensione `oracle_fdw` è supportata su Aurora PostgreSQL 12.7 (Amazon Aurora versione 4.2) e versioni successive.

Argomenti

- Attivazione dell'estensione oracle_fdw
- Esempio: utilizzo di un server esterno collegato a un database Amazon RDS for Oracle
- Utilizzo della crittografia in transito
- Informazioni sulla visualizzazione pg_user_mappings e sulle autorizzazioni

Attivazione dell'estensione oracle_fdw

Per utilizzare l'estensione oracle_fdw, eseguire la procedura riportata di seguito.

Come attivare l'estensione oracle_fdw

- Eseguire il seguente comando utilizzando un account con le autorizzazioni `rds_superuser`.

```
CREATE EXTENSION oracle_fdw;
```

Esempio: utilizzo di un server esterno collegato a un database Amazon RDS for Oracle

L'esempio seguente mostra l'utilizzo di un server esterno collegato a un database Amazon RDS for Oracle.

Come creare un server esterno collegato a un database RDS for Oracle

1. Annotare le seguenti informazioni sull'istanza database RDS for Oracle:
   - Endpoint
   - Porta
   - Nome del database
2. Creare un server esterno.

```
test=> CREATE SERVER oradb FOREIGN DATA WRAPPER oracle_fdw OPTIONS (dbserver
 '//endpoint:port/DB_name');
CREATE SERVER
```

3. Concedere l'utilizzo a un utente che non dispone dei privilegi `rds_superuser`, ad esempio`user1`.

```
test=> GRANT USAGE ON FOREIGN SERVER oradb TO user1;
GRANT
```

4. Connettersi come `user1` e creare una mappatura a un utente Oracle.

```
test=> CREATE USER MAPPING FOR user1 SERVER oradb OPTIONS (user 'oracleuser',
 password 'mypassword');
CREATE USER MAPPING
```

5. Creare una tabella esterna collegata a una tabella Oracle.

```
test=> CREATE FOREIGN TABLE mytab (a int) SERVER oradb OPTIONS (table 'MYTABLE');
CREATE FOREIGN TABLE
```

6. Eseguire una query sulla tabella esterna.

```
test=>  SELECT * FROM mytab;
a
---
1
(1 row)
```

Se la query segnala il seguente errore, controllare il gruppo di sicurezza e la lista di controllo degli accessi (ACL) per assicurarsi che entrambe le istanze possano comunicare.

```
ERROR: connection for foreign table "mytab" cannot be established
DETAIL: ORA-12170: TNS:Connect timeout occurred
```

Utilizzo della crittografia in transito

La crittografia da PostgreSQL a Oracle in transito si basa su una combinazione di parametri di configurazione client e server. Per un esempio di utilizzo di Oracle 21c, consultare Informazioni sui valori per la negoziazione di crittografia e integrità nella documentazione Oracle. Il client utilizzato per oracle_fdw su Amazon RDS è configurato con `ACCEPTED`, il che significa che la crittografia dipende dalla configurazione del server di database Oracle e utilizza Oracle Security Library (libnnz) per la crittografia.

Se il database si trova su RDS for Oracle, consultare Crittografia di rete nativa Oracle per configurare la crittografia.

Informazioni sulla visualizzazione pg_user_mappings e sulle autorizzazioni

Il catalogo PostgreSQL `pg_user_mapping` archivia la mappatura da un utente Aurora PostgreSQL all'utente in un server remoto di dati esterni. L'accesso al catalogo è limitato, ma puoi usare la visualizzazione `pg_user_mappings` per vedere le mappature. Di seguito è possibile trovare un esempio che mostra come si applicano le autorizzazioni con un database Oracle, sebbene le stesse informazioni si applichino più in generale a qualsiasi wrapper di dati esterno.

Nel seguente output sono presenti ruoli e autorizzazioni mappati su tre diversi utenti di esempio. Gli utenti `rdssu1` e `rdssu2` sono membri del ruolo `rds_superuser`, mentre `user1` non lo è. Nell'esempio viene utilizzato il metacomando `psql \du` per elencare i ruoli esistenti.

```
test=>  \du
                                                          List of roles
    Role name    |                         Attributes                         |
                 Member of
-----------------+------------------------------------------------------------
+-----------------------------------------------------------
 rdssu1          |                                                            |
 {rds_superuser}
 rdssu2          |                                                            |
 {rds_superuser}
 user1           |                                                            | {}
```

Tutti gli utenti, inclusi gli utenti che godono dei privilegi `rds_superuser`, sono autorizzati a visualizzare le proprie mappature utente (`umoptions`) nella tabella `pg_user_mappings`. Come mostrato nell'esempio seguente, quando `rdssu1` cerca di ottenere tutte le mappature utente, viene generato un errore anche se gode dei privilegi `rdssu1rds_superuser`:

```
test=> SELECT * FROM pg_user_mapping;
ERROR: permission denied for table pg_user_mapping
```

Di seguito vengono riportati alcuni esempi.

```
test=> SET SESSION AUTHORIZATION rdssu1;
SET
test=> SELECT * FROM pg_user_mappings;
 umid  | srvid | srvname | umuser | usename    |            umoptions
-------+-------+---------+--------+------------+---------------------------------
 16414 | 16411 | oradb   |  16412 | user1      |
 16423 | 16411 | oradb   |  16421 | rdssu1     | {user=oracleuser,password=mypwd}
 16424 | 16411 | oradb   |  16422 | rdssu2     |
 (3 rows)

test=> SET SESSION AUTHORIZATION rdssu2;
SET
test=> SELECT * FROM pg_user_mappings;
 umid  | srvid | srvname | umuser | usename    |            umoptions
-------+-------+---------+--------+------------+---------------------------------
```

```
 16414 | 16411 | oradb   |  16412 | user1      |
 16423 | 16411 | oradb   |  16421 | rdssu1     |
 16424 | 16411 | oradb   |  16422 | rdssu2     | {user=oracleuser,password=mypwd}
 (3 rows)

test=> SET SESSION AUTHORIZATION user1;
SET
test=> SELECT * FROM pg_user_mappings;
 umid  | srvid | srvname | umuser | usename    |           umoptions
-------+-------+---------+--------+------------+-------------------------------
 16414 | 16411 | oradb   |  16412 | user1      | {user=oracleuser,password=mypwd}
 16423 | 16411 | oradb   |  16421 | rdssu1     |
 16424 | 16411 | oradb   |  16422 | rdssu2     |
 (3 rows)
```

A causa delle differenze nell'implementazione di `information_schema._pg_user_mappings` e `pg_catalog.pg_user_mappings`, un `rds_superuser` creato manualmente richiede autorizzazioni aggiuntive per visualizzare le password in `pg_catalog.pg_user_mappings`.

Non sono necessarie autorizzazioni aggiuntive per un `rds_superuser` che desideri visualizzare le password in `information_schema._pg_user_mappings`.

Gli utenti che non dispongono del ruolo `rds_superuser` possono visualizzare le password in `pg_user_mappings` solo nelle seguenti condizioni:

- L'utente corrente è l'utente mappato e possiede il server oppure detiene il privilegio `USAGE` su di esso.
- L'utente corrente è il proprietario del server e la mappatura è per `PUBLIC`.

Interazione con i database MySQL utilizzando l'estensione mysql_fw

È possibile utilizzare l'estensione `tds_fdw` per PostgreSQL per accedere ai database che supportano il protocollo TDS (Tabular Data Stream), ad esempio i database Sybase e Microsoft SQL Server. Questo wrapper di dati esterni consente di connettersi dal proprio cluster di database Aurora PostgreSQL ai database che utilizzano il protocollo TDS, incluso Amazon RDS for Microsoft SQL Server. Per ulteriori informazioni, consultare la documentazione di tds-fdw/tds_fdw su GitHub.

L'estensione `tds_fdw` è supportata su Amazon Aurora PostgreSQL versioni 13.6 e successive.

Configurazione del database Aurora PostgreSQL per l'utilizzo dell'estensione tds_fdw

Nelle procedure seguenti, è possibile trovare un esempio di configurazione e utilizzo di `tds_fdw` con un cluster di database Aurora PostgreSQL. Prima di potersi connettere a un database di SQL Server utilizzando `tds_fdw` è necessario disporre delle seguenti informazioni sull'istanza:

- Nome host o endpoint. Per trovare l'endpoint di un'istanza database RDS for SQL Server è possibile utilizzare la console. Scegliere la scheda Connectivity & security (Connettività e sicurezza) e cercare nella sezione Endpoint and port (Endpoint e porta).
- Numero della porta. Il numero di porta predefinito per Microsoft SQL Server è 1433.
- Nome del database. L'identificatore del database.

È inoltre necessario fornire l'accesso al gruppo di sicurezza o alla lista di controllo degli accessi (ACL) per la porta SQL Server 1433. Sia il cluster di database Aurora PostegreSQL che l'istanza database RDS for MySQL Server necessitano dell'accesso alla porta 1433. Se l'accesso non è configurato correttamente, quando si tenta di eseguire una query su Microsoft SQL Server viene visualizzato il seguente messaggio di errore:

```
ERROR: DB-Library error: DB #: 20009, DB Msg: Unable to connect:
Adaptive Server is unavailable or does not exist (mssql2019.aws-
region.rds.amazonaws.com), OS #: 0, OS Msg: Success, Level: 9
```

Per utilizzare tds_fdw per connettersi a un database di SQL Server

1. Collegarsi all'istanza principale del cluster di database Aurora PostgreSQL utilizzando un account che dispone del ruolo `rds_superuser`:

   ```
   psql --host=your-cluster-name-instance-1.aws-region.rds.amazonaws.com --port=5432
    --username=test --password
   ```

2. Installare l'estensione `tds_fdw`:

   ```
   test=> CREATE EXTENSION tds_fdw;
   CREATE EXTENSION
   ```

Dopo che l'estensione è stata installata sul cluster di database Aurora PostgreSQL , è necessario configurare il server esterno.

Per creare il server esterno

Eseguire queste attività sul cluster di database Aurora PostgreSQL utilizzando un account che dispone dei privilegi `rds_superuser`.

1. Creazione di un server esterno nel cluster di database Aurora PostgreSQL:

```
test=> CREATE SERVER sqlserverdb FOREIGN DATA WRAPPER tds_fdw OPTIONS
 (servername 'mssql2019.aws-region.rds.amazonaws.com', port '1433', database
 'tds_fdw_testing');
CREATE SERVER
```

   Per accedere ai dati non ASCII sul lato SQLServer, crea un collegamento server con l'opzione character_set nel cluster database Aurora PostgreSQL:

```
test=> CREATE SERVER sqlserverdb FOREIGN DATA WRAPPER tds_fdw OPTIONS (servername
 'mssql2019.aws-region.rds.amazonaws.com', port '1433', database 'tds_fdw_testing',
 character_set 'UTF-8');
CREATE SERVER
```

2. Concedere le autorizzazioni a un utente che non dispone del ruolo `rds_superuser`, ad esempio `user1`:

```
test=> GRANT USAGE ON FOREIGN SERVER sqlserverdb TO user1;
```

3. Collegarsi come user1 e quindi creare una mappatura per l'utente SQL Server:

```
test=> CREATE USER MAPPING FOR user1 SERVER sqlserverdb OPTIONS (username
 'sqlserveruser', password 'password');
CREATE USER MAPPING
```

4. Creare una tabella esterna collegata a una tabella SQL Server:

```
test=> CREATE FOREIGN TABLE mytab (a int) SERVER sqlserverdb OPTIONS (table
 'MYTABLE');
CREATE FOREIGN TABLE
```

5. Eseguire una query sulla tabella esterna:

```
test=> SELECT * FROM mytab;
 a
```

```
---
 1
(1 row)
```

Utilizzo della crittografia in transito per la connessione

La connessione da Aurora PostgreSQL verso SQL Server utilizza la crittografia in transito (TLS/SSL) in base alla configurazione del database SQL Server. Se SQL Server non è configurato per la crittografia, il client RDS per PostgreSQL che effettua la richiesta al database di SQL Server torna a comunicare in modalità non crittografata.

È possibile imporre l'utilizzo della crittografia per la connessione alle istanze database RDS for SQL Server impostando il parametro `rds.force_ssl`. Per scoprire come fare, consultare Imposizione dell'utilizzo di SSL per le connessioni all'istanza database. Per ulteriori informazioni sulla configurazione di SSL/TLS per RDS for SQL Server, consultare Utilizzo di SSL con un'istanza database Microsoft SQL Server.

Utilizzo di Trusted Language Extensions per PostgreSQL

Trusted Language Extensions per PostgreSQL è un kit di sviluppo open source per la creazione di estensioni di PostgreSQL. Ti consente di creare estensioni di PostgreSQL ad alte prestazioni ed eseguirle in sicurezza sul cluster database Aurora PostgreSQL. Con Trusted Language Extensions (TLE) per PostgreSQL, puoi creare estensioni di PostgreSQL che seguono l'approccio documentato per estendere le funzionalità di PostgreSQL. Per ulteriori informazioni, consulta Packaging Related Objects into an Extension (Creazione di pacchetti di oggetti correlati in un'estensione) nella documentazione PostgreSQL.

Uno dei principali vantaggi di TLE è che è possibile utilizzarlo in ambienti che non forniscono accesso al file system alla base dell'istanza PostgreSQL. In precedenza, l'installazione di una nuova estensione richiedeva l'accesso al file system. TLE rimuove questo vincolo. Fornisce un ambiente di sviluppo per creare nuove estensioni per qualsiasi database PostgreSQL, compresi quelli in esecuzione sui cluster database Aurora PostgreSQL.

TLE è progettato per impedire l'accesso a risorse non sicure per le estensioni create utilizzando TLE. Il suo ambiente di esecuzione limita l'impatto di qualsiasi difetto dell'estensione a una singola connessione al database. TLE inoltre offre agli amministratori di database un controllo dettagliato su chi può installare le estensioni e fornisce un modello di autorizzazioni per eseguirle.

TLE è supportato su Aurora PostgreSQL versione 14.5 e versioni successive.

L'ambiente di sviluppo e il runtime di Trusted Language Extensions sono compressi come estensione `pg_tle` di PostgreSQL versione 1.0.1. Supporta la creazione di estensioni in JavaScript, Perl, Tcl, PL/pgSQL e SQL. L'estensione `pg_tle` si installa nel cluster database Aurora PostgreSQL nello stesso modo in cui si installano altre estensioni di PostgreSQL. Dopo l'impostazione di `pg_tle`, gli sviluppatori possono utilizzarlo per creare nuove estensioni di PostgreSQL, note come estensioni TLE.

Negli argomenti seguenti sono disponibili informazioni su come impostare Trusted Language Extensions e come iniziare a creare le proprie estensioni TLE.

Argomenti

- Terminology
- Requisiti per l'utilizzo di Trusted Language Extensions per PostgreSQL
- Impostazione di Trusted Language Extensions nel cluster database Aurora PostgreSQL

- Panoramica di Trusted Language Extensions per PostgreSQL
- Creazione di estensioni TLE per Aurora PostgreSQL
- Eliminazione delle estensioni TLE da un database
- Disinstallazione di Trusted Language Extensions per PostgreSQL
- Utilizzo di hook PostgreSQL con le estensioni TLE
- Informazioni di riferimento sulle funzioni per Trusted Language Extensions per PostgreSQL
- Riferimento sugli hook per Trusted Language Extensions per PostgreSQL

Terminology

Per comprendere meglio Trusted Language Extensions, consulta il seguente glossario dei termini usati in questo argomento.

Trusted Language Extensions per PostgreSQL

: Trusted Language Extensions per PostgreSQL è il nome ufficiale del kit di sviluppo open source fornito come estensione `pg_tle`. È disponibile per l'uso su qualsiasi sistema PostgreSQL. Per ulteriori informazioni, consulta aws/pg_tle su GitHub.

Trusted Language Extensions

: Trusted Language Extensions è il nome abbreviato di Trusted Language Extensions per PostgreSQL. Questo nome abbreviato e la sua abbreviazione (TLE) vengono utilizzati anche in questa documentazione.

linguaggio attendibile

: Un linguaggio attendibile è un linguaggio di programmazione o di script con attributi di sicurezza specifici. Ad esempio, i linguaggi attendibili in genere limitano l'accesso al file system e l'uso di proprietà di rete specificate. Il kit di sviluppo TLE è progettato per supportare linguaggi attendibili. PostgreSQL supporta diversi linguaggi utilizzati per creare estensioni attendibili o non attendibili. Per un esempio, vedi Trusted and Untrusted PL/Perl (PL/Perl attendibile e non attendibile) nella documentazione di PostgreSQL. Quando crei un'estensione utilizzando Trusted Language Extensions, l'estensione utilizza intrinsecamente meccanismi di linguaggio attendibile.

Estensione TLE

: Un'estensione TLE è un'estensione di PostgreSQL creata utilizzando il kit di sviluppo Trusted Language Extensions (TLE).

Requisiti per l'utilizzo di Trusted Language Extensions per PostgreSQL

I seguenti sono i requisiti per l'impostazione e l'utilizzo del kit di sviluppo TLE.

- Versioni Aurora PostgreSQL – Trusted Language Extensions supportate su Aurora PostgreSQL versione 14,5 e solo versioni successive.
  - Per aggiornare il cluster database Aurora PostgreSQL, consulta Aggiornamento dei cluster database Amazon Aurora PostgreSQL.
  - Se non disponi ancora di un cluster database Aurora che esegue PostgreSQL, puoi eseguirne la creazione. Per ulteriori informazioni, consulta Creazione e connessione di un cluster di database Aurora PostgreSQL.
- Richiede i privilegi **`rds_superuser`**: per impostare e configurare l'estensione `pg_tle`, il ruolo utente del database deve disporre delle autorizzazioni del ruolo `rds_superuser`. Per impostazione predefinita, questo ruolo viene concesso all'utente `postgres` che crea il cluster database Aurora PostgreSQL.
- Richiede un gruppo di parametri database personalizzato: è necessario configurare il cluster database Aurora PostgreSQL con un gruppo di parametri database personalizzato. Utilizza il gruppo di parametri database personalizzato per l'istanza di scrittura del cluster database Aurora PostgreSQL.
  - Se non si configura il cluster database Aurora PostgreSQL con un gruppo di parametri database personalizzato, è necessario crearne uno e associarlo all'istanza di scrittura del cluster database Aurora PostgreSQL. Per un breve riepilogo dei passaggi, consulta Creazione e applicazione di un gruppo di parametri database personalizzato.
  - Se è già stata eseguita la configurazione del cluster database Aurora PostgreSQL utilizzando un gruppo di parametri database personalizzato, puoi impostare Trusted Language Extensions. Per informazioni dettagliate, consultare Impostazione di Trusted Language Extensions nel cluster database Aurora PostgreSQL.

Creazione e applicazione di un gruppo di parametri database personalizzato

Utilizza i seguenti passaggi per creare un gruppo di parametri database personalizzato e configurare il cluster database Aurora PostgreSQL per utilizzarlo.

Console

Per creare un gruppo di parametri database personalizzato e utilizzarlo con il cluster database Aurora PostgreSQL

1. Accedi alla Console di gestione AWS e apri la console Amazon RDS all'indirizzo https://console.aws.amazon.com/rds/.
2. Nel menu di Amazon RDS scegli Parameter groups (Gruppi di parametri).
3. Scegli Create parameter group (Crea gruppo di parametri).
4. Nella pagina Parameter group details (Dettagli del gruppo di parametri) immetti le seguenti informazioni.
   - Per Parameter group family (Famiglia del gruppo di parametri), scegli aurora-postgresql14.
   - Per Type (Tipo), scegli il gruppo di parametri database.
   - Per Group name (Nome gruppo), assegna al gruppo di parametri un nome significativo nel contesto delle operazioni.
   - In Description (Descrizione), immetti una descrizione utile in modo che gli altri membri del team possano trovarla facilmente.
5. Scegli Create (Crea). Il gruppo di parametri database personalizzato viene creato nella Regione AWS. Ora puoi modificare il cluster database Aurora PostgreSQL che sarà possibile utilizzare seguendo i prossimi passaggi.
6. Scegli Databases (Database) dal menu Amazon RDS.
7. Scegli il cluster database Aurora PostgreSQL che desideri utilizzare con TLE tra le opzioni elencate, quindi scegli Modify (Modifica).
8. Nella pagina Modify DB cluster settings (Modifica le impostazioni del cluster database), trova Database options (Opzioni del database) e usa il selettore per scegliere il gruppo di parametri database personalizzato.
9. Per salvare la modifica seleziona Continua (Continua).
10. Scegli Apply immediately (Applica immediatamente) in modo da poter continuare a impostare il cluster database Aurora PostgreSQL per utilizzare TLE.

Per continuare a impostare il sistema per Trusted Language Extensions, consulta Impostazione di Trusted Language Extensions nel cluster database Aurora PostgreSQL.

Per ulteriori informazioni sull'utilizzo di gruppi di parametri database e cluster database, consulta Gruppi di parametri del cluster di database per i cluster di database Amazon Aurora.

AWS CLI

Puoi evitare di specificare l'argomento `--region` quando usi i comandi dell'interfaccia della linea di comando configurando AWS CLI con la Regione AWS predefinita. Per ulteriori informazioni, consulta Nozioni di base sulla configurazione nella Guida per l'utente di AWS Command Line Interface.

Per creare un gruppo di parametri database personalizzato e utilizzarlo con il cluster database Aurora PostgreSQL

1. Usa il comando AWS CLI create-db-parameter-group per creare un gruppo di parametri database personalizzato basato su aurora-postgresql14 per la tua Regione AWS. Tieni presente che in questo passaggio crei un gruppo di parametri database da applicare all'istanza di scrittura del cluster database Aurora PostgreSQL.

   Per Linux, macOS o Unix:

   ```
   aws rds create-db-parameter-group \
     --region aws-region \
     --db-parameter-group-name custom-params-for-pg-tle \
     --db-parameter-group-family aurora-postgresql14 \
     --description "My custom DB parameter group for Trusted Language Extensions"
   ```

   Per Windows:

   ```
   aws rds create-db-parameter-group ^
     --region aws-region ^
     --db-parameter-group-name custom-params-for-pg-tle ^
     --db-parameter-group-family aurora-postgresql14 ^
     --description "My custom DB parameter group for Trusted Language Extensions"
   ```

   Il gruppo di parametri database personalizzato è disponibile nella Regione AWS, in modo che puoi modificare l'istanza di scrittura del cluster database Aurora PostgreSQL per utilizzarlo.
2. Utilizza il comando AWS CLI modify-db-instance per applicare il gruppo parametri database personalizzato all'istanza di scrittura del cluster database Aurora PostgreSQL. Questo comando riavvia immediatamente l'istanza attiva.

   Per Linux, macOS o Unix:

```
aws rds modify-db-instance \
  --region aws-region \
  --db-instance-identifier your-writer-instance-name \
  --db-parameter-group-name custom-params-for-pg-tle \
  --apply-immediately
```

Per Windows:

```
aws rds modify-db-instance ^
  --region aws-region ^
  --db-instance-identifier your-writer-instance-name ^
  --db-parameter-group-name custom-params-for-pg-tle ^
  --apply-immediately
```

Per continuare a impostare il sistema per Trusted Language Extensions, consulta Impostazione di Trusted Language Extensions nel cluster database Aurora PostgreSQL.

Per ulteriori informazioni, consulta Gruppi di parametri database per istanze database Amazon Aurora .

Impostazione di Trusted Language Extensions nel cluster database Aurora PostgreSQL

I passaggi seguenti si basano sull'ipotesi che il cluster database Aurora PostgreSQL sia associato a un gruppo di parametri personalizzato del cluster database. Puoi usare la Console di gestione AWS o la AWS CLI per questi passaggi.

Quando imposti Trusted Language Extensions nel cluster database Aurora PostgreSQL, lo installi in un database specifico che deve essere utilizzato dagli utenti del database che dispongono delle relative autorizzazioni.

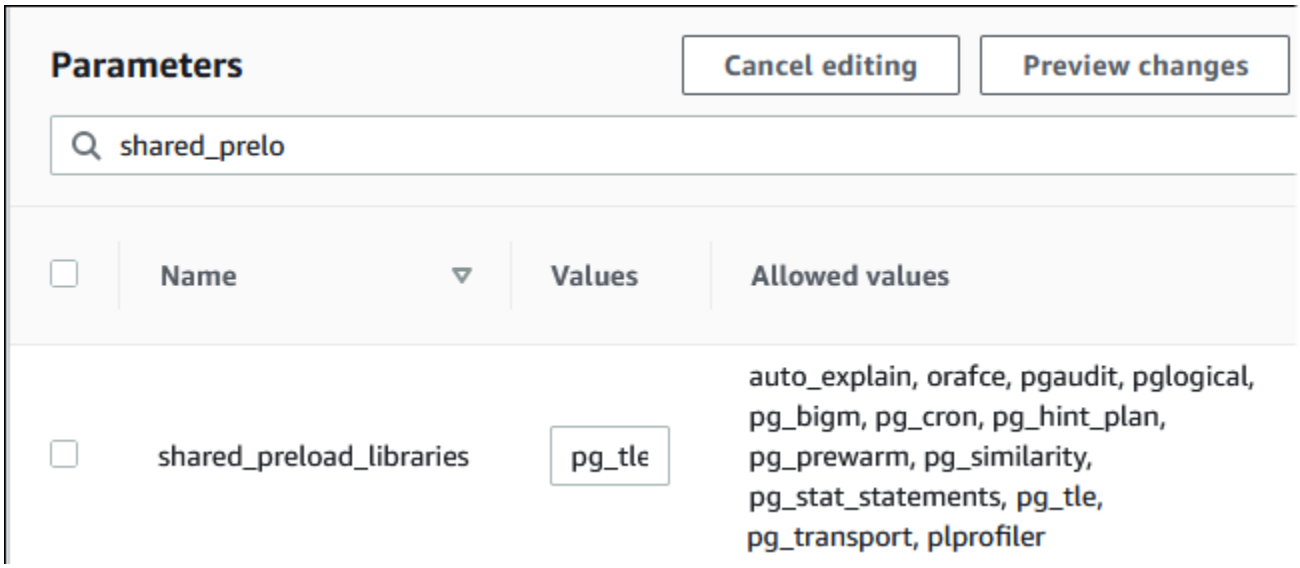
Console

Per impostare Trusted Language Extensions

Esegui i seguenti passaggi utilizzando un account membro del gruppo `rds_superuser` (ruolo).

1. Accedi alla Console di gestione AWS e apri la console Amazon RDS all'indirizzo https://console.aws.amazon.com/rds/.

2. Nel riquadro di navigazione, scegli l'istanza di scrittura del cluster database Aurora PostgreSQL .
3. Apri la scheda Configurazione per l'istanza di scrittura del cluster database Aurora PostgreSQL. Tra i dettagli dell'istanza, individua il collegamento Parameter group (Gruppo di parametri).
4. Scegli il collegamento per aprire i parametri personalizzati associati al cluster database Aurora PostgreSQL.
5. Nel campo di ricerca Parametri, digita `shared_pre` per trovare il parametro `shared_preload_libraries`.
6. Scegli Edit parameters (Modifica parametri) per accedere ai valori delle proprietà.
7. Aggiungi `pg_tle` all'elenco nel campo Values (Valori). Utilizza una virgola per separare gli elementi nell'elenco di valori.



8. Riavvia l'istanza di scrittura del cluster database Aurora PostgreSQL in modo che la modifica al parametro `shared_preload_libraries` diventi effettiva.
9. Quando l'istanza è disponibile, verifica che `pg_tle` sia stato inizializzato. Utilizza `psql` per connetterti all'istanza di scrittura del cluster database Aurora PostgreSQL, quindi esegui il comando seguente.

```
SHOW shared_preload_libraries;
shared_preload_libraries
--------------------------
rdsutils,pg_tle
(1 row)
```

10. Con `pg_tle` inizializzato, puoi ora creare l'estensione.

```
CREATE EXTENSION pg_tle;
```

Per verificare che l'estensione sia installata, puoi usare il seguente metacomando `psql`.

```
labdb=> \dx
                          List of installed extensions
  Name   | Version |   Schema   |                Description
---------+---------+------------+-----------------------------------------
 pg_tle  | 1.0.1   | pgtle      | Trusted-Language Extensions for PostgreSQL
 plpgsql | 1.0     | pg_catalog | PL/pgSQL procedural language
```

11. Assegna il ruolo `pgtle_admin` al nome utente principale che hai creato per il cluster database Aurora PostgreSQL al momento dell'impostazione. Se hai accettato l'impostazione predefinita, il valore è `postgres`.

```
labdb=> GRANT pgtle_admin TO postgres;
GRANT ROLE
```

Per verificare se la concessione è avvenuta, utilizza il metacomando `psql` come illustrato nell'esempio seguente. Nell'output vengono visualizzati solo i ruoli `pgtle_admin` e `postgres`. Per ulteriori informazioni, consulta Informazioni su ruoli e autorizzazioni di PostgreSQL.

```
labdb=> \du
                           List of roles
    Role name    |           Attributes            |               Member of
-----------------+---------------------------------
+-----------------------------------
pgtle_admin     | Cannot login                     | {}
postgres        | Create role, Create DB          +| {rds_superuser,pgtle_admin}
                | Password valid until infinity    |...
```

12. Chiudi la sessione `psql` usando il metacomando `\q`.

```
\q
```

Per iniziare a creare le estensioni TLE, consulta Esempio: creazione di un'estensione Trusted Language Extensions utilizzando SQL.

AWS CLI

Puoi evitare di specificare l'argomento `--region` quando usi i comandi CLI configurando AWS CLI con la Regione AWS predefinita. Per ulteriori informazioni, consulta Nozioni di base sulla configurazione nella Guida per l'utente di AWS Command Line Interface.

Per impostare Trusted Language Extensions

1. Usa il comando AWS CLI modify-db-parameter-group per aggiungere `pg_tle` al parametro `shared_preload_libraries`.

```
aws rds modify-db-parameter-group \
   --db-parameter-group-name custom-param-group-name \
   --parameters
 "ParameterName=shared_preload_libraries,ParameterValue=pg_tle,ApplyMethod=pending-
reboot" \
   --region aws-region
```

2. Usa il comando AWS CLI reboot-db-instance per riavviare l'istanza di scrittura del cluster database Aurora PostgreSQL e inizializzare la libreria `pg_tle`.

```
aws rds reboot-db-instance \
    --db-instance-identifier writer-instance \
    --region aws-region
```

3. Quando l'istanza è disponibile, verifica che `pg_tle` sia stato inizializzato. Utilizza `psql` per connetterti all'istanza di scrittura del cluster database Aurora PostgreSQL, quindi esegui il comando seguente.

```
SHOW shared_preload_libraries;
shared_preload_libraries
--------------------------
rdsutils,pg_tle
(1 row)
```

   Con `pg_tle` inizializzato, puoi ora creare l'estensione.

```
CREATE EXTENSION pg_tle;
```

4. Assegna il ruolo `pgtle_admin` al nome utente principale che hai creato per il cluster database Aurora PostgreSQL al momento dell'impostazione. Se hai accettato l'impostazione predefinita, il valore è `postgres`.

```
GRANT pgtle_admin TO postgres;
GRANT ROLE
```

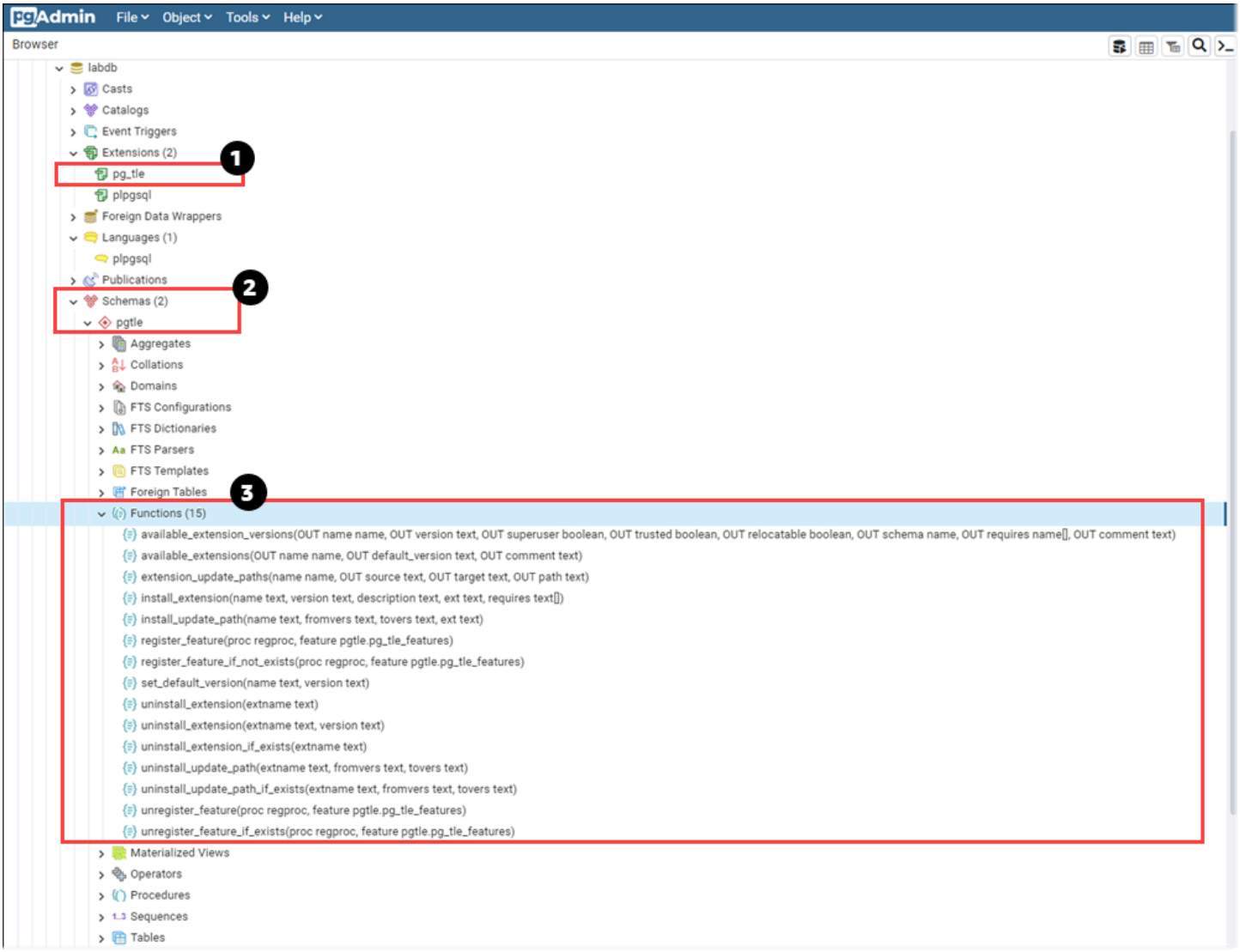
5. Chiudi la sessione `psql` come indicato di seguito.

```
labdb=> \q
```

Per iniziare a creare le estensioni TLE, consulta Esempio: creazione di un'estensione Trusted Language Extensions utilizzando SQL.

Panoramica di Trusted Language Extensions per PostgreSQL

Trusted Language Extensions per PostgreSQL è un'estensione di PostgreSQL che si installa nel cluster database Aurora PostgreSQL nello stesso modo in cui si impostano le altre estensioni di PostgreSQL. Nell'immagine seguente di un database di esempio nello strumento client pgAdmin, è possibile vedere alcuni dei componenti che compongono l'estensione `pg_tle`.



È possibile vedere i dettagli riportati di seguito.

1. Il kit di sviluppo Trusted Language Extensions (TLE) per PostgreSQL è fornito nel pacchetto come estensione `pg_tle`. Pertanto, `pg_tle` viene aggiunto alle estensioni disponibili per il database in cui è installato.
2. TLE ha un proprio schema, `pgtle`. Questo schema contiene funzioni helper (3) per l'installazione e la gestione delle estensioni create.
3. TLE offre oltre una dozzina di funzioni helper per l'installazione, la registrazione e la gestione delle estensioni. Per ulteriori informazioni su queste funzioni, consulta Informazioni di riferimento sulle funzioni per Trusted Language Extensions per PostgreSQL.

Altri componenti dell'estensione `pg_tle` sono:

- Il ruolo **`pgtle_admin`**: il ruolo `pgtle_admin` viene creato quando viene installata l'estensione `pg_tle`. Questo ruolo include privilegi e deve essere trattato come tale. Ti consigliamo vivamente di seguire il principio del privilegio minimo quando concedi il ruolo `pgtle_admin` agli utenti del database. In altre parole, concedi il ruolo `pgtle_admin` solo agli utenti del database autorizzati a creare, installare e gestire nuove estensioni TLE, ad esempio `postgres`.
- La tabella **`pgtle.feature_info`**: la tabella `pgtle.feature_info` è una tabella protetta che contiene informazioni sulle estensioni TLE, sugli hook e sulle stored procedure e funzioni personalizzate che utilizzano. Se disponi di privilegi `pgtle_admin`, usa le seguenti funzioni Trusted Language Extensions per aggiungere e aggiornare le informazioni nella tabella.
  - pgtle.register_feature
  - pgtle.register_feature_if_not_exists
  - pgtle.unregister_feature
  - pgtle.unregister_feature_if_exists

Creazione di estensioni TLE per Aurora PostgreSQL

È possibile installare qualsiasi estensione creata con TLE in qualsiasi cluster database Aurora PostgreSQL in cui è installata l'estensione `pg_tle`. L'estensione `pg_tle` si riferisce al database PostgreSQL in cui è installata. Le estensioni create utilizzando TLE si riferiscono allo stesso database.

Usa le varie funzioni `pgtle` per installare il codice che costituisce la tua estensione TLE. Le seguenti funzioni di Trusted Language Extensions richiedono tutte il ruolo `pgtle_admin`.

- pgtle.install_extension
- pgtle.install_update_path
- pgtle.register_feature
- pgtle.register_feature_if_not_exists
- pgtle.set_default_version
- pgtle.uninstall_extension(name)
- pgtle.uninstall_extension(name, version)
- pgtle.uninstall_extension_if_exists
- pgtle.uninstall_update_path
- pgtle.uninstall_update_path_if_exists

- pgtle.unregister_feature
- pgtle.unregister_feature_if_exists

Esempio: creazione di un'estensione Trusted Language Extensions utilizzando SQL

L'esempio seguente mostra come creare un'estensione TLE denominata `pg_distance` che contiene alcune funzioni SQL per il calcolo delle distanze utilizzando formule diverse. Nell'elenco, puoi trovare la funzione per il calcolo della distanza di Manhattan e la funzione per il calcolo della distanza euclidea. Per ulteriori informazioni sulla differenza tra queste formule, consulta Taxicab geometry (Geometria del taxi) e Euclidean geometry (Geometria euclidea) in Wikipedia.

È possibile utilizzare questo esempio nel cluster database Aurora PostgreSQL se l'estensione `pg_tle` è impostata come descritto in dettaglio in Impostazione di Trusted Language Extensions nel cluster database Aurora PostgreSQL.

> **Note**
>
> È necessario disporre dei privilegi del ruolo `pgtle_admin` per seguire questa procedura.

Per creare l'estensione TLE di esempio

I passaggi seguenti utilizzano un database di esempio denominato `labdb`. Questo database è di proprietà dell'utente `postgres` principale. Il ruolo `postgres` dispone anche delle autorizzazioni del ruolo `pgtle_admin`.

1. Utilizza `psql` per connetterti all'istanza di scrittura del cluster database Aurora PostgreSQL.

   ```
   psql --host=db-instance-123456789012.aws-region.rds.amazonaws.com
   --port=5432 --username=postgres --password --dbname=labdb
   ```

2. Crea un'estensione TLE denominata `pg_distance` copiando il seguente codice e incollandolo nella console della sessione `psql`.

   ```
   SELECT pgtle.install_extension
   (
    'pg_distance',
    '0.1',
     'Distance functions for two points',
   $_pg_tle_$
   ```

```
    CREATE FUNCTION dist(x1 float8, y1 float8, x2 float8, y2 float8, norm int)
    RETURNS float8
    AS $$
      SELECT (abs(x2 - x1) ^ norm + abs(y2 - y1) ^ norm) ^ (1::float8 / norm);
    $$ LANGUAGE SQL;

    CREATE FUNCTION manhattan_dist(x1 float8, y1 float8, x2 float8, y2 float8)
    RETURNS float8
    AS $$
      SELECT dist(x1, y1, x2, y2, 1);
    $$ LANGUAGE SQL;

    CREATE FUNCTION euclidean_dist(x1 float8, y1 float8, x2 float8, y2 float8)
    RETURNS float8
    AS $$
      SELECT dist(x1, y1, x2, y2, 2);
    $$ LANGUAGE SQL;
$_pg_tle_$
);
```

Viene visualizzato l'output riportato di seguito.

```
install_extension
---------------
 t
(1 row)
```

Gli artefatti che costituiscono l'estensione `pg_distance` sono ora installati nel database. Questi artefatti includono il file di controllo e il codice dell'estensione, che devono essere presenti in modo che l'estensione possa essere creata utilizzando il comando `CREATE EXTENSION`. In altre parole, è comunque necessario creare l'estensione per rendere le funzioni disponibili agli utenti del database.

3. Per creare l'estensione, usa il comando `CREATE EXTENSION` come per qualsiasi altra estensione. Come per altre estensioni, l'utente del database deve disporre delle autorizzazioni `CREATE` nel database.

```
CREATE EXTENSION pg_distance;
```

4. Per testare l'estensione TLE `pg_distance`, puoi usarla per calcolare la distanza di Manhattan tra quattro punti.

```
labdb=> SELECT manhattan_dist(1, 1, 5, 5);
8
```

Per calcolare la distanza euclidea tra lo stesso set di punti, puoi usare quanto segue.

```
labdb=> SELECT euclidean_dist(1, 1, 5, 5);
5.656854249492381
```

L'estensione `pg_distance` carica le funzioni nel database e le rende disponibili a tutti gli utenti con le autorizzazioni per il database.

Modifica dell'estensione TLE

Per migliorare le prestazioni delle query per le funzioni contenute nell'estensione TLE, aggiungi i seguenti due attributi PostgreSQL alle specifiche.

- `IMMUTABLE`: l'attributo `IMMUTABLE` garantisce che l'ottimizzatore di query possa utilizzare le ottimizzazioni per migliorare i tempi di risposta delle query. Per ulteriori informazioni, consulta Function Volatility Categories (Categorie della volatilità delle funzioni) nella documentazione di PostgreSQL.
- `PARALLEL SAFE`: l'attributo `PARALLEL SAFE` è un altro attributo che consente a PostgreSQL di eseguire la funzione in modalità parallela. Per ulteriori informazioni, consultare CREATE FUNCTION nella documentazione di PostgreSQL.

Nell'esempio seguente, puoi vedere come viene utilizzata la funzione `pgtle.install_update_path` per aggiungere questi attributi a ogni funzione per creare la versione `0.2` dell'estensione TLE `pg_distance`. Per ulteriori informazioni su questa funzione, consulta pgtle.install_update_path. È necessario avere il ruolo `pgtle_admin` necessario per eseguire questa operazione.

Per aggiornare un'estensione TLE esistente e specificare la versione predefinita

1. Esegui la connessione all'istanza di scrittura del cluster database Aurora PostgreSQL utilizzando `psql` o un altro strumento client, ad esempio pgAdmin.

```
psql --host=db-instance-123456789012.aws-region.rds.amazonaws.com
```

```
--port=5432 --username=postgres --password --dbname=labdb
```

2. Modifica l'estensione TLE esistente copiando il seguente codice e incollandolo nella console della sessione `psql`.

```
SELECT pgtle.install_update_path
(
 'pg_distance',
 '0.1',
 '0.2',
$_pg_tle_$
    CREATE OR REPLACE FUNCTION dist(x1 float8, y1 float8, x2 float8, y2 float8,
 norm int)
    RETURNS float8
    AS $$
      SELECT (abs(x2 - x1) ^ norm + abs(y2 - y1) ^ norm) ^ (1::float8 / norm);
    $$ LANGUAGE SQL IMMUTABLE PARALLEL SAFE;

    CREATE OR REPLACE FUNCTION manhattan_dist(x1 float8, y1 float8, x2 float8, y2
 float8)
    RETURNS float8
    AS $$
      SELECT dist(x1, y1, x2, y2, 1);
    $$ LANGUAGE SQL IMMUTABLE PARALLEL SAFE;

    CREATE OR REPLACE FUNCTION euclidean_dist(x1 float8, y1 float8, x2 float8, y2
 float8)
    RETURNS float8
    AS $$
      SELECT dist(x1, y1, x2, y2, 2);
    $$ LANGUAGE SQL IMMUTABLE PARALLEL SAFE;
$_pg_tle_$
);
```

Viene visualizzata una risposta simile alla seguente.

```
install_update_path
--------------------
 t
(1 row)
```

È possibile impostare questa versione dell'estensione come versione predefinita, in modo che gli utenti del database non debbano specificare una versione quando creano o aggiornano l'estensione nel database.

3. Per specificare che la versione modificata (versione 0.2) dell'estensione TLE è la versione predefinita, usa la funzione `pgtle.set_default_version` come mostrato nell'esempio seguente.

```
SELECT pgtle.set_default_version('pg_distance', '0.2');
```

Per ulteriori informazioni su questa funzione, consulta pgtle.set_default_version.

4. Una volta inserito il codice, puoi aggiornare l'estensione TLE installata nel modo consueto, usando il comando `ALTER EXTENSION ... UPDATE`, come mostrato di seguito:

```
ALTER EXTENSION pg_distance UPDATE;
```

Eliminazione delle estensioni TLE da un database

Puoi eliminare le estensioni TLE usando il comando `DROP EXTENSION` nello stesso modo che impieghi per le altre estensioni di PostgreSQL. L'eliminazione dell'estensione non rimuove i file di installazione che costituiscono l'estensione, il che consente agli utenti di ricrearla. Per rimuovere l'estensione e i relativi file di installazione, esegui la seguente procedura in due passaggi.

Per eliminare l'estensione TLE e rimuovere i file di installazione

1. Utilizza `psql` o un altro strumento cliente per connetterti all'istanza di scrittura del cluster database Aurora PostgreSQL.

```
psql --host=cluster-instance-1.111122223333.aws-region.rds.amazonaws.com --
port=5432 --username=postgres --password --dbname=dbname
```

2. Elimina l'estensione come faresti per qualsiasi estensione di PostgreSQL.

```
DROP EXTENSION your-TLE-extension
```

Ad esempio, se crei l'estensione `pg_distance` come descritto in Esempio: creazione di un'estensione Trusted Language Extensions utilizzando SQL, puoi eliminarla come segue.

```
DROP EXTENSION pg_distance;
```

Viene visualizzato l'output che conferma che l'estensione è stata eliminata, come segue.

```
DROP EXTENSION
```

A questo punto, l'estensione non è più attiva nel database. Tuttavia, i file di installazione e il file di controllo sono ancora disponibili nel database, quindi gli utenti del database possono creare nuovamente l'estensione, se lo desiderano.

- Se vuoi lasciare intatti i file delle estensioni in modo che gli utenti del database possano creare l'estensione TLE, puoi fermarti qui.
- Se desideri rimuovere tutti i file che costituiscono l'estensione, continua con il passaggio successivo.

3. Per rimuovere tutti i file di installazione per l'estensione, usa la funzione `pgtle.uninstall_extension`. Questa funzione rimuove tutto il codice e i file di controllo dell'estensione.

```
SELECT pgtle.uninstall_extension('your-tle-extension-name');
```

Ad esempio, per rimuovere tutti i file di installazione `pg_distance`, utilizza il comando seguente.

```
SELECT pgtle.uninstall_extension('pg_distance');
 uninstall_extension
---------------------
 t
(1 row)
```

Disinstallazione di Trusted Language Extensions per PostgreSQL

Se non desideri più creare le estensioni TLE utilizzando TLE, puoi eliminare l'estensione `pg_tle` e rimuovere tutti gli artefatti. Questa azione include l'eliminazione di qualsiasi estensione TLE nel database e l'eliminazione dello schema `pgtle`.

Per eliminare l'estensione **`pg_tle`** e il relativo schema da un database

1. Utilizza `psql` o un altro strumento cliente per connetterti all'istanza di scrittura del cluster database Aurora PostgreSQL.

   ```
   psql --host=cluster-instance-1.111122223333.aws-region.rds.amazonaws.com --
   port=5432 --username=postgres --password --dbname=dbname
   ```

2. Elimina l'estensione `pg_tle` dal database. Se il database ha le estensioni TLE ancora in esecuzione nel database, devi eliminare anche quelle estensioni. A questo scopo, puoi utilizzare la parola chiave `CASCADE`, come illustrato di seguito.

   ```
   DROP EXTENSION pg_tle CASCADE;
   ```

   Se l'estensione `pg_tle` non è ancora attiva nel database, non è necessario utilizzare la parola chiave `CASCADE`.

3. Elimina lo schema `pgtle`. Questa azione rimuove tutte le funzioni di gestione dal database.

   ```
   DROP SCHEMA pgtle CASCADE;
   ```

   Il comando restituisce quanto segue al termine del processo.

   ```
   DROP SCHEMA
   ```

   L'estensione `pg_tle`, il relativo schema, le funzioni e tutti gli artefatti vengono rimossi. Per creare nuove estensioni utilizzando TLE, ripeti il processo di impostazione. Per ulteriori informazioni, consulta Impostazione di Trusted Language Extensions nel cluster database Aurora PostgreSQL.

Utilizzo di hook PostgreSQL con le estensioni TLE

Un hook è un meccanismo di callback disponibile in PostgreSQL che consente agli sviluppatori di chiamare funzioni personalizzate o altre routine durante le normali operazioni del database. Il kit di sviluppo TLE supporta gli hook PostgreSQL per poter integrare le funzioni personalizzate con il comportamento di PostgreSQL in fase di esecuzione. Ad esempio, puoi utilizzare un hook per associare il processo di autenticazione al codice personalizzato o per modificare il processo di pianificazione ed esecuzione delle query in base alle tue esigenze specifiche.

Le estensioni TLE possono utilizzare gli hook. Se l'ambito dell'hook è globale, si applica a tutti i database. Pertanto, se l'estensione TLE utilizza un hook globale, è necessario crearla in tutti i database a cui gli utenti possono accedere.

Quando usi `pg_tle` per creare le estensioni Trusted Language Extensions, puoi utilizzare gli hook disponibili da un'API SQL per creare le funzioni della tua estensione. È necessario registrare gli hook con `pg_tle`. Per alcuni hook, potrebbe essere necessario impostare anche vari parametri di configurazione. Ad esempio, l'hook di controllo `passcode` può essere impostato su attivo, disattivo oppure obbligatorio. Per ulteriori informazioni sui requisiti specifici per gli hook `pg_tle` disponibili, consulta Riferimento sugli hook per Trusted Language Extensions per PostgreSQL.

Esempio: creazione di un'estensione che utilizza un hook PostgreSQL

L'esempio discusso in questa sezione utilizza un hook PostgreSQL per controllare la password fornita durante specifiche operazioni SQL e impedisce agli utenti del database di impostare le proprie password su una qualsiasi di quelle contenute nella tabella `password_check.bad_passwords`. La tabella contiene le prime dieci opzioni di password più utilizzate, ma facilmente violabili.

Per configurare questo esempio nel cluster database Aurora PostgreSQL, devi aver già installato Trusted Language Extensions. Per informazioni dettagliate, consultare Impostazione di Trusted Language Extensions nel cluster database Aurora PostgreSQL.

Per configurare l'esempio dell'hook di controllo della password

1. Utilizza `psql` per connetterti all'istanza di scrittura del cluster database Aurora PostgreSQL.

```
psql --host=db-instance-123456789012.aws-region.rds.amazonaws.com
--port=5432 --username=postgres --password --dbname=labdb
```

2. Copia il codice da Codice di hook di controllo della password e incollalo nel database.

```
SELECT pgtle.install_extension (
  'my_password_check_rules',
  '1.0',
  'Do not let users use the 10 most commonly used passwords',
$_pgtle_$
  CREATE SCHEMA password_check;
  REVOKE ALL ON SCHEMA password_check FROM PUBLIC;
  GRANT USAGE ON SCHEMA password_check TO PUBLIC;

  CREATE TABLE password_check.bad_passwords (plaintext) AS
```

```
 VALUES
   ('123456'),
   ('password'),
   ('12345678'),
   ('qwerty'),
   ('123456789'),
   ('12345'),
   ('1234'),
   ('111111'),
   ('1234567'),
   ('dragon');
 CREATE UNIQUE INDEX ON password_check.bad_passwords (plaintext);

 CREATE FUNCTION password_check.passcheck_hook(username text, password text,
password_type pgtle.password_types, valid_until timestamptz, valid_null boolean)
 RETURNS void AS $$
   DECLARE
     invalid bool := false;
   BEGIN
     IF password_type = 'PASSWORD_TYPE_MD5' THEN
       SELECT EXISTS(
         SELECT 1
         FROM password_check.bad_passwords bp
         WHERE ('md5' || md5(bp.plaintext || username)) = password
       ) INTO invalid;
       IF invalid THEN
         RAISE EXCEPTION 'Cannot use passwords from the common password
dictionary';
       END IF;
     ELSIF password_type = 'PASSWORD_TYPE_PLAINTEXT' THEN
       SELECT EXISTS(
         SELECT 1
         FROM password_check.bad_passwords bp
         WHERE bp.plaintext = password
       ) INTO invalid;
       IF invalid THEN
         RAISE EXCEPTION 'Cannot use passwords from the common password
dictionary';
       END IF;
     END IF;
   END
 $$ LANGUAGE plpgsql SECURITY DEFINER;

 GRANT EXECUTE ON FUNCTION password_check.passcheck_hook TO PUBLIC;
```

```
  SELECT pgtle.register_feature('password_check.passcheck_hook', 'passcheck');
$_pgtle_$
);
```

Quando l'estensione è stata caricata nel database, viene visualizzato un output come il seguente.

```
 install_extension
-------------------
 t
(1 row)
```

3. Mentre sei ancora connesso al database, puoi creare l'estensione.

```
CREATE EXTENSION my_password_check_rules;
```

4. È possibile confermare che l'estensione è stata creata nel database utilizzando il seguente metacomando `psql`.

```
\dx
                        List of installed extensions
          Name           | Version |   Schema   |
 Description
------------------------+---------+------------
+-------------------------------------------------------------
 my_password_check_rules | 1.0     | public     | Prevent use of any of the top-ten
 most common bad passwords
 pg_tle                  | 1.0.1   | pgtle      | Trusted-Language Extensions for
 PostgreSQL
 plpgsql                 | 1.0     | pg_catalog | PL/pgSQL procedural language
(3 rows)
```

5. Apri un'altra sessione di terminale per utilizzare AWS CLI. È necessario modificare il gruppo di parametri database personalizzato per attivare l'hook di controllo della password. Per fare ciò, utilizza il comando dell'interfaccia della linea di comando modify-db-parameter-group, come indicato nell'esempio seguente.

```
aws rds modify-db-parameter-group \
    --region aws-region \
    --db-parameter-group-name your-custom-parameter-group \
```

```
    --parameters
 "ParameterName=pgtle.enable_password_check,ParameterValue=on,ApplyMethod=immediate"
```

Per rendere effettive le modifiche all'impostazione del gruppo di parametri possono essere necessari alcuni minuti. Tuttavia, questo parametro è dinamico, quindi non è necessario riavviare l'istanza di scrittura del cluster database Aurora PostgreSQL perché l'impostazione diventi effettiva.

6. Apri la sessione `psql` ed esegui una query sul database per verificare che l'hook di controllo della password sia stato attivato.

```
labdb=> SHOW pgtle.enable_password_check;
pgtle.enable_password_check
----------------------------
on
(1 row)
```

L'hook di controllo della password è ora attivo. Puoi testarlo creando un nuovo ruolo e utilizzando una delle password errate, come illustrato nell'esempio seguente.

```
CREATE ROLE test_role PASSWORD 'password';
ERROR:  Cannot use passwords from the common password dictionary
CONTEXT:  PL/pgSQL function
 password_check.passcheck_hook(text,text,pgtle.password_types,timestamp with time
 zone,boolean) line 21 at RAISE
SQL statement "SELECT password_check.passcheck_hook(
    $1::pg_catalog.text,
    $2::pg_catalog.text,
    $3::pgtle.password_types,
    $4::pg_catalog.timestamptz,
    $5::pg_catalog.bool)"
```

L'output è stato formattato per ragioni di leggibilità.

L'esempio seguente mostra che il comportamento `\password` del metacomando interattivo `pgsql` è influenzato anche dall'hook di controllo della password.

```
postgres=> SET password_encryption TO 'md5';
SET
postgres=> \password
Enter new password for user "postgres":*****
```

```
Enter it again:*****
ERROR:  Cannot use passwords from the common password dictionary
CONTEXT:  PL/pgSQL function
 password_check.passcheck_hook(text,text,pgtle.password_types,timestamp with time
 zone,boolean) line 12 at RAISE
SQL statement "SELECT password_check.passcheck_hook($1::pg_catalog.text,
 $2::pg_catalog.text, $3::pgtle.password_types, $4::pg_catalog.timestamptz,
 $5::pg_catalog.bool)"
```

Puoi eliminare questa estensione TLE e disinstallare i file di origine, se lo desideri. Per ulteriori informazioni, consulta Eliminazione delle estensioni TLE da un database.

Codice di hook di controllo della password

Il codice di esempio mostrato qui definisce le specifiche per l'estensione TLE `my_password_check_rules`. Quando copi questo codice e lo incolli nel database, il codice dell'estensione `my_password_check_rules` viene caricato nel database e l'hook `password_check` viene registrato per essere utilizzato dall'estensione.

```
SELECT pgtle.install_extension (
  'my_password_check_rules',
  '1.0',
  'Do not let users use the 10 most commonly used passwords',
$_pgtle_$
  CREATE SCHEMA password_check;
  REVOKE ALL ON SCHEMA password_check FROM PUBLIC;
  GRANT USAGE ON SCHEMA password_check TO PUBLIC;

  CREATE TABLE password_check.bad_passwords (plaintext) AS
  VALUES
    ('123456'),
    ('password'),
    ('12345678'),
    ('qwerty'),
    ('123456789'),
    ('12345'),
    ('1234'),
    ('111111'),
    ('1234567'),
    ('dragon');
  CREATE UNIQUE INDEX ON password_check.bad_passwords (plaintext);
```

```
  CREATE FUNCTION password_check.passcheck_hook(username text, password text,
 password_type pgtle.password_types, valid_until timestamptz, valid_null boolean)
  RETURNS void AS $$
    DECLARE
      invalid bool := false;
    BEGIN
      IF password_type = 'PASSWORD_TYPE_MD5' THEN
        SELECT EXISTS(
          SELECT 1
          FROM password_check.bad_passwords bp
          WHERE ('md5' || md5(bp.plaintext || username)) = password
        ) INTO invalid;
        IF invalid THEN
          RAISE EXCEPTION 'Cannot use passwords from the common password dictionary';
        END IF;
      ELSIF password_type = 'PASSWORD_TYPE_PLAINTEXT' THEN
        SELECT EXISTS(
          SELECT 1
          FROM password_check.bad_passwords bp
          WHERE bp.plaintext = password
        ) INTO invalid;
        IF invalid THEN
          RAISE EXCEPTION 'Cannot use passwords from the common password dictionary';
        END IF;
      END IF;
    END
  $$ LANGUAGE plpgsql SECURITY DEFINER;

  GRANT EXECUTE ON FUNCTION password_check.passcheck_hook TO PUBLIC;

  SELECT pgtle.register_feature('password_check.passcheck_hook', 'passcheck');
$_pgtle_$
);
```

Informazioni di riferimento sulle funzioni per Trusted Language Extensions per PostgreSQL

Esamina la seguente documentazione di riferimento sulle funzioni disponibili in Trusted Language Extensions per PostgreSQL. Usa queste funzioni per installare, registrare, aggiornare e gestire le tue estensioni TLE, ovvero le estensioni di PostgreSQL che sviluppi utilizzando il kit di sviluppo Trusted Language Extensions.

Funzioni

- pgtle.available_extensions
- pgtle.available_extension_versions
- pgtle.extension_update_paths
- pgtle.install_extension
- pgtle.install_update_path
- pgtle.register_feature
- pgtle.register_feature_if_not_exists
- pgtle.set_default_version
- pgtle.uninstall_extension(name)
- pgtle.uninstall_extension(name, version)
- pgtle.uninstall_extension_if_exists
- pgtle.uninstall_update_path
- pgtle.uninstall_update_path_if_exists
- pgtle.unregister_feature
- pgtle.unregister_feature_if_exists

pgtle.available_extensions

La funzione `pgtle.available_extensions` è progettata per restituire un set. Restituisce tutte le estensioni TLE disponibili nel database. Ogni riga restituita contiene informazioni su una singola estensione TLE.

Prototipo di funzione

```
pgtle.available_extensions()
```

Ruolo

Nessuna.

Argomenti

Nessuna.

Output

- `name`: il nome dell’estensione TLE.
- `default_version`: la versione dell'estensione TLE da usare quando la funzione `CREATE EXTENSION` viene chiamata senza una versione specifica.
- `description`: una descrizione più dettagliata dell'estensione TLE.

Esempio di utilizzo

```
SELECT * FROM pgtle.available_extensions();
```

pgtle.available_extension_versions

La funzione `available_extension_versions` è progettata per restituire un set. Restituisce l'elenco di tutte le estensioni TLE disponibili e le relative versioni. Ogni riga contiene informazioni su una versione specifica dell'estensione TLE indicata, incluso se è richiesto un ruolo specifico.

Prototipo di funzione

```
pgtle.available_extension_versions()
```

Ruolo

Nessuna.

Argomenti

Nessuna.

Output

- `name`: il nome dell’estensione TLE.
- `version`: la versione dell’estensione TLE.
- `superuser`: questo valore è sempre `false` per le estensioni TLE. Le autorizzazioni necessarie per creare o aggiornare l'estensione TLE sono uguali a quelle per creare altri oggetti nel database specificato.
- `trusted`: questo valore è sempre `false` per un'estensione TLE.

- `relocatable`: questo valore è sempre `false` per un'estensione TLE.
- `schema`: specifica il nome dello schema in cui è installata l'estensione TLE.
- `requires`: un array contenente i nomi di altre estensioni necessarie a questa estensione TLE.
- `description`: una descrizione dettagliata dell'estensione TLE.

Per ulteriori informazioni sulle estensioni di PostgreSQL, consulta Packaging Related Objects into an Extension > Extension Files (Creazione di pacchetti di oggetti correlati in un'estensione > File di estensione) nella documentazione PostgreSQL.

Esempio di utilizzo

```
SELECT * FROM pgtle.available_extension_versions();
```

pgtle.extension_update_paths

La funzione `extension_update_paths` è progettata per restituire un set. Restituisce l'elenco di tutti i possibili percorsi di aggiornamento per un'estensione TLE. Ogni riga include gli aggiornamenti o i downgrade disponibili per l'estensione TLE.

Prototipo di funzione

```
pgtle.extension_update_paths(name)
```

Ruolo

Nessuna.

Argomenti

`name`: il nome dell'estensione TLE da cui ottenere i percorsi di aggiornamento.

Output

- `source`: la versione di origine di un aggiornamento.
- `target`: la versione di destinazione di un aggiornamento.
- `path`: il percorso di aggiornamento utilizzato per aggiornare un'estensione TLE dalla versione `source` alla versione `target`, ad esempio`0.1--0.2`.

Esempio di utilizzo

```
SELECT * FROM pgtle.extension_update_paths('your-TLE');
```

pgtle.install_extension

La funzione `install_extension` consente di installare gli artefatti che costituiscono l'estensione TLE nel database, dopodiché può essere creata utilizzando il comando `CREATE EXTENSION`.

Prototipo di funzione

```
pgtle.install_extension(name text, version text, description text, ext text, requires
 text[] DEFAULT NULL::text[])
```

Ruolo

Nessuna.

Argomenti

- `name`: il nome dell'estensione TLE. Questo valore viene utilizzato per la chiamata `CREATE EXTENSION`.
- `version`: la versione dell'estensione TLE.
- `description`: una descrizione dettagliata dell'estensione TLE. Questa descrizione viene visualizzata nel campo `comment` in `pgtle.available_extensions()`.
- `ext`: il contenuto dell'estensione TLE. Questo valore include gli oggetti, come le funzioni.
- `requires`: un parametro facoltativo che specifica le dipendenze per l'estensione TLE. L'estensione `pg_tle` viene aggiunta automaticamente come dipendenza.

Molti di questi argomenti sono uguali a quelli inclusi in un file di controllo delle estensioni per l'installazione di un'estensione di PostgreSQL nel file system di un'istanza PostgreSQL. Per ulteriori informazioni, consulta Extension Files (File di estensione) in Packaging Related Objects into an Extension (Creazione di pacchetti di oggetti correlati in un'estensione) nella documentazione PostgreSQL.

Output

Questa funzione restituisce `OK` in caso di esito positivo e `NULL` in caso di errore.

- `OK`: l'estensione TLE è stata installata correttamente nel database.
- `NULL`: l'estensione TLE non è stata installata correttamente nel database.

Esempio di utilizzo

```
SELECT pgtle.install_extension(
 'pg_tle_test',
 '0.1',
 'My first pg_tle extension',
$_pgtle_$
  CREATE FUNCTION my_test()
  RETURNS INT
  AS $$
    SELECT 42;
  $$ LANGUAGE SQL IMMUTABLE;
$_pgtle_$
);
```

pgtle.install_update_path

La funzione `install_update_path` fornisce il percorso di aggiornamento tra due diverse versioni di un'estensione TLE. Questa funzione consente agli utenti dell'estensione TLE di aggiornarne la versione utilizzando la sintassi `ALTER EXTENSION ... UPDATE`.

Prototipo di funzione

```
pgtle.install_update_path(name text, fromvers text, tovers text, ext text)
```

Ruolo

`pgtle_admin`

Argomenti

- `name`: il nome dell'estensione TLE. Questo valore viene utilizzato per la chiamata `CREATE EXTENSION`.
- `fromvers`: la versione di origine dell'estensione TLE per l'aggiornamento.
- `tovers`: la versione di destinazione dell'estensione TLE per l'aggiornamento.
- `ext`: i contenuti dell'aggiornamento. Questo valore include gli oggetti, come le funzioni.

Output

Nessuna.

Esempio di utilizzo

```
SELECT pgtle.install_update_path('pg_tle_test', '0.1', '0.2',
  $_pgtle_$
    CREATE OR REPLACE FUNCTION my_test()
    RETURNS INT
    AS $$
      SELECT 21;
    $$ LANGUAGE SQL IMMUTABLE;
  $_pgtle_$
);
```

pgtle.register_feature

La funzione `register_feature` aggiunge la funzionalità PostgreSQL interna specificata alla tabella `pgtle.feature_info`. Gli hook PostgreSQL sono un esempio di funzionalità interna di PostgreSQL. Il kit di sviluppo Trusted Language Extensions supporta l'uso degli hook PostgreSQL. Attualmente, questa funzione supporta la seguente funzionalità.

- `passcheck`: registra l'hook di verifica della password con la procedura o la funzione che personalizza il comportamento di verifica della password di PostgreSQL.

Prototipo di funzione

```
pgtle.register_feature(proc regproc, feature pg_tle_feature)
```

Ruolo

`pgtle_admin`

Argomenti

- `proc`: il nome di una procedura o funzione memorizzata da utilizzare per la funzionalità.
- `feature`: il nome della funzionalità `pg_tle` (ad esempio `passcheck`) da registrare con la funzione.

Output

Nessuna.

Esempio di utilizzo

```
SELECT pgtle.register_feature('pw_hook', 'passcheck');
```

pgtle.register_feature_if_not_exists

La funzione `pgtle.register_feature_if_not_exists` aggiunge la funzionalità PostgreSQL specificata alla tabella `pgtle.feature_info` e identifica l'estensione TLE o un'altra procedura o funzione che utilizza la funzionalità. Per ulteriori informazioni sugli hook e su Trusted Language Extensions, consulta Utilizzo di hook PostgreSQL con le estensioni TLE.

Prototipo di funzione

```
pgtle.register_feature_if_not_exists(proc regproc, feature pg_tle_feature)
```

Ruolo

`pgtle_admin`

Argomenti

- `proc`: il nome di una stored procedure o una funzione che contiene la logica (codice) da utilizzare come funzionalità dell'estensione TLE. Ad esempio, il codice `pw_hook`.
- `feature`: il nome della funzionalità PostgreSQL da registrare per la funzione TLE. Attualmente, l'unica funzionalità disponibile è l'hook `passcheck`. Per ulteriori informazioni, consulta Hook di verifica della password (passcheck).

Output

Restituisce `true` dopo aver registrato la funzionalità per l'estensione specificata. Restituisce `false` se la funzionalità è già registrata.

Esempio di utilizzo

```
SELECT pgtle.register_feature_if_not_exists('pw_hook', 'passcheck');
```

pgtle.set_default_version

La funzione `set_default_version` ti consente di specificare una `default_version` per la tua estensione TLE. È possibile utilizzare questa funzione per definire un percorso di aggiornamento e designare la versione come predefinita per l'estensione TLE. Quando gli utenti del database specificano l'estensione TLE nei comandi `CREATE EXTENSION` e `ALTER EXTENSION ... UPDATE`, la versione specificata dell'estensione TLE viene creata nel database per tali utenti.

Questa funzione restituisce `true` in caso di esito positivo. Se l'estensione TLE specificata nell'argomento `name` non esiste, la funzione restituisce un errore. Analogamente, se la `version` dell'estensione TLE non esiste, la funzione restituisce un errore.

Prototipo di funzione

```
pgtle.set_default_version(name text, version text)
```

Ruolo

`pgtle_admin`

Argomenti

- `name`: il nome dell'estensione TLE. Questo valore viene utilizzato per la chiamata `CREATE EXTENSION`.
- `version`: la versione dell'estensione TLE da impostare come predefinita.

Output

- `true`: quando l'impostazione della versione predefinita ha esito positivo, la funzione restituisce `true`.
- `ERROR`: restituisce un messaggio di errore se non esiste un'estensione TLE con il nome o la versione specificati.

Esempio di utilizzo

```
SELECT * FROM pgtle.set_default_version('my-extension', '1.1');
```

pgtle.uninstall_extension(name)

La funzione `uninstall_extension` rimuove tutte le versioni di un'estensione TLE da un database. Questa funzione impedisce alle future chiamate `CREATE EXTENSION` di installare l'estensione TLE. Se l'estensione TLE non esiste nel database, viene generato un errore.

La funzione `uninstall_extension` non elimina un'estensione TLE attualmente attiva nel database. Per rimuovere un'estensione TLE attualmente attiva, è necessario chiamare esplicitamente `DROP EXTENSION` per rimuoverla.

Prototipo di funzione

```
pgtle.uninstall_extension(extname text)
```

Ruolo

`pgtle_admin`

Argomenti

- `extname`: il nome dell'estensione TLE da disinstallare. Questo nome è quello utilizzato con `CREATE EXTENSION` per caricare l'estensione TLE da usare in un determinato database.

Output

Nessuna.

Esempio di utilizzo

```
SELECT * FROM pgtle.uninstall_extension('pg_tle_test');
```

pgtle.uninstall_extension(name, version)

La funzione `uninstall_extension(name, version)` rimuove la versione specificata dell'estensione TLE dal database. Questa funzione impedisce alle chiamate `CREATE EXTENSION` e `ALTER EXTENSION` di installare o aggiornare un'estensione TLE alla versione specificata. Questa funzione rimuove anche tutti i percorsi di aggiornamento per la versione specificata dell'estensione TLE. La funzione non disinstalla l'estensione TLE se è attualmente attiva nel database. È necessario chiamare esplicitamente `DROP EXTENSION` per rimuovere l'estensione TLE. Per disinstallare tutte le versioni di un'estensione TLE, consulta pgtle.uninstall_extension(name).

Prototipo di funzione

```
pgtle.uninstall_extension(extname text, version text)
```

Ruolo

`pgtle_admin`

Argomenti

- `extname`: il nome dell'estensione TLE. Questo valore viene utilizzato per la chiamata `CREATE EXTENSION`.
- `version`: la versione dell'estensione TLE da disinstallare dal database.

Output

Nessuna.

Esempio di utilizzo

```
SELECT * FROM pgtle.uninstall_extension('pg_tle_test', '0.2');
```

pgtle.uninstall_extension_if_exists

La funzione `uninstall_extension_if_exists` rimuove tutte le versioni di un'estensione TLE da un determinato database. Se l'estensione TLE non esiste, la funzione non restituisce alcun avviso (non viene generato alcun messaggio di errore). Se l'estensione specificata è attualmente attiva in un database, non viene eliminata dalla funzione. È necessario chiamare esplicitamente `DROP EXTENSION` per rimuovere l'estensione TLE prima di utilizzare questa funzione per disinstallarne gli artefatti.

Prototipo di funzione

```
pgtle.uninstall_extension_if_exists(extname text)
```

Ruolo

`pgtle_admin`

Argomenti

- `extname`: il nome dell'estensione TLE. Questo valore viene utilizzato per la chiamata `CREATE EXTENSION`.

Output

La funzione `uninstall_extension_if_exists` restituisce `true` dopo aver disinstallato l'estensione specificata. Se l'estensione specificata non esiste, la funzione restituisce `false`.

- `true`: restituisce `true` dopo aver disinstallato l'estensione TLE.
- `false`: restituisce `false` quando l'estensione TLE non esiste nel database.

Esempio di utilizzo

```
SELECT * FROM pgtle.uninstall_extension_if_exists('pg_tle_test');
```

pgtle.uninstall_update_path

La funzione `uninstall_update_path` rimuove il percorso di aggiornamento specificato da un'estensione TLE. In tal modo `ALTER EXTENSION ... UPDATE TO` non può utilizzarlo come percorso di aggiornamento.

Se l'estensione TLE è attualmente utilizzata da una delle versioni di questo percorso di aggiornamento, rimane nel database.

Se il percorso di aggiornamento specificato non esiste, la funzione genera un errore.

Prototipo di funzione

```
pgtle.uninstall_update_path(extname text, fromvers text, tovers text)
```

Ruolo

`pgtle_admin`

Argomenti

- `extname`: il nome dell'estensione TLE. Questo valore viene utilizzato per la chiamata `CREATE EXTENSION`.

- `fromvers`: la versione di origine dell'estensione TLE utilizzata nel percorso di aggiornamento.
- `tovers`: la versione di destinazione dell'estensione TLE utilizzata nel percorso di aggiornamento.

Output

Nessuna.

Esempio di utilizzo

```
SELECT * FROM pgtle.uninstall_update_path('pg_tle_test', '0.1', '0.2');
```

pgtle.uninstall_update_path_if_exists

La funzione `uninstall_update_path_if_exists` è simile a `uninstall_update_path` in quanto rimuove il percorso di aggiornamento specificato da un'estensione TLE. Tuttavia, se il percorso di aggiornamento non esiste, questa funzione non genera un messaggio di errore e restituisce `false`.

Prototipo di funzione

```
pgtle.uninstall_update_path_if_exists(extname text, fromvers text, tovers text)
```

Ruolo

`pgtle_admin`

Argomenti

- `extname`: il nome dell'estensione TLE. Questo valore viene utilizzato per la chiamata `CREATE EXTENSION`.
- `fromvers`: la versione di origine dell'estensione TLE utilizzata nel percorso di aggiornamento.
- `tovers`: la versione di destinazione dell'estensione TLE utilizzata nel percorso di aggiornamento.

Output

- `true`: la funzione ha aggiornato correttamente il percorso dell'estensione TLE.
- `false`: la funzione non è stata in grado di aggiornare il percorso dell'estensione TLE.

Esempio di utilizzo

```
SELECT * FROM pgtle.uninstall_update_path_if_exists('pg_tle_test', '0.1', '0.2');
```

pgtle.unregister_feature

La funzione `unregister_feature` fornisce un modo per rimuovere le funzioni registrate per utilizzare la funzionalità `pg_tle`, ad esempio gli hook. Per ulteriori informazioni sulla registrazione di una funzionalità, consulta pgtle.register_feature.

Prototipo di funzione

```
pgtle.unregister_feature(proc regproc, feature pg_tle_features)
```

Ruolo

`pgtle_admin`

Argomenti

- `proc`: il nome di una funzione memorizzata da registrare con una funzionalità `pg_tle`.
- `feature`: il nome della funzionalità `pg_tle` da registrare con la funzione. Ad esempio, `passcheck` è una funzionalità che può essere registrata per essere utilizzata dalle estensioni Trusted Language Extensions sviluppate. Per ulteriori informazioni, consulta Hook di verifica della password (passcheck).

Output

Nessuna.

Esempio di utilizzo

```
SELECT * FROM pgtle.unregister_feature('pw_hook', 'passcheck');
```

pgtle.unregister_feature_if_exists

La funzione `unregister_feature` fornisce un modo per rimuovere le funzioni registrate per utilizzare la funzionalità `pg_tle`, ad esempio gli hook. Per ulteriori informazioni, consulta Utilizzo di

hook PostgreSQL con le estensioni TLE. Restituisce `true` dopo aver completato l'annullamento della registrazione della funzionalità. Restituisce `false` se la funzionalità non è stata registrata.

Per informazioni sulla registrazione delle funzionalità `pg_tle` per le estensioni TLE, consulta pgtle.register_feature.

Prototipo di funzione

```
pgtle.unregister_feature_if_exists('proc regproc', 'feature pg_tle_features')
```

Ruolo

`pgtle_admin`

Argomenti

- `proc`: il nome della funzione memorizzata che è stata registrata per includere una funzionalità `pg_tle`.
- `feature`: il nome della funzionalità `pg_tle` registrata con l'estensione Trusted Language Extensions.

Output

Restituisce `true` o `false`, come indicato di seguito.

- `true`: la funzione ha completato l'annullamento della registrazione della funzionalità dall'estensione.
- `false`: la funzione non è stata in grado di annullare la registrazione della funzionalità dall'estensione TLE.

Esempio di utilizzo

```
SELECT * FROM pgtle.unregister_feature_if_exists('pw_hook', 'passcheck');
```

Riferimento sugli hook per Trusted Language Extensions per PostgreSQL

Trusted Language Extensions per PostgreSQL supporta gli hook PostgreSQL. Un hook è un meccanismo di callback interno che gli sviluppatori possono usare per estendere le funzionalità di base di PostgreSQL. Utilizzando gli hook, gli sviluppatori possono implementare le proprie funzioni

o procedure da utilizzare durante varie operazioni del database, modificando così il comportamento di PostgreSQL in qualche modo. Ad esempio, puoi usare un hook `passcheck` per personalizzare il modo in cui PostgreSQL gestisce le password fornite durante la creazione o la modifica delle password per gli utenti (ruoli).

Consultare la seguente documentazione per informazioni sull'hook passcheck disponibile per le estensioni TLE. Per ulteriori informazioni sugli hook disponibili, incluso l'hook di autenticazione del client, consulta Trusted Language Extensions hooks.

Hook di verifica della password (passcheck)

L'hook `passcheck` viene utilizzato per personalizzare il comportamento di PostgreSQL durante il processo di verifica della password per i seguenti comandi SQL e il metacomando `psql`.

- `CREATE ROLE username ...PASSWORD`: per ulteriori informazioni, consulta CREATE ROLE nella documentazione di PostgreSQL.
- `ALTER ROLE username...PASSWORD`: per ulteriori informazioni, consulta ALTER ROLE nella documentazione di PostgreSQL.
- `\password username`: questo metacomando `psql` interattivo modifica in modo sicuro la password per l'utente specificato eseguendo l'hashing della password prima di utilizzare in modo trasparente la sintassi `ALTER ROLE ... PASSWORD`. Il metacomando è un wrapper sicuro per il comando `ALTER ROLE ... PASSWORD`, quindi l'hook si applica al comportamento del metacomando `psql`.

Per vedere un esempio, consulta Codice di hook di controllo della password.

Indice

- Prototipo di funzione
- Argomenti
- Configurazione
- Note per l'utilizzo

Prototipo di funzione

```
passcheck_hook(username text, password text, password_type pgtle.password_types,
 valid_until timestamptz, valid_null boolean)
```

Argomenti

La funzione dell'hook `passcheck` accetta i seguenti argomenti:

- `username`: il nome (come testo) del ruolo (nome utente) che imposta una password.
- `password`: la password in chiaro o con hash. La password immessa deve corrispondere al tipo specificato in `password_type`.
- `password_type`: specifica il formato `pgtle.password_type` della password, che può essere costituito da una delle seguenti opzioni.
  - `PASSWORD_TYPE_PLAINTEXT`: una password in testo semplice.
  - `PASSWORD_TYPE_MD5`: una password che è stata sottoposta a hash utilizzando l'algoritmo MD5 (message digest 5).
  - `PASSWORD_TYPE_SCRAM_SHA_256`: una password che è stata sottoposta a hash utilizzando l'algoritmo SCRAM-SHA-256.
- `valid_until`: specifica l'ora in cui la password diventa non valida. Questo argomento è facoltativo. Se utilizzi questo argomento, specifica l'ora come valore `timestamptz`.
- `valid_null`: se questo booleano è impostato su `true`, l'opzione `valid_until` è impostata su `NULL`.

Configurazione

La funzione `pgtle.enable_password_check` controlla se l'hook passcheck è attivo. L'hook passcheck ha tre possibili impostazioni.

- `off`: disattiva l'hook `passcheck` di verifica della password. Si tratta del valore di default.
- on: attiva l'hook `passcode` di verifica della password in modo che le password vengano confrontate con la tabella.
- `require`: richiede la definizione di un hook di verifica della password.

Note per l’utilizzo

Per attivare o disattivare l'hook `passcheck`, è necessario modificare il gruppo di parametri database personalizzato per l'istanza di scrittura del cluster database Aurora PostgreSQL.

Per Linux, macOS o Unix:

```
aws rds modify-db-parameter-group \
```

```
    --region aws-region \
    --db-parameter-group-name your-custom-parameter-group \
    --parameters
 "ParameterName=pgtle.enable_password_check,ParameterValue=on,ApplyMethod=immediate"
```

Per Windows:

```
aws rds modify-db-parameter-group ^
    --region aws-region ^
    --db-parameter-group-name your-custom-parameter-group ^
    --parameters
 "ParameterName=pgtle.enable_password_check,ParameterValue=on,ApplyMethod=immediate"
```

Riferimento Amazon Aurora PostgreSQL

Nei seguenti argomenti sono disponibili informazioni su regole di confronto, funzioni, parametri ed eventi di attesa in Amazon Aurora PostgreSQL.

Argomenti

- Fascicolazioni Aurora PostgreSQL per EBCDIC e altre migrazioni del mainframe
- Regole di confronto supportate in Aurora PostgreSQL
- Riferimenti relativi alle funzioni Aurora PostgreSQL
- Amazon Aurora PostgreSQL parametri
- Eventi di attesa Amazon Aurora PostgreSQL

Fascicolazioni Aurora PostgreSQL per EBCDIC e altre migrazioni del mainframe

La migrazione delle applicazioni mainframe a nuove piattaforme quali AWS preserva idealmente il comportamento dell'applicazione. Per mantenere il comportamento dell'applicazione su una nuova piattaforma esattamente come era sul mainframe, è necessario che i dati migrati vengano fascicolati utilizzando le stesse regole di fascicolazione e ordinamento. Ad esempio, molte soluzioni di migrazione Db2 spostano valori nulli in u0180 (posizione Unicode 0180), pertanto queste fascicolazioni ordinano prima u0180. Questo è un esempio di come le fascicolazioni possono variare rispetto all'origine del mainframe e perché è necessario scegliere una fascicolazione che si meglio mappata alla fascicolazione EBCDIC originale.

Aurora PostgreSQL 14.3 e versioni successive forniscono molte fascicolazioni ICU ed EBCDIC per supportare tale migrazione verso AWS utilizzando il servizio Modernizzazione del mainframe AWS. Per ulteriori informazioni su questo servizio, consulta Cos'è Modernizzazione del mainframe AWS?

Nella tabella seguente, sono disponibili le fascicolazioni fornite da Aurora PostgreSQL. Queste fascicolazioni seguono le regole EBCDIC e garantiscono che le applicazioni mainframe funzionino allo stesso modo su AWS come nell'ambiente mainframe. Il nome della fascicolazione include la pagina di codice pertinente, (cpnnnn), per poter scegliere la fascicolazione appropriata per l'origine del mainframe. Ad esempio, utilizza `en-US-cp037-x-icu` per ottenere il comportamento di fascicolazione per i dati EBCDIC provenienti da un'applicazione mainframe che utilizza la pagina codice 037.

Fascicolazioni EBCDIC	Fascicolazioni AWS Blu Age	Fascicolazioni AWS Micro Focus
da-DK-cp1142-x-icu	da-DK-cp1142-x-icu	da-DK-cp1142m-x-icu
da-DK-cp277-x-icu	da-DK-cp277b-x-icu	–
de-DE-cp1141-x-icu	de-DE-cp1141b-x-icu	de-DE-cp1141m-x-icu
de-DE-cp273-x-icu	de-DE-cp273b-x-icu	–
en-GB-cp1146-x-icu	en-GB-cp1146b-x-icu	en-GB-cp1146m-x-icu
en-GB-cp285-x-icu	en-GB-cp285b-x-icu	–
en-US-cp037-x-icu	en-US-cp037b-x-icu	–
en-US-cp1140-x-icu	en-US-cp1140b-x-icu	en-US-cp1140m-x-icu
es-ES-cp1145-x-icu	es-ES-cp1145b-x-icu	es-ES-cp1145b-x-icu
es-ES-cp284-x-icu	es-ES-cp284b-x-icu	–
fi-FI-cp1143-x-icu	fi-FI-cp1143b-x-icu	fi-FI-cp1143m-x-icu
fi-FI-cp278-x-icu	fi-FI-cp278b-x-icu	–
FR-FR-CP1147-x-ICU	fr-FR-cp1147b-x-icu	fr-FR-cp1147m-x-icu
fr-FR-cp297-x-icu	fr-FR-cp297b-x-icu	–
it-IT-cp1144-x-icu	it-IT-cp1144b-x-icu	it-IT-cp1144b-x-icu
it-IT-cp280-x-icu	it-IT-cp280b-x-icu	–
nl-BE-cp1148-x-icu	nl-BE-cp1148b-x-icu	nl-BE-cp1148m-x-icu
nl-BE-cp500-x-icu	nl-BE-cp500b-x-icu	–

Per ulteriori informazioni su AWS Blu Age, consulta Tutorial: Runtime gestito per AWS Blu Age nella Guida per l'utente di Modernizzazione mainframe AWS.

Per ulteriori informazioni sull'utilizzo di AWS Micro Focus, consulta Tutorial: Managed Runtime for Micro Focus nella Guida per l'utente di Modernizzazione mainframe AWS.

Per ulteriori informazioni sulla gestione di fascicolazioni in PostgreSQL, consulta Collation Support nella documentazione di PostgreSQL.

Regole di confronto supportate in Aurora PostgreSQL

Le regole di confronto sono un insieme di regole che determinano il modo in cui le stringhe di caratteri archiviate nel database vengono ordinate e confrontate. Le regole di confronto svolgono un ruolo fondamentale nel sistema del computer e sono incluse come parte del sistema operativo. Le regole di confronto cambiano nel tempo quando vengono aggiunti nuovi caratteri alle lingue o quando vengono modificate le regole di ordinamento.

Le librerie di regole di confronto definiscono regole e algoritmi specifici per una regola di confronto. Le librerie di regole di confronto più popolari utilizzate in PostgreSQL sono GNU C (glibc) e Internationalization components for Unicode (ICU). Per impostazione predefinita, Aurora PostgreSQL utilizza la regola di confronto glibc che include le sequenze di ordinamento dei caratteri unicode per sequenze di caratteri multibyte.

Quando si crea un nuovo cluster database Aurora PostgreSQL, viene cercata la regola di confronto disponibile nel sistema operativo. I parametri PostgreSQL `LC_COLLATE` e `LC_CTYPE` del comando `CREATE DATABASE` vengono utilizzati per specificare una regola di confronto, che rappresenta la regola di confronto predefinita nel database. In alternativa, puoi anche usare il parametro `LOCALE` in `CREATE DATABASE` per impostare questi parametri e determinare la regole di confronto predefinita per le stringhe di caratteri nel database e le regole per classificare i caratteri come lettere, numeri o simboli. Puoi anche scegliere una regola di confronto da utilizzare per una colonna, un indice o una query.

Aurora PostgreSQL dipende dalla libreria glibc del sistema operativo per il supporto della regola di confronto. L'istanza Aurora PostgreSQL viene aggiornata periodicamente con le versioni più recenti del sistema operativo. Questi aggiornamenti a volte includono una nuova versione della libreria glibc. Raramente, le versioni più recenti della libreria glibc modificano l'ordinamento o la regola di confronto di alcuni caratteri e pertanto i dati possono essere ordinati in modo diverso o produrre voci di indice non valide. Se si riscontrano problemi di ordinamento per la regola di confronto durante un aggiornamento, potrebbe essere necessario ricostruire gli indici.

Per ridurre il possibile impatto degli aggiornamenti della libreria glibc, Aurora PostgreSQL ora include una libreria di regole di confronto predefinita indipendente. Questa libreria di regole confronto è

disponibile in Aurora PostgreSQL 14.6, 13.9, 12.13, 11.18 e versioni secondarie successive. È compatibile con glibc 2.26-59.amzn2 e fornisce stabilità dell'ordinamento per evitare risultati errati delle query.

Riferimenti relativi alle funzioni Aurora PostgreSQL

Di seguito, è possibile trovare un elenco delle funzioni Aurora PostgreSQL disponibili per i cluster Aurora DB che eseguono il motore di database dell'edizione compatibile con Aurora PostgreSQL. Queste funzioni Aurora PostgreSQL sono in aggiunta alle funzioni standard PostgreSQL. Per ulteriori informazioni sulle funzioni PostgreSQL standard, consulta PostgreSQL: funzioni e operatori.

Panoramica

È possibile utilizzare le seguenti funzioni per le istanze database Amazon RDS che eseguono Aurora PostgreSQL:

- aurora_db_instance_identifier
- aurora_ccm_status
- aurora_global_db_instance_status
- aurora_global_db_status
- aurora_list_builtins
- aurora_replica_status
- aurora_stat_activity
- aurora_stat_backend_waits
- aurora_stat_bgwriter
- aurora_stat_database
- aurora_stat_dml_activity
- aurora_stat_get_db_commit_latency
- aurora_stat_logical_wal_cache
- aurora_stat_memctx_usage
- aurora_stat_optimized_reads_cache
- aurora_stat_plans
- aurora_stat_reset_wal_cache
- aurora_stat_statements

- aurora_stat_system_waits
- aurora_stat_wait_event
- aurora_stat_wait_type
- aurora_version
- aurora_volume_logical_start_lsn
- aurora_wait_report

aurora_db_instance_identifier

Comunica il nome dell'istanza database cui si è connessi.

Sintassi

```
aurora_db_instance_identifier()
```

Argomenti

Nessuno

Tipo restituito

Stringa VARCHA

Note per l'utilizzo

Questa funzione visualizza il nome dell'istanza database del cluster Aurora edizione compatibile con PostgreSQL per la connessione al client di database o all'applicazione.

Questa funzione è disponibile a partire dal rilascio di PostgreSQL versioni 13.7, 12.11, 11.16, 10.21 e per tutte le altre versioni successive.

Esempi

Nell'esempio seguente vengono mostrati i risultati della chiamata della funzione `aurora_db_instance_identifier`.

```
=> SELECT aurora_db_instance_identifier();
```

```
aurora_db_instance_identifier
------------------------------
 test-my-instance-name
```

È possibile unire i risultati di questa funzione con la funzione `aurora_replica_status` per ottenere i dettagli relativi all'istanza database per la connessione. Il parametro aurora_replica_status da solo non mostra l'istanza database in uso. Nell'esempio seguente viene illustrato come fare.

```
=> SELECT *
    FROM aurora_replica_status() rt,
         aurora_db_instance_identifier() di
    WHERE rt.server_id = di;
-[ RECORD 1 ]----------------------+----------------------
server_id                         | test-my-instance-name
session_id                        | MASTER_SESSION_ID
durable_lsn                       | 88492069
highest_lsn_rcvd                  |
current_read_lsn                  |
cur_replay_latency_in_usec        |
active_txns                       |
is_current                        | t
last_transport_error              | 0
last_error_timestamp              |
last_update_timestamp             | 2022-06-03 11:18:25+00
feedback_xmin                     |
feedback_epoch                    |
replica_lag_in_msec               |
log_stream_speed_in_kib_per_second | 0
log_buffer_sequence_number        | 0
oldest_read_view_trx_id           |
oldest_read_view_lsn              |
pending_read_ios                  | 819
```

aurora_ccm_status

Visualizza lo stato del gestore della cache del cluster.

Sintassi

```
aurora_ccm_status()
```

Argomenti

Nessuna.

Tipo restituito

Record SETOF avente le seguenti colonne:

- `buffers_sent_last_minute`: il numero di buffer inviati all'istanza database di lettura designata nell'ultimo minuto.
- `buffers_found_last_minute`: il numero di buffer con accesso frequente identificati durante l'ultimo minuto.
- `buffers_sent_last_scan`: il numero di buffer inviati all'istanza di lettura designata durante l'ultima scansione completa della cache.
- `buffers_found_last_scan`: il numero di buffer ad accesso frequente inviati durante l'ultima scansione completa della cache. I buffer già memorizzati sull'istanza database di lettura designata non vengono inviati.
- `buffers_sent_current_scan`: il numero di buffer inviati durante la scansione corrente.
- `buffers_found_current_scan`: il numero di buffer ad accesso frequente identificati nella scansione corrente.
- `current_scan_progress`: il numero di buffer analizzati finora durante la scansione corrente.

Note per l'utilizzo

È possibile utilizzare questa funzione per controllare e monitorare la funzionalità di gestione della cache del cluster (CCM). Ciò può avvenire solo se la funzionalità di gestione della cache del cluster è attiva sul cluster di database Aurora PostgreSQL. Per utilizzare questa funzione, connettersi all'istanza DB di scrittura sul cluster di database Aurora PostgreSQL.

Per attivare la funzionalità di gestione della cache del cluster per un cluster di database Aurora PostgreSQL, impostare `apg_ccm_enabled` su 1 nel gruppo di parametri del cluster di database personalizzato. Per scoprire come, consulta Configurazione della gestione della cache del cluster.

La gestione della cache del cluster è attiva su un cluster di database Aurora PostgreSQL quando il cluster dispone di un'istanza di lettura Aurora configurata come segue:

- L'istanza di lettura Aurora utilizza lo stesso tipo di classe di istanza database e la stessa dimensione dell'istanza di scrittura del cluster.

- L'istanza di lettura Aurora è configurata come Tier-0 per il cluster. Se il cluster include più di un'istanza di lettura, questa istanza sarà l'unica istanza di lettura di tipo Tier-0.

L'impostazione di più di un'istanza di lettura su Tier-0 disabilita la funzionalità di gestione della cache del cluster. Quando la funzionalità di gestione della cache del cluster è disabilitata, una chiamata a questa funzione restituirà il seguente messaggio di errore:

```
ERROR: Cluster Cache Manager is disabled
```

È inoltre possibile impostare l'estensione PostgreSQL pg_buffercache affinché analizzi la cache del buffer. Per ulteriori informazioni, consulta pg_buffercache nella documentazione di PostgreSQL.

Per ulteriori informazioni, consulta Introduzione alla gestione della cache del cluster Aurora PostgreSQL.

Esempi

Nell'esempio seguente vengono mostrati i risultati della chiamata della funzione `aurora_ccm_status`. Questo primo esempio mostra le statistiche restituite dalla funzionalità di gestione della cache del cluster.

```
=> SELECT * FROM aurora_ccm_status();
 buffers_sent_last_minute | buffers_found_last_minute | buffers_sent_last_scan |
 buffers_found_last_scan | buffers_sent_current_scan | buffers_found_current_scan |
 current_scan_progress
--------------------------+---------------------------+------------------------
+-------------------------+---------------------------+---------------------------
+------------------------
                  2242000 |                   2242003 |               17920442 |
         17923410 |                  14098000 |                   14100964 |
   15877443
```

Per dettagli più completi, è possibile utilizzare la visualizzazione espansa, come illustrato di seguito:

```
\x
Expanded display is on.
SELECT *  FROM aurora_ccm_status();
[ RECORD 1 ]----------------------+---------
buffers_sent_last_minute          | 2242000
buffers_found_last_minute         | 2242003
```

```
buffers_sent_last_scan              | 17920442
buffers_found_last_scan             | 17923410
buffers_sent_current_scan           | 14098000
buffers_found_current_scan          | 14100964
current_scan_progress               | 15877443
```

Questo esempio mostra come controllare la frequenza e la percentuale degli avvii a caldo.

```
=> SELECT buffers_sent_last_minute * 8/60 AS warm_rate_kbps,
100 * (1.0-buffers_sent_last_scan/buffers_found_last_scan) AS warm_percent
FROM aurora_ccm_status ();
 warm_rate_kbps | warm_percent
----------------+--------------
 16523 |        100.0
```

aurora_global_db_instance_status

Visualizza lo stato di tutte le istanze Aurora, incluse le repliche in un cluster di database globale Aurora.

Sintassi

```
aurora_global_db_instance_status()
```

Argomenti

Nessuno

Tipo restituito

Record SETOF avente le seguenti colonne:

- `server_id`: l'identificatore dell'istanza database.
- `session_id`: un identificatore univoco per la sessione. Il valore di `MASTER_SESSION_ID` identifica l'istanza database di lettura (primaria).
- `aws_region`: l'Regione AWS in cui viene eseguita l'istanza database globale corrente. Per l'elenco delle regioni , consulta Disponibilità nelle regioni.
- `durable_lsn`: il numero di sequenza di log (LSN) reso durevole nell'archiviazione. Numero di sequenza di log (LSN) è un numero sequenziale univoco che identifica un record nel log delle

transazioni del database. Gli LSN sono ordinati in modo tale che un LSN più grande rappresenti una transazione successiva.

- `highest_lsn_rcvd`: l'LSN più alto che l'istanza database ha ricevuto dall'istanza database di scrittura.
- `feedback_epoch`: l'epoca utilizzata dall'istanza database quando genera informazioni standby a caldo. Uno standby a caldo è un'istanza database che supporta connessioni e query mentre il database primario è in modalità di ripristino o standby. Le informazioni relative allo standby a caldo includono l'epoca (point-in-time) e altri dettagli sull'istanza database utilizzata come standby a caldo. Per ulteriori informazioni, consulta Hot Standby nella documentazione di PostgreSQL.
- `feedback_xmin`: l'ID della transazione attiva più basso utilizzato dall'istanza database.
- `oldest_read_view_lsn`: l'LSN più vecchio utilizzato dall'istanza database per le operazioni di lettura dallo storage.
- `visibility_lag_in_msec`: il ritardo dell'istanza database corrente rispetto all'istanza database di scrittura in millisecondi.

Note per l'utilizzo

Questa funzione mostra le statistiche di replica per un cluster di database Aurora. Per ogni istanza database Aurora PostgreSQL nel cluster, la funzione mostra una riga di dati che include eventuali repliche tra regioni in una configurazione di database globale.

È possibile eseguire questa funzione da qualsiasi istanza in un cluster di database Aurora PostgreSQL o Aurora PostgreSQL. La funzione restituisce dettagli sul ritardo per tutte le istanze di replica.

Per ulteriori informazioni sul monitoraggio del ritardo utilizzando questa funzione (`aurora_global_db_instance_status`) o `aurora_global_db_status`, consulta Monitoraggio dei database globali Aurora basati su PostgreSQL.

Per informazioni sui database globali Aurora, consulta Panoramica di Database globale Amazon Aurora.

Per iniziare a utilizzare i database globali Aurora, consulta Nozioni di base sul Database globale Amazon Aurora o Domande frequenti su Amazon Aurora.

Esempi

Questo esempio mostra le statistiche delle istanze tra regioni.

```
=> SELECT *
  FROM aurora_global_db_instance_status();
               server_id                |              session_id              |
  aws_region  | durable_lsn | highest_lsn_rcvd | feedback_epoch | feedback_xmin |
 oldest_read_view_lsn | visibility_lag_in_msec
----------------------------------------+--------------------------------------
+-------------+------------+------------------+----------------+---------------
+----------------------+------------------------
 db-119-001-instance-01                  | MASTER_SESSION_ID                    | eu-
west-1    |  2534560273 |           [NULL] |         [NULL] |        [NULL] |
     [NULL] |                [NULL]
 db-119-001-instance-02                  | 4ecff34d-d57c-409c-ba28-278b31d6fc40 | eu-
west-1    |  2534560266 |       2534560273 |              0 |      19669196 |
 2534560266 |                     6
 db-119-001-instance-03                  | 3e8a20fc-be86-43d5-95e5-bdf19d27ad6b | eu-
west-1    |  2534560266 |       2534560273 |              0 |      19669196 |
 2534560266 |                     6
 db-119-001-instance-04                  | fc1b0023-e8b4-4361-bede-2a7e926cead6 | eu-
west-1    |  2534560266 |       2534560273 |              0 |      19669196 |
 2534560254 |                    23
 db-119-001-instance-05                  | 30319b74-3f08-4e13-9728-e02aa1aa8649 | eu-
west-1    |  2534560266 |       2534560273 |              0 |      19669196 |
 2534560254 |                    23
 db-119-001-global-instance-1            | a331ffbb-d982-49ba-8973-527c96329c60 | eu-
central-1 |  2534560254 |       2534560266 |              0 |      19669196 |
 2534560247 |                   996
 db-119-001-global-instance-1            | e0955367-7082-43c4-b4db-70674064a9da | eu-
west-2    |  2534560254 |       2534560266 |              0 |      19669196 |
 2534560247 |                    14
 db-119-001-global-instance-1-eu-west-2a  | 1248dc12-d3a4-46f5-a9e2-85850491a897 | eu-
west-2    |  2534560254 |       2534560266 |              0 |      19669196 |
 2534560247 |                     0
```

Questo esempio mostra come controllare il ritardo delle repliche globali in millisecondi.

```
=> SELECT CASE
          WHEN 'MASTER_SESSION_ID' = session_id THEN 'Primary'
         ELSE 'Secondary'
      END AS global_role,
      aws_region,
      server_id,
      visibility_lag_in_msec
  FROM aurora_global_db_instance_status()
```

```
  ORDER BY 1, 2, 3;
   global_role |  aws_region  |               server_id                |
 visibility_lag_in_msec
-------------+-------------+----------------------------------------
+------------------------
 Primary     | eu-west-1    | db-119-001-instance-01                 |
 [NULL]
 Secondary   | eu-central-1 | db-119-001-global-instance-1           |
     13
 Secondary   | eu-west-1    | db-119-001-instance-02                 |
     10
 Secondary   | eu-west-1    | db-119-001-instance-03                 |
      9
 Secondary   | eu-west-1    | db-119-001-instance-04                 |
      2
 Secondary   | eu-west-1    | db-119-001-instance-05                 |
     18
 Secondary   | eu-west-2    | db-119-001-global-instance-1           |
     14
 Secondary   | eu-west-2    | db-119-001-global-instance-1-eu-west-2a |
     13
```

Questo esempio mostra come controllare il ritardo minimo, massimo e medio per Regione AWS rispetto alla configurazione del database globale.

```
=> SELECT 'Secondary' global_role,
       aws_region,
       min(visibility_lag_in_msec) min_lag_in_msec,
       max(visibility_lag_in_msec) max_lag_in_msec,
       round(avg(visibility_lag_in_msec),0) avg_lag_in_msec
  FROM aurora_global_db_instance_status()
 WHERE aws_region NOT IN (SELECT   aws_region
                            FROM aurora_global_db_instance_status()
                           WHERE session_id='MASTER_SESSION_ID')
                        GROUP BY aws_region
UNION ALL
SELECT  'Primary' global_role,
        aws_region,
        NULL,
        NULL,
        NULL
    FROM aurora_global_db_instance_status()
   WHERE session_id='MASTER_SESSION_ID'
```

```
ORDER BY 1, 5;
 global_role |  aws_region  | min_lag_in_msec | max_lag_in_msec | avg_lag_in_msec
-------------+--------------+-----------------+-----------------+-----------------
 Primary     | eu-west-1    |          [NULL] |          [NULL] |          [NULL]
 Secondary   | eu-central-1 |             133 |             133 |             133
 Secondary   | eu-west-2    |               0 |             495 |             248
```

aurora_global_db_status

Visualizza informazioni su vari aspetti del ritardo del database globale Aurora, in particolare il ritardo dell'archiviazione Aurora sottostante (il cosiddetto ritardo di durabilità) e il ritardo rispetto all'obiettivo del punto di ripristino (RPO).

Sintassi

```
aurora_global_db_status()
```

Argomenti

Nessuna.

Tipo restituito

Record SETOF avente le seguenti colonne:

- `aws_region`: la Regione AWS in cui si trova il cluster di database corrente. Per un elenco completo di Regioni AWS per motore, consulta Regioni e zone di disponibilità.
- `highest_lsn_written`: il numero di sequenza di log (LSN) più alto attualmente esistente in questo cluster di database. Numero di sequenza di log (LSN) è un numero sequenziale univoco che identifica un record nel log delle transazioni del database. Gli LSN sono ordinati in modo tale che un LSN più grande rappresenti una transazione successiva.
- `durability_lag_in_msec`: la differenza tra i valori di timestamp tra `highest_lsn_written` su un cluster di database secondario e `highest_lsn_written` sul cluster di database primario. Un valore -1 identifica il cluster di database globale principale di un database globale Aurora.
- `rpo_lag_in_msec`: il ritardo dell'obiettivo del punto di ripristino (RPO). Il ritardo dell'obiettivo del punto di ripristino (RPO) è il tempo necessario per memorizzare il COMMIT delle transazioni utente più recenti dopo la sua memorizzazione nel cluster di database primario del database globale Aurora. Un valore -1 indica il cluster di database primario e quindi il ritardo non è rilevante.

In sintesi, questo parametro calcola l'obiettivo del punto di ripristino per ciascun cluster di database Aurora PostgreSQL nel database globale Aurora, ovvero quanti dati potrebbero andare perduti in caso di interruzione. Come per il ritardo, l'obiettivo del punto di ripristino (RPO) viene misurato nel tempo.

- `last_lag_calculation_time`: il timestamp che specifica quando sono stati calcolati i valori per `durability_lag_in_msec` e `rpo_lag_in_msec`. Il valore temporale `1970-01-01 00:00:00+00` indica che questo è il cluster di database primario.
- `feedback_epoch`: l'epoca utilizzata dal cluster di database secondario quando genera informazioni sullo standby a caldo. Uno standby a caldo è un'istanza database che supporta connessioni e query mentre il database primario è in modalità di ripristino o standby. Le informazioni relative allo standby a caldo includono l'epoca (point-in-time) e altri dettagli sull'istanza database utilizzata come standby a caldo. Per ulteriori informazioni, consulta Hot Standby nella documentazione di PostgreSQL.
- `feedback_xmin`: l'ID minimo della transazione attiva (meno recente) utilizzato da un cluster di database secondario.

Note per l'utilizzo

Tutte le versioni di Aurora PostgreSQL attualmente disponibili supportano questa funzione. Questa funzione mostra le statistiche di replica per un database globale Aurora. Mostra una riga per ciascun cluster di database in un database globale Aurora PostgreSQL. È possibile eseguire questa funzione da qualsiasi istanza del database globale Aurora PostgreSQL.

Per valutare il ritardo di replica del database globale Aurora, ovvero il ritardo visibile dei dati, consulta aurora_global_db_instance_status.

Per ulteriori informazioni sull'uso di `aurora_global_db_status` e `aurora_global_db_instance_status` per monitorare il ritardo del database globale Aurora, consulta Monitoraggio dei database globali Aurora basati su PostgreSQL. Per informazioni sui database globali Aurora, consulta Panoramica di Database globale Amazon Aurora.

Esempi

Questo esempio mostra come visualizzare le statistiche di archiviazione tra regioni.

```
=> SELECT CASE
```

```
          WHEN '-1' = durability_lag_in_msec THEN 'Primary'
          ELSE 'Secondary'
       END AS global_role,
       *
  FROM aurora_global_db_status();
 global_role | aws_region | highest_lsn_written | durability_lag_in_msec |
 rpo_lag_in_msec | last_lag_calculation_time  | feedback_epoch | feedback_xmin
-------------+------------+---------------------+------------------------
+-----------------+----------------------------+----------------+---------------
 Primary     | eu-west-1  |           131031557 |                     -1 |
 -1 | 1970-01-01 00:00:00+00     |              0 |             0
 Secondary   | eu-west-2  |           131031554 |                    410 |
  0 | 2021-06-01 18:59:36.124+00 |              0 |         12640
 Secondary   | eu-west-3  |           131031554 |                    410 |
  0 | 2021-06-01 18:59:36.124+00 |              0 |         12640
```

aurora_list_builtins

Elenca tutte le funzioni integrate di Aurora PostgreSQL disponibili, insieme a brevi descrizioni e dettagli delle funzioni.

Sintassi

```
aurora_list_builtins()
```

Argomenti

Nessuno

Tipo restituito

Record SETOF

Esempi

Nell'esempio seguente vengono mostrati i risultati della chiamata alla funzione `aurora_list_builtins`.

```
=> SELECT *
FROM aurora_list_builtins();
```

```
              Name                | Result data type |                  Argument data
 types                     | Type | Volatility |  Parallel  | Security |
            Description
-----------------------------------+------------------
+-------------------------------------------------------------+------+------------
+------------+----------
+-----------------------------------------------------------------------
 aurora_version                    | text             |
                           | func | stable     | safe       | invoker  | Amazon Aurora
 PostgreSQL-Compatible Edition version string
 aurora_stat_wait_type             | SETOF record     | OUT type_id smallint, OUT
 type_name text                 | func | volatile   | restricted | invoker  | Lists all
 supported wait types
 aurora_stat_wait_event            | SETOF record     | OUT type_id smallint, OUT
 event_id integer, OUT event_na.| func | volatile   | restricted | invoker  | Lists all
 supported wait events
                                   |                  |.me text
                          |      |           |            |          |
 aurora_list_builtins              | SETOF record     | OUT "Name" text, OUT "Result
 data type" text, OUT "Argum.| func | stable     | safe       | invoker  | Lists all
 Aurora built-in functions
                                   |                  |.ent data types" text, OUT
 "Type" text, OUT "Volatility" .|      |           |            |          |
                                   |                  |.text, OUT "Parallel" text, OUT
 "Security" text, OUT "Des.|      |            |            |          |
                                   |                  |.cription" text
                          |      |           |            |          |
 .
 .
 .
 aurora_stat_file                  | SETOF record     | OUT filename text, OUT
 allocated_bytes bigint, OUT used_.| func | stable     | safe       | invoker  | Lists
 all files present in Aurora storage
                                   |                  |.bytes bigint
                          |      |           |            |          |
 aurora_stat_get_db_commit_latency | bigint           | oid
                           | func | stable     | restricted | invoker  | Per DB commit
 latency in microsecs
```

aurora_replica_status

Visualizza lo stato di tutti i nodi del lettore Aurora PostgreSQL.

Sintassi

```
aurora_replica_status()
```

Argomenti

Nessuno

Tipo restituito

Record SETOF avente le seguenti colonne:

- `server_id`: ID dell'istanza database (identificatore).
- `session_id`: un identificatore univoco per la sessione corrente, restituito per l'istanza primaria e le istanze di lettura nel modo seguente:
  - Per l'istanza primaria, il `session_id` è sempre `` `MASTER_SESSION_ID' ``.
  - Per le istanze di lettura, il `session_id` è sempre l'`UUID` (universally unique identifier, identificatore univoco universale) dell'istanza di lettura.
- `durable_lsn`: numero di sequenza di log (log sequence number, LSN) archiviato nello storage.
  - Per il volume primario, l'LSN (VDL) durevole del primario attualmente in uso.
  - Per tutti i volumi secondari, il VDL del primario a cui è stato applicato correttamente il secondario.

> **Note**
>
> Numero di sequenza di log (LSN) è un numero sequenziale univoco che identifica un record nel log delle transazioni del database. Gli LSN sono ordinati in modo tale che quelli più grandi rappresentino una transizione successiva.

- `highest_lsn_rcvd`: l'LSN più alto (più recente) che l'istanza database ha ricevuto dall'istanza database di scrittura.
- `current_read_lsn`: l'LSN della snapshot più recente applicato a tutte le operazioni di lettura.
- `cur_replay_latency_in_usec`: tempo previsto in millisecondi per riprodurre il log sul secondario.
- `active_txns`: numero di transazioni attualmente attive.
- `is_current`: non utilizzato.

- `last_transport_error`: codice di errore dell'ultima replica.
- `last_error_timestamp`: timestamp dell'ultimo errore di replica.
- `last_update_timestamp`: timestamp dell'ultimo aggiornamento allo stato della replica. Da Aurora PostgreSQL versione 13.9, il valore `last_update_timestamp` per l'istanza database a cui si è connessi è impostato su `NULL`.
- `feedback_xmin`: feedback_xmin della replica in standby a caldo. L'ID della transazione attiva più basso (meno recente) utilizzato dall'istanza database.
- `feedback_epoch`: l'epoca utilizzata dall'istanza database quando genera informazioni standby a caldo.
- `replica_lag_in_msec`: ritardo dell'istanza di lettura rispetto a quella di scrittura misurato in millisecondi.
- `log_stream_speed_in_kib_per_second`: la velocità effettiva del flusso di log in kilobyte al secondo.
- `log_buffer_sequence_number`: il numero di sequenza del buffer di registro.
- `oldest_read_view_trx_id`: non utilizzato.
- `oldest_read_view_lsn`: l'LSN più vecchio utilizzato dall'istanza database per le operazioni di lettura dallo storage.
- `pending_read_ios`: le letture di pagina in sospeso sulla replica.
- `read_ios`: il numero totale di letture di pagina sulla replica.
- `iops`: non utilizzato.
- `cpu`: utilizzo della CPU di Aurora Storage Daemon per ogni nodo del cluster. Per informazioni sull'utilizzo della CPU da parte dell'istanza, consulta Parametri a livello di istanza per Amazon Aurora.

Note per l'utilizzo

Tutte le versioni di Aurora PostgreSQL attualmente disponibili supportano questa funzione. La funzione `aurora_replica_status` restituisce i valori del gestore di stato della replica di un cluster database Aurora PostgreSQL. Utilizza questa funzione per ottenere informazioni sullo stato della replica sul cluster database Aurora PostgreSQL, inclusi i parametri per tutte le istanze database nel cluster database Aurora. Ad esempio, puoi eseguire le operazioni seguenti:

- Ottenere informazioni sul tipo di istanza (di scrittura, di lettura) nel cluster database Aurora PostgreSQL. Per ottenere queste informazioni, controlla i valori contenuti nelle colonne seguenti:

  - `server_id`: contiene il nome dell'istanza specificato durante la sua creazione. In alcuni casi, ad esempio per l'istanza primaria (di scrittura), il nome viene generato automaticamente aggiungendo -instance-1 al nome creato per il cluster database Aurora PostgreSQL.
  - `session_id`: il campo `session_id` indica se l'istanza è di lettura o di scrittura. Per un'istanza di scrittura, `session_id` è sempre impostato su `"MASTER_SESSION_ID"`. Per un'istanza di lettura, `session_id` è impostato sull'`UUID` del lettore specifico.
- Diagnostica dei problemi comuni correlati alle repliche, come il ritardo di replica. Il ritardo di replica indica il ritardo in millisecondi della cache di pagina di un'istanza di lettura rispetto a quella dell'istanza di scrittura. Questo ritardo si verifica perché i cluster Aurora utilizzano la replica asincrona, come descritto in Replica con Amazon Aurora. Nella colonna `replica_lag_in_msec` sono mostrati i risultati restituiti da questa funzione. Il ritardo può verificarsi anche quando una query viene annullata a causa di conflitti con il ripristino su un server in standby. È possibile verificare se il ritardo di replica è causato o meno da questo conflitto controllando `pg_stat_database_conflicts()`. Per ulteriori informazioni, consulta Raccolta di statistiche nella Documentazione di PostgreSQL. Per ulteriori informazioni sulla disponibilità elevata e la replica, consulta Domande frequenti su Amazon Aurora.

  Amazon CloudWatch archivia i risultati di `replica_lag_in_msec` nel tempo, come parametro `AuroraReplicaLag`. Per ulteriori informazioni sull'utilizzo dei parametri CloudWatch per Aurora, consulta Monitoraggio dei parametri di Amazon Aurora con Amazon CloudWatch

Per ulteriori informazioni sulla risoluzione dei problemi relativi alla lettura e al riavvio di Aurora, consulta Perché la mia replica Amazon Aurora è in ritardo e si è riavviata? nel Centro Supporto AWS.

Esempi

L'esempio seguente mostra come ottenere lo stato di replica di tutte le istanze in un cluster database Aurora PostgreSQL:

```
=> SELECT *
FROM aurora_replica_status();
```

L'esempio seguente mostra l'istanza di lettura nel cluster database `docs-lab-apg-main` Aurora PostgreSQL:

```
=> SELECT server_id,
    CASE
```

```
        WHEN 'MASTER_SESSION_ID' = session_id THEN 'writer'
        ELSE 'reader'
    END AS instance_role
FROM aurora_replica_status()
WHERE session_id = 'MASTER_SESSION_ID';
        server_id       | instance_role
------------------------+---------------
 db-119-001-instance-01 | writer
```

Nell'esempio seguente sono elencate tutte le istanze di lettura in un cluster:

```
=> SELECT server_id,
    CASE
        WHEN 'MASTER_SESSION_ID' = session_id THEN 'writer'
        ELSE 'reader'
    END AS instance_role
FROM aurora_replica_status()
WHERE session_id <> 'MASTER_SESSION_ID';
        server_id       | instance_role
------------------------+---------------
db-119-001-instance-02  | reader
db-119-001-instance-03  | reader
db-119-001-instance-04  | reader
db-119-001-instance-05  | reader
(4 rows)
```

Nell'esempio seguente sono elencate tutte le istanze, il ritardo di ognuna di esse rispetto all'istanza di scrittura e quanto tempo è trascorso dall'ultimo aggiornamento:

```
=> SELECT server_id,
    CASE
        WHEN 'MASTER_SESSION_ID' = session_id THEN 'writer'
        ELSE 'reader'
    END AS instance_role,
    replica_lag_in_msec AS replica_lag_ms,
    round(extract (epoch FROM (SELECT age(clock_timestamp(), last_update_timestamp))) *
 1000) AS last_update_age_ms
FROM aurora_replica_status()
ORDER BY replica_lag_in_msec NULLS FIRST;
       server_id        | instance_role | replica_lag_ms | last_update_age_ms
------------------------+---------------+----------------+--------------------
 db-124-001-instance-03 | writer        |         [NULL] |               1756
 db-124-001-instance-01 | reader        |             13 |               1756
```

```
 db-124-001-instance-02 | reader        |             13 |              1756
(3 rows)
```

aurora_stat_activity

Restituisce una riga per processo del server, mostrando le informazioni relative all'attività corrente di quel processo.

Sintassi

```
aurora_stat_activity();
```

Argomenti

Nessuno

Tipo restituito

Restituisce una riga per processo del server. Oltre alle colonne `pg_stat_activity`, viene aggiunto il seguente campo:

- planid: identificatore del piano

Note per l'utilizzo

Una vista supplementare di `pg_stat_activity` che restituisce le stesse colonne con una colonna `plan_id` aggiuntiva che mostra il piano di esecuzione della query corrente.

`aurora_compute_plan_id` deve essere abilitato affinché la vista restituisca un plan_id.

Questa funzione è disponibile dalle versioni 14.10, 15.5 di Aurora PostgreSQL e per tutte le altre versioni successive.

Esempi

La query di esempio riportata di seguito aggrega il carico massimo per query_id e plan_id.

```
db1=# select count(*), query_id, plan_id
db1-# from aurora_stat_activity() where state = 'active'
db1-# and pid <> pg_backend_pid()
db1-# group by query_id, plan_id
db1-# order by 1 desc;

count |  query_id             |  plan_id
-------+----------------------+-------------
 11    | -5471422286312252535 | -2054628807
 3     | -6907107586630739258 | -815866029
 1     | 5213711845501580017  |  300482084
(3 rows)
```

Se il piano utilizzato per query_id cambia, aurora_stat_activity riporterà un nuovo plan_id.

```
count  |  query_id             |  plan_id
-------+----------------------+-------------
 10    | -5471422286312252535 | 1602979607
 1     | -6907107586630739258 | -1809935983
 1     | -2446282393000597155 | -207532066
(3 rows)
```

aurora_stat_backend_waits

Visualizza le statistiche per l'attività di attesa per un processo di back-end specifico.

Sintassi

```
aurora_stat_backend_waits(pid)
```

Argomenti

`pid`: l'ID per il processo di back-end. È possibile ottenere gli ID di processo utilizzando la vista `pg_stat_activity`.

Tipo restituito

Record SETOF avente le seguenti colonne:

- `type_id`: un numero che indica il tipo di evento di attesa, ad esempio 1 per un blocco leggero (`LWLock`), 3 per un blocco o 6 per una sessione client. Questi valori diventano significativi quando i risultati di questa funzione vengono abbinati alle colonne della funzione `aurora_stat_wait_type`, come descritto nella sezione Esempi.
- `event_id`: un numero identificativo per l'evento di attesa. Abbina questo valore alle colonne di `aurora_stat_wait_event` per ottenere nomi significativi per gli eventi.
- `waits`: conteggio del numero di attese accumulate per l'ID di processo specificato.
- `wait_time`: tempo di attesa in millisecondi.

Note per l'utilizzo

È possibile utilizzare questa funzione per analizzare eventi di attesa di back-end (sessione) specifici che si sono verificati dall'apertura di una connessione. Per ottenere informazioni più significative sui nomi e sui tipi di eventi di attesa, è possibile combinare `aurora_stat_wait_type` e `aurora_stat_wait_event`, utilizzando JOIN come mostrato negli esempi.

Esempi

Questo esempio mostra tutte le attese, tutti i tipi e tutti i nomi di evento per l'ID processo back-end 3027.

```
=> SELECT type_name, event_name, waits, wait_time
        FROM aurora_stat_backend_waits(3027)
NATURAL JOIN aurora_stat_wait_type()
NATURAL JOIN aurora_stat_wait_event();
type_name |       event_name       | waits | wait_time
----------+------------------------+-------+------------
 LWLock    | ProcArrayLock          |     3 |         27
 LWLock    | wal_insert             |   423 |      16336
 LWLock    | buffer_content         | 11840 |    1033634
 LWLock    | lock_manager           | 23821 |    5664506
 Lock      | tuple                  | 10258 |  152280165
 Lock      | transactionid          | 78340 | 1239808783
 Client    | ClientRead             | 34072 |   17616684
 IO        | ControlFileSyncUpdate  |     2 |          0
 IO        | ControlFileWriteUpdate |     4 |         32
```

```
 IO        | RelationMapRead        |     2 |        795
 IO        | WALWrite               | 36666 |      98623
 IO        | XactSync               |  4867 |    7331963
```

Questo esempio mostra i tipi di attesa correnti e cumulativi e gli eventi di attesa per tutte le sessioni attive (`pg_stat_activity state <> 'idle'`) (ma senza la sessione corrente che richiama la funzione (`pid <> pg_backend_pid()`).

```
=> SELECT a.pid,
            a.usename,
            a.app_name,
            a.current_wait_type,
            a.current_wait_event,
            a.current_state,
            wt.type_name AS wait_type,
            we.event_name AS wait_event,
            a.waits,
            a.wait_time
       FROM (SELECT pid,
                    usename,
                    left(application_name,16) AS app_name,
                    coalesce(wait_event_type,'CPU') AS current_wait_type,
                    coalesce(wait_event,'CPU') AS current_wait_event,
                    state AS current_state,
                    (aurora_stat_backend_waits(pid)).*
               FROM pg_stat_activity
              WHERE pid <> pg_backend_pid()
                AND state <> 'idle') a
NATURAL JOIN aurora_stat_wait_type() wt
NATURAL JOIN aurora_stat_wait_event() we;
  pid  | usename  | app_name | current_wait_type | current_wait_event | current_state |
 wait_type |       wait_event       | waits | wait_time
-------+----------+----------+-------------------+--------------------+---------------
+-----------+------------------------+-------+-----------
 30099 | postgres | pgbench  | Lock              | transactionid      | active        |
 LWLock    | wal_insert             |  1937 |     29975
 30099 | postgres | pgbench  | Lock              | transactionid      | active        |
 LWLock    | buffer_content         | 22903 |    760498
 30099 | postgres | pgbench  | Lock              | transactionid      | active        |
 LWLock    | lock_manager           | 10012 |    223207
 30099 | postgres | pgbench  | Lock              | transactionid      | active        |
 Lock      | tuple                  | 20315 |  63081529
 .
```

```
.
.
30099 | postgres | pgbench  | Lock              | transactionid      | active        |
IO        | WALWrite               | 93293 |    237440
30099 | postgres | pgbench  | Lock              | transactionid      | active        |
IO        | XactSync               | 13010 |  19525143
30100 | postgres | pgbench  | Lock              | transactionid      | active        |
LWLock    | ProcArrayLock          |     6 |        53
30100 | postgres | pgbench  | Lock              | transactionid      | active        |
LWLock    | wal_insert             |  1913 |     25450
30100 | postgres | pgbench  | Lock              | transactionid      | active        |
LWLock    | buffer_content         | 22874 |    778005
.
.
.
30109 | postgres | pgbench  | IO                | XactSync           | active        |
LWLock    | ProcArrayLock          |     3 |        71
30109 | postgres | pgbench  | IO                | XactSync           | active        |
LWLock    | wal_insert             |  1940 |     27741
30109 | postgres | pgbench  | IO                | XactSync           | active        |
LWLock    | buffer_content         | 22962 |    776352
30109 | postgres | pgbench  | IO                | XactSync           | active        |
LWLock    | lock_manager           |  9879 |    218826
30109 | postgres | pgbench  | IO                | XactSync           | active        |
Lock      | tuple                  | 20401 |  63581306
30109 | postgres | pgbench  | IO                | XactSync           | active        |
Lock      | transactionid          | 50769 | 211645008
30109 | postgres | pgbench  | IO                | XactSync           | active        |
Client    | ClientRead             | 89901 |  44192439
```

Questo esempio mostra il tipo di attesa corrente e i primi tre (3) eventi di attesa cumultativi, nonché gli eventi di attesa per tutte le sessioni attive (`pg_stat_activity state <> 'idle'`) esclusa la sessione corrente (`pid <>pg_backend_pid()`).

```
=> SELECT top3.*
       FROM (SELECT a.pid,
                    a.usename,
                    a.app_name,
                    a.current_wait_type,
                    a.current_wait_event,
                    a.current_state,
                    wt.type_name AS wait_type,
                    we.event_name AS wait_event,
```

```
                    a.waits,
                    a.wait_time,
                    RANK() OVER (PARTITION BY pid ORDER BY a.wait_time DESC)
                FROM (SELECT pid,
                             usename,
                             left(application_name,16) AS app_name,
                             coalesce(wait_event_type,'CPU') AS current_wait_type,
                             coalesce(wait_event,'CPU') AS current_wait_event,
                             state AS current_state,
                             (aurora_stat_backend_waits(pid)).*
                        FROM pg_stat_activity
                       WHERE pid <> pg_backend_pid()
                         AND state <> 'idle') a
          NATURAL JOIN aurora_stat_wait_type() wt
          NATURAL JOIN aurora_stat_wait_event() we) top3
 WHERE RANK <=3;
  pid  | usename  | app_name | current_wait_type | current_wait_event | current_state |
 wait_type |   wait_event    |  waits  | wait_time  | rank
-------+----------+----------+-------------------+--------------------+---------------
+-----------+-----------------+---------+------------+------
 20567 | postgres | psql     | CPU               | CPU                | active        |
 LWLock    | wal_insert      |   25000 |   67512003 |    1
 20567 | postgres | psql     | CPU               | CPU                | active        |
 IO        | WALWrite        | 3071758 |    1016961 |    2
 20567 | postgres | psql     | CPU               | CPU                | active        |
 IO        | BufFileWrite    |   20750 |     184559 |    3
 27743 | postgres | pgbench  | Lock              | transactionid      | active        |
 Lock      | transactionid   |  237350 | 1265580011 |    1
 27743 | postgres | pgbench  | Lock              | transactionid      | active        |
 Lock      | tuple           |   93641 |  341472318 |    2
 27743 | postgres | pgbench  | Lock              | transactionid      | active        |
 Client    | ClientRead      |  417556 |  204796837 |    3
 .
 .
 .
 27745 | postgres | pgbench  | IO                | XactSync           | active        |
 Lock      | transactionid   |  238068 | 1265816822 |    1
 27745 | postgres | pgbench  | IO                | XactSync           | active        |
 Lock      | tuple           |   93210 |  338312247 |    2
 27745 | postgres | pgbench  | IO                | XactSync           | active        |
 Client    | ClientRead      |  419157 |  207836533 |    3
 27746 | postgres | pgbench  | Lock              | transactionid      | active        |
 Lock      | transactionid   |  237621 | 1264528811 |    1
```

```
 27746 | postgres | pgbench  | Lock              | transactionid      | active        |
 Lock      | tuple           |   93563 |  339799310 |    2
 27746 | postgres | pgbench  | Lock              | transactionid      | active        |
 Client    | ClientRead      |  417304 |  208372727 |    3
```

aurora_stat_bgwriter

`aurora_stat_bgwriter` è una vista delle statistiche che mostra informazioni sulle scritture nella cache di Letture ottimizzate.

Sintassi

```
aurora_stat_bgwriter()
```

Argomenti

Nessuno

Tipo restituito

Record SETOF con tutte le colonne `pg_stat_bgwriter` e le seguenti colonne aggiuntive. Per ulteriori informazioni sulle colonne `pg_stat_bgwriter`, consulta `pg_stat_bgwriter`.

È possibile ripristinare le statistiche per questa funzione utilizzando `pg_stat_reset_shared("bgwriter")`.

- `orcache_blks_written`: numero totale di blocchi di dati della cache di Letture ottimizzate scritti.
- `orcache_blk_write_time`: se `track_io_timing` è abilitato, tiene traccia del tempo totale in millisecondi impiegato per scrivere blocchi di dati di file della cache di Letture ottimizzate. Per ulteriori informazioni, consulta track_io_timing.

Note per l'utilizzo

Questa funzione è disponibile per le seguenti versioni di Aurora PostgreSQL:

- 15.4 e versioni successive
- 14.9 e versioni successive

Esempi

```
=> select * from aurora_stat_bgwriter();
-[ RECORD 1 ]-----------------+-----------
orcache_blks_written          | 246522
orcache_blk_write_time        | 339276.404
```

aurora_stat_database

Contiene tutte le colonne di pg_stat_database e aggiunge nuove colonne alla fine.

Sintassi

```
aurora_stat_database()
```

Argomenti

Nessuno

Tipo restituito

Record SETOF con tutte le colonne `pg_stat_database` e le seguenti colonne aggiuntive. Per ulteriori informazioni sulle colonne `pg_stat_database`, consulta `pg_stat_database`.

- `storage_blks_read`: numero totale di blocchi condivisi letti dall'archiviazione Aurora in questo database.
- `orcache_blks_hit`: numero totale di accessi alla cache di Letture ottimizzate in questo database.
- `local_blks_read`: numero totale di blocchi locali letti in questo database.
- `storage_blk_read_time`: se `track_io_timing` è abilitato, tiene traccia del tempo totale impiegato in millisecondi per leggere i blocchi di file di dati dall'archiviazione Aurora altrimenti il valore è zero. Per ulteriori informazioni, consulta track_io_timing.
- `local_blk_read_time`: se `track_io_timing` è abilitato, tiene traccia del tempo totale impiegato in millisecondi per leggere i blocchi di file di dati locali, altrimenti il valore è zero. Per ulteriori informazioni, consulta track_io_timing.
- `orcache_blk_read_time`: se `track_io_timing` è abilitato, tiene traccia del tempo totale impiegato in millisecondi per leggere i blocchi di file di dati dalla cache di Letture ottimizzate, altrimenti il valore è zero. Per ulteriori informazioni, consulta track_io_timing.

> **Note**
>
> Il valore di `blks_read` è la somma di `storage_blks_read`, `orcache_blks_hit` e `local_blks_read`.
> Il valore di `blk_read_time` è la somma di `storage_blk_read_time`, `orcache_blk_read_time` e `local_blk_read_time`.

Note per l'utilizzo

Questa funzione è disponibile per le seguenti versioni di Aurora PostgreSQL:

- 15.4 e versioni successive
- 14.9 e versioni successive

Esempi

L'esempio seguente mostra come contiene tutte le colonne `pg_stat_database` e aggiunge 6 nuove colonne alla fine:

```
=> select * from aurora_stat_database() where datid=14717;
-[ RECORD 1 ]------------+------------------------------
datid                    | 14717
datname                  | postgres
numbackends              | 1
xact_commit              | 223
xact_rollback            | 4
blks_read                | 1059
blks_hit                 | 11456
tup_returned             | 27746
tup_fetched              | 5220
tup_inserted             | 165
tup_updated              | 42
tup_deleted              | 91
conflicts                | 0
temp_files               | 0
temp_bytes               | 0
deadlocks                | 0
checksum_failures        |
checksum_last_failure    |
blk_read_time            | 3358.689
```

```
blk_write_time           | 0
session_time             | 1076007.997
active_time              | 3684.371
idle_in_transaction_time | 0
sessions                 | 10
sessions_abandoned       | 0
sessions_fatal           | 0
sessions_killed          | 0
stats_reset              | 2023-01-12 20:15:17.370601+00
orcache_blks_hit         | 425
orcache_blk_read_time    | 89.934
storage_blks_read        | 623
storage_blk_read_time    | 3254.914
local_blks_read          | 0
local_blk_read_time      | 0
```

aurora_stat_dml_activity

Riporta l'attività cumulativa per ogni tipo di operazione di linguaggio di manipolazione dei dati (DML) su un database in un cluster Aurora PostgreSQL.

Sintassi

```
aurora_stat_dml_activity(database_oid)
```

Argomenti

database_oid

L'ID oggetto (OID) del database nel cluster Aurora PostgreSQL.

Tipo restituito

Record SETOF

Note per l'utilizzo

La funzione `aurora_stat_dml_activity` è disponibile solo con Aurora PostgreSQL versione 3.1 compatibile con il motore PostgreSQL 11.6 e versioni successive.

Utilizzare questa funzione sui cluster Aurora PostgreSQL con un gran numero di database per identificare quali database hanno un'attività DML più veloce o più lenta, o entrambi.

La funzione `aurora_stat_dml_activity` restituisce il numero di volte in cui le operazioni sono state eseguite e la latenza cumulativa in microsecondi per le operazioni SELECT, INSERT, UPDATE e DELETE. Il report include solo le operazioni DML riuscite.

È possibile reimpostare questa statistica utilizzando la funzione di accesso alle statistiche PostgreSQL `pg_stat_reset`. È possibile controllare l'ultima volta che questa statistica è stata reimpostata utilizzando la funzione `pg_stat_get_db_stat_reset_time`. Per ulteriori informazioni sulle funzioni di accesso alle statistiche PostgreSQL, consulta Raccolta di statistiche nella documentazione di PostgreSQL.

Esempi

L'esempio seguente mostra come eseguire il report delle statistiche di attività DML per il database connesso.

```
--Define the oid variable from connected database by using \gset
=> SELECT oid,
          datname
     FROM pg_database
    WHERE datname=(select current_database()) \gset
=> SELECT *
     FROM aurora_stat_dml_activity(:oid);
select_count | select_latency_microsecs | insert_count | insert_latency_microsecs |
 update_count | update_latency_microsecs | delete_count | delete_latency_microsecs
--------------+--------------------------+--------------+--------------------------
+--------------+--------------------------+--------------+--------------------------
       178957 |                 66684115 |       171065 |                 28876649 |
    519538 |            1454579206167 |            1 |                    53027


-- Showing the same results with expanded display on
=> SELECT *
     FROM aurora_stat_dml_activity(:oid);
-[ RECORD 1 ]------------+-------------
select_count             | 178957
select_latency_microsecs | 66684115
insert_count             | 171065
insert_latency_microsecs | 28876649
update_count             | 519538
update_latency_microsecs | 1454579206167
delete_count             | 1
delete_latency_microsecs | 53027
```

L'esempio seguente mostra le statistiche di attività DML per tutti i database nel cluster Aurora PostgreSQL. Questo cluster dispone di due database, `postgres` e `mydb`. L'elenco separato da virgole corrisponde ai campi `select_count`, `select_latency_microsecs`, `insert_count`, `insert_latency_microsecs`, `update_count`, `update_latency_microsecs`, `delete_count` e `delete_latency_microsecs`.

Aurora PostgreSQL crea e utilizza un database di sistema denominato `rdsadmin` per supportare operazioni amministrative quali backup, ripristini, controlli di integrità, replica e così via. Queste operazioni DML non hanno alcun impatto sul cluster Aurora PostgreSQL.

```
=> SELECT oid,
    datname,
    aurora_stat_dml_activity(oid)
    FROM pg_database;
oid  |    datname     |                   aurora_stat_dml_activity
-------+---------------
+-----------------------------------------------------------------
 14006 | template0      | (,,,,,,,)
 16384 | rdsadmin       | (2346623,1211703821,4297518,817184554,0,0,0,0)
     1 | template1      | (,,,,,,,)
 14007 | postgres       |
 (178961,66716329,171065,28876649,519538,1454579206167,1,53027)
 16401 | mydb           | (200246,64302436,200036,107101855,600000,83659417514,0,0)
```

Nell'esempio seguente vengono illustrate le statistiche di attività DML per tutti i database, organizzate in colonne per una migliore leggibilità.

```
SELECT db.datname,
       BTRIM(SPLIT_PART(db.asdmla::TEXT, ',', 1), '()') AS select_count,
       BTRIM(SPLIT_PART(db.asdmla::TEXT, ',', 2), '()') AS select_latency_microsecs,
       BTRIM(SPLIT_PART(db.asdmla::TEXT, ',', 3), '()') AS insert_count,
       BTRIM(SPLIT_PART(db.asdmla::TEXT, ',', 4), '()') AS insert_latency_microsecs,
       BTRIM(SPLIT_PART(db.asdmla::TEXT, ',', 5), '()') AS update_count,
       BTRIM(SPLIT_PART(db.asdmla::TEXT, ',', 6), '()') AS update_latency_microsecs,
       BTRIM(SPLIT_PART(db.asdmla::TEXT, ',', 7), '()') AS delete_count,
       BTRIM(SPLIT_PART(db.asdmla::TEXT, ',', 8), '()') AS delete_latency_microsecs
FROM  (SELECT datname,
              aurora_stat_dml_activity(oid) AS asdmla
         FROM pg_database
      ) AS db;
```

```
    datname     | select_count | select_latency_microsecs | insert_count |
 insert_latency_microsecs | update_count | update_latency_microsecs | delete_count |
 delete_latency_microsecs
---------------+--------------+--------------------------+--------------
+--------------------------+--------------+--------------------------+--------------
+--------------------------
 template0      |              |                          |              |
             |              |                          |              |
 rdsadmin       | 4206523      | 2478812333               | 7009414      | 1338482258
             | 0            | 0                        | 0            | 0
 template1      |              |                          |              |
             |              |                          |              |
 fault_test     | 66           | 452099                   | 0            | 0
             | 0            | 0                        | 0            | 0
 db_access_test | 1            | 5982                     | 0            | 0
             | 0            | 0                        | 0            | 0
 postgres       | 42035        | 95179203                 | 5752         | 2678832898
             | 21157        | 441883182488             | 2            | 1520
 mydb           | 71           | 453514                   | 0            | 0
             | 1            | 190                      | 1            | 152
```

Nell'esempio seguente viene illustrata la latenza cumulativa media (latenza cumulativa divisa per conteggio) per ogni operazione DML per il database con OID 16401.

```
=> SELECT select_count,
          select_latency_microsecs,
          select_latency_microsecs/NULLIF(select_count,0) select_latency_per_exec,
          insert_count,
          insert_latency_microsecs,
          insert_latency_microsecs/NULLIF(insert_count,0) insert_latency_per_exec,
          update_count,
          update_latency_microsecs,
          update_latency_microsecs/NULLIF(update_count,0) update_latency_per_exec,
          delete_count,
          delete_latency_microsecs,
          delete_latency_microsecs/NULLIF(delete_count,0) delete_latency_per_exec
     FROM aurora_stat_dml_activity(16401);
-[ RECORD 1 ]------------+-------------
select_count             | 451312
select_latency_microsecs | 80205857
select_latency_per_exec  | 177
insert_count             | 451001
insert_latency_microsecs | 123667646
```

```
insert_latency_per_exec  | 274
update_count             | 1353067
update_latency_microsecs | 200900695615
update_latency_per_exec  | 148478
delete_count             | 12
delete_latency_microsecs | 448
delete_latency_per_exec  | 37
```

aurora_stat_get_db_commit_latency

Ottiene la latenza di commit cumulativa in microsecondi per i database Aurora PostgreSQL. La Latenza commit viene misurata come il tempo che intercorre tra il momento in cui un client invia una richiesta di commit e il momento in cui riceve la conferma di commit.

Sintassi

```
aurora_stat_get_db_commit_latency(database_oid)
```

Argomenti

database_oid

L'ID oggetto (OID) del database Aurora PostgreSQL.

Tipo restituito

Record SETOF

Note per l'utilizzo

Amazon CloudWatch utilizza questa funzione per calcolare la latenza media di commit. Per ulteriori informazioni sui parametri di Amazon CloudWatch e sulla risoluzione dei problemi di elevata latenza di commit, consulta Visualizzazione delle metriche nella console Amazon RDS e Prendere decisioni migliori su Amazon RDS con i parametri di Amazon CloudWatch.

È possibile reimpostare questa statistica utilizzando la funzione di accesso alle statistiche PostgreSQL `pg_stat_reset`. È possibile controllare l'ultima volta che questa statistica è stata reimpostata utilizzando la funzione `pg_stat_get_db_stat_reset_time`. Per ulteriori informazioni sulle funzioni di accesso alle statistiche PostgreSQL, consulta Raccolta di statistiche nella documentazione di PostgreSQL.

Esempi

Nell'esempio seguente viene ottenuta la latenza di commit cumulativa per ogni database nel cluster `pg_database`.

```
=> SELECT oid,
    datname,
    aurora_stat_get_db_commit_latency(oid)
    FROM pg_database;

  oid  |    datname     | aurora_stat_get_db_commit_latency
-------+----------------+-----------------------------------
 14006 | template0      |                                 0
 16384 | rdsadmin       |                         654387789
     1 | template1      |                                 0
 16401 | mydb           |                            229556
 69768 | postgres       |                             22011
```

Nell'esempio seguente viene ottenuta la latenza di commit cumulativa per il database attualmente connesso. Prima di chiamare la funzione `aurora_stat_get_db_commit_latency`, l'esempio utilizza `\gset` per definire una variabile per `oid` e imposta il suo valore dal database connesso.

```
--Get the oid value from the connected database before calling
 aurora_stat_get_db_commit_latency
=> SELECT oid
     FROM pg_database
    WHERE datname=(SELECT current_database()) \gset
=> SELECT *
     FROM aurora_stat_get_db_commit_latency(:oid);

 aurora_stat_get_db_commit_latency
-----------------------------------
                        1424279160
```

Nell'esempio seguente viene ottenuta la latenza di commit cumulativa per il database `mydb` nel cluster `pg_database`. Quindi, reimposta questa statistica utilizzando la funzione `pg_stat_reset` e mostra i risultati. Infine, utilizza la funzione `pg_stat_get_db_stat_reset_time` per controllare l'ultima volta che questa statistica è stata reimpostata.

```
=> SELECT oid,
    datname,
```

```
    aurora_stat_get_db_commit_latency(oid)
    FROM pg_database
    WHERE datname = 'mydb';

  oid  |  datname  | aurora_stat_get_db_commit_latency
-------+-----------+-----------------------------------
 16427 | mydb      |                           3320370


=> SELECT pg_stat_reset();
 pg_stat_reset
---------------


=> SELECT oid,
          datname,
          aurora_stat_get_db_commit_latency(oid)
     FROM pg_database
    WHERE datname = 'mydb';
  oid  |  datname  | aurora_stat_get_db_commit_latency
-------+-----------+-----------------------------------
 16427 | mydb      |                                 6


=> SELECT *
     FROM pg_stat_get_db_stat_reset_time(16427);

 pg_stat_get_db_stat_reset_time
--------------------------------
 2021-04-29 21:36:15.707399+00
```

aurora_stat_logical_wal_cache

Mostra l'utilizzo della cache WAL (write-ahead log) logica per slot.

Sintassi

```
SELECT * FROM aurora_stat_logical_wal_cache()
```

Argomenti

Nessuno

Tipo restituito

Record SETOF avente le seguenti colonne:

- `name`: il nome dello slot di replica.
- `active_pid`: ID del processo walsender.
- `cache_hit`: il numero totale di riscontri della cache wal dall'ultimo ripristino.
- `cache_miss`: il numero totale di riscontri mancati della cache wal dall'ultimo ripristino.
- `blks_read`: il numero totale di richieste di lettura della cache wal.
- `hit_rate`: la percentuale di riscontri nella cache WAL (cache_hit/blks_read).
- `last_reset_timestamp`: data e ora dell'ultimo ripristino del contatore.

Note per l'utilizzo

Questa funzione è disponibile per le seguenti versioni di Aurora PostgreSQL.

- 15.2 e tutte le versioni successive
- 14.7 e versioni successive
- 13.8 e versioni successive
- 12.12 e versioni successive
- 11.17 e versioni successive

Esempi

L'esempio seguente mostra due slot di replica con una sola funzione `aurora_stat_logical_wal_cache`attiva.

```
=> SELECT *
     FROM aurora_stat_logical_wal_cache();
    name    | active_pid | cache_hit | cache_miss | blks_read | hit_rate |
 last_reset_timestamp
------------+------------+-----------+------------+-----------+----------
+-------------------------------
 test_slot1 |      79183 |        24 |          0 |        24 | 100.00%  | 2022-08-05
 17:39:56.830635+00
 test_slot2 |            |         1 |          0 |         1 | 100.00%  | 2022-08-05
 17:34:04.036795+00
```

```
(2 rows)
```

aurora_stat_memctx_usage

Segnala l'utilizzo del contesto della memoria per ciascun processo PostgreSQL.

Sintassi

```
aurora_stat_memctx_usage()
```

Argomenti

Nessuno

Tipo restituito

Record SETOF avente le seguenti colonne:

- `pid`: l'ID del processo.
- `name`: il nome del contesto della memoria.
- `allocated`: il numero di byte ottenuti dal sottosistema della memoria sottostante mediante il contesto della memoria.
- `used`: il numero di byte vincolati per i client del contesto della memoria.
- `instances`: il numero di contesti di questo tipo attualmente esistenti.

Note per l'utilizzo

Questa funzione visualizza l'utilizzo del contesto della memoria per ciascun processo PostgreSQL. Alcuni processi sono etichettati `anonymous`. I processi non sono esposti perché contengono parole chiave limitate.

Questa funzione è disponibile a partire dalle seguenti versioni di Aurora PostgreSQL:

- 15.3 o versioni successive alla 15
- 14.8 o versioni successive alla 14
- 13.11 o versioni successive alla 13
- 12.15 e versioni successive alla 12
- 11.20 e versioni successive alla 11

Esempi

Nell'esempio seguente vengono mostrati i risultati della chiamata della funzione `aurora_stat_memctx_usage`.

```
=> SELECT *
     FROM aurora_stat_memctx_usage();

    pid| name                              | allocated |   used  | instances
-------+-----------------------------------+-----------+---------+-----------
123864 | Miscellaneous                     |     19520 |   15064 |         3
123864 | Aurora File Context               |      8192 |     616 |         1
123864 | Aurora WAL Context                |      8192 |     296 |         1
123864 | CacheMemoryContext                |    524288 |  422600 |         1
123864 | Catalog tuple context             |     16384 |   13736 |         1
123864 | ExecutorState                     |     32832 |   28304 |         1
123864 | ExprContext                       |      8192 |    1720 |         1
123864 | GWAL record construction          |      1024 |     832 |         1
123864 | MdSmgr                            |      8192 |     296 |         1
123864 | MessageContext                    |    532480 |  353832 |         1
123864 | PortalHeapMemory                  |      1024 |     488 |         1
123864 | PortalMemory                      |      8192 |     576 |         1
123864 | printtup                          |      8192 |     296 |         1
123864 | RelCache hash table entries       |      8192 |    8152 |         1
123864 | RowDescriptionContext             |      8192 |    1344 |         1
123864 | smgr relation context             |      8192 |     296 |         1
123864 | Table function arguments          |      8192 |     352 |         1
123864 | TopTransactionContext             |      8192 |     632 |         1
123864 | TransactionAbortContext           |     32768 |     296 |         1
123864 | WAL record construction           |     50216 |   43904 |         1
123864 | hash table                        |     65536 |   52744 |         6
123864 | Relation metadata                 |    191488 |  124240 |        87
104992 | Miscellaneous                     |      9280 |    7728 |         3
104992 | Aurora File Context               |      8192 |     376 |         1
104992 | Aurora WAL Context                |      8192 |     296 |         1
104992 ||Autovacuum Launcher               |      8192 |     296 |         1
104992 | Autovacuum database list          |     16384 |     744 |         2
104992 | CacheMemoryContext                |    262144 |  140288 |         1
104992 | Catalog tuple context             |      8192 |     296 |         1
104992 | GWAL record construction          |      1024 |     832 |         1
104992 | MdSmgr                            |      8192 |     296 |         1
104992 | PortalMemory                      |      8192 |     296 |         1
104992 | RelCache hash table entries       |      8192 |     296 |         1
104992 | smgr relation context             |      8192 |     296 |         1
```

```
104992 | Autovacuum start worker (tmp)   |      8192 |     296 |         1
104992 | TopTransactionContext           |     16384 |     592 |         2
104992 | TransactionAbortContext         |     32768 |     296 |         1
104992 | WAL record construction         |     50216 |   43904 |         1
104992 | hash table                      |     49152 |   34024 |         4
(39 rows)
```

Alcune parole chiave limitate verranno nascoste e l'aspetto dell'output sarà il seguente:

```
postgres=>SELECT *
     FROM aurora_stat_memctx_usage();

    pid| name                            | allocated |   used  | instances
-------+---------------------------------+-----------+---------+-----------
  5482 | anonymous                       |      8192 |     456 |         1
  5482 | anonymous                       |      8192 |     296 |         1
```

aurora_stat_optimized_reads_cache

Questa funzione mostra le statistiche relative alla cache a più livelli.

Sintassi

```
aurora_stat_optimized_reads_cache()
```

Argomenti

Nessuno

Tipo restituito

Record SETOF avente le seguenti colonne:

- `total_size`: dimensione totale della cache di Letture ottimizzate.
- `used_size`: dimensione della pagina utilizzata nella cache di Letture ottimizzate.

Note per l'utilizzo

Questa funzione è disponibile per le seguenti versioni di Aurora PostgreSQL:

- 15.4 e versioni successive

- 14.9 e versioni successive

Esempi

L'esempio seguente mostra l'output di un'istanza r6gd.8xlarge:

```
=> select pg_size_pretty(total_size) as total_size, pg_size_pretty(used_size)
                as used_size from aurora_stat_optimized_reads_cache();
total_size | used_size
-----------+-----------
1054 GB    | 975 GB
```

aurora_stat_plans

Restituisce una riga per ogni piano di esecuzione monitorato.

Sintassi

```
aurora_stat_plans(
    showtext
)
```

Argomenti

- showtext: mostra il testo della query e del piano. I valori validi sono NULL, true o false. True mostrerà il testo della query e del piano.

Tipo restituito

Restituisce una riga per ogni piano monitorato che contiene tutte le colonne di `aurora_stat_statements` e le seguenti colonne aggiuntive.

- planid: identificatore del piano
- explain_plan: spiega il testo del piano
- plan_type:
  - `no plan`: nessun piano è stato acquisito
  - `estimate`: piano acquisito con costi stimati
  - `actual`: piano acquisito con EXPLAIN ANALYZE

- plan_captured_time: ultima acquisizione di un piano

Note per l’utilizzo

`aurora_compute_plan_id` deve essere abilitato e `pg_stat_statements` deve essere in `shared_preload_libraries` affinché i piani possano essere monitorati.

Il numero di piani disponibili è controllato dal valore impostato nel parametro `pg_stat_statements.max`. Quando `aurora_compute_plan_id` è abilitato, è possibile monitorare i piani fino al valore specificato in `aurora_stat_plans`.

Questa funzione è disponibile dalle versioni 14.10, 15.5 di Aurora PostgreSQL e per tutte le altre versioni successive.

Esempi

Nell’esempio seguente, vengono acquisiti i due piani relativi all’identificatore di query -5471422286312252535 e le statistiche delle istruzioni vengono monitorate dal planid.

```
db1=# select calls, total_exec_time, planid, plan_captured_time, explain_plan
db1-# from aurora_stat_plans(true)
db1-# where queryid = '-5471422286312252535'

calls    |  total_exec_time   |   planid    |      plan_captured_time       |
                 explain_plan
---------+--------------------+-------------+-------------------------------
+-----------------------------------------------------------------
 1532632 |  3209846.097107853 |  1602979607 | 2023-10-31 03:27:16.925497+00 | Update on
 pgbench_branches                                        +
         |                    |             |                               |   ->
 Bitmap Heap Scan on pgbench_branches                       +
         |                    |             |                               |
 Recheck Cond: (bid = 76)                                 +
         |                    |             |                               |         -
>  Bitmap Index Scan on pgbench_branches_pkey          +
         |                    |             |                               |
     Index Cond: (bid = 76)
   61365 | 124078.18012200127 | -2054628807 | 2023-10-31 03:20:09.85429+00  | Update on
 pgbench_branches                                        +
         |                    |             |                               |   ->
 Index Scan using pgbench_branches_pkey on pgbench_branches+
```

```
        |                    |            |                                |
Index Cond: (bid = 17)
```

aurora_stat_reset_wal_cache

Reimposta il contatore per la cache wal logica.

Sintassi

Per reimpostare uno slot specifico

```
SELECT * FROM aurora_stat_reset_wal_cache('slot_name')
```

Per ripristinare tutti gli slot

```
SELECT * FROM aurora_stat_reset_wal_cache(NULL)
```

Argomenti

`NULL` o `slot_name`

Tipo restituito

Messaggio di stato, stringa di testo

- Reimposta il contatore logico della cache wal. Messaggio di esito positivo. Questo testo viene restituito quando la funzione ha esito positivo.
- Lo slot di replica non è stato trovato Riprova. Messaggio di errore Questo testo viene restituito quando la funzione non riesce.

Note per l'utilizzo

Questa funzione è disponibile per le seguenti versioni.

- Aurora PostgreSQL 14.5 e versioni successive
- Aurora PostgreSQL 13.8 e versioni successive
- Aurora PostgreSQL 12.12 e versioni successive

- Aurora PostgreSQL 11.7 e versioni successive

Esempi

L'esempio seguente utilizza la funzione `aurora_stat_reset_wal_cache` per reimpostare uno slot denominato `test_results` e quindi tenta di reimpostare uno slot che non esiste.

```
=> SELECT *
     FROM aurora_stat_reset_wal_cache('test_slot');
aurora_stat_reset_wal_cache
------------------------------------
 Reset the logical wal cache counter.
(1 row)
=> SELECT *
     FROM aurora_stat_reset_wal_cache('slot-not-exist');
aurora_stat_reset_wal_cache
-----------------------------------------------
 Replication slot not found. Please try again.
(1 row)
```

aurora_stat_resource_usage

Riporta l’utilizzo delle risorse in tempo reale, che comprende le metriche delle risorse backend e l’utilizzo della CPU per tutti i processi backend Aurora PostgreSQL.

Sintassi

```
aurora_stat_resource_usage()
```

Argomenti

Nessuno

Tipo restituito

Record SETOF con colonne:

- pid: identificatore del processo
- allocated_memory: memoria totale allocata dal processo in byte
- used_memory: memoria effettivamente utilizzata per processo in byte
- cpu_usage_percent: percentuale di utilizzo della CPU del processo

Note per l'utilizzo

Questa funzione mostra l'utilizzo delle risorse backend per ogni processo backend Aurora PostgreSQL.

Questa funzione è disponibile a partire dalle seguenti versioni di Aurora PostgreSQL:

- Aurora PostgreSQL 17.5 e versioni 17 successive
- Aurora PostgreSQL 16.9 e versioni 16 successive
- Aurora PostgreSQL 15.13 e versioni 15 successive
- Aurora PostgreSQL 14.18 e versioni 14 successive
- Aurora PostgreSQL 13.21 e versioni 13 successive

Esempi

Il seguente esempio mostra l'output della funzione `aurora_stat_resource_usage`.

```
=> select * from aurora_stat_resource_usage();
 pid  | allocated_memory | used_memory |   cpu_usage_percent
------+------------------+-------------+-----------------------
  666 |          1074032 |      333544 |   0.00729274882897963
  667 |           787312 |      287360 | 0.0029263928146372746
  668 |          3076776 |     1563488 |  0.006013116835953961
  684 |           803744 |      307480 |  0.002226855426881142
 2401 |          1232992 |      943144 |                     0
  647 |             8000 |         944 |   0.48853387812429855
  659 |           319344 |      243000 | 0.0004135602076683591
  663 |           262000 |      185736 |  0.008181301476644002
  664 |             9024 |        1216 |   0.10992313082653653
(9 rows)
```

aurora_stat_statements

Mostra tutte le colonne `pg_stat_statements` e ne aggiunge altre alla fine.

Sintassi

```
aurora_stat_statements(showtext boolean)
```

Argomenti

showtext boolean

Tipo restituito

Record SETOF con tutte le colonne `pg_stat_statements` e le seguenti colonne aggiuntive. Per ulteriori informazioni sulle colonne `pg_stat_statements`, consulta `pg_stat_statements`.

È possibile ripristinare le statistiche per questa funzione utilizzando `pg_stat_statements_reset()`.

- `storage_blks_read`: numero totale di blocchi condivisi letti dall'archiviazione Aurora con questa istruzione.
- `orcache_blks_hit`: numero totale di accessi alla cache di Letture ottimizzate con questa istruzione.
- `storage_blk_read_time`: se `track_io_timing` è abilitato, tiene traccia del tempo totale, in millisecondi, impiegato dall'istruzione per leggere i blocchi condivisi dall'archiviazione Aurora, altrimenti il valore è zero. Per ulteriori informazioni, consulta track_io_timing.
- `local_blk_read_time`: se `track_io_timing` è abilitato, tiene traccia del tempo totale, in millisecondi, impiegato dall'istruzione per leggere i blocchi locali, altrimenti il valore è zero. Per ulteriori informazioni, consulta track_io_timing.
- `orcache_blk_read_time`: se `track_io_timing` è abilitato, tiene traccia del tempo totale, in secondi, impiegato dall'istruzione per leggere i blocchi condivisi dalla cache di letture ottimizzate, altrimenti il valore è zero. Per ulteriori informazioni, consulta track_io_timing.
- `total_plan_peakmem`: somma totale dei valori di memoria di picco durante la fase di pianificazione per tutte le chiamate a questa istruzione. Per visualizzare la memoria di picco media durante la pianificazione dell'istruzione, dividi questo valore per il numero di chiamate.
- `min_plan_peakmem`: il valore di memoria di picco più basso rilevato durante la pianificazione di tutte le chiamate a questa istruzione.
- `max_plan_peakmem`: il valore di memoria di picco più alto durante la pianificazione di tutte le chiamate a questa istruzione.
- `total_exec_peakmem`: somma totale dei valori di memoria di picco durante la fase di esecuzione per tutte le chiamate a questa istruzione. Per visualizzare la memoria di picco media durante l'esecuzione dell'istruzione, dividi questo valore per il numero di chiamate.
- `min_exec_peakmem`: il valore di memoria di picco più basso, in byte, rilevato durante l'esecuzione di tutte le chiamate a questa istruzione.

- `max_exec_peakmem`: il valore di memoria di picco più alto, in byte, rilevato durante l'esecuzione di tutte le chiamate a questa istruzione.

> **Note**
>
> `total_plan_peakmen`, `min_plan_peakmem` e `max_plan_peakmem` vengono monitorati solo quando l'impostazione `pg_stat_statements.track_planning` è attivata.

Note per l'utilizzo

Per utilizzare la funzione aurora_stat_statements(), è necessario includere l'estensione `pg_stat_statements` nel parametro `shared_preload_libraries`.

Questa funzione è disponibile per le seguenti versioni di Aurora PostgreSQL:

- 15.4 e versioni successive
- 14.9 e versioni successive

Le colonne che mostrano i picchi di memoria sono disponibili nelle seguenti versioni:

- 16.3 e versioni successive
- 15.7 e versioni successive
- 14.12 e versioni successive

Esempi

L'esempio seguente mostra come contiene tutte le colonne pg_stat_statements e aggiunge 11 nuove colonne alla fine:

```
=> select * from aurora_stat_statements(true) where query like 'with window_max%';
-[ RECORD 1 ]----------
+-----------------------------------------------------------------------------------------------
userid                 | 16409
dbid                   | 5
toplevel               | t
queryid                | -8347523682669847482
query                  | with window_max as (select custid, max(scratch) over (order by
 scratch rows between $1 preceding
```

```
and $2 following) wmax from ts) select sum(wmax), max(custid) from window_max
plans                   | 0
total_plan_time         | 0
min_plan_time           | 0
max_plan_time           | 0
mean_plan_time          | 0
stddev_plan_time        | 0
calls                   | 4
total_exec_time         | 254.105121
min_exec_time           | 57.503164000000005
max_exec_time           | 68.687418
mean_exec_time          | 63.52628025
stddev_exec_time        | 5.150765359979643
rows                    | 4
shared_blks_hit         | 200192
shared_blks_read        | 0
shared_blks_dirtied     | 0
shared_blks_written     | 0
local_blks_hit          | 0
local_blks_read         | 0
local_blks_dirtied      | 0
local_blks_written      | 0
temp_blks_read          | 0
temp_blks_written       | 0
blk_read_time           | 0
blk_write_time          | 0
temp_blk_read_time      | 0
temp_blk_write_time     | 0
wal_records             | 0
wal_fpi                 | 0
wal_bytes               | 0
jit_functions           | 0
jit_generation_time     | 0
jit_inlining_count      | 0
jit_inlining_time       | 0
jit_optimization_count  | 0
jit_optimization_time   | 0
jit_emission_count      | 0
jit_emission_time       | 0
storage_blks_read       | 0
orcache_blks_hit        | 0
storage_blk_read_time   | 0
local_blk_read_time     | 0
orcache_blk_read_time   | 0
```

```
total_plan_peakmem     | 0
min_plan_peakmem       | 0
max_plan_peakmem       | 0
total_exec_peakmem     | 6356224
min_exec_peakmem       | 1589056
max_exec_peakmem       | 1589056
```

aurora_stat_system_waits

Segnala le informazioni sugli eventi di attesa per l'istanza database di Aurora PostgreSQL.

Sintassi

```
aurora_stat_system_waits()
```

Argomenti

Nessuno

Tipo restituito

Record SETOF

Note per l'utilizzo

Questa funzione restituisce il numero cumulativo di attese e tempo di attesa cumulativo per ogni evento di attesa generato dall'istanza database a cui si è attualmente connessi.

Il recordset restituito include i seguenti campi:

- `type_id`: ID del tipo di evento di attesa.
- `event_id`: ID dell'evento di attesa.
- `waits`: numero di volte in cui si è verificato l'evento di attesa.
- `wait_time`: tempo di attesa totale in microsecondi per questo evento.

Le statistiche restituite da questa funzione vengono reimpostate al riavvio di un'istanza database.

Esempi

Nell'esempio seguente vengono mostrati i risultati della chiamata alla funzione `aurora_stat_system_waits`.

```
=> SELECT *
    FROM aurora_stat_system_waits();
 type_id | event_id  |   waits   |  wait_time
---------+-----------+-----------+--------------
       1 |  16777219 |        11 |        12864
       1 |  16777220 |       501 |       174473
       1 |  16777270 |     53171 |     23641847
       1 |  16777271 |        23 |       319668
       1 |  16777274 |        60 |        12759
.
.
.
      10 | 167772231 |    204596 |    790945212
      10 | 167772232 |         2 |        47729
      10 | 167772234 |         1 |          888
      10 | 167772235 |         2 |           64
```

L'esempio seguente mostra come puoi utilizzare questa funzione insieme a `aurora_stat_wait_event` e `aurora_stat_wait_type` per produrre risultati più leggibili.

```
=> SELECT type_name,
            event_name,
            waits,
            wait_time
        FROM aurora_stat_system_waits()
NATURAL JOIN aurora_stat_wait_event()
NATURAL JOIN aurora_stat_wait_type();

 type_name |       event_name       |  waits   |  wait_time
-----------+------------------------+----------+--------------
 LWLock    | XidGenLock             |       11 |        12864
 LWLock    | ProcArrayLock          |      501 |       174473
 LWLock    | buffer_content         |    53171 |     23641847
 LWLock    | rdsutils               |        2 |        12764
 Lock      | tuple                  |    75686 |   2033956052
 Lock      | transactionid          |  1765147 |  47267583409
 Activity  | AutoVacuumMain         |   136868 |  56305604538
 Activity  | BgWriterHibernate      |     7486 |  55266949471
 Activity  | BgWriterMain           |     7487 |   1508909964
.
.
.
 IO        | SLRURead               |        3 |        11756
```

```
IO        | WALWrite               | 52544463 |    388850428
IO        | XactSync               |   187073 |    597041642
IO        | ClogRead               |        2 |        47729
IO        | OutboundCtrlRead       |        1 |          888
IO        | OutboundCtrlWrite      |        2 |           64
```

aurora_stat_wait_event

Elenca tutti gli eventi di attesa supportati per Aurora PostgreSQL. Per ulteriori informazioni sulla sicurezza con Aurora PostgreSQL, consulta Eventi di attesa Amazon Aurora PostgreSQL.

Sintassi

```
aurora_stat_wait_event()
```

Argomenti

Nessuno

Tipo restituito

Record SETOF avente le seguenti colonne:

- type_id: ID del tipo di evento di attesa.
- event_id: ID dell'evento di attesa.
- type_name: nome del tipo di attesa
- event_name: nome dell'evento di attesa

Note per l'utilizzo

Per visualizzare i nomi degli eventi con tipi di evento anziché ID, utilizzare questa funzione insieme ad altre funzioni, ad esempio `aurora_stat_wait_type` e `aurora_stat_system_waits`. I nomi degli eventi di attesa restituiti da questa funzione sono gli stessi di quelli restituiti dalla funzione `aurora_wait_report`.

Esempi

Nell'esempio seguente vengono mostrati i risultati della chiamata alla funzione `aurora_stat_wait_event`.

```
=>  SELECT *
    FROM aurora_stat_wait_event();

 type_id | event_id  |               event_name
---------+-----------+-----------------------------------------
       1 |  16777216 | <unassigned:0>
       1 |  16777217 | ShmemIndexLock
       1 |  16777218 | OidGenLock
       1 |  16777219 | XidGenLock
.
.
.
       9 | 150994945 | PgSleep
       9 | 150994946 | RecoveryApplyDelay
      10 | 167772160 | BufFileRead
      10 | 167772161 | BufFileWrite
      10 | 167772162 | ControlFileRead
.
.
.
      10 | 167772226 | WALInitWrite
      10 | 167772227 | WALRead
      10 | 167772228 | WALSync
      10 | 167772229 | WALSyncMethodAssign
      10 | 167772230 | WALWrite
      10 | 167772231 | XactSync
.
.
.
      11 | 184549377 | LsnAllocate
```

L'esempio seguente unisce `aurora_stat_wait_type` e `aurora_stat_wait_event` per restituire nomi di tipo e nomi di eventi per una migliore leggibilità.

```
=> SELECT *
    FROM aurora_stat_wait_type() t
    JOIN aurora_stat_wait_event() e
      ON t.type_id = e.type_id;

 type_id | type_name | type_id | event_id  |               event_name
---------+-----------+---------+-----------+-----------------------------------------
       1 | LWLock    |       1 |  16777216 | <unassigned:0>
       1 | LWLock    |       1 |  16777217 | ShmemIndexLock
```

```
       1 | LWLock     |       1 |  16777218 | OidGenLock
       1 | LWLock     |       1 |  16777219 | XidGenLock
       1 | LWLock     |       1 |  16777220 | ProcArrayLock
.
.
.
       3 | Lock       |       3 |  50331648 | relation
       3 | Lock       |       3 |  50331649 | extend
       3 | Lock       |       3 |  50331650 | page
       3 | Lock       |       3 |  50331651 | tuple
.
.
.
      10 | IO         |      10 | 167772214 | TimelineHistorySync
      10 | IO         |      10 | 167772215 | TimelineHistoryWrite
      10 | IO         |      10 | 167772216 | TwophaseFileRead
      10 | IO         |      10 | 167772217 | TwophaseFileSync
.
.
.
      11 | LSN        |      11 | 184549376 | LsnDurable
```

aurora_stat_wait_type

Elenca tutti i tipi di attesa supportati per Aurora PostgreSQL.

Sintassi

```
aurora_stat_wait_type()
```

Argomenti

Nessuno

Tipo restituito

Record SETOF avente le seguenti colonne:

- type_id: ID del tipo di evento di attesa.
- type_name: nome del tipo di attesa

Note per l'utilizzo

Per visualizzare i nomi degli eventi di attesa con tipi di eventi di attesa anziché ID, utilizzare questa funzione insieme ad altre funzioni come `aurora_stat_wait_event` e `aurora_stat_system_waits`. I nomi dei tipi di attesa restituiti da questa funzione sono gli stessi di quelli restituiti dalla funzione `aurora_wait_report`.

Esempi

Nell'esempio seguente vengono mostrati i risultati della chiamata alla funzione `aurora_stat_wait_type`.

```
=> SELECT *
     FROM aurora_stat_wait_type();
 type_id | type_name
---------+-----------
       1 | LWLock
       3 | Lock
       4 | BufferPin
       5 | Activity
       6 | Client
       7 | Extension
       8 | IPC
       9 | Timeout
      10 | IO
      11 | LSN
```

aurora_version

Restituisce il valore di stringa del numero di versione di Amazon Aurora PostgreSQL-Compatible Edition.

Sintassi

```
aurora_version()
```

Argomenti

Nessuno

Tipo restituito

Stringa CHAR o VARCHAR

Note per l'utilizzo

Questa funzione visualizza la versione del motore di database di Amazon Aurora PostgreSQL-Compatible Edition. Il numero di versione viene restituito come stringa formattata come *principale*.*secondaria*.*patch*. Per ulteriori informazioni sui numeri di versione del motore Aurora PostgreSQL, consulta Numero versione Aurora

Puoi scegliere quando applicare aggiornamenti di versione secondaria impostando la finestra di manutenzione per il cluster database Aurora PostgreSQL. Per scoprire come fare, consulta Manutenzione di un cluster di database Amazon Aurora.

A partire dal rilascio di PostgreSQL versioni 13.3, 12.8, 11.13, 10.18 e per tutte le altre versioni successive, i numeri di versione di Aurora seguono i numeri di versione di PostgreSQL. Per ulteriori informazioni su questa versione, consulta Aggiornamenti di Amazon Aurora PostgreSQL nelle Note di rilascio di Aurora PostgreSQL.

Esempi

L'esempio seguente mostra i risultati della chiamata della funzione `aurora_version` su un cluster database Aurora PostgreSQL in esecuzione prima su PostgreSQL 12.7, Aurora PostgreSQL versione 4.2 e poi su Aurora PostgreSQL versione 13.3.

```
=> SELECT * FROM aurora_version();
aurora_version
----------------
 4.2.2
SELECT * FROM aurora_version();
aurora_version
----------------
 13.3.0
```

Questo esempio mostra varie opzioni di utilizzo della funzione per ottenere maggiori dettagli sulla versione di Aurora PostgreSQL. In questo esempio Aurora presenta un numero di versione distinto rispetto a PostgreSQL.

```
=> SHOW SERVER_VERSION;
```

```
 server_version
----------------
 12.7
(1 row)

=> SELECT * FROM aurora_version();
aurora_version
----------------
 4.2.2
(1 row)

=> SELECT current_setting('server_version') AS "PostgreSQL Compatiblility";
PostgreSQL Compatiblility
--------------------------
 12.7
(1 row)

=> SELECT version() AS "PostgreSQL Compatiblility Full String";
PostgreSQL Compatiblility Full String
-------------------------------------------------------------------------------------------------
 PostgreSQL 12.7 on aarch64-unknown-linux-gnu, compiled by aarch64-unknown-linux-gnu-
gcc (GCC) 7.4.0, 64-bit
(1 row)

=> SELECT 'Aurora: '
    || aurora_version()
    || ' Compatible with PostgreSQL: '
    || current_setting('server_version') AS "Instance Version";
Instance Version
-----------------------------------------------
 Aurora: 4.2.2 Compatible with PostgreSQL: 12.7
(1 row)
```

Nell'esempio successivo, la funzione viene utilizzata con le stesse opzioni dell'esempio precedente. In questo esempio il numero di versione di Aurora non è distinto rispetto al numero di versione di PostgreSQL.

```
=> SHOW SERVER_VERSION;
server_version
----------------
 13.3
```

```
=> SELECT * FROM aurora_version();
aurora_version
---------------
 13.3.0
=> SELECT current_setting('server_version') AS "PostgreSQL Compatiblility";
PostgreSQL Compatiblility
--------------------------
 13.3

=> SELECT version() AS "PostgreSQL Compatiblility Full String";
PostgreSQL Compatiblility Full String
-------------------------------------------------------------------------------------------------
 PostgreSQL 13.3 on x86_64-pc-linux-gnu, compiled by x86_64-pc-linux-gnu-gcc (GCC)
 7.4.0, 64-bit
=> SELECT 'Aurora: '
    || aurora_version()
    || ' Compatible with PostgreSQL: '
    || current_setting('server_version') AS "Instance Version";
Instance Version
-----------------------------------------------------
 Aurora: 13.3.0 Compatible with PostgreSQL: 13.3
```

aurora_volume_logical_start_lsn

Restituisce il numero di sequenza di log (LSN) utilizzato per identificare l'inizio di un record nel flusso WAL (write-ahead log) logico del volume del cluster Aurora.

Sintassi

```
aurora_volume_logical_start_lsn()
```

Argomenti

Nessuno

Tipo restituito

`pg_lsn`

Note per l'utilizzo

Questa funzione identifica l'inizio del record nel flusso WAL logico per un determinato volume del cluster Aurora. È possibile utilizzare questa funzione durante l'aggiornamento della versione

principale utilizzando la replica logica e la clonazione rapida Aurora per determinare il numero di sequenza di log in corrispondenza del quale viene eseguito uno snapshot o un clone del database. È quindi possibile utilizzare la replica logica per trasmettere continuamente i record di dati più recenti dopo il numero di sequenza di log e sincronizzare le modifiche da editore a sottoscrittore.

Per ulteriori informazioni sull'utilizzo della replica logica per l'aggiornamento di una versione principale, consulta Utilizzo della replica logica per eseguire l'aggiornamento a una versione principale per Aurora PostgreSQL.

Questa funzione è disponibile per le seguenti versioni di Aurora PostgreSQL:

- 15.2 o versioni successive alla 15
- 14.3 o versioni successive alla 14
- 13.6 o versioni successive alla 13
- 12.10 e versioni successive alla 12
- 11.15 e versioni successive alla 11
- 10.20 e versioni successive alla 10

Esempi

È possibile ottenere il numero di sequenza di log utilizzando la seguente query:

```
postgres=> SELECT aurora_volume_logical_start_lsn();

aurora_volume_logical_start_lsn
---------------
0/402E2F0
(1 row)
```

aurora_wait_report

Questa funzione mostra l'attività dell'evento di attesa durante un periodo di tempo.

Sintassi

```
aurora_wait_report([time])
```

Argomenti

tempo (facoltativo)

Il tempo in secondi. Il valore predefinito è 10 secondi.

Tipo restituito

Record SETOF avente le seguenti colonne:

- type_name: nome del tipo di attesa
- event_name: nome dell'evento di attesa
- wait: numero di attese
- wait_time: tempo di attesa in millisecondi
- ms_per_wait: durata media di un'attesa in millisecondi
- waits_per_xact: media delle attese per ogni transazione
- ms_per_xact: durata media di una transizione in millisecondi

Note per l'utilizzo

Questa funzione è disponibile a partire da Aurora PostgreSQL versione 1.1 compatibile con PostgreSQL 9.6.6 e versioni successive.

Per utilizzare questa funzione è necessario innanzitutto creare l'estensione `aurora_stat_utils` Aurora PostgreSQL, nel modo seguente:

```
=> CREATE extension aurora_stat_utils;
CREATE EXTENSION
```

Per ulteriori informazioni sulle versioni dell'estensione Aurora PostgreSQL disponibili, consulta Versioni delle estensioni per Amazon Aurora PostgreSQL in Note di rilascio di Aurora PostgreSQL.

Questa funzione calcola gli eventi di attesa a livello di istanza confrontando i dati statistici di due snapshot della funzione aurora_stat_system_waits () e di pg_stat_database, acquisite dalla vista delle statistiche di PostgreSQL.

Per ulteriori informazioni su `aurora_stat_system_waits()` e `pg_stat_database`, consulta Raccolta di statistiche nella documentazione di PostgreSQL.

Quando è in esecuzione, questa funzione acquisisce una snapshot iniziale, attende il numero di secondi specificato e acquisisce una seconda snapshot. La funzione confronta le due snapshot e restituisce la differenza. Questa differenza rappresenta l'attività dell'istanza in quell'intervallo di tempo.

Sull'istanza di scrittura, la funzione visualizza anche il numero di transazioni di cui è stato eseguito il commit e il TPS (transazioni al secondo). Questa funzione restituisce informazioni a livello di istanza e include tutti i database sull'istanza.

Esempi

Questo esempio mostra come creare l'estensione aurora_stat_utils per poter utilizzare la funzione aurora_wait_report.

```
=> CREATE extension aurora_stat_utils;
CREATE EXTENSION
```

Questo esempio mostra come controllare il report delle attese per 10 secondi.

```
=> SELECT *
     FROM aurora_wait_report();
NOTICE:  committed 34 transactions in 10 seconds (tps 3)
 type_name |    event_name     | waits | wait_time | ms_per_wait | waits_per_xact |
 ms_per_xact
-----------+-------------------+-------+-----------+-------------+----------------
+-------------
 Client    | ClientRead        |    26 |  30003.00 |    1153.961 |           0.76 |
 882.441
 Activity  | WalWriterMain     |    50 |  10051.32 |     201.026 |           1.47 |
 295.627
 Timeout   | PgSleep           |     1 |  10049.52 |   10049.516 |           0.03 |
 295.574
 Activity  | BgWriterHibernate |     1 |  10048.15 |   10048.153 |           0.03 |
 295.534
 Activity  | AutoVacuumMain    |    18 |   9941.66 |     552.314 |           0.53 |
 292.402
 Activity  | BgWriterMain      |     1 |    201.09 |     201.085 |           0.03 |
   5.914
 IO        | XactSync          |    15 |     25.34 |       1.690 |           0.44 |
   0.745
 IO        | RelationMapRead   |    12 |      0.54 |       0.045 |           0.35 |
   0.016
```

```
 IO        | WALWrite          |    84 |      0.21 |       0.002 |           2.47 |
   0.006
 IO        | DataFileExtend    |     1 |      0.02 |       0.018 |           0.03 |
   0.001
```

Questo esempio mostra come controllare il report delle attese per 60 secondi.

```
=> SELECT *
     FROM aurora_wait_report(60);
NOTICE:  committed 1544 transactions in 60 seconds (tps 25)
 type_name |       event_name       |  waits  | wait_time | ms_per_wait |
 waits_per_xact | ms_per_xact
-----------+------------------------+---------+-----------+-------------
+----------------+-------------
 Lock      | transactionid          |    6422 | 477000.53 |      74.276 |
 4.16 |     308.938
 Client    | ClientRead             |    8265 | 270752.99 |      32.759 |
 5.35 |     175.358
 Activity  | CheckpointerMain       |       1 |  60100.25 |   60100.246 |
 0.00 |      38.925
 Timeout   | PgSleep                |       1 |  60098.49 |   60098.493 |
 0.00 |      38.924
 Activity  | WalWriterMain          |     296 |  60010.99 |     202.740 |
 0.19 |      38.867
 Activity  | AutoVacuumMain         |     107 |  59827.84 |     559.139 |
 0.07 |      38.749
 Activity  | BgWriterMain           |     290 |  58821.83 |     202.834 |
 0.19 |      38.097
 IO        | XactSync               |    1295 |  55220.13 |      42.641 |
 0.84 |      35.764
 IO        | WALWrite               | 6602259 |  47810.94 |       0.007 |
 4276.07 |      30.966
 Lock      | tuple                  |     473 |  29880.67 |      63.173 |
 0.31 |      19.353
 LWLock    | buffer_mapping         |     142 |   3540.13 |      24.930 |
 0.09 |       2.293
 Activity  | BgWriterHibernate      |     290 |   1124.15 |       3.876 |
 0.19 |       0.728
 IO        | BufFileRead            |    7615 |    618.45 |       0.081 |
 4.93 |       0.401
 LWLock    | buffer_content         |      73 |    345.93 |       4.739 |
 0.05 |       0.224
```

```
LWLock    | lock_manager           |      62 |    191.44 |       3.088 |
0.04 |       0.124
IO        | RelationMapRead        |      72 |      5.16 |       0.072 |
0.05 |       0.003
LWLock    | ProcArrayLock          |       1 |      2.01 |       2.008 |
0.00 |       0.001
IO        | ControlFileWriteUpdate |       2 |      0.03 |       0.013 |
0.00 |       0.000
IO        | DataFileExtend         |       1 |      0.02 |       0.018 |
0.00 |       0.000
IO        | ControlFileSyncUpdate  |       1 |      0.00 |       0.000 |
0.00 |       0.000
```

Amazon Aurora PostgreSQL parametri

La gestione del cluster di database di Amazon Aurora è uguale a quella delle istanze database Amazon RDS, ovvero utilizza i parametri di un gruppo parametri del database. Tuttavia, Amazon Aurora si differenzia da Amazon RDS in quanto un cluster DB Aurora ha più istanze DB. Alcuni parametri che utilizzi per gestire il cluster DB Amazon Aurora DB si applicano a tutto il cluster, mentre altri si applicano soltanto a una specifica istanza database nel cluster DB, come segue.

- Gruppo di parametri del cluster DB: un gruppo di parametri del cluster DB contiene il set di parametri di configurazione del motore applicabili in tutto il cluster DB Aurora. Ad esempio, la gestione della cache del cluster è una caratteristica di un cluster DB Aurora controllato dal parametro `apg_ccm_enabled` che fa parte del gruppo di parametri del cluster di database. Il gruppo di parametri del cluster DB contiene anche le impostazioni predefinite per il gruppo parametri del database per le istanze database che costituiscono il cluster.
- Gruppo parametri del database: un gruppo di parametri del database è l'insieme di valori di configurazione del motore che si applicano a un'istanza database specifica di quel tipo di motore. I gruppi parametri del database per il motore PostgreSQL DB vengono utilizzati da un'istanza database RDS per PostgreSQL e da un cluster database Aurora PostgreSQL. Queste impostazioni di configurazione si applicano alle proprietà che possono variare tra le istanze database all’interno di un cluster Aurora, ad esempio le dimensioni dei buffer di memoria.

I parametri a livello di cluster sono gestiti nei gruppi di parametri del cluster database. I parametri a livello di istanza sono gestiti nei gruppi parametri del database. È possibile gestire i parametri tramite la console Amazon RDS, la AWS CLI o l'API Amazon RDS. Sono disponibili comandi separati per la gestione dei parametri a livello di cluster e a livello di istanza.

- Per gestire i parametri a livello di cluster in un gruppo di parametri del cluster di database, utilizzare il comando AWS CLI modify-db-cluster-parameter-group.
- Per gestire i parametri a livello di istanza in un gruppo di parametri database per un'istanza database in un cluster di database, utilizzare il comando AWS CLI modify-db-parameter-group.

Per ulteriori informazioni sulla AWS CLI, consulta Utilizzo della AWS CLI nella Guida per l'utente della AWS Command Line Interface.

Per ulteriori informazioni sui gruppi di parametri, consultare Gruppi di parametri per Amazon Aurora.

Visualizzazione del cluster di database Aurora PostgreSQL e dei parametri del database

È possibile visualizzare tutti i gruppi di parametri di default disponibili per le istanze DB Amazon RDS for PostgreSQL e per i cluster DB Aurora PostgreSQL nella Console di gestione AWS. I gruppi di parametri di default per tutti i motori di database e i tipi e le versioni di cluster DB sono elencati per ciascuna regione AWS. Vengono elencati anche tutti i gruppi di parametri personalizzati.

Piuttosto che visualizzarli nella Console di gestione AWS, è inoltre possibile elencare i parametri contenuti nei gruppi di parametri di cluster di database e gruppi parametri del database utilizzando la AWS CLIo l'API Amazon RDS. Ad esempio, per elencare i parametri in un gruppo di parametri del cluster di database, utilizza il comando della AWS CLI describe-db-cluster-parameters come segue:

```
aws rds describe-db-cluster-parameters --db-cluster-parameter-group-name
 default.aurora-postgresql12
```

Il comando restituisce descrizioni JSON dettagliate di ciascun parametro. Per ridurre la quantità di informazioni restituite, è possibile specificare le preferenze utilizzando l'opzione `--query`. Ad esempio, puoi ottenere il nome del parametro, la relativa descrizione e i valori consentiti per il gruppo di parametri del cluster di database Aurora PostgreSQL 12 di default come segue:

Per Linux, macOS o Unix:

```
aws rds describe-db-cluster-parameters --db-cluster-parameter-group-name
 default.aurora-postgresql12 \
    --query 'Parameters[].
[{ParameterName:ParameterName,Description:Description,ApplyType:ApplyType,AllowedValues:Allowed
```

Per Windows:

```
aws rds describe-db-cluster-parameters --db-cluster-parameter-group-name
 default.aurora-postgresql12 ^
    --query "Parameters[].
[{ParameterName:ParameterName,Description:Description,ApplyType:ApplyType,AllowedValues:Allowed
```

Un gruppo di parametri del cluster di DB Aurora include il gruppo di parametri dell'istanza database e i valori di default per un determinato motore di Aurora DB. È possibile ottenere l'elenco dei parametri del database dallo stesso gruppo di parametri di default di Aurora PostgreSQL utilizzando il comando della AWS CLI describe-db-parameters come mostrato di seguito.

Per Linux, macOS o Unix:

```
aws rds describe-db-parameters --db-parameter-group-name default.aurora-postgresql12 \
    --query 'Parameters[].
[{ParameterName:ParameterName,Description:Description,ApplyType:ApplyType,AllowedValues:Allowed
```

Per Windows:

```
aws rds describe-db-parameters --db-parameter-group-name default.aurora-postgresql12 ^
    --query "Parameters[].
[{ParameterName:ParameterName,Description:Description,ApplyType:ApplyType,AllowedValues:Allowed
```

I comandi precedenti restituiscono elenchi di parametri dal cluster di database o dal gruppo parametri del database con descrizioni e altri dettagli specificati nella query. Di seguito è riportata una risposta di esempio:

```
[
    [
        {
            "ParameterName": "apg_enable_batch_mode_function_execution",
            "ApplyType": "dynamic",
            "Description": "Enables batch-mode functions to process sets of rows at a
 time.",
            "AllowedValues": "0,1"
        }
    ],
    [
        {
            "ParameterName": "apg_enable_correlated_any_transform",
            "ApplyType": "dynamic",
```

```
            "Description": "Enables the planner to transform correlated ANY Sublink
(IN/NOT IN subquery) to JOIN when possible.",
            "AllowedValues": "0,1"
        }
    ],...
```

Di seguito sono riportate le tabelle contenenti i valori per il parametro del cluster database di default e il parametro database per Aurora PostgreSQL versione 14.

Parametri a livello di cluster Aurora PostgreSQL

È possibile visualizzare i parametri a livello di cluster disponibili per una versione specifica di Aurora PostgreSQL con la Console di gestione AWS, l'Interfaccia della linea di comando AWS (AWS CLI) o l'API Amazon RDS. Per informazioni sulla visualizzazione dei parametri in un gruppo di parametri del cluster database Aurora PostgreSQL nella console RDS, consultare Visualizzazione dei valori dei parametri per un gruppo di parametri del cluster di database in Amazon Aurora.

Alcuni parametri a livello di cluster non sono disponibili in tutte le versioni e alcuni sono obsoleti. Per informazioni sulla visualizzazione dei parametri di una versione specifica di Aurora PostgreSQL, consultare Visualizzazione del cluster di database Aurora PostgreSQL e dei parametri del database.

Ad esempio, nella tabella seguente sono elencati i parametri disponibili nel gruppo di parametri del cluster database di default per Aurora PostgreSQL versione 14. Se si crea un cluster di database Aurora PostgreSQL senza specificare il proprio gruppo parametri del database personalizzato, il cluster di database viene creato utilizzando il gruppo di parametri cluster di database Aurora predefinito per la versione scelta, ad esempio `default.aurora-postgresql14`, `default.aurora-postgresql13` e così via.

Per un elenco dei parametri dell'istanza database per lo stesso gruppo di parametri del cluster database, consulta Parametri a livello di istanza Aurora PostgreSQL.

Nome del parametro	Descrizione	Default
ansi_constraint_trigger_ordering	Modifica l'ordine di attivazione dei trigger di vincolo in modo che siano compatibili con lo standard ANSI SQL.	–
ansi_force_foreign_key_checks	Assicurati che le operazioni referenziali come l'eliminazione a cascata o l'aggiornamento a cascata si verifichino sempre indipendentemente dai vari contesti di attivazione esistenti per l'operazione.	–
ansi_qualified_update_set_target	Tabella di supporto e qualificatori dello schema nelle istruzioni UPDATE ... SET.	–
apg_ccm_enabled	Abilita o disabilita la gestione della cache del cluster per il cluster.	–

Nome del parametro	Descrizione	Default
apg_enable_batch_mode_function_execution	Consente alle funzioni in modalità batch di elaborare serie di righe in una volta.	–
apg_enable_correlated_any_transform	Consente al pianificatore di trasformare il link secondario ANY (sottoquery IN/NOT IN) in JOIN quando possibile.	–
apg_enable_function_migration	Consente al pianificatore di migrare le funzioni scalari idonee alla clausola FROM.	–
apg_enable_not_in_transform	Consente al pianificatore di trasformare la query secondaria NOT IN in ANTI JOIN quando possibile.	–
apg_enable_remove_redundant_inner_joins	Consente al pianificatore di rimuovere i join interni ridondanti.	–
apg_enable_semijoin_push_down	Consente l'utilizzo di filtri semijoin per gli hash join.	–
apg_plan_mgmt.capture_plan_baselines	Modalità baseline di acquisizione del piano. manual (manuale): abilita l'acquisizione del piano per qualsiasi istruzione SQL, off: disabilita l'acquisizione del piano, automatic (automatica): abilita l'acquisizione del piano per le istruzioni in pg_stat_statements che soddisfano i criteri di idoneità.	off
apg_plan_mgmt.max_databases	Imposta il numero massimo di database che possono gestire le query utilizzando apg_plan_mgmt.	10
apg_plan_mgmt.max_plans	Imposta il numero massimo di piani che possono essere memorizzati nella cache da apg_plan_mgmt.	10000

Nome del parametro	Descrizione	Default
apg_plan_mgmt.plan_retention_period	Numero massimo di giorni dall'ultimo utilizzo di un piano prima che venga eliminato automaticamente.	32
apg_plan_mgmt.unapproved_plan_execution_threshold	Costo totale stimato del piano al di sotto del quale verrà eseguito un piano non approvato.	0
apg_plan_mgmt.use_plan_baselines	Utilizzare solo piani approvati o fissi per le istruzioni gestite.	false
application_name	Imposta il nome dell'applicazione da riportare nelle statistiche e nei registri.	–
array_nulls	Abilita l'inserimento di elementi NULL negli array.	–
aurora_compute_plan_id	Monitora i piani di esecuzione delle query per rilevare i piani di esecuzione che contribuiscono al carico corrente del database e per tenere traccia delle statistiche sulle prestazioni dei piani di esecuzione nel tempo. Per ulteriori informazioni, consulta Monitoring query execution plans for Aurora PostgreSQL.	on
authentication_timeout	(s) Imposta il tempo massimo concesso per completare l'autenticazione del client.	–
auto_explain.log_analyze	Usa EXPLAIN ANALYZE per la registrazione del piano.	–
auto_explain.log_buffers	Registra l'utilizzo dei buffer.	–
auto_explain.log_format	Formato EXPLAIN da utilizzare per la registrazione del piano.	–

Nome del parametro	Descrizione	Default
auto_explain.log_min_duration	Imposta il tempo minimo di esecuzione al di sopra del quale verranno registrati i piani.	–
auto_explain.log_nested_statements	Registra le istruzioni nidificate.	–
auto_explain.log_timing	Raccogli i dati temporali, non solo il numero di righe.	–
auto_explain.log_triggers	Includi statistiche di attivazione nei piani.	–
auto_explain.log_verbose	Usa EXPLAIN VERBOSE per la registrazione del piano.	–
auto_explain.sample_rate	Frazione di query da elaborare.	–
autovacuum	Avvia il sottoprocesso autovacuum.	–
autovacuum_analyze_scale_factor	Numero di inserimenti, aggiornamenti o eliminazioni di tupla prima dell'analisi come una frazione di reltuple.	0,05
autovacuum_analyze_threshold	Numero minimo di inserti, aggiornamenti o eliminazioni di tupla prima dell'analisi.	–
autovacuum_freeze_max_age	Età nella quale eseguire l'autovacuum in una tabella per impedire il wraparound ID della transazione.	–
autovacuum_max_workers	Imposta il numero massimo di processi dipendenti di autovacuum in esecuzione simultanea	GREATEST(DBInstanceClassMemory/64371566592,3)

Nome del parametro	Descrizione	Default
autovacuum_multixact_freeze_max_age	Età Multixact a cui eseguire l'autovacuum in una tabella per impedire il wraparound multixact.	–
autovacuum_naptime	(s) Periodo di inattività tra le esecuzioni di autovacuum.	5
autovacuum_vacuum_cost_delay	(ms) Ritardo del costo del vacuum, in millisecondi, per l'autovacuum.	5
autovacuum_vacuum_cost_limit	Quantità del costo del vacuum disponibile prima del napping, per l'autovacuum.	GREATEST(log(DBInstanceClassMemory/21474836480)*600,200)
autovacuum_vacuum_insert_scale_factor	Numero di inserti, aggiornamenti o eliminazioni di tupla prima di eseguire il vacuum come una frazione di reltuple.	–
autovacuum_vacuum_insert_threshold	Numero minimo di inserimenti tupla prima del vacuum o -1 per disabilitare i vacuum.	–
autovacuum_vacuum_scale_factor	Numero di aggiornamenti o eliminazioni di tupla prima del vacuum come una frazione di reltuple.	0.1
autovacuum_vacuum_threshold	Numero minimo di aggiornamenti o eliminazioni di tupla prima del vacuum.	–
autovacuum_work_mem	(kB) Imposta la memoria massima da utilizzare per ogni processo di dipendente autovacuum.	GREATEST(DBInstanceClassMemory/32768,131072)
babelfishpg_tds.default_server_name	Il nome del server Babelfish predefinito	Microsoft SQL Server

Nome del parametro	Descrizione	Default
babelfishpg_tds.listen_address	Imposta il nome host o l'indirizzo o gli indirizzi IP su cui ascoltare TDS.	*
babelfishpg_tds.port	Imposta la porta TDS TCP su cui il server è in ascolto.	1433
babelfishpg_tds.tds_debug_log_level	Imposta il livello di registrazione in TDS, 0 disabilita la registrazione	1
babelfishpg_tds.tds_default_numeric_precision	Imposta la precisione predefinita del tipo numerico da inviare nei metadati della colonna TDS se il motore non ne specifica uno.	38
babelfishpg_tds.tds_default_numeric_scale	Imposta la scala predefinita di tipo numerico da inviare nei metadati della colonna TDS se il motore non ne specifica uno.	8
babelfishpg_tds.tds_default_packet_size	Imposta la dimensione predefinita del pacchetto per tutti i client SQL Server in fase di connessione	4096
babelfishpg_tds.tds_default_protocol_version	Imposta una versione del protocollo TDS predefinita per tutti i client in fase di connessione	DEFAULT
babelfishpg_tds.tds_ssl_encrypt	Imposta l'opzione di crittografia SSL	0
babelfishpg_tds.tds_ssl_max_protocol_version	Imposta la versione massima del protocollo SSL/TLS da utilizzare per la sessione TDS.	TLSv1.2

Nome del parametro	Descrizione	Default
babelfishpg_tds.tds_ssl_min_protocol_version	Imposta la versione minima del protocollo SSL/TLS da utilizzare per la sessione TDS.	TLSv1.2 da Aurora PostgreSQL versione 16, TLSv1 per versioni precedenti ad Aurora PostgreSQL versione 16
babelfishpg_tsql.default_locale	Localizzazione predefinita da utilizzare per le regole di confronto create tramite CREATE COLLATION.	it-IT
babelfishpg_tsql.migration_mode	Definisce se sono supportati più database utente	multi-db da Aurora PostgreSQL versione 16, single-db per versioni precedenti ad Aurora PostgreSQL versione 16
babelfishpg_tsql.server_collation_name	Nome delle regole di confronto del server predefinite	SQL_Latin1_General_CP1_CI_AS (default)
babelfishpg_tsql.versione	Imposta l'output della variabile @ @VERSION	default
backend_flush_after	(8Kb) Numero di pagine dopo le quali le scritture eseguite in precedenza vengono riportate su disco.	–
backslash_quote	Imposta se una doppia barra rovesciata \\ è consentita nelle stringhe letterali.	–
backtrace_functions	Registra backtrace per errori in queste funzioni.	–
bytea_output	Imposta il formato di output per byte.	–
check_function_bodies	Controlla i corpi delle funzioni durante CREATE FUNCTION.	–

Nome del parametro	Descrizione	Default
client_connection_check_interval	Imposta l'intervallo di tempo tra i controlli di disconnessione durante l'esecuzione di query.	–
client_encoding	Imposta la codifica del set di caratteri del client.	UTF8
client_min_messages	Imposta i livelli dei messaggi che vengono inviati al client.	–
compute_query_id	Calcola gli identificatori delle query.	auto
config_file	Imposta il file di configurazione principale del server.	/rdsdbdata/config/postgresql.conf
constraint_exclusion	Consente al pianificatore di utilizzare i vincoli per ottimizzare le query.	–
cpu_index_tuple_cost	Imposta la stima del pianificatore del costo di elaborazione di ciascuna voce di indice durante una scansione dell’indice.	–
cpu_operator_cost	Imposta la stima del pianificatore del costo di elaborazione di ciascuna chiamata dell’operatore o della funzione.	–
cpu_tuple_cost	Imposta la stima del pianificatore del costo di elaborazione di ciascuna tupla (riga).	–
cron.database_name	Imposta il database per archiviare le tabelle di metadati pg_cron	postgres
cron.log_run	Registra tutti i processi eseguiti nella tabella job_run_details	on
cron.log_statement	Registra tutte le istruzioni cron prima dell’esecuzione.	off
cron.max_running_jobs	Numero massimo di processi che possono essere eseguiti contemporaneamente.	5

Nome del parametro	Descrizione	Default
cron.use_background_workers	Abilita i dipendente in background per pg_cron	on
cursor_tuple_fraction	Imposta la stima del pianificatore della frazione delle righe del cursore che verranno recuperate.	–
data_directory	Imposta la directory dei dati dei server.	/rdsdbdata/db
datestyle	Imposta il formato del display per i valori di data e ora.	–
db_user_namespace	Abilita i nomi utente per database.	–
deadlock_timeout	(ms) Imposta il tempo di attesa su un lock prima di verificare il deadlock.	–
debug_pretty_print	I trattini analizzano e visualizzano le visualizzazioni dell'albero.	–
debug_print_parse	Registra ogni albero di analisi della query.	–
debug_print_plan	Registra ogni programma di esecuzione della query.	–
debug_print_rewritten	Registra ogni albero di analisi riscritto della query.	–
default_statistics_target	Imposta la destinazione della statistica predefinita.	–
default_tablespace	Imposta il tablespace predefinito per la creazione di tabelle e indici.	–
default_toast_compression	Imposta il metodo di compressione predefinito per i valori comprimibili.	–

Nome del parametro	Descrizione	Default
default_transaction_deferrable	Imposta lo stato differibile predefinito delle nuove transazioni.	–
default_transaction_isolation	Imposta il livello di isolamento della transazione di ogni nuova transazione.	–
default_transaction_read_only	Imposta lo stato di sola lettura predefinito delle nuove transazioni.	–
effective_cache_size	(8 kB) Imposta l'ipotesi del pianificatore sulla dimensione della cache del disco.	SUM(DBInstanceClassMemory/12038,-50003)
effective_io_concurrency	Numero di richieste simultanee che possono essere gestite in modo efficace dal sottosistema del disco.	–
enable_async_append	Consente ai pianificatori di utilizzare piani di aggiunta asincroni.	–
enable_bitmapscan	Abilita l’utilizzo da parte del pianificatore di piani di scansione bitmap.	–
enable_gathermerge	Abilita l’utilizzo da parte del pianificatore di piani di unione di raccolta.	–
enable_hashagg	Abilita l’utilizzo da parte del pianificatore di piani di aggregazione hash.	–
enable_hashjoin	Abilita l’utilizzo da parte del pianificatore di piani di unione hash.	–
enable_incremental_sort	Consente ai pianificatori di utilizzare fasi incrementali di ordinamento.	–
enable_indexonlyscan	Abilita l’utilizzo da parte del pianificatore di piani di sola scansione dell’indice.	–

Nome del parametro	Descrizione	Default
enable_indexscan	Abilita l’utilizzo da parte del pianificatore di piani di scansione dell’indice.	–
enable_material	Abilita l’utilizzo da parte del pianificatore della materializzazione.	–
enable_memoize	Abilita l'uso della memorizzazione da parte dei pianificatori	–
enable_mergejoin	Abilita l’utilizzo da parte del pianificatore di piani di unione.	–
enable_nestloop	Abilita l’utilizzo da parte del pianificatore di piani di unione a ciclo nested.	–
enable_parallel_append	Consente ai pianificatori di utilizzare piani di aggiunta paralleli.	–
enable_parallel_hash	Consente ai pianificatori di utilizzare piani di hash paralleli.	–
enable_partition_pruning	Abilita l’eliminazione delle partizioni del tempo di pianificazione e del runtime.	–
enable_partitionwise_aggregate	Consente l’aggregazione e il raggruppamento a livello di partizione.	–
enable_partitionwise_join	Attiva il join a livello di partizione.	–
enable_seqscan	Abilita l’utilizzo da parte del pianificatore di piani di scansione sequenziali.	–
enable_sort	Abilita l’utilizzo da parte del pianificatore di passaggi di ordinamento espliciti.	–
enable_tidscan	Abilita l’utilizzo da parte del pianificatore di piani di scansione TID.	–

Nome del parametro	Descrizione	Default
escape_string_warning	Avvisa circa la perdita di barre rovesciate nelle stringhe letterali ordinarie.	–
exit_on_error	Termina la sessione in caso di errore.	–
extra_float_digits	Imposta il numero di cifre visualizzate per i valori del punto variabile.	–
force_parallel_mode	Forza l'uso di strutture di query parallele.	–
from_collapse_limit	Imposta la dimensione dell'elenco FROM oltre la quale le sottoquery non vengono compresse.	–
geqo	Abilita l'ottimizzazione genetica delle query.	–
geqo_effort	GEQO: lo sforzo viene utilizzato per impostare il valore predefinito per altri parametri GEQO.	–
geqo_generations	GEQO: numero di iterazioni dell'algoritmo.	–
geqo_pool_size	GEQO: numero di individui nella popolazione.	–
geqo_seed	GEQO: seme per la selezione casuale del percorso.	–
geqo_selection_bias	GEQO: pressione selettiva all'interno della popolazione.	–
geqo_threshold	Imposta la soglia degli elementi FROM oltre i quali viene utilizzato GEQO.	–
gin_fuzzy_search_limit	Imposta il risultato massimo consentito per la ricerca esatta da GIN.	–
gin_pending_list_limit	(kB) Imposta la dimensione massima dell'elenco in sospeso per l'indice GIN.	–

Nome del parametro	Descrizione	Default
hash_mem_multiplier	Multipli di work_mem da utilizzare per le tabelle hash.	–
hba_file	Imposta il file di configurazione hba dei server.	/rdsdbdata/config/pg_hba.conf
hot_standby_feedback	Consente il feedback da uno standby a caldo all'elemento primario che eviterà conflitti di query.	on
huge_pages	Riduce il sovraccarico quando un'istanza database lavora con grandi blocchi di memoria contigui, come quelli utilizzati dai buffer condivisi. È attivato per impostazione predefinita per tutte le classi di istanza database diverse dalle classi di istanza t3.medium, db.t3.large, db.t4g.medium, db.t4g.large	on
ident_file	Imposta il file di configurazione ident dei server.	/rdsdbdata/config/pg_ident.conf
idle_in_transaction_session_timeout	(ms) Imposta la durata massima concessa di ogni transazione inattiva.	86400000
idle_session_timeout	Termina qualsiasi sessione che è rimasta inattiva (ovvero in attesa di una query del client) per un periodo di tempo specificato, ma che non è all'interno di una transazione aperta	–
intervalstyle	Imposta il formato del display per i valori dell'intervallo.	–
join_collapse_limit	Imposta la dimensione dell'elenco FROM oltre la quale i costrutti JOIN non vengono appiattiti.	–

Nome del parametro	Descrizione	Default
krb_caseins_users	Indica se i nomi utente GSSAPI (Generic Security Service API) devono essere trattati senza distinzione tra maiuscole e minuscole (true) o meno. Per impostazione predefinita, questo parametro è impostato su false, quindi Kerberos suppone che i nomi utente applichin o la distinzione tra maiuscole e minuscole. Per ulteriori informazioni, consulta la pagina GSSAPI Authentication (Autenticazione GSSAPI) nella documentazione di PostgreSQL.	false
lc_messages	Imposta la lingua nella quale vengono visualizz ati i messaggi.	–
lc_monetary	Imposta l'ambientazione per la formattazione degli importi monetari.	–
lc_numeric	Imposta l'ambientazione per la formattazione degli numeri.	–
lc_time	Imposta l'ambientazione per la formattazione dei valori di data e ora.	–
listen_addresses	Imposta il nome host o l'indirizzo o gli indirizzi IP da ascoltare.	*
lo_compat_privileges	Abilita la modalità di compatibilità con le versioni precedenti per i controlli dei privilegi su oggetti di grandi dimensioni.	0
log_autovacuum_min_duration	(ms) Imposta il tempo minimo di esecuzione al di sopra del quale verranno registrate le azioni di autovacuum.	10000
log_connections	Registra ogni connessione riuscita.	–

Nome del parametro	Descrizione	Default
log_destination	Imposta la destinazione per l’output del registro del server.	stderr
log_directory	Imposta la directory di destinazione per i file di file di log.	/rdsdbdata/log/error
log_disconnections	Registra la fine di una sessione, compresa la durata.	–
log_duration	Registra la durata di ogni istruzione SQL completata.	–
log_error_verbosity	Imposta la verbosità dei messaggi registrati.	–
log_executor_stats	Scrive le statistiche sulla prestazione degli esecutori nel registro del server.	–
log_file_mode	Imposta le autorizzazioni del file per i file di file di log.	0644
log_filename	Imposta il modello del nome del file per i file di log.	postgresql.log.%Y-%m-%d-%H%M
logging_collector	Avvia un processo secondario per acquisire l’output stderr e/o csvlogs nei file di log.	1
log_hostname	Registra il nome host nei registri delle connessioni.	0
logical_decoding_work_mem	(kB) Questa quantità di memoria può essere utilizzata da ogni buffer interno di riordino prima del versamento su disco.	–
log_line_prefix	Controlla le informazioni con prefisso su ciascuna riga di registro.	%t:%r:%u@%d:%p]:
log_lock_waits	Registra lunghe attese di lock.	–

Nome del parametro	Descrizione	Default
log_duration_sample	(ms) Imposta il tempo minimo di esecuzione al di sopra del quale verrà registrato un campione di istruzioni. Il campionamento è determinato da log_statement_sample_rate.	–
log_min_duration_statement	(ms) Imposta il tempo minimo di esecuzione al di sopra del quale verranno registrate le istruzioni.	–
log_min_error_statement	Fa sì che tutte le istruzioni che generano un errore pari o superiore a questo livello vengano registrate.	–
log_min_messages	Imposta i livelli dei messaggi che vengono registrati.	–
log_parameter_max_length	(B) Quando si registrano le istruzioni, limita i valori dei parametri registrati ai primi N byte.	–
log_parameter_max_length_on_error	(B) Quando si segnala un errore, limita i valori dei parametri registrati ai primi N byte.	–
log_parser_stats	Scrive le statistiche sulla prestazione del decodificatore nel registro del server.	–
log_planner_stats	Scrive le statistiche sulla prestazione del programmatore nel registro del server.	–
log_replication_commands	Registra ogni comando di replica.	–
log_rotation_age	(min) La rotazione del file di log automatico avverrà dopo N minuti.	60
log_rotation_size	(kB) La rotazione del file di log automatico avverrà dopo N kilobyte.	100000

Nome del parametro	Descrizione	Default
log_statement	Imposta il tipo di istruzioni registrate.	–
log_statement_sample_rate	Frazione di istruzioni che superano log_min_duration_sample da registrare.	–
log_statement_stats	Scrive le statistiche cumulative sulla prestazione nel registro del server.	–
log_temp_files	(kB) Registra l'uso di file temporanei più grandi di questo numero di kilobyte.	–
log_timezone	Imposta il fuso orario da utilizzare nei messaggi di registro.	UTC
log_transaction_sample_rate	Imposta la frazione di transazioni da registrare per le nuove transazioni.	–
log_truncate_on_rotation	Tronca i file di log esistenti con lo stesso nome durante la rotazione del registro.	0
maintenance_io_concurrency	Una variante di effective_io_concurrency utilizzata per i lavori di manutenzione.	1
maintenance_work_mem	(kB) Imposta la memoria massima da utilizzare per le operazioni di manutenzione.	GREATEST(DBInstanceClassMemory/63963136*1024,65536)
max_connections	Imposta il numero massimo di connessioni simultanee.	LEAST(DBInstanceClassMemory/9531392,5000)
max_files_per_process	Imposta il numero massimo di file aperti in modo simultaneo per ogni processo del server.	–
max_locks_per_transaction	Imposta il numero massimo di lock per transazione.	64

Nome del parametro	Descrizione	Default
max_logical_replication_workers	Numero massimo di processi di dipendente di replica logica.	–
max_parallel_maintenance_workers	Imposta il numero massimo di processi paralleli per operazione di manutenzione.	–
max_parallel_workers	Imposta il numero massimo di worker paralleli che possono essere attivi in una sola volta.	GREATEST($DBInstanceVCPU/2, 8)
max_parallel_workers_per_gather	Imposta il numero massimo di processi paralleli per nodo executor.	–
max_pred_locks_per_page	Imposta il numero massimo di tuple bloccate dal predicato per pagina.	–
max_pred_locks_per_relation	Imposta il numero massimo di pagine e tuple bloccate dal predicato per relazione.	–
max_pred_locks_per_transaction	Imposta il numero massimo di lock del predicato per transazione.	–
max_prepared_transactions	Imposta il numero massimo di transazioni preparati in modo simultaneo.	0
max_replication_slots	Imposta il numero massimo di slot di replica che il server può supportare.	20
max_slot_wal_keep_size	(MB) Gli slot di replica saranno contrassegnati come non riusciti e i segmenti rilasciati per l'eliminazione o il riciclaggio se questo spazio è occupato dai WAL su disco.	–
max_stack_depth	(kB) Imposta la profondità massima della pila in kilobyte.	6144

Nome del parametro	Descrizione	Default
max_standby_streaming_delay	(ms) Imposta il ritardo massimo prima di annullare le query quando un server con standby a caldo elabora i dati WAL in streaming .	14000
max_sync_workers_per_subscription	Numero massimo di worker di sincronizzazione per abbonamento	2
max_wal_senders	Imposta il numero massimo di processi del mittente WAL in esecuzione simultanea	10
max_worker_processes	Imposta il numero massimo di processi dipendente simultanei.	GREATEST($DBInstanceVCPU*2, 8)
min_dynamic_shared_memory	(MB) Quantità di memoria condivisa dinamica riservata all'avvio.	–
min_parallel_index_scan_size	(8 kB) Imposta la quantità minima di dati di indicizzazione per una scansione parallela.	–
min_parallel_table_scan_size	(8 kB) Imposta la quantità minima di dati della tabella per una scansione parallela.	–
old_snapshot_threshold	(min) Tempo prima che uno snapshot sia troppo vecchio per leggere le pagine modificate dopo l'acquisizione dello snapshot.	–
orafce.nls_date_format	Emula il comportamento di output oracles date.	–
orafce.timezone	Specifica il fuso orario utilizzato per la funzione sysdate.	–
parallel_leader_participation	Controlla se Gather e Gather Merge eseguono anche piani secondari.	–

Nome del parametro	Descrizione	Default
parallel_setup_cost	Imposta la stima dei pianificatori del costo di avvio dei processi dei dipendenti per la query parallela.	–
parallel_tuple_cost	Imposta la stima del pianificatore del costo del passaggio di ciascuna tupla (riga) dal back-end dipendente a quello principale.	–
password_encryption	Crittografa le password.	–
pgaudit.log	Specifica quali classi di istruzioni verranno registrate per registrazione di verifica della sessione.	–
pgaudit.log_catalog	Specifica che la registrazione delle sessioni deve essere abilitata nel caso in cui tutte le relazioni in un'istruzione siano in pg_catalog.	–
pgaudit.log_level	Specifica il livello di log che verrà utilizzato per le voci di registro.	–
pgaudit.log_parameter	Specifica che la registrazione di log di audit deve includere i parametri passati con l'istruzi one.	–
pgaudit.log_relation	Specifica se la registrazione di verifica della sessione deve creare una voce di registro separata per ogni relazione (TABLE, VIEW, ecc.) a cui fa riferimento in un'istruzione SELECT o DML.	–
pgaudit.log_statem ent_once	Specifica se la registrazione includerà il testo dell'istruzione e i parametri con la prima voce di registro per una combinazione istruzione/istruzi one secondaria o con ogni voce.	–

Nome del parametro	Descrizione	Default
pgaudit.role	Specifica il ruolo principale da utilizzare per la registrazione di verifica degli oggetti.	–
pg_bigm.enable_recheck	Specifica se eseguire Recheck, che è un processo interno di ricerca full-text.	on
pg_bigm.gin_key_limit	Specifica il numero massimo di 2-grammi della parola chiave di ricerca da utilizzare per la ricerca full-text.	0
pg_bigm.last_update	Riporta l'ultima data aggiornata del modulo pg_bigm.	2013.11.22
pg_bigm.similarity_limit	Specifica la soglia minima utilizzata dalla ricerca di somiglianza.	0.3
pg_hint_plan.debug_print	Registra i risultati dell'analisi dei suggerimenti.	–
pg_hint_plan.enable_hint	Forza il pianificatore a utilizzare i piani specificati nel commento del suggerimento precedente alla query.	–
pg_hint_plan.enable_hint_table	Forza il pianificatore a non ottenere indicazioni utilizzando le ricerche nella tabella.	–
pg_hint_plan.message_level	Livello di messaggio dei messaggi di debug.	–
pg_hint_plan.parse_messages	Livello di messaggio degli errori di analisi.	–
pglogical.batch_inserts	Inserimenti batch, se possibile	–
pglogical.conflict_log_level	Imposta il livello di registro utilizzato per la registrazione dei conflitti risolti.	–

Nome del parametro	Descrizione	Default
pglogical.conflict_resolution	Imposta il metodo utilizzato per la risoluzione dei conflitti per i conflitti risolvibili.	–
pglogical.extra_connection_options	opzioni di connessione da aggiungere a tutte le connessioni dei nodi peer	–
pglogical.synchronous_commit	Valore di commit sincrono specifico pglogical	–
pglogical.use_spi	Usa SPI invece dell'API di basso livello per applicare le modifiche	–
pgtle.clientauth_databases_to_skip	Elenco di database da saltare per la funzionalità clientauth.	–
pgtle.clientauth_db_name	Controlla quale database viene utilizzato per la funzionalità clientauth.	–
pgtle.clientauth_num_parallel_workers	Numero di worker in background utilizzati per la funzionalità clientauth.	–
pgtle.clientauth_users_to_skip	Elenco di utenti da saltare per la funzionalità clientauth.	–
pgtle.enable_clientauth	Abilita la funzionalità clientauth.	–
pgtle.passcheck_db_name	Imposta quale database viene utilizzato per la funzionalità passcheck a livello di cluster.	–
pg_prewarm.autoprewarm	Avvia il dipendente di pre-riscaldamento automatico.	–
pg_prewarm.autoprewarm_interval	Imposta l'intervallo tra i dump dei buffer condivisi	–
pg_similarity.block_is_normalized	Imposta se il valore del risultato è normalizzato o meno.	–

Nome del parametro	Descrizione	Default
pg_similarity.block_threshold	Imposta la soglia utilizzata dalla funzione di similarità dei blocchi.	–
pg_similarity.block_tokenizer	Imposta il tokenizzatore per la funzione di similarità dei blocchi.	–
pg_similarity.cosine_is_normalized	Imposta se il valore del risultato è normalizzato o meno.	–
pg_similarity.cosine_threshold	Imposta la soglia utilizzata dalla funzione di similarità del coseno.	–
pg_similarity.cosine_tokenizer	Imposta il tokenizzatore per la funzione di similarità del coseno.	–
pg_similarity.dice_is_normalized	Imposta se il valore del risultato è normalizzato o meno.	–
pg_similarity.dice_threshold	Imposta la soglia utilizzata dalla misura di similarità di Dice.	–
pg_similarity.dice_tokenizer	Imposta il tokenizzatore per la misura di similarità di Dice.	–
pg_similarity.euclidean_is_normalized	Imposta se il valore del risultato è normalizzato o meno.	–
pg_similarity.euclidean_threshold	Imposta la soglia utilizzata dalla misura di similarità euclidea.	–
pg_similarity.euclidean_tokenizer	Imposta il tokenizzatore per la misura di similarità euclidea.	–
pg_similarity.hamming_is_normalized	Imposta se il valore del risultato è normalizzato o meno.	–
pg_similarity.hamming_threshold	Imposta la soglia utilizzata dal parametro di similarità dei blocchi.	–

Nome del parametro	Descrizione	Default
pg_similarity.jaccard_is_normalized	Imposta se il valore del risultato è normalizzato o meno.	–
pg_similarity.jaccard_threshold	Imposta la soglia utilizzata dalla misura di similarità di Jaccard.	–
pg_similarity.jaccard_tokenizer	Imposta il tokenizzatore per la misura di similarità di Jaccard.	–
pg_similarity.jaro_is_normalized	Imposta se il valore del risultato è normalizzato o meno.	–
pg_similarity.jaro_threshold	Imposta la soglia utilizzata dalla misura di similarità di Jaro.	–
pg_similarity.jarowinkler_is_normalized	Imposta se il valore del risultato è normalizzato o meno.	–
pg_similarity.jarowinkler_threshold	Imposta la soglia utilizzata dalla misura di similarità di Jarowinkler.	–
pg_similarity.levenshtein_is_normalized	Imposta se il valore del risultato è normalizzato o meno.	–
pg_similarity.levenshtein_threshold	Imposta la soglia utilizzata dalla misura di similarità di Levenshtein.	–
pg_similarity.matching_is_normalized	Imposta se il valore del risultato è normalizzato o meno.	–
pg_similarity.matching_threshold	Imposta la soglia utilizzata dalla misura di similarità del coefficiente di corrispondenza.	–
pg_similarity.matching_tokenizer	Imposta il tokenizzatore per la misura di similarità del coefficiente di corrispondenza.	–
pg_similarity.mongeelkan_is_normalized	Imposta se il valore del risultato è normalizzato o meno.	–

Nome del parametro	Descrizione	Default
pg_similarity.mongeelkan_threshold	Imposta la soglia utilizzata dalla misura di similarità di Monge-Elkan.	–
pg_similarity.mongeelkan_tokenizer	Imposta il tokenizzatore per la misura di similarità di Monge-Elkan.	–
pg_similarity.nw_gap_penalty	Imposta la penalità del gap utilizzata dalla misura di similitudine di Needleman-Wunsch.	–
pg_similarity.nw_is_normalized	Imposta se il valore del risultato è normalizzato o meno.	–
pg_similarity.nw_threshold	Imposta la soglia utilizzata dalla misura di similitudine di Needleman-Wunsch.	–
pg_similarity.overlap_is_normalized	Imposta se il valore del risultato è normalizzato o meno.	–
pg_similarity.overlap_threshold	Imposta la soglia utilizzata dalla misura di similarità del coefficiente di sovrapposizione.	–
pg_similarity.overlap_tokenizer	Imposta il tokenizzatore per la misura di similarità del coefficiente di sovrapposizione.	–
pg_similarity.qgram_is_normalized	Imposta se il valore del risultato è normalizzato o meno.	–
pg_similarity.qgram_threshold	Imposta la soglia utilizzata dalla misura di similitudine Q-Gram.	–
pg_similarity.qgram_tokenizer	Imposta il tokenizzatore per la misura Q-Gram.	–
pg_similarity.swg_is_normalized	Imposta se il valore del risultato è normalizzato o meno.	–
pg_similarità.swg_threshold	Imposta la soglia utilizzata dalla misura di similitudine di Smith-Waterman-Gotoh.	–

Nome del parametro	Descrizione	Default
pg_similarity.sw_is_normalized	Imposta se il valore del risultato è normalizzato o meno.	–
pg_similarity.sw_threshold	Imposta la soglia utilizzata dalla misura di similitudine di Smith-Waterman.	–
pg_stat_statements.max	Imposta il numero massimo di istruzioni monitorate da pg_stat_statements.	–
pg_stat_statements.save	Salva le statistiche pg_stat_statements attraverso le interruzioni del server.	–
pg_stat_statements.track	Seleziona le istruzioni che vengono monitorate da pg_stat_statements.	–
pg_stat_statements.track_planning	Seleziona se la durata della pianificazione è monitorata da pg_stat_statements.	–
pg_stat_statements.track_utility	Seleziona se i comandi di utilità sono monitorati da pg_stat_statements.	–
plan_cache_mode	Controlla la selezione da parte del pianificatore di un piano personalizzato o generico.	–
port	Imposta la porta TCP su cui il server è in ascolto.	EndPointPort
postgis.gdal_enabled_drivers	Abilita o disabilita i driver GDAL utilizzati con PostGIS in Postgres 9.3.5 e versioni successive.	ENABLE_ALL
quote_all_identifiers	Aggiungi le virgolette a tutti gli identificatori quando si generano i frammenti SQL.	–
random_page_cost	Imposta la stima del pianificatore del costo di una pagina del disco recuperata in modo non sequenziale.	–

Nome del parametro	Descrizione	Default
rdkit.dice_threshold	Soglia inferiore di somiglianza Dice. Le molecole con somiglianza inferiore alla soglia non sono simili per operazione #.	–
rdkit.do_chiral_sss	È bene prendere in considerazione la stereochimica nella corrispondenza delle sottostrutture. Se false, non vengono utilizzate informazioni stereochimiche nelle corrispondenze della sottostruttura.	–
rdkit.tanimoto_threshold	Soglia inferiore di somiglianza Tanimoto. Le molecole con somiglianza inferiore alla soglia non sono simili per operazione %.	–
rds.accepted_password_auth_method	Forza l'autenticazione per le connessioni con password archiviata localmente.	md5+scram
rds.adaptive_autovacuum	Parametro RDS per abilitare/disabilitare l'autovacuum adattivo.	1
rds.babelfish_status	Parametro RDS per abilitare/disabilitare Babelfish per Aurora PostgreSQL.	off
rds.enable_plan_management	Abilita o disabilita l'estensione apg_plan_mgmt.	0

Nome del parametro	Descrizione	Default
rds.extensions	Elenco di estensioni fornite da RDS	address_standardizer, address_standardizer_data_us, apg_plan_mgmt, aurora_stat_utils, amcheck, autoinc, aws_commons, aws_ml, aws_s3, aws_lambda, bool_plperl, bloom, btree_gin, btree_gist, citext, cube, dblink, dict_int, dict_xsyn, earthdistance, fuzzystrmatch, hll, hstore, hstore_plperl, insert_username, intagg, intarray, ip4r, isn, jsonb_plperl, lo, log_fdw, ltree, moddatetime, old_snapshot, oracle_fdw, orafce, pgaudit, pgcrypto, pglogical, pgrouting, pgrowlocks, pgstattuple, pgtap, pg_bigm, pg_buffercache, pg_cron, pg_freespacemap, pg_hint_plan, pg_partman, pg_prewarm, pg_proctab, pg_repack, pg_simila

Nome del parametro	Descrizione	Default
		rity, pg_stat_s tatements, pg_trgm, pg_visibility, plcoffee, plls, plperl, plpgsql, plprofiler, pltcl, plv8, postgis, postgis_t iger_geocoder, postgis_raster, postgis_topology, postgres_fdw, prefix, rdkit, rds_tools, refint, sslinfo, tablefunc, tds_fdw, test_parser, tsm_system_rows, tsm_system_time, unaccent, uuid-ossp
rds.force_admin_lo gging_level	Visualizza i messaggi di registro per le azioni dell'utente amministratore di RDS nei database dei clienti.	–
rds.force_autovacu um_logging_level	Visualizza i messaggi dei registri relativi alle operazioni di autovacuum.	WARNING
rds.force_ssl	Forza le connessioni SSL.	0

Nome del parametro	Descrizione	Default
rds.global_db_rpo	(s) Soglia dell'obiettivo del punto di ripristino, in secondi, che blocca i commit dell'utente quando viene violata. **Important** Questo parametro è destinato ai database globali Aurora basati su PostgreSQL. Per un database non globale, lasciare il valore predefinito. Per ulteriori informazioni su questo parametro, consulta the section called “Gestione degli RPO (Aurora PostgreSQL)”.	–
rds.logical_replication	Abilita la decodifica logica.	0
rds.logically_replicate_unlogged_tables	Le tabelle non registrate vengono replicate logicamente.	1
rds.log_retention_period	Amazon RDS elimina i registri PostgreSQL più vecchi di N minuti.	4320
rds.pg_stat_ramdisk_size	Dimensioni del ramdisk delle statistiche in MB. Un valore diverso da zero configurerà il ramdisk. Questo parametro è disponibile solo in Aurora PostgreSQL 14 e versioni precedenti.	0
rds.rds_superuser_reserved_connections	Imposta il numero di slot di connessione riservati a rds_superusers. Questo parametro è disponibile solo nella versione 15 e precedenti. Per ulteriori informazioni, consulta le connessioni riservate nella documentazione di PostgreSQL.	2

Nome del parametro	Descrizione	Default
rds.restrict_password_commands	limita i comandi relativi alla password ai membri di rds_password	–
rds.superuser_variables	Elenco delle variabili del solo utente con privilegi avanzati per le quali eleviamo le istruzioni di modifica rds_superuser.	session_replication_role
recovery_init_sync_method	Imposta il metodo per sincronizzare la directory dei dati prima del ripristino da arresto anomalo.	syncfs
remove_temp_files_after_crash	Rimuove i file temporanei dopo l'arresto anomalo del back-end	0
restart_after_crash	Reinizializza il server dopo l’arresto anomalo del back-end.	–
row_security	Abilita la sicurezza delle righe.	–
search_path	Imposta l’ordine di ricerca dello schema per i nomi che non sono qualificati come schema.	–
seq_page_cost	Imposta la stima del pianificatore del costo di una pagina del disco recuperata in modo sequenziale.	–
session_replication_role	Imposta il comportamento delle sessioni per i trigger e le regole di riscrittura.	–
shared_buffers	(8 kB) Imposta il numero di buffer di memoria condivisa utilizzati dal server.	SUM(DBInstanceClassMemory/12038,-50003)
shared_preload_libraries	Elenca le librerie condivise da precaricare nel server.	pg_stat_statements
ssl	Abilita le connessioni SSL.	1

Nome del parametro	Descrizione	Default
ssl_ca_file	Posizione del file dell'autorità del server SSL.	/rdsdbdata/rds-metadata/ca-cert.pem
ssl_cert_file	Posizione del file del certificato del server SSL.	/rdsdbdata/rds-metadata/server-cert.pem
ssl_ciphers	Imposta l'elenco delle crittografie TLS consentite da utilizzare su connessioni sicure.	–
ssl_crl_dir	Posizione della directory dell'elenco di revoche di certificati SSL.	/rdsdbdata/rds-metadata/ssl_crl_dir/
ssl_key_file	Posizione del file della chiave privata del server SSL	/rdsdbdata/rds-metadata/server-key.pem
ssl_max_protocol_version	Imposta la versione del protocollo SSL/TLS massima consentita	–
ssl_min_protocol_version	Imposta la versione del protocollo SSL/TLS minima consentita	TLSv1.2
standard_conforming_strings	Fa sì che le stringhe ... trattino letteralmente le barre rovesciate.	–
statement_timeout	(ms) Imposta la durata massima concessa di ogni istruzione.	–
stats_temp_directory	Scrive i file di statistiche temporanei nella directory specificata.	/rdsdbdata/db/pg_stat_tmp
superuser_reserved_connections	Imposta il numero di slot di connessione riservati agli utenti con privilegi avanzati.	3
synchronize_seqscans	Abilita le scansioni sequenziali sincronizzate.	–
synchronous_commit	Imposta il livello di sincronizzazione delle transazioni correnti.	on

Nome del parametro	Descrizione	Default
tcp_keepalives_count	Numero massimo di ritrasmissioni keepalive TCP.	–
tcp_keepalives_idle	(s) Tempo tra l'emissione di keepalive TCP.	–
tcp_keepalives_interval	(s) Tempo tra la ritrasmissione di keepalive TCP.	–
temp_buffers	(8 kB) Imposta il numero massimo di buffer temporanei utilizzati da ogni sessione.	–
temp_file_limit	Vincola la quantità totale di spazio su disco in kilobyte che un determinato processo PostgreSQL può utilizzare per i file temporanei, escludendo lo spazio utilizzato per tabelle temporanee esplicite	-1
temp_tablespaces	Imposta i tablespace da usare per le tabelle temporanee e i file di ordinamento.	–
timezone	Imposta il fuso orario per la visualizzazione e l'interpretazione dei timestamp.	UTC
track_activities	Raccoglie informazioni sui comandi di esecuzione.	–
track_activity_query_size	Imposta la dimensione riservata per pg_stat_activity.current_query, in byte.	4096
track_commit_timestamp	Raccoglie l'ora di commit della transazione.	–
track_counts	Raccoglie statistiche sull’attività del database.	–
track_functions	Raccoglie statistiche a livello di funzione sull’attività del database.	pl

Nome del parametro	Descrizione	Default
track_io_timing	Raccoglie statistiche di temporizzazione sull'attività di I/O del database.	1
track_wal_io_timing	Raccoglie statistiche di temporizzazione per l'attività di I/O WAL.	–
transform_null_equals	Tratta expr=NULL come expr È NULL.	–
update_process_title	Aggiorna il titolo del processo per mostrare il comando SQL attivo.	–
vacuum_cost_delay	(ms) Ritardo del costo del vacuum, in millisecondi.	–
vacuum_cost_limit	Quantità del costo del vacuum disponibile prima del napping.	–
vacuum_cost_page_dirty	Costo del vacuum per una pagina sporcata dal vacuum.	–
vacuum_cost_page_hit	Costo del vacuum per una pagina trovata nella cache del buffer.	–
vacuum_cost_page_miss	Costo del vacuum per una non pagina trovata nella cache del buffer.	0
vacuum_defer_cleanup_age	Numero di transazioni in base alle quali il VACUUM e la pulizia A CALDO devono essere posticipati, se presenti.	–
vacuum_failsafe_age	Età in cui il VACUUM deve attivare una misura di sicurezza per evitare un'interruzione del wraparound.	1200000000
vacuum_freeze_min_age	Età minima in cui il VACUUM dovrebbe congelare una riga della tabella.	–

Nome del parametro	Descrizione	Default
vacuum_freeze_table_age	Età in cui il VACUUM dovrebbe eseguire la scansione di un'intera tabella per congelare le tuple.	–
vacuum_multixact_failsafe_age	Età multixact in cui il VACUUM deve attivare una misura di sicurezza per evitare un'interruzione del wraparound.	1200000000
vacuum_multixact_freeze_min_age	Età minima in cui il VACUUM dovrebbe congelare un MultiXactId in una riga della tabella.	–
vacuum_multixact_freeze_table_age	Età del Multixact in cui il VACUUM dovrebbe eseguire la scansione di un'intera tabella per congelare le tuple.	–
wal_buffer	(8 kB) Imposta il numero di buffer della pagina del disco nella memoria condivisa per WAL.	–
wal_receiver_create_temp_slot	Imposta se un ricevitore WAL deve creare uno slot di replica temporaneo se non è configurato uno slot permanente.	0
wal_receiver_status_interval	(s) Imposta l'intervallo massimo tra i report di stato del ricevitore WAL al nodo primario.	–
wal_receiver_timeout	(ms) Imposta il tempo di attesa massimo per la ricezione dei dati dal nodo primario.	30000
wal_sender_timeout	(ms) Imposta il tempo massimo di attesa della replica WAL.	–
work_mem	(kB) Imposta la memoria massima da utilizzare per gli spazi di lavoro delle query.	–
xmlbinary	Imposta come i valori binari devono essere codificati in XML.	–

Nome del parametro	Descrizione	Default
xmloption	Imposta se i dati XML nelle operazioni di analisi e serializzazione implicite devono essere considerati come documenti o frammenti di contenuto.	–

Parametri a livello di istanza Aurora PostgreSQL

È possibile visualizzare i parametri a livello di istanza disponibili per una versione specifica di Aurora PostgreSQL con la Console di gestione AWS, l'Interfaccia della linea di comando AWS (AWS CLI) o l'API Amazon RDS. Per informazioni sulla visualizzazione dei parametri in un gruppo di parametri del database Aurora PostgreSQL nella console RDS, consultare Visualizzazione dei valori dei parametri per un gruppo di parametri del database in Amazon Aurora.

Alcuni parametri a livello di istanza non sono disponibili in tutte le versioni e alcuni sono obsoleti. Per informazioni sulla visualizzazione dei parametri di una versione specifica di Aurora PostgreSQL, consultare Visualizzazione del cluster di database Aurora PostgreSQL e dei parametri del database.

Ad esempio, nella tabella seguente sono elencati i parametri applicabili a una specifica istanza database di un cluster database Aurora PostgreSQL. Questo elenco è stato generato eseguendo il comando describe-db-parameters della AWS CLI con `default.aurora-postgresql14`per il valore `--db-parameter-group-name`.

Per un elenco dei parametri del cluster database per lo stesso gruppo di parametri DB, consulta Parametri a livello di cluster Aurora PostgreSQL.

Nome del parametro	Descrizione	Default
apg_enable_batch_mode_function_execution	Consente alle funzioni in modalità batch di elaborare serie di righe in una volta.	–
apg_enable_correlated_any_transform	Consente al pianificatore di trasformare il link secondario ANY (sottoquery IN/NOT IN) in JOIN quando possibile.	–
apg_enable_function_migration	Consente al pianificatore di migrare le funzioni scalari idonee alla clausola FROM.	–
apg_enable_not_in_transform	Consente al pianificatore di trasformare la query secondaria NOT IN in ANTI JOIN quando possibile.	–
apg_enable_remove_redundant_inner_joins	Consente al pianificatore di rimuovere i join interni ridondanti.	–

Nome del parametro	Descrizione	Default
apg_enable_semijoin_push_down	Consente l’utilizzo di filtri semijoin per gli hash join.	–
apg_plan_mgmt.capture_plan_baselines	Modalità baseline di acquisizione del piano. manual (manuale): abilita l'acquisizione del piano per qualsiasi istruzione SQL, off: disabilita l'acquisizione del piano, automatic (automatica): abilita l'acquisizione del piano per le istruzioni in pg_stat_statements che soddisfano i criteri di idoneità.	off
apg_plan_mgmt.max_databases	Imposta il numero massimo di database che possono gestire le query utilizzando apg_plan_mgmt.	10
apg_plan_mgmt.max_plans	Imposta il numero massimo di piani che possono essere memorizzati nella cache da apg_plan_mgmt.	10000
apg_plan_mgmt.plan_retention_period	Numero massimo di giorni dall’ultimo utilizzo di un piano prima che venga eliminato automaticamente.	32
apg_plan_mgmt.unapproved_plan_execution_threshold	Costo totale stimato del piano al di sotto del quale verrà eseguito un piano non approvato.	0
apg_plan_mgmt.use_plan_baselines	Utilizzare solo piani approvati o fissi per le istruzioni gestite.	false
application_name	Imposta il nome dell’applicazione da riportare nelle statistiche e nei registri.	–

Nome del parametro	Descrizione	Default
aurora_compute_plan_id	Monitora i piani di esecuzione delle query per rilevare i piani di esecuzione che contribuiscono al carico corrente del database e per tenere traccia delle statistiche sulle prestazioni dei piani di esecuzione nel tempo. Per ulteriori informazioni, consulta Monitoring query execution plans for Aurora PostgreSQL.	on
aurora_temp_space_size	(MB) Imposta la dimensione dello spazio allocato per gli oggetti temporanei abilitati alle letture ottimizzate su cluster Aurora ottimizzati per I/O con classi di istanze supportate.	DBInstanceClassMemory/524288
authentication_timeout	(s) Imposta il tempo massimo concesso per completare l'autenticazione del client.	–
auto_explain.log_analyze	Usa EXPLAIN ANALYZE per la registrazione del piano.	–
auto_explain.log_buffers	Registra l'utilizzo dei buffer.	–
auto_explain.log_format	Formato EXPLAIN da utilizzare per la registrazione del piano.	–
auto_explain.log_min_duration	Imposta il tempo minimo di esecuzione al di sopra del quale verranno registrati i piani.	–
auto_explain.log_nested_statements	Registra le istruzioni nidificate.	–
auto_explain.log_timing	Raccogli i dati temporali, non solo il numero di righe.	–
auto_explain.log_triggers	Includi statistiche di attivazione nei piani.	–

Nome del parametro	Descrizione	Default
auto_explain.log_verbose	Usa EXPLAIN VERBOSE per la registrazione del piano.	–
auto_explain.sample_rate	Frazione di query da elaborare.	–
babelfishpg_tds.listen_address	Imposta il nome host o l'indirizzo/gli indirizzi IP su cui ascoltare TDS.	*
babelfishpg_tds.tds_debug_log_level	Imposta il livello di registrazione in TDS, 0 disabilita la registrazione	1
backend_flush_after	(8Kb Numero di pagine dopo le quali le scritture eseguite in precedenza vengono riportate su disco.	–
bytea_output	Imposta il formato di output per byte.	–
check_function_bodies	Controlla i corpi delle funzioni durante CREATE FUNCTION.	–
client_connection_check_interval	Imposta l'intervallo di tempo tra i controlli di disconnessione durante l'esecuzione di query.	–
client_min_messages	Imposta i livelli dei messaggi che vengono inviati al client.	–
config_file	Imposta il file di configurazione principale del server.	/rdsdbdata/config/postgresql.conf
constraint_exclusion	Consente al pianificatore di utilizzare i vincoli per ottimizzare le query.	–
cpu_index_tuple_cost	Imposta la stima del pianificatore del costo di elaborazione di ciascuna voce di indice durante una scansione dell’indice.	–

Nome del parametro	Descrizione	Default
cpu_operator_cost	Imposta la stima del pianificatore del costo di elaborazione di ciascuna chiamata dell'operatore o della funzione.	–
cpu_tuple_cost	Imposta la stima del pianificatore del costo di elaborazione di ciascuna tupla (riga).	–
cron.database_name	Imposta il database per archiviare le tabelle di metadati pg_cron	postgres
cron.log_run	Registra tutti i processi eseguiti nella tabella job_run_details	on
cron.log_statement	Registra tutte le istruzioni cron prima dell'esecuzione.	off
cron.max_running_jobs	Numero massimo di processi che possono essere eseguiti contemporaneamente.	5
cron.use_background_workers	Abilita i dipendente in background per pg_cron	on
cursor_tuple_fraction	Imposta la stima del pianificatore della frazione delle righe del cursore che verranno recuperate.	–
db_user_namespace	Abilita i nomi utente per database.	–
deadlock_timeout	(ms) Imposta il tempo di attesa su un lock prima di verificare il deadlock.	–
debug_pretty_print	I trattini analizzano e visualizzano le visualizzazioni dell'albero.	–
debug_print_parse	Registra ogni albero di analisi della query.	–
debug_print_plan	Registra ogni programma di esecuzione della query.	–

Nome del parametro	Descrizione	Default
debug_print_rewritten	Registra ogni albero di analisi riscritto della query.	–
default_statistics_target	Imposta la destinazione della statistica predefinita.	–
default_transaction_deferrable	Imposta lo stato differibile predefinito delle nuove transazioni.	–
default_transaction_isolation	Imposta il livello di isolamento della transazione di ogni nuova transazione.	–
default_transaction_read_only	Imposta lo stato di sola lettura predefinito delle nuove transazioni.	–
effective_cache_size	(8 kB) Imposta l'ipotesi dei pianificatori sulla dimensione della cache del disco.	SUM(DBInstanceClassMemory/12038,-50003
effective_io_concurrency	Numero di richieste simultanee che possono essere gestite in modo efficace dal sottosistema del disco.	–
enable_async_append	Consente ai pianificatori di utilizzare piani di aggiunta asincroni.	–
enable_bitmapscan	Abilita l’utilizzo da parte del pianificatore di piani di scansione bitmap.	–
enable_gathermerge	Abilita l’utilizzo da parte del pianificatore di piani di unione di raccolta.	–
enable_hashagg	Abilita l’utilizzo da parte del pianificatore di piani di aggregazione hash.	–
enable_hashjoin	Abilita l’utilizzo da parte del pianificatore di piani di unione hash.	–

Nome del parametro	Descrizione	Default
enable_incremental_sort	Consente ai pianificatori di utilizzare fasi incrementali di ordinamento.	–
enable_indexonlyscan	Abilita l’utilizzo da parte del pianificatore di piani di sola scansione dell’indice.	–
enable_indexscan	Abilita l’utilizzo da parte del pianificatore di piani di scansione dell’indice.	–
enable_material	Abilita l’utilizzo da parte del pianificatore della materializzazione.	–
enable_memoize	Abilita l'uso della memorizzazione da parte dei pianificatori	–
enable_mergejoin	Abilita l’utilizzo da parte del pianificatore di piani di unione.	–
enable_nestloop	Abilita l’utilizzo da parte del pianificatore di piani di unione a ciclo nested.	–
enable_parallel_append	Consente ai pianificatori di utilizzare piani di aggiunta paralleli.	–
enable_parallel_hash	Consente ai pianificatori di utilizzare piani di hash paralleli.	–
enable_partition_pruning	Abilita l’eliminazione delle partizioni del tempo di pianificazione e del runtime.	–
enable_partitionwise_aggregate	Consente l’aggregazione e il raggruppamento a livello di partizione.	–
enable_partitionwise_join	Attiva il join a livello di partizione.	–
enable_seqscan	Abilita l’utilizzo da parte del pianificatore di piani di scansione sequenziali.	–

Nome del parametro	Descrizione	Default
enable_sort	Abilita l'utilizzo da parte del pianificatore di passaggi di ordinamento espliciti.	–
enable_tidscan	Abilita l'utilizzo da parte del pianificatore di piani di scansione TID.	–
escape_string_warning	Avvisa circa la perdita di barre rovesciate nelle stringhe letterali ordinarie.	–
exit_on_error	Termina la sessione in caso di errore.	–
force_parallel_mode	Forza l'uso di strutture di query parallele.	–
from_collapse_limit	Imposta la dimensione dell'elenco FROM oltre la quale le sottoquery non vengono compresse.	–
geqo	Abilita l'ottimizzazione genetica delle query.	–
geqo_effort	GEQO: lo sforzo viene utilizzato per impostare il valore predefinito per altri parametri GEQO.	–
geqo_generations	GEQO: numero di iterazioni dell'algoritmo.	–
geqo_pool_size	GEQO: numero di individui nella popolazione.	–
geqo_seed	GEQO: seme per la selezione casuale del percorso.	–
geqo_selection_bias	GEQO: pressione selettiva all'interno della popolazione.	–
geqo_threshold	Imposta la soglia degli elementi FROM oltre i quali viene utilizzato GEQO.	–
gin_fuzzy_search_limit	Imposta il risultato massimo consentito per la ricerca esatta da GIN.	–

Nome del parametro	Descrizione	Default
gin_pending_list_limit	(kB) Imposta la dimensione massima dell'elenco in sospeso per l'indice GIN.	–
hash_mem_multiplier	Multipli di work_mem da utilizzare per le tabelle hash.	–
hba_file	Imposta il file di configurazione hba dei server.	/rdsdbdata/config/pg_hba.conf
hot_standby_feedback	Consente il feedback da uno standby a caldo all’elemento primario che eviterà conflitti di query.	on
ident_file	Imposta il file di configurazione ident dei server.	/rdsdbdata/config/pg_ident.conf
idle_in_transaction_session_timeout	(ms) Imposta la durata massima concessa di ogni transazione inattiva.	86400000
idle_session_timeout	Termina qualsiasi sessione che è rimasta inattiva (ovvero, in attesa di una richiesta da parte del client), ma che non rientra in una transazione aperta, per un periodo di tempo specificato	–
join_collapse_limit	Imposta la dimensione dell’elenco FROM oltre la quale i costrutti JOIN non vengono appiattiti.	–
lc_messages	Imposta la lingua nella quale vengono visualizzati i messaggi.	–
listen_addresses	Imposta il nome host o l'indirizzo/gli indirizzi IP da ascoltare.	*
lo_compat_privileges	Abilita la modalità di compatibilità con le versioni precedenti per i controlli dei privilegi su oggetti di grandi dimensioni.	0

Nome del parametro	Descrizione	Default
log_connections	Registra ogni connessione riuscita.	–
log_destination	Imposta la destinazione per l'output del registro del server.	stderr
log_directory	Imposta la directory di destinazione per i file di file di log.	/rdsdbdata/log/error
log_disconnections	Registra la fine di una sessione, compresa la durata.	–
log_duration	Registra la durata di ogni istruzione SQL completata.	–
log_error_verbosity	Imposta la verbosità dei messaggi registrati.	–
log_executor_stats	Scrive le statistiche sulla prestazione degli esecutori nel registro del server.	–
log_file_mode	Imposta le autorizzazioni del file per i file di file di log.	0644
log_filename	Imposta il modello del nome del file per i file di log.	postgresql.log.%Y-%m-%d-%H%M
logging_collector	Avvia un processo secondario per acquisire l'output stderr e/o csvlogs nei file di log.	1
log_hostname	Registra il nome host nei registri delle connessioni.	0
logical_decoding_work_mem	(kB) Questa quantità di memoria può essere utilizzata da ogni buffer interno di riordino prima del versamento su disco.	–
log_line_prefix	Controlla le informazioni con prefisso su ciascuna riga di registro.	%t:%r:%u@%d:%p]:

Nome del parametro	Descrizione	Default
log_lock_waits	Registra lunghe attese di lock.	–
log_duration_sample	(ms) Imposta il tempo minimo di esecuzione al di sopra del quale verrà registrato un campione di istruzioni. Il campionamento è determinato da log_statement_sample_rate.	–
log_min_duration_statement	(ms) Imposta il tempo minimo di esecuzione al di sopra del quale verranno registrate le istruzioni.	–
log_min_error_statement	Fa sì che tutte le istruzioni che generano un errore pari o superiore a questo livello vengano registrate.	–
log_min_messages	Imposta i livelli dei messaggi che vengono registrati.	–
log_parameter_max_length	(B) Quando si registrano le istruzioni, limita i valori dei parametri registrati ai primi N byte.	–
log_parameter_max_length_on_error	(B) Quando si segnala un errore, limita i valori dei parametri registrati ai primi N byte.	–
log_parser_stats	Scrive le statistiche sulla prestazione del decodificatore nel registro del server.	–
log_planner_stats	Scrive le statistiche sulla prestazione del programmatore nel registro del server.	–
log_replication_commands	Registra ogni comando di replica.	–
log_rotation_age	(min) La rotazione del file di log automatico avverrà dopo N minuti.	60

Nome del parametro	Descrizione	Default
log_rotation_size	(kB) La rotazione del file di log automatico avverrà dopo N kilobyte.	100000
log_statement	Imposta il tipo di istruzioni registrate.	–
log_statement_sample_rate	Frazione di istruzioni che superano log_min_duration_sample da registrare.	–
log_statement_stats	Scrive le statistiche cumulative sulla prestazione nel registro del server.	–
log_temp_files	(kB) Registra l'uso di file temporanei più grandi di questo numero di kilobyte.	–
log_timezone	Imposta il fuso orario da utilizzare nei messaggi di registro.	UTC
log_truncate_on_rotation	Tronca i file di log esistenti con lo stesso nome durante la rotazione del registro.	0
maintenance_io_concurrency	Una variante di effective_io_concurrency utilizzata per i lavori di manutenzione.	1
maintenance_work_mem	(kB) Imposta la memoria massima da utilizzare per le operazioni di manutenzione.	GREATEST(DBInstanceClassMemory/63963136*1024, 65536
max_connections	Imposta il numero massimo di connessioni simultanee.	LEAST(DBInstanceClassMemory/9531392, 5000
max_files_per_process	Imposta il numero massimo di file aperti in modo simultaneo per ogni processo del server.	–
max_locks_per_transaction	Imposta il numero massimo di lock per transazione.	64

Nome del parametro	Descrizione	Default
max_parallel_maintenance_workers	Imposta il numero massimo di processi paralleli per operazione di manutenzione.	–
max_parallel_workers	Imposta il numero massimo di worker paralleli che possono essere attivi in una sola volta.	GREATEST($DBInstanceVCPU/2, 8
max_parallel_workers_per_gather	Imposta il numero massimo di processi paralleli per nodo executor.	–
max_pred_locks_per_page	Imposta il numero massimo di tuple bloccate dal predicato per pagina.	–
max_pred_locks_per_relation	Imposta il numero massimo di pagine e tuple bloccate dal predicato per relazione.	–
max_pred_locks_per_transaction	Imposta il numero massimo di lock del predicato per transazione.	–
max_slot_wal_keep_size	(MB) Gli slot di replica saranno contrassegnati come non riusciti e i segmenti rilasciati per l'eliminazione o il riciclaggio, se questo spazio è occupato dai WAL su disco.	–
max_stack_depth	(kB) Imposta la profondità massima della pila in kilobyte.	6144
max_standby_streaming_delay	(ms) Imposta il ritardo massimo prima di annullare le query quando un server con standby a caldo elabora i dati WAL in streaming .	14000
max_worker_processes	Imposta il numero massimo di processi dipendente simultanei.	GREATEST($DBInstanceVCPU*2, 8

Nome del parametro	Descrizione	Default
min_dynamic_shared_memory	(MB) Quantità di memoria condivisa dinamica riservata all'avvio.	–
min_parallel_index_scan_size	(8 kB) Imposta la quantità minima di dati di indicizzazione per una scansione parallela.	–
min_parallel_table_scan_size	(8 kB) Imposta la quantità minima di dati della tabella per una scansione parallela.	–
old_snapshot_threshold	(min) Tempo prima che uno snapshot sia troppo vecchio per leggere le pagine modificate dopo l'acquisizione dello snapshot.	–
parallel_leader_participation	Controlla se Gather e Gather Merge eseguono anche piani secondari.	–
parallel_setup_cost	Imposta la stima dei pianificatori del costo di avvio dei processi dei dipendenti per la query parallela.	–
parallel_tuple_cost	Imposta la stima del pianificatore del costo del passaggio di ciascuna tupla (riga) dal back-end dipendente a quello principale.	–
pgaudit.log	Specifica quali classi di istruzioni verranno registrate per registrazione di verifica della sessione.	–
pgaudit.log_catalog	Specifica che la registrazione delle sessioni deve essere abilitata nel caso in cui tutte le relazioni in un’istruzione siano in pg_catalog.	–
pgaudit.log_level	Specifica il livello di log che verrà utilizzato per le voci di registro.	–

Nome del parametro	Descrizione	Default
pgaudit.log_parameter	Specifica che la registrazione di log di audit deve includere i parametri passati con l'istruzione.	–
pgaudit.log_relation	Specifica se la registrazione di verifica della sessione deve creare una voce di registro separata per ogni relazione (TABLE, VIEW, ecc.) a cui fa riferimento in un'istruzione SELECT o DML.	–
pgaudit.log_statement_once	Specifica se la registrazione includerà il testo dell'istruzione e i parametri con la prima voce di registro per una combinazione istruzione/istruzione secondaria o con ogni voce.	–
pgaudit.role	Specifica il ruolo principale da utilizzare per la registrazione di verifica degli oggetti.	–
pg_bigm.enable_recheck	Specifica se eseguire Recheck, che è un processo interno di ricerca full-text.	on
pg_bigm.gin_key_limit	Specifica il numero massimo di 2-grammi della parola chiave di ricerca da utilizzare per la ricerca full-text.	0
pg_bigm.last_update	Riporta l'ultima data aggiornata del modulo pg_bigm.	2013.11.22
pg_bigm.similarity_limit	Specifica la soglia minima utilizzata dalla ricerca di somiglianza.	0.3
pg_hint_plan.debug_print	Registra i risultati dell'analisi dei suggerimenti.	–
pg_hint_plan.enable_hint	Forza il pianificatore a utilizzare i piani specificati nel commento del suggerimento precedente alla query.	–

Nome del parametro	Descrizione	Default
pg_hint_plan.enable_hint_table	Forza il pianificatore a non ottenere indicazioni utilizzando le ricerche nella tabella.	–
pg_hint_plan.message_level	Livello di messaggio dei messaggi di debug.	–
pg_hint_plan.parse_messages	Livello di messaggio degli errori di analisi.	–
pglogical.batch_inserts	Inserimenti batch, se possibile	–
pglogical.conflict_log_level	Imposta il livello di registro utilizzato per la registrazione dei conflitti risolti.	–
pglogical.conflict_resolution	Imposta il metodo utilizzato per la risoluzione dei conflitti per i conflitti risolvibili.	–
pglogical.extra_connection_options	opzioni di connessione da aggiungere a tutte le connessioni dei nodi peer	–
pglogical.synchronous_commit	Valore di commit sincrono specifico pglogical	–
pglogical.use_spi	Usa SPI invece dell'API di basso livello per applicare le modifiche	–
pg_similarity.block_is_normalized	Imposta se il valore del risultato è normalizzato o meno.	–
pg_similarity.block_threshold	Imposta la soglia utilizzata dalla funzione di similarità dei blocchi.	–
pg_similarity.block_tokenizer	Imposta il tokenizzatore per la funzione di similarità dei blocchi.	–
pg_similarity.cosine_is_normalized	Imposta se il valore del risultato è normalizzato o meno.	–

Nome del parametro	Descrizione	Default
pg_similarity.cosine_threshold	Imposta la soglia utilizzata dalla funzione di similarità del coseno.	–
pg_similarity.cosine_tokenizer	Imposta il tokenizzatore per la funzione di similarità del coseno.	–
pg_similarity.dice_is_normalized	Imposta se il valore del risultato è normalizzato o meno.	–
pg_similarity.dice_threshold	Imposta la soglia utilizzata dalla misura di similarità di Dice.	–
pg_similarity.dice_tokenizer	Imposta il tokenizzatore per la misura di similarità di Dice.	–
pg_similarity.euclidean_is_normalized	Imposta se il valore del risultato è normalizzato o meno.	–
pg_similarity.euclidean_threshold	Imposta la soglia utilizzata dalla misura di similarità euclidea.	–
pg_similarity.euclidean_tokenizer	Imposta il tokenizzatore per la misura di similarità euclidea.	–
pg_similarity.hamming_is_normalized	Imposta se il valore del risultato è normalizzato o meno.	–
pg_similarity.hamming_threshold	Imposta la soglia utilizzata dal parametro di similarità dei blocchi.	–
pg_similarity.jaccard_is_normalized	Imposta se il valore del risultato è normalizzato o meno.	–
pg_similarity.jaccard_threshold	Imposta la soglia utilizzata dalla misura di similarità di Jaccard.	–
pg_similarity.jaccard_tokenizer	Imposta il tokenizzatore per la misura di similarità di Jaccard.	–

Nome del parametro	Descrizione	Default
pg_similarity.jaro_is_normalized	Imposta se il valore del risultato è normalizzato o meno.	–
pg_similarity.jaro_threshold	Imposta la soglia utilizzata dalla misura di similarità di Jaro.	–
pg_similarity.jaro winkler_is_normalized	Imposta se il valore del risultato è normalizzato o meno.	–
pg_similarity.jaro winkler_threshold	Imposta la soglia utilizzata dalla misura di similarità di Jarowinkler.	–
pg_similarity.leve nshtein_is_normalized	Imposta se il valore del risultato è normalizzato o meno.	–
pg_similarity.leve nshtein_threshold	Imposta la soglia utilizzata dalla misura di similarità di Levenshtein.	–
pg_similarity.matc hing_is_normalized	Imposta se il valore del risultato è normalizzato o meno.	–
pg_similarity.matc hing_threshold	Imposta la soglia utilizzata dalla misura di similarità del coefficiente di corrispondenza.	–
pg_similarity.matc hing_tokenizer	Imposta il tokenizzatore per la misura di similarità del coefficiente di corrispondenza.	–
pg_similarity.mong eelkan_is_normalized	Imposta se il valore del risultato è normalizzato o meno.	–
pg_similarity.mong eelkan_threshold	Imposta la soglia utilizzata dalla misura di similarità di Monge-Elkan.	–
pg_similarity.mong eelkan_tokenizer	Imposta il tokenizzatore per la misura di similarità di Monge-Elkan.	–
pg_similarity.nw_g ap_penalty	Imposta la penalità del gap utilizzata dalla misura di similitudine di Needleman-Wunsch.	–

Nome del parametro	Descrizione	Default
pg_similarity.nw_is_normalized	Imposta se il valore del risultato è normalizzato o meno.	–
pg_similarity.nw_threshold	Imposta la soglia utilizzata dalla misura di similitudine di Needleman-Wunsch.	–
pg_similarity.overlap_is_normalized	Imposta se il valore del risultato è normalizzato o meno.	–
pg_similarity.overlap_threshold	Imposta la soglia utilizzata dalla misura di similarità del coefficiente di sovrapposizione.	–
pg_similarity.overlap_tokenizer	Imposta il tokenizzatore per la misura di similarità del coefficiente di sovrapposizione.	–
pg_similarity.qgram_is_normalized	Imposta se il valore del risultato è normalizzato o meno.	–
pg_similarity.qgram_threshold	Imposta la soglia utilizzata dalla misura di similitudine Q-Gram.	–
pg_similarity.qgram_tokenizer	Imposta il tokenizzatore per la misura Q-Gram.	–
pg_similarity.swg_is_normalized	Imposta se il valore del risultato è normalizzato o meno.	–
pg_similarità.swg_threshold	Imposta la soglia utilizzata dalla misura di similitudine di Smith-Waterman-Gotoh.	–
pg_similarity.sw_is_normalized	Imposta se il valore del risultato è normalizzato o meno.	–
pg_similarity.sw_threshold	Imposta la soglia utilizzata dalla misura di similitudine di Smith-Waterman.	–
pg_stat_statements.max	Imposta il numero massimo di istruzioni monitorate da pg_stat_statements.	–

Nome del parametro	Descrizione	Default
pg_stat_statements.save	Salva le statistiche pg_stat_statements attraverso le interruzioni del server.	–
pg_stat_statements.track	Seleziona le istruzioni che vengono monitorate da pg_stat_statements.	–
pg_stat_statements.track_planning	Seleziona se la durata della pianificazione è monitorata da pg_stat_statements.	–
pg_stat_statements.track_utility	Seleziona se i comandi di utilità sono monitorati da pg_stat_statements.	–
postgis.gdal_enabled_drivers	Abilita o disabilita i driver GDAL utilizzati con PostGIS in Postgres 9.3.5 e versioni successive.	ENABLE_ALL
quote_all_identifiers	Aggiungi le virgolette a tutti gli identificatori quando si generano i frammenti SQL.	–
random_page_cost	Imposta la stima del pianificatore del costo di una pagina del disco recuperata in modo non sequenziale.	–
rds.enable_memory_management	Migliora le funzionalità di gestione della memoria in Aurora PostgreSQL versioni 12.17, 13.13, 14.10, 15.5 e versioni successive, prevenendo i problemi di stabilità e i riavvii del database causati da una memoria libera insufficiente. Per ulteriori informazioni, consulta Migliore gestione della memoria in Aurora PostgreSQL.	True
rds.force_admin_logging_level	Visualizza i messaggi di registro per le azioni dell'utente amministratore di RDS nei database dei clienti.	–

Nome del parametro	Descrizione	Default
rds.log_retention_period	Amazon RDS elimina i registri PostgreSQL più vecchi di N minuti.	4320
rds.memory_allocation_guard	Migliora le funzionalità di gestione della memoria in Aurora PostgreSQL 11.21, 12.16, 13.12, 14.9, 15.4 e versioni precedenti, prevenendo i problemi di stabilità e i riavvii del database causati da una memoria libera insufficiente. Per ulteriori informazioni, consulta Migliore gestione della memoria in Aurora PostgreSQL.	False
rds.pg_stat_ramdisk_size	Dimensioni del ramdisk delle statistiche in MB. Un valore diverso da zero configurerà il ramdisk.	0
rds.rds_superuser_reserved_connections	Imposta il numero di slot di connessione riservati a rds_superusers. Questo parametro è disponibile solo nella versione 15 e precedenti. Per ulteriori informazioni, consulta le connessioni riservate nella documentazione di PostgreSQL.	2
rds.superuser_variables	Elenco delle variabili del solo utente con privilegi avanzati per le quali eleviamo le istruzioni di modifica rds_superuser.	session_replication_role
remove_temp_files_after_crash	Rimuove i file temporanei dopo l'arresto anomalo del back-end	0
restart_after_crash	Reinizializza il server dopo l’arresto anomalo del back-end.	–
row_security	Abilita la sicurezza delle righe.	–
search_path	Imposta l’ordine di ricerca dello schema per i nomi che non sono qualificati come schema.	–

Nome del parametro	Descrizione	Default
seq_page_cost	Imposta la stima del pianificatore del costo di una pagina del disco recuperata in modo sequenziale.	–
session_replicatio n_role	Imposta il comportamento delle sessioni per i trigger e le regole di riscrittura.	–
shared_buffers	(8 kB) Imposta il numero di buffer di memoria condivisa utilizzati dal server.	SUM(DBInstanceClas sMemory/12038,-500 03)
shared_preload_lib raries	Elenca le librerie condivise da precaricare nel server.	pg_stat_statements
ssl_ca_file	Posizione del file dell'autorità del server SSL.	/rdsdbdata/rds-met adata/ca-cert.pem
ssl_cert_file	Posizione del file del certificato del server SSL.	/rdsdbdata/rds-met adata/server-cert.pem
ssl_crl_dir	Posizione della directory dell'elenco di revoche di certificati SSL.	/rdsdbdata/rds-met adata/ssl_crl_dir/
ssl_key_file	Posizione del file della chiave privata del server SSL	/rdsdbdata/rds-met adata/server-key.pem
standard_conformin g_strings	Fa sì che le stringhe ... trattino letteralmente le barre rovesciate.	–
statement_timeout	(ms) Imposta la durata massima concessa di ogni istruzione.	–
stats_temp_directory	Scrive i file di statistiche temporanei nella directory specificata.	/rdsdbdata/db/pg_s tat_tmp
superuser_reserved _connections	Imposta il numero di slot di connessione riservati agli utenti con privilegi avanzati.	3

Nome del parametro	Descrizione	Default
synchronize_seqscans	Abilita le scansioni sequenziali sincronizzate.	–
tcp_keepalives_count	Numero massimo di ritrasmissioni keepalive TCP.	–
tcp_keepalives_idle	(s) Tempo tra l'emissione di keepalive TCP.	–
tcp_keepalives_interval	(s) Tempo tra la ritrasmissione di keepalive TCP.	–
temp_buffers	(8 kB) Imposta il numero massimo di buffer temporanei utilizzati da ogni sessione.	–
temp_file_limit	Vincola la quantità totale di spazio su disco in kilobyte che un determinato processo PostgreSQL può utilizzare per i file temporanei, escludendo lo spazio utilizzato per tabelle temporanee esplicite	-1
temp_tablespaces	Imposta i tablespace da usare per le tabelle temporanee e i file di ordinamento.	–
track_activities	Raccoglie informazioni sui comandi di esecuzione.	–
track_activity_query_size	Imposta la dimensione riservata per pg_stat_activity.current_query, in byte.	4096
track_counts	Raccoglie statistiche sull’attività del database.	–
track_functions	Raccoglie statistiche a livello di funzione sull’attività del database.	pl
track_io_timing	Raccoglie statistiche di temporizzazione sull’attività di I/O del database.	1
transform_–_equals	Treats expr=– as expr IS –.	–

Nome del parametro	Descrizione	Default
update_process_title	Aggiorna il titolo del processo per mostrare il comando SQL attivo.	–
wal_receiver_status_interval	(s) Imposta l'intervallo massimo tra i report di stato del ricevitore WAL al nodo primario.	–
work_mem	(kB) Imposta la memoria massima da utilizzare per gli spazi di lavoro delle query.	–
xmlbinary	Imposta come i valori binari devono essere codificati in XML.	–
xmloption	Imposta se i dati XML nelle operazioni di analisi e serializzazione implicite devono essere considerati come documenti o frammenti di contenuto.	–

Eventi di attesa Amazon Aurora PostgreSQL

Di seguito sono riportati alcuni tra gli eventi di attesa più frequenti di Aurora PostgreSQL. Per ulteriori informazioni sugli eventi di attesa e l'ottimizzazione del cluster database Aurora PostgreSQL, consulta Sintonizzazione degli eventi di attesa per Aurora PostgreSQL.

Attività: ArchiverMain

Il processo di archiviazione è in attesa di attività.

Attività: AutoVacuumMain

Il processo del programma di avvio di autovacuum è in attesa di attività.

Attività: BgWriterHibernate

Il processo di scrittura in background è in ibernazione in attesa dell'attività.

Attività: BgWriterMain

Il processo di scrittura in background è in attesa di attività.

Attività: CheckpointerMain

Il processo di checkpointer è in attesa di attività.

Attività: LogicalApplyMain

Il processo di applicazione della replica logica è in attesa di attività.

Attività: LogicalLauncherMain

Il processo di avvio della replica logica è in attesa di attività.

Attività: PgStatMain

Il processo di raccolta statistiche è in attesa di attività.

Attività: RecoveryWalAll

Un processo è in attesa del WAL (write-ahead log) da un flusso al momento del ripristino.

Attività: RecoveryWalStream

Il processo di avvio è in attesa che il WAL (write-ahead log) arrivi durante il ripristino dello streaming.

Attività: SysLoggerMain

Il processo syslogger è in attesa di attività.

Attività: WalReceiverMain

Il processo WAL (write-ahead log) del destinatario è in attesa di attività.

Attività: WalSenderMain

Il processo WAL (write-ahead log) del mittente è in attesa di attività.

Attività: WalWriterMain

Il processo di scrittura WAL (write-ahead log) è in attesa di attività.

BufferPin:BufferPin

Un processo è in attesa di acquisire un pin esclusivo su un buffer.

Client: GSSOpen Server

Un processo è in attesa di leggere i dati dal client nel tentativo di stabilire una sessione GSSAPI (Generic Security Service Application Program Interface).

Cliente: ClientRead

Un processo di backend è in attesa di ricevere dati da un client PostgreSQL. Per ulteriori informazioni, consulta Client:ClientRead.

Cliente: ClientWrite

Un processo di backend è in attesa di inviare più dati a un client PostgreSQL. Per ulteriori informazioni, consulta Client:ClientWrite.

Cliente: Lib PQWal ReceiverConnect

Un processo è in attesa nel destinatario WAL (write-ahead log) per stabilire la connessione al server remoto.

Cliente: Lib PQWal ReceiverReceive

Un processo è in attesa nel destinatario WAL (write-ahead log) per ricevere i dati dal server remoto.

Cliente: Server SSLOpen

Un processo è in attesa di Secure Sockets Layer (SSL) durante il tentativo di connessione.

Cliente: WalReceiverWaitStart

Un processo è in attesa che il processo di avvio invii i dati iniziali per la replica in streaming.

Cliente: WalSenderWaitFor WAL

Un processo è in attesa che il WAL (write-ahead log) venga svuotato nel processo mittente WAL.

Cliente: WalSenderWriteData

Un processo è in attesa di qualsiasi attività durante l'elaborazione delle risposte dal destinatario WAL (write-ahead log) nel processo mittente WAL.

CPU

Un processo di backend è attivo o è in attesa della CPU. Per ulteriori informazioni, consulta CPU.

Extension:extension

Un processo di backend è in attesa di una condizione definita da un'estensione o da un modulo.

IO: AuroraEnhancedLogical WALRead

Un processo di backend sta recuperando i record di log dal volume CDC (Change Data Capture).

IO: AuroraOptimizedReadsCacheRead

Un processo è in attesa di una lettura dalla cache a più livelli di Letture ottimizzate perché la pagina non è disponibile nella memoria condivisa.

IO: AuroraOptimizedReads CacheSegmenttronca

Un processo è in attesa che un file di segmento della cache a più livelli di Letture ottimizzate venga troncato.

IO: AuroraOptimizedReadsCacheWrite

Il processo di scrittura in background è in attesa di scrivere nella cache a più livelli di Letture ottimizzate.

IO: AuroraStorageLogAllocate

Una sessione sta allocando i metadati e si sta preparando per la scrittura nel registro delle transazioni.

IO: BufFileRead

Quando le operazioni richiedono più memoria della quantità definita dai parametri di memoria di lavoro, il motore crea file temporanei su disco. Questo evento di attesa si verifica quando le operazioni vengono lette dai file temporanei. Per ulteriori informazioni, consulta IO: BufFileRead e IO: BufFileWrite.

IO: BufFileWrite

Quando le operazioni richiedono più memoria della quantità definita dai parametri di memoria di lavoro, il motore crea file temporanei su disco. Questo evento di attesa si verifica quando le operazioni scrivono nei file temporanei. Per ulteriori informazioni, consulta IO: BufFileRead e IO: BufFileWrite.

IO: ControlFileRead

Un processo è in attesa di una lettura dal file `pg_control`.

IO: ControlFileSync

Un processo è in attesa del file `pg_control` per raggiungere un'archiviazione durevole.

IO: ControlFileSyncUpdate

Un processo è in attesa di un aggiornamento del file `pg_control` per raggiungere un'archiviazione durevole.

IO: ControlFileWrite

Un processo è in attesa di una scrittura sul file `pg_control`.

IO: ControlFileWriteUpdate

Un processo è in attesa di una scrittura per aggiornare il file `pg_control`.

IO: CopyFileRead

Un processo è in attesa di una lettura durante un'operazione di copia dei file.

IO: CopyFileWrite

Un processo è in attesa di una scrittura durante un'operazione di copia dei file.

IO: DataFileExtend

Un processo è in attesa dell'estensione di un file di dati di relazione.

IO: DataFileFlush

Un processo è in attesa che un file di dati di relazione raggiunga un’archiviazione durevole.

IO: DataFileImmediateSync

Un processo è in attesa di una sincronizzazione immediata di un file di dati di relazione a un’archiviazione durevole.

IO: DataFilePrefetch

Un processo è in attesa di un prefetch asincrono da un file di dati di relazione.

IO: DataFileSync

Un processo è in attesa di modifiche a un file di dati di relazione per raggiungere un’archiviazione durevole.

IO: DataFileRead

Un processo di back-end ha cercato di trovare una pagina nei buffer condivisi, non l'ha trovata e quindi l'ha letta dall’archiviazione. Per ulteriori informazioni, consulta IO:DataFileRead.

IO: DataFileTruncate

Un processo è in attesa che un file di dati di relazione venga troncato.

IO: DataFileWrite

Un processo è in attesa di una scrittura su un file di dati di relazione.

IO: DSMFill ZeroWrite

Un processo è in attesa di scrivere zero byte su un file dinamico di backup della memoria condivisa.

IO: LockFileAddToDataDirRead

Un processo è in attesa di una lettura durante l'aggiunta di una riga al file da bloccare della directory dati.

IO: LockFileAddToDataDirSync

Un processo è in attesa che i dati raggiungano un'archiviazione durevole durante l'aggiunta di una riga al file da bloccare della directory dati.

IO: LockFileAddToDataDirWrite

Un processo è in attesa di scrittura durante l'aggiunta di una riga al file da bloccare della directory dati.

IO: LockFileCreateRead

Un processo è in attesa di lettura durante la creazione del file da bloccare della directory dati.

IO: LockFileCreateSync

Un processo è in attesa che i dati raggiungano un'archiviazione durevole durante la creazione del file da bloccare della directory dati.

IO: LockFileCreateWrite

Un processo è in attesa di scrittura durante la creazione del file da bloccare della directory dati.

IO: LockFileReCheckDataDirRead

Un processo è in attesa di una lettura durante il ricontrollo del file da bloccare della directory dati.

IO: LogicalRewriteCheckpointSync

Un processo è in attesa che le mappature di riscrittura logica raggiungano un'archiviazione durevole durante un checkpoint.

IO: LogicalRewriteMappingSync

Un processo è in attesa che i dati di mappatura raggiungano un'archiviazione durevole durante una riscrittura logica.

IO: LogicalRewriteMappingWrite

Un processo è in attesa di una scrittura dei dati di mappatura durante una riscrittura logica.

IO: LogicalRewriteSync

Un processo è in attesa che le mappature di riscrittura logica raggiungano un'archiviazione durevole.

IO: LogicalRewriteTruncate

Un processo è in attesa del troncamento dei dati di mappatura durante una riscrittura logica.

IO: LogicalRewriteWrite

Un processo è in attesa di una scrittura di mappature di riscrittura logica.

IO: RelationMapRead

Un processo è in attesa di una lettura del file di mappa della relazione.

IO: RelationMapSync

Un processo è in attesa che il file di mappa della relazione raggiunga un'archiviazione durevole.

IO: RelationMapWrite

Un processo è in attesa di una scrittura sul file di mappatura della relazione.

IO: ReorderBufferRead

Un processo è in attesa di una lettura durante la gestione del buffer da riordinare.

IO: ReorderBufferWrite

Un processo è in attesa di una scrittura durante la gestione del buffer da riordinare.

IO: ReorderLogicalMappingRead

Un processo è in attesa di una lettura di una mappatura logica durante la gestione del buffer da riordinare.

IO: ReplicationSlotRead

Un processo è in attesa di una lettura da un file di controllo dello slot di replica.

IO: ReplicationSlotRestoreSync

Un processo è in attesa che un file di controllo dello slot di replica raggiunga un'archiviazione durevole durante il ripristino in memoria.

IO: ReplicationSlotSync

Un processo è in attesa che un file di controllo dello slot di replica raggiunga un'archiviazione durevole.

IO: ReplicationSlotWrite

Un processo è in attesa di una scrittura su un file di controllo dello slot di replica.

IO: SLRUFlush sincronizzazione

Un processo è in attesa che i dati semplici utilizzati meno di recente (SLRU) raggiungano un'archiviazione durevole durante un checkpoint o uno spegnimento del database.

IO: SLRURead

Un processo è in attesa di una lettura di una pagina semplice utilizzata meno di recente (SLRU).

IO: SLRUSync

Un processo è in attesa che i dati semplici utilizzati meno di recente (SLRU) raggiungano un'archiviazione durevole dopo la scrittura di una pagina.

IO: SLRUWrite

Un processo è in attesa della scrittura di una pagina semplice utilizzata meno di recente (SLRU).

IO: SnapbuildRead

Un processo è in attesa di una lettura di uno snapshot del catalogo storico serializzato.

IO: SnapbuildSync

Un processo è in attesa che uno snapshot del catalogo storico serializzato raggiunga un'archiviazione durevole.

IO: SnapbuildWrite

Un processo è in attesa della scrittura di uno snapshot del catalogo storico serializzato.

IO: TimelineHistoryFileSync

Un processo è in attesa che un file della cronologia della timeline ricevuto tramite la replica in streaming raggiunga un'archiviazione durevole.

IO: TimelineHistoryFileWrite

Un processo è in attesa di una scrittura di un file di cronologia della timeline ricevuto tramite la replica in streaming.

IO: TimelineHistoryRead

Un processo è in attesa di una lettura di un file cronologico della timeline.

IO: TimelineHistorySync

Un processo è in attesa che un file cronologico della timeline appena creato raggiunga un'archiviazione durevole.

IO: TimelineHistoryWrite

Un processo è in attesa di una scrittura di un file di cronologia della timeline appena creato.

IO: TwophaseFileRead

Un processo è in attesa di una lettura di un file di stato in due fasi.

IO: TwophaseFileSync

Un processo è in attesa che un file di stato in due fasi raggiunga un'archiviazione durevole.

IO: TwophaseFileWrite

Un processo è in attesa di scrittura di un file di stato in due fasi.

IO: WALBootstrap sincronizzazione

Un processo è in attesa che il WAL (write-ahead log) raggiunga un'archiviazione durevole durante il processo di bootstrap.

IO: WALBootstrap Scrivi

Un processo è in attesa di scrittura di una pagina WAL (write-ahead log) durante il processo di bootstrap.

IO: WALCopy Leggi

Un processo è in attesa di una lettura durante la creazione di un nuovo segmento WAL (write-ahead log) tramite copia di uno esistente.

IO: WALCopy sincronizzazione

Un processo è in attesa di un nuovo segmento WAL (write-ahead log) creato tramite copia di uno esistente per raggiungere un'archiviazione durevole.

IO: WALCopy Scrivi

Un processo è in attesa di una scrittura quando si crea un nuovo segmento WAL (write-ahead log) tramite copia di uno esistente.

IO: WALInit sincronizzazione

Un processo è in attesa che un file WAL (write-ahead log) di recente inizializzazione raggiunga l'archiviazione durevole.

IO: WALInit Scrivi

Un processo è in attesa di una scrittura durante l'inizializzazione di un nuovo file WAL (write-ahead log).

IO: WALRead

Un processo è in attesa di una lettura da un file WAL (write-ahead log).

IO: WALSender TimelineHistoryRead

Un processo è in attesa di una lettura da un file cronologico della timeline durante un comando della timeline del mittente WAL (write-ahead log).

IO: WALSync

Un processo è in attesa che un file WAL (write-ahead log) raggiunga un'archiviazione durevole.

IO: WALSync MethodAssign

Un processo è in attesa che i dati raggiungano un'archiviazione durevole durante l'assegnazione di un nuovo metodo di sincronizzazione WAL (write-ahead log).

IO: WALWrite

Un processo è in attesa di una scrittura su un file WAL (write-ahead log).

IO: XactSync

Un processo di backend è in attesa che il sottosistema archiviazione Aurora riconosca il commit di una transazione regolare o il commit o il rollback di una transazione preparata. Per ulteriori informazioni, consulta IO:XactSync.

IPC: AuroraLogicalSchemaUpdate

Due processi di backend stanno tentando di inserire la stessa voce nella cache dello schema. Un processo continuerà mentre l'altro attende il completamento.

IPC: AuroraOptimizedReadsCacheWriteStop

Un processo è in attesa che l'istanza di scrittura in background interrompa la scrittura nella cache a più livelli di letture ottimizzate.

IPC: BackupWaitWalArchive

Un processo è in attesa dei file WAL (write-ahead log) necessari per l'archiviazione corretta di un backup.

IPC: BgWorkerShutdown

Un processo è in attesa dell'arresto del lavoro in background.

IPC: BgWorkerStartup

Un processo è in attesa dell'avvio di un lavoro in background.

IPC: BtreePage

Un processo è in attesa che il numero di pagina necessario per continuare la scansione parallela del B-tree diventi disponibile.

IPC: CheckpointDone

Un processo è in attesa del completamento di un checkpoint.

IPC: CheckpointStart

Un processo è in attesa dell'avvio di un checkpoint.

IPC: ClogGroupUpdate

Un processo è in attesa che il leader del gruppo aggiorni lo stato della transazione al termine di una transazione.

IPC: DamRecordTxAck

Un processo di backend ha generato un evento di flussi di attività del database ed è in attesa che l'evento diventi durevole. Per ulteriori informazioni, consulta IPC:DamRecordTxAck.

IPC: ExecuteGather

Un processo è in attesa di attività da un processo figlio durante l'esecuzione di un nodo del piano Gather.

IPC: Hash/Batch/Allocating

Un processo è in attesa dell'allocazione di una tabella hash da parte di un partecipante all'hash parallelo eletto.

IPC: Hash/Batch/Electing

Un processo sta eleggendo un partecipante all'hash parallelo per allocare una tabella hash.

IPC: Hash/Batch/Loading

Un processo è in attesa che altri partecipanti all’hash parallelo finiscano il caricamento di una tabella hash.

IPC: Hash/Build/Allocating

Un processo è in attesa dell’allocazione della tabella hash iniziale da parte di un partecipante all’hash parallelo eletto.

IPC: Hash/Build/Electing

Un processo sta eleggendo un partecipante all’hash parallelo per allocare la tabella hash iniziale.

IPC: Hash/Build/HashingInner

Un processo è in attesa che altri partecipanti all’hash parallelo finiscano di eseguire l'hashing della relazione interna.

IPC: Hash/Build/HashingOuter

Un processo è in attesa che altri partecipanti all’hash parallelo finiscano il partizionamento della relazione esterna.

IPC: Hash/GrowBatches/Allocating

Un processo è in attesa dell’allocazione di più batch da parte di un partecipante all’hash parallelo eletto.

IPC: Hash/GrowBatches/Deciding

Un processo sta eleggendo un partecipante all'hash parallelo per decidere sull’aumento futuro dei batch.

IPC: Hash/GrowBatches/Electing

Un processo sta eleggendo un partecipante all’hash parallelo per l’allocazione di più batch.

IPC: Hash/GrowBatches/Finishing

Un processo è in attesa che un partecipante all'hash parallelo eletto decida sulla crescita futura dei batch.

IPC: Hash/GrowBatches/Repartitioning

Un processo è in attesa che altri partecipanti all'hash parallelo finiscano il ripartizionamento.

IPC: Hash/GrowBuckets/Allocating

Un processo è in attesa che un partecipante all'hash parallelo eletto finisca l'allocazione di più bucket.

IPC: Hash/GrowBuckets/Electing

Un processo sta eleggendo un partecipante all'hash parallelo per l'allocazione di più bucket.

IPC: Hash/GrowBuckets/Reinserting

Un processo è in attesa che altri partecipanti all'hash parallelo finiscano di inserire tuple in nuovi bucket.

IPC: HashBatchAllocate

Un processo è in attesa dell'allocazione di una tabella hash da parte di un partecipante all'hash parallelo eletto.

IPC: HashBatchElect

Un processo è in attesa di eleggere un partecipante all'hash parallelo per allocare una tabella hash.

IPC: HashBatchLoad

Un processo è in attesa che altri partecipanti all'hash parallelo finiscano il caricamento di una tabella hash.

IPC: HashBuildAllocate

Un processo è in attesa dell'allocazione della tabella hash iniziale da parte di un partecipante all'hash parallelo eletto.

IPC: HashBuildElect

Un processo è in attesa di eleggere un partecipante all'hash parallelo per allocare la tabella hash iniziale.

IPC: HashBuildHashInner

Un processo è in attesa che altri partecipanti all'hash parallelo finiscano di eseguire l'hashing della relazione interna.

IPC:» HashBuildHashOuter

Un processo è in attesa che altri partecipanti all'hash parallelo finiscano il partizionamento della relazione esterna.

IPC: HashGrowBatchesAllocate

Un processo è in attesa dell'allocazione di più batch da parte di un partecipante all'hash parallelo eletto.

IPC:» HashGrowBatchesDecide

Un processo è in attesa di eleggere un partecipante all'hash parallelo per decidere sulla crescita futura dei batch.

IPC: HashGrowBatchesElect

Un processo è in attesa di eleggere un partecipante all'hash parallelo per allocare più batch.

IPC: HashGrowBatchesFinish

Un processo è in attesa che un partecipante all'hash parallelo eletto decida sulla crescita futura dei batch.

IPC: HashGrowBatchesRepartition

Un processo è in attesa che altri partecipanti all'hash parallelo finiscano il ripartizionamento.

IPC: HashGrowBucketsAllocate

Un processo è in attesa che un partecipante all'hash parallelo eletto finisca l'allocazione di più bucket.

IPC: HashGrowBucketsElect

Un processo è in attesa di eleggere un partecipante all'hash parallelo per allocare più bucket.

IPC: HashGrowBucketsReinsert

Un processo è in attesa che altri partecipanti all'hash parallelo finiscano di inserire tuple in nuovi bucket.

IPC: LogicalSyncData

Un processo è in attesa che un server remoto di replica logica invii dati per la sincronizzazione iniziale della tabella.

IPC: LogicalSyncStateChange

Un processo è in attesa che un server remoto di replica logica modifichi lo stato.

IPC: MessageQueueInternal

Un processo è in attesa che un altro processo sia allegato a una coda di messaggi condivisa.

IPC: MessageQueuePutMessage

Un processo è in attesa di scrivere un messaggio di protocollo in una coda di messaggi condivisa.

IPC: MessageQueueReceive

Un processo è in attesa di ricevere byte da una coda di messaggi condivisa.

IPC: MessageQueueSend

Un processo è in attesa di inviare byte a una coda di messaggi condivisa.

IPC: ParallelBitmapScan

Un processo è in attesa di inizializzazione da parte di una scansione bitmap parallela.

IPC: ParallelCreateIndexScan

Un processo è in attesa che i lavori CREATE INDEX (CREA INDICE) paralleli finiscano una scansione heap.

IPC: ParallelFinish

Un processo è in attesa che i lavori paralleli finiscano l'elaborazione.

IPC: ProcArrayGroupUpdate

Un processo è in attesa che il leader del gruppo cancelli l'ID transazione al termine di una transazione.

IPC: ProcSignalBarrier

Un processo è in attesa che un evento barriera venga elaborato da tutti i backend.

IPC:Promote

Un processo è in attesa di una promozione in standby.

IPC: RecoveryConflictSnapshot

Un processo è in attesa della risoluzione dei conflitti di ripristino per una pulizia vacuum.

IPC: RecoveryConflictTablespace

Un processo è in attesa della risoluzione dei conflitti di ripristino per l'eliminazione di uno spazio di tabella.

IPC: RecoveryPause

Un processo è in attesa di riavvio del ripristino.

IPC: ReplicationOriginDrop

Un processo è in attesa che un'origine di replica diventi inattiva in modo che possa essere eliminata.

IPC: ReplicationSlotDrop

Un processo è in attesa che uno slot di replica diventi inattivo in modo che possa essere eliminato.

IPC: SafeSnapshot

Un processo è in attesa di ottenere uno snapshot valido per una transazione READ ONLY DEFERRABLE (SOLA LETTURA DIFFERIBILE).

IPC: SyncRep

Un processo è in attesa di conferma da un server remoto durante la replica sincrona.

IPC: XactGroupUpdate

Un processo è in attesa che il leader del gruppo aggiorni lo stato della transazione al termine di una transazione.

Lock:advisory

Un processo di back-end è in attesa del blocco di consulenza che ha richiesto. Per ulteriori informazioni, consulta Lock:advisory.

Lock:extend

Un processo di backend è in attesa del rilascio di un blocco in modo che possa estendere una relazione. Questo blocco è necessario perché solo un processo di back-end può estendere una relazione alla volta. Per ulteriori informazioni, consulta Lock:extend.

Lock:frozenid

Un processo è in attesa di aggiornare `pg_database.datfrozenxid` e `pg_database.datminmxid`.

Lock:object

Un processo è in attesa di ottenere un blocco su un oggetto di database non relazionale.

Lock:page

Un processo è in attesa di ottenere un blocco su una pagina di una relazione.

Lock:Relation

Un processo di backend è in attesa di acquisire un blocco su una relazione bloccata da un'altra transazione. Per ulteriori informazioni, consulta Lock:Relation.

Lock:spectoken

Un processo è in attesa di ottenere un blocco di inserimento speculativo.

Token Lock:speculative

Un processo è in attesa di acquisire un blocco di inserimento speculativo.

Lock:transactionid

Una transazione è in attesa di un blocco a livello di riga. Per ulteriori informazioni, consulta Lock:transactionid.

Lock:tuple

Un processo di backend è in attesa di acquisire un blocco su una tupla mentre un altro processo di backend tiene un blocco in conflitto sulla stessa tupla. Per ulteriori informazioni, consulta Lock:tuple.

Lock:userlock

Un processo è in attesa di ottenere un blocco utente.

Lock:virtualxid

Un processo è in attesa di ottenere un blocco ID di transazione virtuale.

LWLock:AddinShmemInit

Un processo è in attesa di gestire l'allocazione dello spazio di un'estensione nella memoria condivisa.

LWLock:AddinShmemInitLock

Un processo è in attesa di gestire l'allocazione dello spazio nella memoria condivisa.

LWLockIPC: ----sep----:async

Un processo è in attesa I/O su un buffer asincrono (notify).

LWLock:AsyncCtlLock

Un processo è in attesa di leggere o aggiornare uno stato di notifica condiviso.

LWLock:AsyncQueueLock

Un processo è in attesa di leggere o aggiornare i messaggi di notifica.

LWLock:AuroraOptimizedReadsCacheMapping

Un processo è in attesa di associare un blocco di dati a una pagina nella cache a più livelli di Letture ottimizzate.

LWLock:AutoFile

Un processo è in attesa di aggiornare il file `postgresql.auto.conf`.

LWLock:AutoFileLock

Un processo è in attesa di aggiornare il file `postgresql.auto.conf`.

LWLockUn processo è in attesa su un buffer asincrono (notifica). ----sep----:AutoVacuum

Un processo è in attesa di leggere o aggiornare lo stato attuale dei lavori autovacuum.

LWLock:AutovacuumLock

Un lavoro o un programma di avvio autovacuum è in attesa di aggiornare o leggere lo stato attuale dei lavori autovacuum.

LWLock:AutovacuumSchedule

Un processo è in attesa di garantire che una tabella selezionata per l'autovacuum necessita ancora di vacuum.

LWLock:AutovacuumScheduleLock

Un processo è in attesa di garantire che la tabella selezionata per il vacuum necessiti ancora di vacuum.

LWLock:BackendRandomLock

Un processo è in attesa di generare un numero casuale.

LWLock:BackgroundWorker

Un processo è in attesa di leggere o aggiornare lo stato del lavoro in background.

LWLock:BackgroundWorkerLock

Un processo è in attesa di leggere o aggiornare lo stato del lavoro in background.

LWLock:BtreeVacuum

Un processo è in attesa di leggere o aggiornare le informazioni relative al vacuum per un indice B-tree.

LWLock:BtreeVacuumLock

Un processo è in attesa di leggere o aggiornare le informazioni relative al vacuum per un indice B-tree.

LWLock:Autovacuum ----sep----:buffer_content

Un processo di backend è in attesa di acquisire un blocco leggero sui contenuti di un buffer di memoria condiviso. Per ulteriori informazioni, consulta LWLock:buffer_content (BufferContent).

LWLock:buffer_content ----sep----:buffer_mapping

Un processo di backend è in attesa di associare un blocco di dati a un buffer nel pool di buffer condiviso. Per ulteriori informazioni, consulta LWLock:buffer_mapping.

LWLock:buffer_mapping ----Sep----:Bufferio

Un processo di back-end vuole leggere una pagina nella memoria condivisa. Il processo è in attesa che gli altri processi finiscano I/O la pagina. Per ulteriori informazioni, consulta LWLock:BufferIO (IPC:BufferIO).

LWLockIl processo è in attesa che altri processi finiscano la pagina. ----sep----:Checkpoint

Un processo è in attesa di iniziare un checkpoint.

LWLock:CheckpointLock

Un processo è in attesa di eseguire il checkpoint.

LWLock:CheckpointerComm

Un processo è in attesa di gestire le richieste `fsync`.

LWLock:CheckpointerCommLock

Un processo è in attesa di gestire le richieste `fsync`.

LWLock:Punto di controllo ----sep----:zoccolo

Un processo è in attesa I/O su un buffer di clog (status della transazione).

LWLock:CLogControlLock

Un processo è in attesa di leggere o aggiornare lo stato della transazione.

LWLock:CLogTruncationLock

Un processo è in attesa di eseguire `txid_status` o di aggiornare l'ID di transazione meno recente disponibile.

LWLockUn processo è in attesa su un buffer di clog (status della transazione). ----sep----:commit_timestamp

Un processo è in attesa su un buffer di timestamp di I/O commit.

LWLock:CommitTs

Un processo è in attesa di leggere o aggiornare l'ultimo valore impostato per un timestamp commit della transazione.

LWLock:CommitTsBuffer

Un processo è in attesa di un timestamp di I/O commit su un semplice buffer SLRU (Least-Recently Used).

LWLock:CommitTsControlLock

Un processo è in attesa di leggere o aggiornare i timestamp commit della transazione.

LWLock:CommitTsLock

Un processo è in attesa di leggere o aggiornare l'ultimo valore impostato per il timestamp della transazione.

LWLockCommitTs: SLRU

Un processo è in attesa di accedere alla cache semplice utilizzata meno di recente (SLRU) per un timestamp commit.

LWLock:ControlFile

Un processo è in attesa di leggere o aggiornare il file `pg_control` o di creare un nuovo file WAL (write-ahead log).

LWLock:ControlFileLock

Un processo è in attesa di leggere o aggiornare il control file o la creazione di un nuovo file WAL (write-ahead log).

LWLock:DynamicSharedMemoryControl

Un processo è in attesa di leggere o aggiornare le informazioni dinamiche di allocazione della memoria condivisa.

LWLock:DynamicSharedMemoryControlLock

Un processo è in attesa di leggere o aggiornare lo stato dinamico della memoria condivisa.

LWLock: SLRU ----sep----:lock_manager

Un processo di backend è in attesa di aggiungere o esaminare i blocchi per i processi di backend. Oppure è in attesa di unirsi o uscire da un gruppo di blocco utilizzato dalla query parallela. Per ulteriori informazioni, consulta LWLock:lock_manager.

LWLock:LockFastPath

Un processo è in attesa di leggere o aggiornare le informazioni di blocco del percorso rapido di un processo.

LWLock:LogicalRepWorker

Un processo è in attesa di leggere o aggiornare lo stato dei lavori di replica logica.

LWLock:LogicalRepWorkerLock

Un processo è in attesa del termine di un'azione su un lavoro di replica logica.

LWLock:LogicalSchemaCache

Un processo ha modificato la cache dello schema.

LWLock:lock_manager ----sep----:multixact_member

Un processo è in attesa su un buffer multixact_member I/O .

LWLockUn processo è in attesa su un buffer multixact_member. ----sep----:multixact_offset

Un processo è in attesa su un buffer di offset multixact I/O .

LWLock:MultiXactGen

Un processo è in attesa di leggere o aggiornare lo stato multixact condiviso.

LWLock:MultiXactGenLock

Un processo è in attesa di leggere o aggiornare uno stato multixact condiviso.

LWLock:MultiXactMemberBuffer

Un processo è in attesa I/O su un buffer SLRU (Simple Least-Recently Used) per un membro multixact. Per ulteriori informazioni, consulta LWLock:MultiXact.

LWLock:MultiXactMemberControlLock

Un processo è in attesa di leggere o aggiornare le mappature dei membri multixact.

LWLockMultiXactMember: SLRU

Un processo è in attesa di accedere alla cache semplice utilizzata meno di recente (SLRU) per un membro multixact. Per ulteriori informazioni, consulta LWLock:MultiXact.

LWLock:MultiXactOffsetBuffer

Un processo è in attesa di un offset I/O multixact su un semplice buffer SLRU (Least-Recently Used). Per ulteriori informazioni, consulta LWLock:MultiXact.

LWLock:MultiXactOffsetControlLock

Un processo è in attesa di leggere o aggiornare le mappature offset multixact.

LWLockMultiXactOffset: SLRU

Un processo è in attesa di accedere alla cache semplice utilizzata meno di recente (SLRU) per un offset multixact. Per ulteriori informazioni, consulta LWLock:MultiXact.

LWLock:MultiXactTruncation

Un processo è in attesa di leggere o troncare informazioni multixact.

LWLock:MultiXactTruncationLock

Un processo è in attesa di leggere o troncare informazioni multixact.

LWLock:NotifyBuffer

Un processo è in attesa di un messaggio NOTIFY I/O sul buffer Simple Least-Recently Used (SLRU).

LWLock:NotifyQueue

Un processo è in attesa di leggere o aggiornare i messaggi NOTIFY (NOTIFICA).

LWLock:NotifyQueueTail

Un processo è in attesa di aggiornare un limite per l’archiviazione dei messaggi NOTIFY (NOTIFICA).

LWLock:NotifyQueueTailLock

Un processo è in attesa di aggiornare il limite sull’archiviazione dei messaggi di notifica.

LWLockUn processo è in attesa sul buffer SLRU (simple least-recently used) per un messaggio NOTIFY. ----sep----:notifySLRU

Un processo è in attesa di accedere alla cache semplice utilizzata meno di recente (SLRU) per un messaggio NOTIFY (NOTIFICA).

LWLock:OidGen

Un processo è in attesa di allocare un nuovo ID oggetto (OID).

LWLock:OidGenLock

Un processo è in attesa di allocare o assegnare un ID oggetto (OID).

LWLock:notifySLRU ----sep----:oldserxid

Un processo è in attesa su un buffer oldserxid I/O .

LWLock:OldSerXidLock

Un processo è in attesa di leggere o registrare transazioni serializzabili in conflitto.

LWLock:OldSnapshotTimeMap

Un processo è in attesa di leggere o aggiornare le informazioni meno recenti sul controllo degli snapshot.

LWLock:OldSnapshotTimeMapLock

Un processo è in attesa di leggere o aggiornare le informazioni meno recenti sul controllo degli snapshot.

LWLockUn processo è in attesa su un buffer oldserxid. ----sep----:parallel_append

Un processo è in attesa di scegliere il subplan successivo durante l'esecuzione parallela del piano di aggiunta.

LWLock:parallel_append ----sep----:parallel_hash_join

Un processo è in attesa di allocare o scambiare un blocco di memoria o di aggiornare i contatori durante l'esecuzione di un piano hash parallelo.

LWLock:parallel_hash_join ----sep----:parallel_query_dsa

Un processo è in attesa di un blocco sull'allocazione dinamica della memoria condivisa per una query parallela.

LWLock:ParallelAppend

Un processo è in attesa di scegliere il subplan successivo durante l'esecuzione parallela del piano di aggiunta.

LWLock:ParallelHashJoin

Un processo è in attesa di sincronizzare i lavori durante l'esecuzione del piano per un hash join parallelo.

Lucchetto: DSA ParallelQuery

Un processo è in attesa di un'allocazione dinamica della memoria condivisa per una query parallela.

Lwlock: DSA PerSession

Un processo è in attesa di un'allocazione dinamica della memoria condivisa per una query parallela.

Serratura: PerSessionRecordType

Un processo è in attesa di accedere alle informazioni di una query parallela sui tipi compositi.

Serratura: PerSessionRecordTypmod

Un processo è in attesa di accedere alle informazioni di una query parallela sui modificatori di tipo che identificano i tipi di record anonimi.

Serratura: PerXactPredicateList

Un processo è in attesa di accedere all'elenco dei blocchi del predicato tenuti dalla transazione serializzabile corrente durante una query parallela.

Lwlock:predicate_lock_manager

Un processo è in attesa di aggiungere o esaminare le informazioni sul blocco dei predicati.

Serratura: PredicateLockManager

Un processo è in attesa di accedere alle informazioni sul blocco predicato utilizzate dalle transazioni serializzabili.

Lwlock:proc

Un processo è in attesa di leggere o aggiornare le informazioni sul blocco del percorso rapido.

LWLock:ProcArray

Un processo è in attesa di accedere alle strutture dati condivise per processo (in genere, per ottenere uno snapshot o segnalare l'ID di transazione di una sessione).

LWLock:ProcArrayLock

Un processo è in attesa di ottenere uno snapshot o di cancellare un'ID di transazione alla fine di una transazione.

LWLock:RelationMapping

Un processo è in attesa di leggere o aggiornare un file `pg_filenode.map` (utilizzato monitorare le assegnazioni file-nodo di alcuni cataloghi di sistema).

LWLock:RelationMappingLock

Un processo è in attesa di aggiornare il file della mappa delle relazioni utilizzato per catalog-to-file-node memorizzare la mappatura.

LWLock:RelCacheInit

Un processo è in attesa di leggere o aggiornare un file `pg_internal.init` (un file di inizializzazione della cache delle relazioni).

LWLock:RelCacheInitLock

Un processo è in attesa di leggere o scrivere un file di inizializzazione della cache delle relazioni.

LWLockUn processo è in attesa di aggiornare il file della mappa delle relazioni utilizzato per memorizzare la mappatura. ----sep----:replication_origin

Un processo è in attesa di leggere o aggiornare l'avanzamento della replica.

LWLock:replication_origin ----sep----:replication_slot_io

Un processo è in attesa su uno slot di replica. I/O

LWLock:ReplicationOrigin

Un processo è in attesa di creare, eliminare o utilizzare un'origine di replica.

LWLock:ReplicationOriginLock

Un processo è in attesa di configurare, eliminare o utilizzare un'origine di replica.

LWLock:ReplicationOriginState

Un processo è in attesa di leggere o aggiornare l'avanzamento di un'origine di replica.

LWLock:ReplicationSlotAllocation

Un processo è in attesa di allocare o liberare uno slot di replica.

LWLock:ReplicationSlotAllocationLock

Un processo è in attesa di allocare o liberare uno slot di replica.

LWLock:ReplicationSlotControl

Un processo è in attesa di leggere o aggiornare lo stato di uno slot di replica.

LWLock:ReplicationSlotControlLock

Un processo è in attesa di leggere o aggiornare lo stato dello slot di replica.

LWLock: IO ReplicationSlot

Un processo è in attesa I/O su uno slot di replica.

LWLock:SerialBuffer

Un processo è in attesa di un conflitto di transazioni serializzabile I/O su un semplice buffer SLRU (Least-Recently Used).

LWLock:SerializableFinishedList

Un processo è in attesa di accedere all'elenco delle transazioni serializzabili finite.

LWLock:SerializableFinishedListLock

Un processo è in attesa di accedere all'elenco delle transazioni serializzabili finite.

LWLock:SerializablePredicateList

Un processo è in attesa di accedere all'elenco dei blocchi del predicato tenuti dalle transazioni serializzabili.

LWLock:SerializablePredicateLockListLock

Un processo è in attesa di eseguire un'operazione su un elenco di blocchi tenuti dalle transazioni serializzabili.

LWLock:SerializableXactHash

Un processo è in attesa di leggere o aggiornare le informazioni sulle transazioni serializzabili.

LWLock:SerializableXactHashLock

Un processo è in attesa di recuperare o archiviare informazioni sulle transazioni serializzabili.

LWLockUn processo è in attesa su un semplice buffer SLRU (Least-Recently Used) per un conflitto di transazioni serializzabile. ----SEP----:SerialSLRU

Un processo è in attesa di accedere alla cache semplice utilizzata meno di recente (SLRU) per un conflitto di transazioni serializzabile.

LWLock:SharedTidBitmap

Un processo è in attesa di accedere a una bitmap TID (shared tuple identifier) durante una scansione parallela dell'indice bitmap.

LWLock:SharedTupleStore

Un processo è in attesa di accedere a un archivio tupla condiviso durante una query parallela.

LWLock:ShmemIndex

Un processo è in attesa di trovare o allocare spazio nella memoria condivisa.

LWLock:ShmemIndexLock

Un processo è in attesa di trovare o allocare spazio nella memoria condivisa.

LWLock: Leggi SInval

Un processo è in attesa di recuperare i messaggi dalla coda di invalidamento del catalogo condiviso.

LWLock:SInvalReadLock

Un processo è in attesa di recuperare o rimuovere messaggi da una coda di invalidazione condivisa.

LWLock: SInval Scrivi

Un processo è in attesa di aggiungere un messaggio alla coda di invalidazione del catalogo condivisa.

LWLock:SInvalWriteLock

Un processo è in attesa di aggiungere un messaggio in una coda di invalidazione condivisa.

LWLock:SyncRep

Un processo è in attesa di leggere o aggiornare le informazioni sullo stato della replica sincrona.

LWLock:SyncRepLock

Un processo è in attesa di leggere o aggiornare le informazioni sulle repliche sincrone.

LWLock:SyncScan

Un processo è in attesa di selezionare la posizione iniziale di una scansione sincronizzata della tabella.

LWLock:SyncScanLock

Un processo è in attesa di ottenere la posizione iniziale di una scansione su una tabella per le scansioni sincronizzate.

LWLock:TablespaceCreate

Un processo è in attesa di creare o eliminare uno spazio tabella.

LWLock:TablespaceCreateLock

Un processo è in attesa di creare o eliminare uno spazio tabella.

LWLock: Scrivi ----sep----:tbm

Un processo è in attesa di un blocco iteratore condiviso su una bitmap ad albero (TBM).

LWLock:TwoPhaseState

Un processo è in attesa di leggere o aggiornare lo stato delle transazioni preparate.

LWLock:TwoPhaseStateLock

Un processo è in attesa di leggere o aggiornare lo stato delle transazioni preparate.

LWLock:tbm ----sep----:wal_insert

Un processo è in attesa di inserire il WAL (write-ahead log) in un buffer di memoria.

LWLock: Mappatura WALBuf

Un processo è in attesa di sostituire una pagina nei buffer WAL (write-ahead log).

LWLock:WALBufMappingLock

Un processo è in attesa di sostituire una pagina nei buffer WAL (write-ahead log).

LWLock:WALInsert

Un processo è in attesa di inserire i dati WAL (write-ahead log) in un buffer di memoria.

LWLock:WALWrite

Un processo è in attesa della scrittura dei buffer WAL (write-ahead log) su disco.

LWLock: Serratura WALWrite

Un processo è in attesa della scrittura dei buffer WAL (write-ahead log) su disco.

LWLock:WrapLimitsVacuum

Un processo è in attesa di aggiornare i limiti relativi all'ID di transazione e al consumo multixact.

LWLock:WrapLimitsVacuumLock

Un processo è in attesa di aggiornare i limiti relativi all'ID di transazione e al consumo multixact.

LWLock:XactBuffer

Un processo è in attesa dello stato della I/O transazione su un buffer SLRU (Simple Least-Recently Used).

LWLockUn processo è in attesa su un semplice buffer SLRU (Least-Recently Used) per lo stato della transazione. ----sep----:xACTSLRU

Un processo è in attesa di accedere alla cache semplice utilizzata meno di recente (SLRU) per lo stato di una transazione.

LWLock:XactTruncation

Un processo è in attesa di eseguire pg_xact_status o di aggiornare l'ID di transazione meno recente disponibile.

LWLock:XidGen

Un processo è in attesa di allocare un nuovo ID di transazione.

LWLock:XidGenLock

Un processo è in attesa di allocare o assegnare un ID di transazione.

Timeout: BaseBackupThrottle

Un processo è in attesa durante il backup di base quando si limita l'attività.

Tempo scaduto: PgSleep

Un processo di backend ha chiamato la funzione pg_sleep ed è in attesa della scadenza del timeout di riposo. Per ulteriori informazioni, consulta Timeout: PG Sleep.

Tempo scaduto: RecoveryApplyDelay

Un processo è in attesa di applicare il WAL (write-ahead log) durante il ripristino a causa di un'impostazione di ritardo.

Tempo scaduto: RecoveryRetrieveRetryInterval

Un processo è in attesa durante il ripristino quando i dati WAL (write-ahead log) non sono disponibili da nessuna fonte (pg_wal, archivio o flusso).

Tempo scaduto: VacuumDelay

Un processo è in attesa in un punto di ritardo del vacuum basato sui costi.

Per un elenco completo degli eventi di attesa PostgreSQL, consulta la pagina relativa al processo di raccolta delle statistiche e alle tabelle degli eventi di attesa nella documentazione di PostgreSQL.

Aggiornamenti del motore di database Amazon Aurora PostgreSQL per 02/06/2020

Di seguito puoi trovare informazioni sulle versioni e sugli aggiornamenti della versione del motore di Amazon Aurora PostgreSQL. Puoi anche trovare informazioni su come aggiornare il tuo motore Aurora PostgreSQL. Per ulteriori informazioni generali sulle versioni di Aurora, consulta Versioni di Amazon Aurora.

> **Tip**
>
> È possibile ridurre al minimo i tempi di inattività necessari per l'aggiornamento di un cluster di database utilizzando un'implementazione blu/verde. Per ulteriori informazioni, consulta Utilizzo di Deployments Blue/Green per gli aggiornamenti del database.

Argomenti

- Identificazione delle versioni di Amazon Aurora PostgreSQL
- versioni di Amazon Aurora PostgreSQL e versioni del motore
- Versioni delle estensioni per Amazon Aurora PostgreSQL
- Aggiornamento dei cluster database Amazon Aurora PostgreSQL
- Utilizzo del rilascio con supporto a lungo termine (LTS) di Aurora PostgreSQL

Identificazione delle versioni di Amazon Aurora PostgreSQL

Amazon Aurora include alcune funzionalità generiche per Aurora e disponibili per tutti i cluster database Aurora. Aurora include altre funzionalità specifiche per un particolare motore del database supportato da Aurora. Queste caratteristiche sono disponibili solo per i cluster di database Aurora che utilizzano tale motore del database, ad esempio Aurora PostgreSQL.

Una versione del database Aurora in genere possiede due numeri di versione: il numero di versione del motore del motore del database e il numero di versione di Aurora. Se una versione di Aurora PostgreSQL ha un numero di versione Aurora, viene incluso dopo il numero di versione del motore nell'elenco versioni di Amazon Aurora PostgreSQL e versioni del motore.

Argomenti

- Numero versione Aurora

- Numeri di versione del motore PostgreSQL

Numero versione Aurora

Per i numeri di versione di Aurora viene utilizzato lo schema di denominazione *principale*.*secondario*.*patch*. Una versione della patch di Aurora include importanti correzioni di bug aggiunte a una versione secondaria dopo il rilascio. Per ulteriori informazioni sulle versioni principale, secondaria e patch di Amazon Aurora, consulta Versioni principali di Amazon Aurora, Versioni secondarie di Amazon Aurora e Versioni delle patch di Amazon Aurora.

Puoi trovare il numero di versione di Aurora della tua istanza database Aurora PostgreSQL con la seguente query SQL:

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

A partire dal rilascio di PostgreSQL versioni 13.3, 12.8, 11.13, 10.18 e per tutte le altre versioni successive, i numeri di versione di Aurora si allineano maggiormente alla versione del motore PostgreSQL. Ad esempio, l'esecuzione di query su un cluster database Aurora PostgreSQL 13.3 restituisce quanto segue:

```
aurora_version
----------------
 13.3.1
(1 row)
```

Le versioni precedenti, come il cluster database Aurora PostgreSQL 10.14, restituiscono numeri di versione simili ai seguenti:

```
aurora_version
----------------
 2.7.3
(1 row)
```

Numeri di versione del motore PostgreSQL

A partire dalla versione PostgreSQL 10, le versioni del motore del database PostgreSQL utilizzano uno schema di numerazione *principale*.*secondaria* per tutte le versioni. Alcuni esempi includono PostgreSQL 10.18, PostgreSQL 12.7 e PostgreSQL 13.3.

Le versioni precedenti alla versione PostgreSQL 10 utilizzano uno schema di numerazione *principale*.*principale*.*secondario* in cui le prime due cifre costituiscono il numero di versione principale e una terza cifra indica una versione secondaria. Ad esempio, PostgreSQL 9.6 è una versione principale, con le versioni secondarie 9.6.19 o 9.6.21 indicate dalla terza cifra.

> **Note**
>
> La versione del motore PostgreSQL 9.6 non è più supportata. Per eseguire l'aggiornamento, consulta Aggiornamento dei cluster database Amazon Aurora PostgreSQL. Per le policy delle versioni e le tempistiche delle versioni, consulta Per quanto tempo le versioni principali di Amazon Aurora rimangono disponibili.

Puoi trovare il numero di versione del motore del database PostgreSQL con la seguente query SQL:

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

Per un cluster database di Aurora PostgreSQL 13.3, i risultati sono i seguenti:

```
version
------------------------------------------------------------------------------------------------
 PostgreSQL 13.3 on x86_64-pc-linux-gnu, compiled by x86_64-pc-linux-gnu-gcc (GCC)
 7.4.0, 64-bit
(1 row)
```

versioni di Amazon Aurora PostgreSQL e versioni del motore

Versioni di Amazon Aurora edizione compatibile con PostgreSQL vengono aggiornate regolarmente. Gli aggiornamenti vengono applicati ai cluster di database di Aurora PostgreSQL durante le finestre di manutenzione del sistema. L'applicazione degli aggiornamenti dipende dal tipo di aggiornamento, dalla Regione AWS e dall'impostazione della finestra di manutenzione del cluster database. Molte delle versioni elencate includono sia un numero di versione PostgreSQL che un numero di versione Amazon Aurora. Tuttavia, a partire dal rilascio di PostgreSQL versioni 13.3, 12.8, 11.13, 10.18 e per tutte le altre versioni successive, i numeri di versione di Aurora non vengono utilizzati. Per identificare i numeri di versione del database Aurora PostgreSQL, consulta Identificazione delle versioni di Amazon Aurora PostgreSQL.

Per informazioni su estensioni e moduli, consulta Versioni delle estensioni per Amazon Aurora PostgreSQL.

> **Note**
>
> Per ulteriori informazioni sulle policy delle versioni Amazon Aurora e sulle tempistiche delle versioni, consulta Per quanto tempo le versioni principali di Amazon Aurora rimangono disponibili.

Per ulteriori informazioni sul supporto per Amazon Aurora, consulta Domande frequenti su Amazon RDS.

Per determinare quali versioni del motore del database Aurora PostgreSQL sono disponibili in una Regione AWS, utilizza il comando AWS CLI describe-db-engine-versions. Ad esempio:

```
aws rds describe-db-engine-versions --engine aurora-postgresql --query '*[].
[EngineVersion]' --output text --region aws-region
```

Per un elenco di Regioni AWS, consulta Disponibilità nelle regioni Aurora PostgreSQL.

Per informazioni dettagliate sulle versioni di PostgreSQL disponibili in Aurora PostgreSQL, consulta Release Notes for Aurora PostgreSQL.

Versioni delle estensioni per Amazon Aurora PostgreSQL

È possibile installare e configurare varie estensioni PostgreSQL da utilizzare con cluster di database Aurora PostgreSQL. Ad esempio, è possibile utilizzare l'estensione PostgreSQL `pg_partman` per automatizzare la creazione e la manutenzione delle partizioni di tabella. Per ulteriori informazioni su questa e altre estensioni disponibili per Aurora PostgreSQL, consulta Utilizzo di estensioni e wrapper di dati esterni.

Per informazioni dettagliate sulle versioni di PostgreSQL supportate su Aurora PostgreSQL, consultare Versioni delle estensioni di Amazon Aurora PostgreSQL nelle Note di rilascio di Aurora PostgreSQL.

Aggiornamento dei cluster database Amazon Aurora PostgreSQL

Amazon Aurora rende disponibili nuove versioni del motore di database PostgreSQL in Regioni AWS solo dopo test approfonditi. Puoi aggiornare i cluster database Aurora PostgreSQL alla nuova versione quando è disponibile nella tua regione.

A seconda della versione di Aurora PostgreSQL attualmente in esecuzione sul cluster database, un aggiornamento alla nuova versione è un aggiornamento secondario o principale. Ad esempio, l'aggiornamento di un cluster database Aurora PostgreSQL 11.15 ad Aurora PostgreSQL 13.6, è un aggiornamento della versione principale. L'aggiornamento di un cluster database Aurora PostgreSQL 13.3 ad Aurora PostgreSQL 13.7, è un aggiornamento della versione secondaria. Negli argomenti seguenti sono disponibili informazioni su come eseguire entrambi i tipi di aggiornamenti.

Indice

- Panoramica dei processi di aggiornamento di Aurora PostgreSQL
- Ottenere un elenco di versioni disponibili nel tuo Regione AWS
- Esecuzione di un aggiornamento alla versione principale
  - Test di un aggiornamento del cluster database di produzione a una nuova versione principale
  - Raccomandazioni successive all'aggiornamento
  - Aggiornamento del motore Aurora PostgreSQL a una nuova versione principale
    - Principali aggiornamenti per database globali
- Esecuzione di un aggiornamento a una versione secondaria
  - Operazioni preliminari all'aggiornamento a una versione secondaria
  - Come eseguire aggiornamenti della versione secondaria e applicare patch
  - Aggiornamenti della versione secondaria e applicazione di patch senza tempi di inattività
  - Limitazioni dell'applicazione di patch senza tempo di inattività
  - Aggiornamento del motore Aurora PostgreSQL a una nuova versione secondaria
- Aggiornamento estensioni PostgreSQL
- Tecnica alternativa di aggiornamento blu/verde

Panoramica dei processi di aggiornamento di Aurora PostgreSQL

Le differenze tra aggiornamenti della versione principale e secondaria sono le seguenti:

Aggiornamenti della versione secondaria e patch

> Aggiornamenti della versione secondaria e patch includono solo le modifiche compatibili con le versioni precedenti delle applicazioni esistenti. Gli aggiornamenti della versione secondaria e patch diventano disponibili solo dopo i test Aurora PostgreSQ e l'approvazione.

Aurora può applicare aggiornamenti della versione secondaria automaticamente. Quando crei un nuovo cluster di database Aurora PostgreSQL, l'opzione Abilita aggiornamento versione secondaria è abilitata in modo predefinito. A meno che questa opzione non venga disattivata manualmente, Aurora applica periodicamente aggiornamenti automatici della versione secondaria durante la finestra di manutenzione pianificata. Per ulteriori informazioni sull'opzione Aggiornamento automatico delle versioni minori (AmVU) e su come modificare il cluster database Aurora per utilizzarla, consulta Aggiornamenti automatici delle versioni secondarie per i cluster di database Aurora.

Se l'aggiornamento automatico della versione secondaria non è abilitato per il cluster di database Aurora PostgreSQL, Aurora PostgreSQL non viene automaticamente aggiornato a una nuova versione secondaria. Invece, quando una nuova versione secondaria viene rilasciata nella Regione AWS e il cluster database Aurora PostgreSQL sta eseguendo una versione secondaria precedente, Aurora richiede di eseguire l'aggiornamento. A questo scopo, viene aggiunto un suggerimento alle attività di manutenzione per il cluster.

Le patch non sono considerate un aggiornamento e non vengono applicate automaticamente. Aurora PostgreSQL richiede di applicare eventuali patch aggiungendo una raccomandazione alle attività di manutenzione per il cluster database Aurora PostgreSQL. Per ulteriori informazioni, consulta Come eseguire aggiornamenti della versione secondaria e applicare patch.

> **Note**
>
> Anche le patch che risolvono la sicurezza o altri problemi critici vengono aggiunte come attività di manutenzione. Tuttavia, queste patch sono obbligatorie. Assicurati di applicare le patch di sicurezza al cluster database Aurora PostgreSQL quando diventano disponibili nelle attività di manutenzione in sospeso.
> Gli aggiornamenti automatici delle versioni secondarie vengono eseguiti sulla versione secondaria predefinita.

Il processo di aggiornamento implica la possibilità di brevi interruzioni poiché ogni istanza nel cluster viene aggiornata alla nuova versione. Tuttavia, dopo Aurora PostgreSQL versioni 14.3.3, 13.7.3, 12.11.3, 11.16.3, 10.21.3 e altre versioni successive di queste versioni secondarie e delle versioni principali più recenti, il processo di aggiornamento utilizza la funzionalità di applicazione di patch senza tempi di inattività (ZDP). Questa funzionalità riduce al minimo le interruzioni e nella maggior parte dei casi le elimina completamente. Per ulteriori informazioni, consulta Aggiornamenti della versione secondaria e applicazione di patch senza tempi di inattività. Per

ulteriori informazioni sulle funzionalità e le limitazioni supportate di ZDP, consulta Limitazioni dell'applicazione di patch senza tempo di inattività.

Aggiornamenti a versioni principali

A differenza degli aggiornamenti della versione secondaria e delle patch, Aurora PostgreSQL non dispone di un'opzione di aggiornamento automatico della versione principale. Nuove versioni PostgreSQL principali potrebbero contenere modifiche al database non compatibili con le versioni precedenti delle applicazioni esistenti. La nuova funzionalità può causare l'interruzione del corretto funzionamento delle applicazioni esistenti.

Per evitare problemi, è fortemente consigliato seguire il processo descritto in Test di un aggiornamento del cluster database di produzione a una nuova versione principale prima di aggiornare le istanze database nei cluster database Aurora PostgreSQL. Assicurati innanzitutto che le applicazioni possano essere eseguite sulla nuova versione seguendo tale procedura. Quindi, puoi aggiornare manualmente il cluster database Aurora PostgreSQL alla nuova versione.

Il processo di aggiornamento implica la possibilità di brevi interruzioni poiché tutte le istanze nel cluster vengono aggiornate alla nuova versione. Anche il processo di pianificazione preliminare richiede tempo. Si consiglia di eseguire sempre attività di aggiornamento durante la finestra di manutenzione del cluster o quando le operazioni sono minime. Per ulteriori informazioni, consulta Esecuzione di un aggiornamento alla versione principale.

> **Note**
>
> Gli aggiornamenti della versione secondaria e quelli della versione principale potrebbero entrambi comportare brevi interruzioni. Per questo motivo, è fortemente consigliato eseguire o pianificare gli aggiornamenti durante la finestra di manutenzione o durante altri periodi di scarso utilizzo.

I cluster di database Aurora PostgreSQL richiedono occasionalmente gli aggiornamenti del sistema operativo. Questi aggiornamenti a volte includono una nuova versione della libreria glibc. Durante gli aggiornamenti, ti consigliamo di seguire le linee guida descritte in Regole di confronto supportate in Aurora PostgreSQL.

Ottenere un elenco di versioni disponibili nel tuo Regione AWS

È possibile ottenere un elenco di tutte le versioni del motore disponibili come obiettivi di aggiornamento per il cluster Aurora PostgreSQL DB eseguendo una query utilizzando il comando, come segue. Regione AWS describe-db-engine-versionsAWS CLI

Per Linux, macOS o Unix:

```
aws rds describe-db-engine-versions \
  --engine aurora-postgresql \
  --engine-version version-number \
  --query 'DBEngineVersions[*].ValidUpgradeTarget[*].{EngineVersion:EngineVersion}' \
  --output text
```

Per Windows:

```
aws rds describe-db-engine-versions ^
  --engine aurora-postgresql ^
  --engine-version version-number ^
  --query "DBEngineVersions[*].ValidUpgradeTarget[*].{EngineVersion:EngineVersion}" ^
  --output text
```

Ad esempio, per identificare gli obiettivi di aggiornamento validi per un cluster DB Aurora PostgreSQL versione 12.10, esegui il comando seguente: AWS CLI

Per Linux, macOS o Unix:

```
aws rds describe-db-engine-versions \
  --engine aurora-postgresql \
  --engine-version 12.10 \
  --query 'DBEngineVersions[*].ValidUpgradeTarget[*].{EngineVersion:EngineVersion}' \
  --output text
```

Per Windows:

```
aws rds describe-db-engine-versions ^
  --engine aurora-postgresql ^
  --engine-version 12.10 ^
  --query "DBEngineVersions[*].ValidUpgradeTarget[*].{EngineVersion:EngineVersion}" ^
  --output text
```

Nella tabella seguente sono disponibili le destinazioni degli aggiornamenti della versione principale e secondaria per differenti versioni di database Aurora PostgreSQL. Per mantenere la compatibilità, non tutte le versioni sono offerte come obiettivi di aggiornamento. Aurora PostgreSQL introduce nuove funzionalità e correzioni di bug con ogni versione trimestrale secondaria. Per ulteriori informazioni sulle versioni secondarie di Aurora PostgreSQL, consulta Note di rilascio Aurora PostgreSQL.

Versione di origine corrente	Destinazioni di aggiornamento
17.7	Nessuno
17,6	17,7
17,5	17,7, 17,6
17,4	17,7, 17,6, 17,5
16,11	Nessuno
16,10	17,7 16,11
16,9	17,7 16,11 , 17,6, 16,10, 17,5
16,8	17,7 16,11 , 17,6, 16,10, 17,5, 17,4, 16,9
16,6	17,7 16,11 , 17,6, 16,10, 17,5, 17,4, 16,9, 16,8
16,4	17,7 16,11 , 17,6, 16,10, 17,5, 17,4, 16,9, 16,8, 16,6
16,3	17,7

Versione di origine corrente	Destinazioni di aggiornamento
	16,11 , 17,6, 16,10, 17,5, 17,4, 16,9, 16,8, 16,6, 16,4
16,2	17,7 16,11 , 17,6, 16,10, 17,5, 17,4, 16,9, 16,8, 16,6, 16,4, 16,3
16,1	17,7 16,11 , 17,6, 16,10, 17,5, 17,4, 16,9, 16,8, 16,6, 16,4, 16,3, 16,2
15,15	Nessuno
15,14	17,7 16,11 15,15
15,13	17,7 16,11 , 15,15, 17,6, 16,10, 15,14, 17,5, 16,9
15,12	17,7 16,11 , 15,15, 17,6, 16,10, 15,14, 15,13
15,10	17,7 16,11 , 15,15, 17,6, 16,10, 15,14, 17,5, 17,4, 16,9, 16,8, 15,13
15,11	17,7 16,11 , 15,15, 17,6, 16,10, 15,14, 17,5, 17,4, 16,9, 16,8, 16,6, 15,13, 15,12
14,20	Nessuno

Per qualsiasi versione che stai considerando, controlla sempre la disponibilità della classe di istanza database del cluster. Ad esempio, `db.r4` non è supportato per Aurora PostgreSQL 13. Se il cluster Aurora PostgreSQL DB utilizza attualmente una classe di istanza db.r4, è necessario modificarla per utilizzare una classe di istanza DB supportata prima di poter eseguire l'aggiornamento ad Aurora PostgreSQL 13. Per ulteriori informazioni sulle classi di istanze del database Aurora PostgreSQL disponibili, incluse quelle basate su Graviton2 e quelle basate su Intel, consulta. Classi di istanze database Amazon Aurora

Esecuzione di un aggiornamento alla versione principale

Gli aggiornamenti a una versione principale potrebbero contenere modifiche al database non compatibili con le versioni precedenti del database. La nuova funzionalità in una nuova versione può causare l'interruzione del funzionamento corretto delle applicazioni esistenti. Per evitare problemi, Amazon Aurora non applica automaticamente aggiornamenti alla versione principale. Si consiglia, invece, di pianificare attentamente un aggiornamento alla versione principale seguendo questi passaggi:

1. Scegli la versione principale desiderata dall'elenco delle destinazioni disponibili tra quelle elencate per la versione nella tabella. Si ottiene un elenco preciso delle versioni disponibili in Regione AWS per la versione corrente utilizzando la AWS CLI. Per informazioni dettagliate, consultare Ottenere un elenco di versioni disponibili nel tuo Regione AWS.
2. Verifica che le applicazioni funzionino come previsto su un'implementazione di prova della nuova versione. Per informazioni sul processo completo, consulta Test di un aggiornamento del cluster database di produzione a una nuova versione principale.
3. Dopo aver verificato che le applicazioni funzionano come previsto nell'implementazione di prova, puoi aggiornare il cluster. Per informazioni dettagliate, consultare Aggiornamento del motore Aurora PostgreSQL a una nuova versione principale.

> **Note**
>
> È possibile eseguire un aggiornamento della versione principale da Babelfish per Aurora PostgreSQL dalle versioni di Aurora PostgreSQL 13 a partire dalla 13.6 alle versioni di Aurora PostgreSQL 14 a partire dalla 14.6. Babelfish per Aurora PostgreSQL 13.4 e 13.5 non supporta l'aggiornamento della versione principale.

Puoi ottenere l'elenco delle versioni del motore disponibili come destinazione di aggiornamento della versione principale per il cluster di database Aurora PostgreSQL eseguendo query sulla Regione AWS utilizzando il comando describe-db-engine-versions della AWS CLI, come segue.

Per Linux, macOS o Unix:

```
aws rds describe-db-engine-versions \
  --engine aurora-postgresql \
  --engine-version version-number \
  --query 'DBEngineVersions[].ValidUpgradeTarget[?IsMajorVersionUpgrade == `true`].
{EngineVersion:EngineVersion}' \
  --output text
```

Per Windows:

```
aws rds describe-db-engine-versions ^
  --engine aurora-postgresql ^
  --engine-version version-number ^
  --query "DBEngineVersions[].ValidUpgradeTarget[?IsMajorVersionUpgrade == `true`].
{EngineVersion:EngineVersion}" ^
  --output text
```

In alcuni casi, la versione di destinazione dell'aggiornamento non è una destinazione per la versione corrente. In questi casi, utilizza le informazioni contenute in versions table per eseguire aggiornamenti della versione secondaria fino a quando la versione del cluster non dispone della destinazione scelta nella relativa riga di destinazioni.

Test di un aggiornamento del cluster database di produzione a una nuova versione principale

Ogni nuova versione principale include miglioramenti all'ottimizzatore di query progettati per migliorare le prestazioni. Tuttavia, il carico di lavoro può includere query che restituiscono prestazioni peggiori nella nuova versione. Ecco perché ti consigliamo di testare e rivedere le prestazioni prima di eseguire l'aggiornamento in produzione. È possibile gestire la stabilità del piano di query tra le varie versioni utilizzando l'estensione Query Plan Management (QPM), come descritto in Garantire la stabilità del piano dopo un aggiornamento di versione principale.

Prima di aggiornare i cluster database Aurora PostgreSQL di produzione a una nuova versione principale, è fortemente consigliato eseguire il test dell'aggiornamento per verificare che tutte le applicazioni funzionino correttamente:

1. Tieni a portata di mano un gruppo di parametri compatibile con la versione.

   Se utilizzi un'istanza database personalizzata o un gruppo di parametri del cluster database, puoi scegliere tra due opzioni:

   a. Puoi specificare l'istanza database predefinita, il gruppo di parametri del cluster di database o entrambi per la nuova versione del motore di database.

   b. Oppure è possibile creare un gruppo di parametri personalizzato per la nuova versione del motore database.

2. Verifica la presenza di database non validi ed elimina quelli esistenti.

   La colonna `datconnlimit` nel catalogo `pg_database` include il valore `-2` per contrassegnare come non validi tutti i database che sono stati interrotti durante un'operazione `DROP DATABASE`. Utilizza la seguente query per verificare se esistono database non validi.

   ```
   SELECT
       datname
   FROM
       pg_database
   WHERE
       datconnlimit = - 2;
   ```

   Se la query restituisce nomi di database, questi database non sono validi. Utilizza l'istruzione `DROP DATABASE invalid_db_name` per eliminare i database non validi. Per eliminare tutti i database non validi, puoi utilizzare la seguente istruzione dinamica.

   ```
   SELECT
       'DROP DATABASE ' || quote_ident(datname) || ';'
   FROM
       pg_database
   WHERE
       datconnlimit = -2 \gexec
   ```

3. Controllare l'utilizzo non supportato:

   - Eseguire il commit o il rollback di tutte le transazioni preparate aperte prima di provare a eseguire un aggiornamento. È possibile utilizzare la seguente query per verificare che sull'istanza non siano presenti transazioni preparate aperte.

     ```
     SELECT count(*) FROM pg_catalog.pg_prepared_xacts;
     ```

- Rimuovere tutti gli utilizzi dei tipi di dati reg* prima di tentare un aggiornamento. Ad eccezione di `regtype` e `regclass`, non è possibile aggiornare i tipi di dati reg*. L'utilità pg_upgrade (utilizzata da Amazon Aurora per eseguire l'aggiornamento) non può preservare questo tipo di dati. Per ulteriori informazioni su questa utilità, consulta pg_upgrade nella documentazione di PostgreSQL.

  Per verificare che non siano presenti utilizzi di tipi di dati reg* non supportati, utilizzare la query seguente per ogni database.

```
SELECT count(*) FROM pg_catalog.pg_class c, pg_catalog.pg_namespace n,
 pg_catalog.pg_attribute a
  WHERE c.oid = a.attrelid
      AND NOT a.attisdropped
      AND a.atttypid IN ('pg_catalog.regproc'::pg_catalog.regtype,
                         'pg_catalog.regprocedure'::pg_catalog.regtype,
                         'pg_catalog.regoper'::pg_catalog.regtype,
                         'pg_catalog.regoperator'::pg_catalog.regtype,
                         'pg_catalog.regconfig'::pg_catalog.regtype,
                         'pg_catalog.regdictionary'::pg_catalog.regtype)
      AND c.relnamespace = n.oid
      AND n.nspname NOT IN ('pg_catalog', 'information_schema');
```

- Se stai eseguendo l'aggiornamento di un cluster di database Aurora PostgreSQL versione 10.18 o successive e l'estensione `pgRouting` è installata, rimuovila prima di eseguire l'aggiornamento alla versione 12.4 o successive.

  Se stai eseguendo l'aggiornamento di Aurora PostgreSQL 10.x che ha l'estensione `pg_repack` versione 1.4.3 installata, rimuovi l'estensione prima di eseguire l'aggiornamento a una versione successiva.

4. Controllare i database template1 e template0.

   Per un aggiornamento corretto, i database template1 e template0 devono esistere e devono essere elencati come modello. Per eseguire questa verifica, utilizzare il seguente comando:

```
SELECT datname, datistemplate FROM pg_database;

datname    | datistemplate
-----------+---------------
template0  | t
rdsadmin   | f
template1  | t
```

```
postgres   | f
```

Nell'output del comando, il valore `datistemplate` per i database template1 e template0 deve essere `t`.

5. Rimuovi slot di replica logica.

   Il processo di aggiornamento non può continuare se il cluster database Aurora PostgreSQL utilizza slot di replica logici. Gli slot di replica logica vengono generalmente utilizzati per attività di migrazione dei dati a breve termine, come la migrazione di dati utilizzando AWS DMS o per la replica di tabelle dal database a data lake, strumenti di business intelligence o altre destinazioni. Prima di eseguire l'aggiornamento, assicurati di conoscere lo scopo degli eventuali slot di replica logica esistenti e verifica che sia corretto eliminarli. Puoi verificare gli slot di replica logica utilizzando la seguente query:

   ```
   SELECT * FROM pg_replication_slots;
   ```

   Se gli slot di replica logica sono ancora in uso, non devono essere eliminati e non è possibile procedere con l'aggiornamento. Tuttavia, se gli slot di replica logica non sono necessari, è possibile eliminarli utilizzando il seguente SQL:

   ```
   SELECT pg_drop_replication_slot(slot_name);
   ```

   Negli scenari di replica logica che utilizzano l'estensione `pglogical` devono inoltre essere eliminati gli slot dal nodo publisher per un corretto aggiornamento della versione principale su quel nodo. Tuttavia, è possibile riavviare il processo di replica dal nodo sottoscrittore dopo l'aggiornamento. Per ulteriori informazioni, consulta Riconnessione della replica logica dopo un aggiornamento principale.

6. Eseguire un backup.

   Il processo di aggiornamento crea uno snapshot del cluster di database durante l'aggiornamento. Se desideri eseguire anche un backup manuale prima del processo di aggiornamento, consulta Creazione di uno snapshot del cluster database per ulteriori informazioni.

7. Aggiornare alcune estensioni alla versione più recente disponibile prima di eseguire l'aggiornamento della versione principale. Le estensioni da aggiornare includono le seguenti:

   - `pgRouting`
   - `postgis_raster`

- `postgis_tiger_geocoder`
- `postgis_topology`
- `address_standardizer`
- `address_standardizer_data_us`

Esegui il comando seguente per ogni estensione attualmente installata.

```
ALTER EXTENSION PostgreSQL-extension UPDATE TO 'new-version';
```

Per ulteriori informazioni, consulta Aggiornamento estensioni PostgreSQL. Per ulteriori informazioni sull'aggiornamento di PostGIS, consulta Passaggio 6: Aggiornamento dell'estensione PostGIS.

8. Se si esegue l'aggiornamento alla versione 11.x, eliminare le estensioni che non supporta prima di eseguire l'aggiornamento della versione principale. Le estensioni da eliminare includono:
   - `chkpass`
   - `tsearch2`
9. Eliminare i tipi di dati `unknown`, a seconda della versione di destinazione.

   La versione 10 di PostgreSQL non supporta il tipo di dati `unknown`. Se un database versione 9.6 utilizza il tipo di dati `unknown`, un aggiornamento alla versione 10 mostra un messaggio di errore del tipo seguente:

```
Database instance is in a state that cannot be upgraded: PreUpgrade checks failed:
The instance could not be upgraded because the 'unknown' data type is used in user
 tables.
Please remove all usages of the 'unknown' data type and try again."
```

   Per trovare il tipo di dati `unknown` nel database in modo da poter rimuovere tali colonne o modificarle in un tipo di dati supportato, utilizza il seguente codice SQL per ciascun database.

```
SELECT n.nspname, c.relname, a.attname
    FROM pg_catalog.pg_class c,
    pg_catalog.pg_namespace n,
    pg_catalog.pg_attribute a
    WHERE c.oid = a.attrelid AND NOT a.attisdropped AND
    a.atttypid = 'pg_catalog.unknown'::pg_catalog.regtype AND
    c.relkind IN ('r','m','c') AND
```

```
    c.relnamespace = n.oid AND
    n.nspname !~ '^pg_temp_' AND
    n.nspname !~ '^pg_toast_temp_' AND n.nspname NOT IN ('pg_catalog',
'information_schema');
```

10 Esegui un aggiornamento di prova.

Si consiglia di testare un aggiornamento della versione principale su un duplicato del database di produzione prima di eseguire l'aggiornamento sul database di produzione. È possibile monitorare i piani di esecuzione sull'istanza di test duplicata per eventuali regressioni del piano di esecuzione e valutarne le prestazioni. Per creare un'istanza di test duplicata, è possibile ripristinare il proprio database da uno snapshot recente o clonare il database. Per ulteriori informazioni, consulta Ripristino da uno snapshot o Clonazione di un volume per un cluster di database Amazon Aurora.

Per ulteriori informazioni, consulta Aggiornamento del motore Aurora PostgreSQL a una nuova versione principale.

11 Aggiornare la propria istanza di produzione.

Dopo aver testato correttamente l'aggiornamento a una versione principale, dovrebbe essere possibile aggiornare il database di produzione senza problemi. Per ulteriori informazioni, consulta Aggiornamento del motore Aurora PostgreSQL a una nuova versione principale.

> **Note**
>
> Durante il processo di aggiornamento, Aurora PostgreSQL accetta uno snapshot del cluster di database. Durante questo processo, non puoi eseguire un ripristino point-in-time del cluster. In seguito, puoi effettuare un ripristino point-in-time a momenti precedenti all'inizio dell'aggiornamento e successivi allo snapshot automatico dell'istanza. Tuttavia, non puoi eseguire un ripristino point-in-time a una versione secondaria precedente.

Per informazioni su un aggiornamento in corso, è possibile utilizzare Amazon RDS per visualizzare due log prodotti dall'utilità. Questi sono `pg_upgrade_internal.log` e `pg_upgrade_server.log`. Amazon Aurora RDS accoda un timestamp al nome file per questi log. Puoi visualizzare questi log come qualsiasi altro log. Per ulteriori informazioni, consulta Monitoraggio dei file di log di Amazon Aurora.

12 Aggiornare le estensioni PostgreSQL. Un processo di aggiornamento PostgreSQL non aggiorna alcuna estensione PostgreSQL. Per ulteriori informazioni, consulta Aggiornamento estensioni PostgreSQL.

Raccomandazioni successive all'aggiornamento

Dopo aver completato un aggiornamento della versione principale, è consigliabile quanto segue:

- Eseguire l'operazione `ANALYZE` per aggiornare la tabella `pg_statistic`. È necessario farlo per ogni database su tutte le istanze database di PostgreSQL. Le statistiche di ottimizzazione non vengono trasferite durante un aggiornamento della versione principale, quindi è necessario rigenerare tutte le statistiche per evitare problemi di prestazioni. Esegui il comando senza parametri per generare statistiche per tutte le tabelle regolari del database corrente, come segue:

  ```
  ANALYZE VERBOSE;
  ```

  Il flag `VERBOSE` è facoltativo, ma usandolo viene mostrato lo stato di avanzamento. Per ulteriori informazioni, consulta ANALYZE nella documentazione di PostgreSQL.

  Quando analizzi tabelle specifiche invece di utilizzare ANALYZE VERBOSE, esegui il comando ANALYZE per ogni tabella come segue:

  ```
  ANALYZE table_name;
  ```

  Per le tabelle partizionate, analizza sempre la tabella principale. Il seguente processo:

  - Campiona automaticamente le righe in tutte le partizioni
  - Aggiorna le statistiche per ogni partizione in modo ricorsivo
  - Mantiene le statistiche di pianificazione essenziali a livello principale

  Sebbene le tabelle principale non memorizzino dati effettivi, la loro analisi è fondamentale per l'ottimizzazione delle query. L'esecuzione di ANALYZE solo su singole partizioni può portare a prestazioni di query scadenti poiché lo strumento di ottimizzazione non disporrà delle statistiche complete necessarie per una pianificazione efficiente tra le partizioni.

  > **Note**
  >
  > Esegui ANALYZE sul tuo sistema dopo l'aggiornamento per evitare problemi di prestazioni.

- Se hai eseguito l'aggiornamento a PostgreSQL versione 10, esegui `REINDEX` su qualsiasi indice hash di cui disponi. Gli indici hash sono stati modificati nella versione 10 e devono essere ricostruiti. Per individuare indici hash non validi, eseguire il seguente SQL per ogni database che contiene indici hash.

```
SELECT idx.indrelid::regclass AS table_name,
   idx.indexrelid::regclass AS index_name
FROM pg_catalog.pg_index idx
   JOIN pg_catalog.pg_class cls ON cls.oid = idx.indexrelid
   JOIN pg_catalog.pg_am am ON am.oid = cls.relam
WHERE am.amname = 'hash'
AND NOT idx.indisvalid;
```

- Ti consigliamo di testare l'applicazione sul database aggiornato con un carico di lavoro analogo per verificare che tutto funzioni come previsto. Dopo la verifica dell'aggiornamento è possibile eliminare l'istanza di test.

Aggiornamento del motore Aurora PostgreSQL a una nuova versione principale

Quando avvii il processo di aggiornamento a una nuova versione principale, Aurora PostgreSQL acquisisce uno snapshot del cluster database Aurora prima di apportare eventuali modifiche al cluster. Questo snapshot viene creato solo per gli aggiornamenti della versione principale, non per gli aggiornamenti della versione secondaria. Al termine del processo di aggiornamento, questo snapshot è disponibile tra gli snapshot manuali elencati in Snapshot nella console RDS. Il nome dello snapshot include `preupgrade` come prefisso, il nome del cluster database Aurora PostgreSQL, la versione di origine, la versione di destinazione e la data e il timestamp, come illustrato nell'esempio seguente.

```
preupgrade-docs-lab-apg-global-db-12-8-to-13-6-2022-05-19-00-19
```

Al termine dell'aggiornamento, puoi utilizzare lo snapshot creato e archiviato da Aurora nell'elenco di snapshot manuali per ripristinare il cluster database alla versione precedente, se necessario.

> **Tip**
>
> In generale, gli snapshot forniscono molti modi per ripristinare il cluster database Aurora in momenti diversi. Per ulteriori informazioni, consultare Ripristino da uno snapshot cluster database e Ripristino di un cluster di database a un determinato momento. Tuttavia, Aurora

PostgreSQL non supporta l'utilizzo di uno snapshot per il ripristino a una versione secondaria precedente.

Durante il processo di aggiornamento della versione principale, Aurora alloca un volume e clona il cluster database Aurora PostgreSQL di origine. Se l'aggiornamento non va a buon fine per qualsiasi motivo, Aurora PostgreSQL utilizza il clone per eseguire il rollback dell'aggiornamento. Quando vengono allocati più di 15 cloni di un volume di origine, i cloni successivi diventano copie complete e richiedono più tempo. Di conseguenza anche il processo di aggiornamento richiede più tempo. Se Aurora PostgreSQL esegue il rollback dell'aggiornamento, tenere presente quanto segue:

- È possibile che vengano visualizzate voci di fatturazione e parametri per il volume originale e per il volume clonato allocati durante l'aggiornamento. Aurora PostgreSQL elimina i dati sul volume aggiuntivo quando la finestra di conservazione del backup del cluster supera il termine dell'aggiornamento.
- La successiva copia snapshot tra regioni da questo cluster sarà una copia completa anziché una copia incrementale.

Per aggiornare in modo sicuro le istanze database che compongono il cluster, Aurora PostgreSQL utilizza l'utilità pg_upgrade. Al termine dell'aggiornamento dell'istanza di scrittura, ogni istanza di lettura riscontra una breve interruzione mentre viene aggiornato alla nuova versione principale. Per ulteriori informazioni su questa utilità, consulta pg_upgrade nella documentazione di PostgreSQL.

Puoi aggiornare il cluster database Aurora PostgreSQL a una nuova versione secondaria utilizzando la Console di gestione AWS, la AWS CLI o l'API RDS.

Console

Modificare la versione del motore di un cluster database

1. Accedi alla Console di gestione AWS e apri la console Amazon RDS all'indirizzo https://console.aws.amazon.com/rds/.
2. Nel riquadro di navigazione, scegliere Databases (Database) quindi selezionare il cluster di database da aggiornare.
3. Scegliere Modify (Modifica). Viene visualizzata la pagina Modify DB cluster (Modifica cluster di database).
4. In Engine version (Versione motore) scegliere la nuova versione.

5. Scegliere Continue (Continua) e controllare il riepilogo delle modifiche.
6. Per applicare immediatamente le modifiche, scegliere Apply immediately (Applica immediatamente). In alcuni casi, la chiusura di questa opzione può causare un'interruzione. Per ulteriori informazioni, consulta Modifica di un cluster database Amazon Aurora.
7. Nella pagina di conferma esaminare le modifiche. Se sono corrette, selezionare Modify Cluster (Modifica cluster) per salvare le modifiche.

   Oppure scegliere Back (Indietro) per cambiare le modifiche o Cancel (Annulla) per annullare le modifiche.

AWS CLI

Per aggiornare la versione del motore di un cluster database, utilizza il comando modify-db-cluster della AWS CLI. Specifica i seguenti parametri:

- `--db-cluster-identifier`: il nome del cluster di database.
- `--engine-version` – Numero di versione del motore di database a cui effettuare l'aggiornamento. Per informazioni sulle versioni motore valide, utilizza il comando AWS CLI describe-db-engine-versions.
- `--allow-major-version-upgrade` – Un flag obbligatorio quando il parametro `--engine-version` è una versione principale diversa rispetto alla versione principale corrente del cluster database.
- `--no-apply-immediately` – Applica le modifiche durante la finestra di manutenzione successiva. Per applicare immediatamente le modifiche utilizzare `--apply-immediately`.

Example

Per Linux, macOS o Unix:

```
aws rds modify-db-cluster \
    --db-cluster-identifier mydbcluster \
    --engine-version new_version \
    --allow-major-version-upgrade \
    --no-apply-immediately
```

Per Windows:

```
aws rds modify-db-cluster ^
```

```
    --db-cluster-identifier mydbcluster ^
    --engine-version new_version ^
    --allow-major-version-upgrade ^
    --no-apply-immediately
```

API RDS

Per aggiornare la versione del motore di un cluster di database, utilizza l'operazione ModifyDBCluster. Specifica i seguenti parametri:

- `DBClusterIdentifier` – Nome del cluster database, ad esempio *`mydbcluster`*.
- `EngineVersion` – Numero di versione del motore di database a cui effettuare l'aggiornamento. Per informazioni sulle versioni motore valide, utilizza l'operazione DescribeDBEngineVersions.
- `AllowMajorVersionUpgrade` – Un flag obbligatorio quando il parametro `EngineVersion` è una versione principale diversa rispetto alla versione principale corrente del cluster database.
- `ApplyImmediately` – Indica se applicare le modifiche immediatamente o durante la finestra di manutenzione successiva. Per applicare le modifiche immediatamente, imposta il valore su `true`. Per applicare le modifiche durante la finestra di manutenzione successiva imposta il valore su `false`.

Principali aggiornamenti per database globali

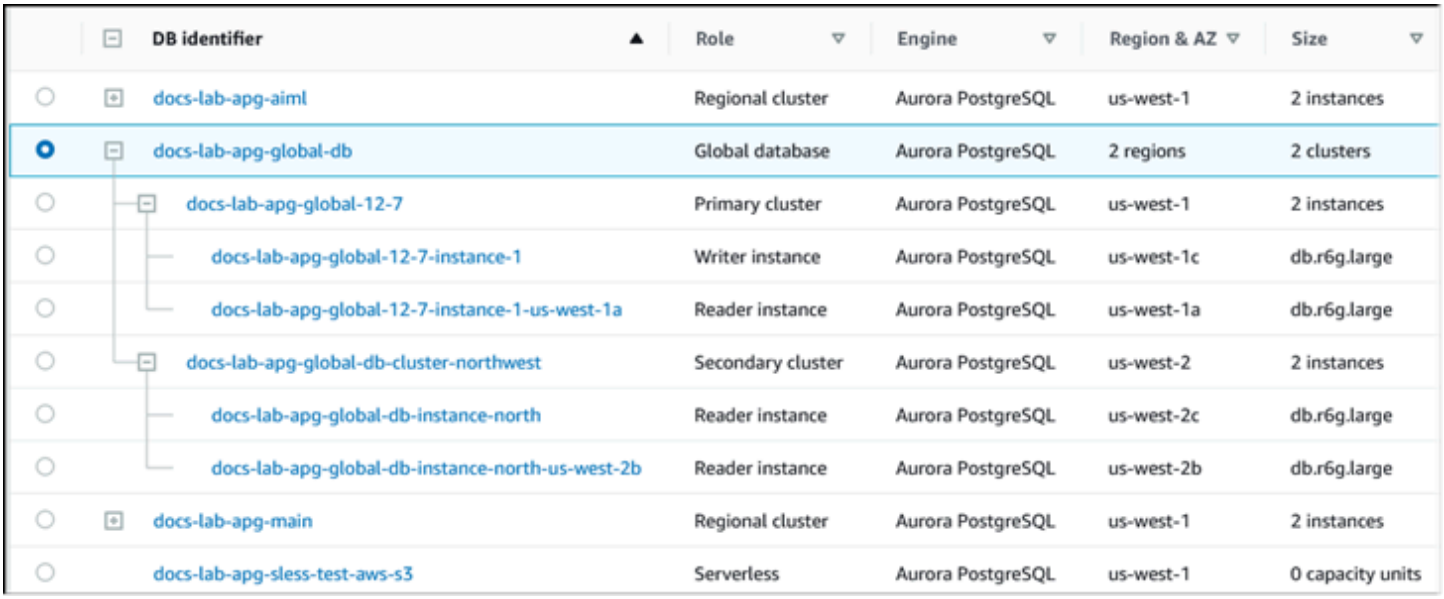
Per un cluster database globale Aurora, il processo di aggiornamento aggiorna tutti i cluster database che compongono contemporaneamente il database globale Aurora. Esegue questa operazione per garantire che ognuno esegua la stessa versione di Aurora PostgreSQL. Inoltre, assicura che le eventuali modifiche apportate alle tabelle di sistema, ai formati di file di dati e così via vengono replicate automaticamente in tutti i cluster secondari.

Per aggiornare un cluster database globale a una nuova versione principale di Aurora PostgreSQL, si consiglia di eseguire il test delle applicazioni nella versione aggiornata, come descritto in Test di un aggiornamento del cluster database di produzione a una nuova versione principale. Assicurati di preparare le impostazioni del gruppo di parametri del cluster database e del gruppo di parametri database per ogni Regione AWS nel database globale Aurora prima dell'aggiornamento come descritto in step 1. di Test di un aggiornamento del cluster database di produzione a una nuova versione principale.

Se il cluster database globale Aurora PostgreSQL dispone di un Obiettivo del punto di ripristino (RPO) impostato per il parametro `rds.global_db_rpo`, assicurati di ripristinare il parametro prima

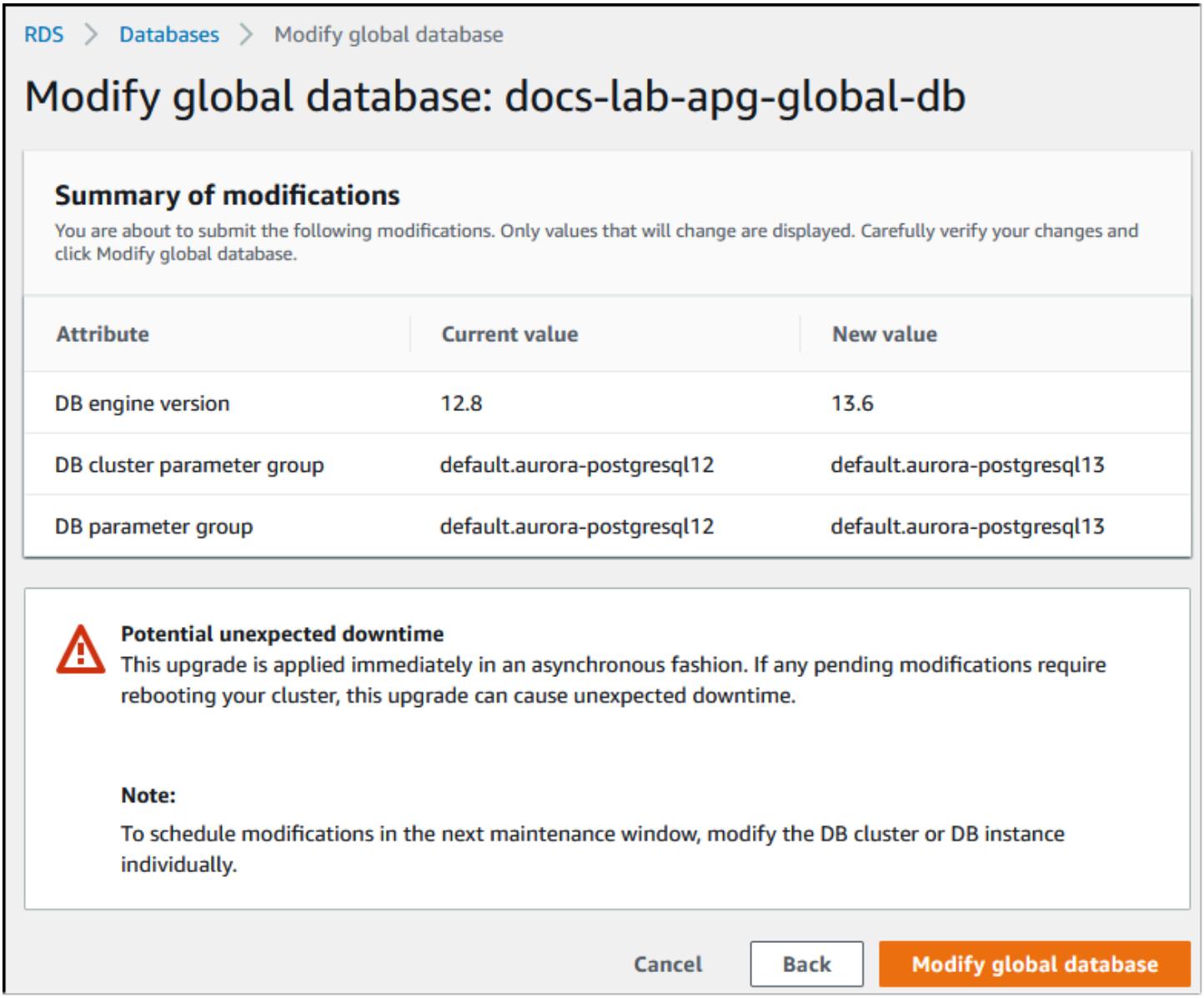
dell'aggiornamento. Il processo di aggiornamento della versione principale non funziona se l'RPO è attivato. Per impostazione predefinita, questo parametro è disattivato. Per ulteriori informazioni su database globali Aurora PostgreSQL e RPO, consulta Gestione degli RPO per database globali basati su Aurora PostgreSQL–.

Se verifichi che le applicazioni possano essere eseguite come previsto durante l'implementazione di prova della nuova versione, puoi avviare il processo di aggiornamento. A questo proposito, consulta Aggiornamento del motore Aurora PostgreSQL a una nuova versione principale. Assicurati di scegliere la voce di primo livello dall'elenco Databases (Database) nella console RDS, Database globale, come mostrato nell'immagine seguente.



DB identifier	Role	Engine	Region & AZ	Size
docs-lab-apg-aiml	Regional cluster	Aurora PostgreSQL	us-west-1	2 instances
docs-lab-apg-global-db	Global database	Aurora PostgreSQL	2 regions	2 clusters
docs-lab-apg-global-12-7	Primary cluster	Aurora PostgreSQL	us-west-1	2 instances
docs-lab-apg-global-12-7-instance-1	Writer instance	Aurora PostgreSQL	us-west-1c	db.r6g.large
docs-lab-apg-global-12-7-instance-1-us-west-1a	Reader instance	Aurora PostgreSQL	us-west-1a	db.r6g.large
docs-lab-apg-global-db-cluster-northwest	Secondary cluster	Aurora PostgreSQL	us-west-2	2 instances
docs-lab-apg-global-db-instance-north	Reader instance	Aurora PostgreSQL	us-west-2c	db.r6g.large
docs-lab-apg-global-db-instance-north-us-west-2b	Reader instance	Aurora PostgreSQL	us-west-2b	db.r6g.large
docs-lab-apg-main	Regional cluster	Aurora PostgreSQL	us-west-1	2 instances
docs-lab-apg-sless-test-aws-s3	Serverless	Aurora PostgreSQL	us-west-1	0 capacity units

Come per qualsiasi modifica, puoi confermare che desideri che il processo proceda quando richiesto.



Invece di utilizzare la console, puoi avviare il processo di aggiornamento utilizzando la AWS CLI o l'API RDS. Come per la console, si opera sul cluster database globale Aurora anziché su uno qualsiasi dei suoi componenti, come segue:

- Utilizza il comando modify-global-cluster della AWS CLI per avviare l'aggiornamento per il database globale Aurora utilizzando la AWS CLI.
- Utilizza l'API ModifyGlobalCluster per avviare l'aggiornamento.

Esecuzione di un aggiornamento a una versione secondaria

Puoi utilizzare i seguenti metodi per aggiornare la versione secondaria di un cluster di database o per applicare le patch a un cluster di database:

Argomenti

- Operazioni preliminari all'aggiornamento a una versione secondaria
- Come eseguire aggiornamenti della versione secondaria e applicare patch
- Aggiornamenti della versione secondaria e applicazione di patch senza tempi di inattività
- Limitazioni dell'applicazione di patch senza tempo di inattività
- Aggiornamento del motore Aurora PostgreSQL a una nuova versione secondaria

Operazioni preliminari all'aggiornamento a una versione secondaria

Consigliamo di eseguire le seguenti azioni per ridurre il tempo di inattività durante un aggiornamento di una versione secondaria:

- La manutenzione del cluster di database Aurora deve essere eseguita durante un periodo di traffico ridotto. Utilizza Approfondimenti sulle prestazioni per identificare questi periodi di tempo al fine di configurare correttamente le finestre di manutenzione. Per ulteriori informazioni su Approfondimenti sulle prestazioni, consulta Monitoraggio del carico DB con Performance Insights su Amazon RDS. Per ulteriori informazioni sulla finestra di manutenzione dei cluster di database, Impostazione della finestra di manutenzione preferita del cluster database.
- Utilizza gli SDK AWS che supportano il backoff esponenziale e il jitter come best practice. Per ulteriori informazioni, consulta Backoff esponenziale e jitter.

Come eseguire aggiornamenti della versione secondaria e applicare patch

Aggiornamenti della versione secondaria e patch diventano disponibili in Regioni AWS solo dopo test rigorosi. Prima di rilasciare aggiornamenti e patch, Aurora PostgreSQL verifica per garantire che problemi di sicurezza noti, bug e altri problemi che emergono dopo il rilascio della versione Community secondaria non compromettano la stabilità del parco istanze Aurora PostgreSQL.

Se si attiva Abilita aggiornamento automatico versione secondaria, Aurora PostgreSQL effettua periodicamente aggiornamenti automatici di versioni secondarie, il cluster di database viene aggiornato durante la finestra di manutenzione specificata. Assicurati che l'opzione Abilita

aggiornamento automatico versione secondaria sia attivata per tutte le istanze nel cluster di database Aurora PostgreSQL. Per informazioni su come impostare Aggiornamento automatico della versione secondaria e su come funziona l'impostazione quando viene applicata a livello di cluster e istanza, consulta Aggiornamenti automatici delle versioni secondarie per i cluster di database Aurora.

Verifica il valore dell'opzione Abilita aggiornamento automatico versione secondaria per tutti i cluster di database Aurora PostgreSQL utilizzando il comando describe-db-instances AWS CLI.

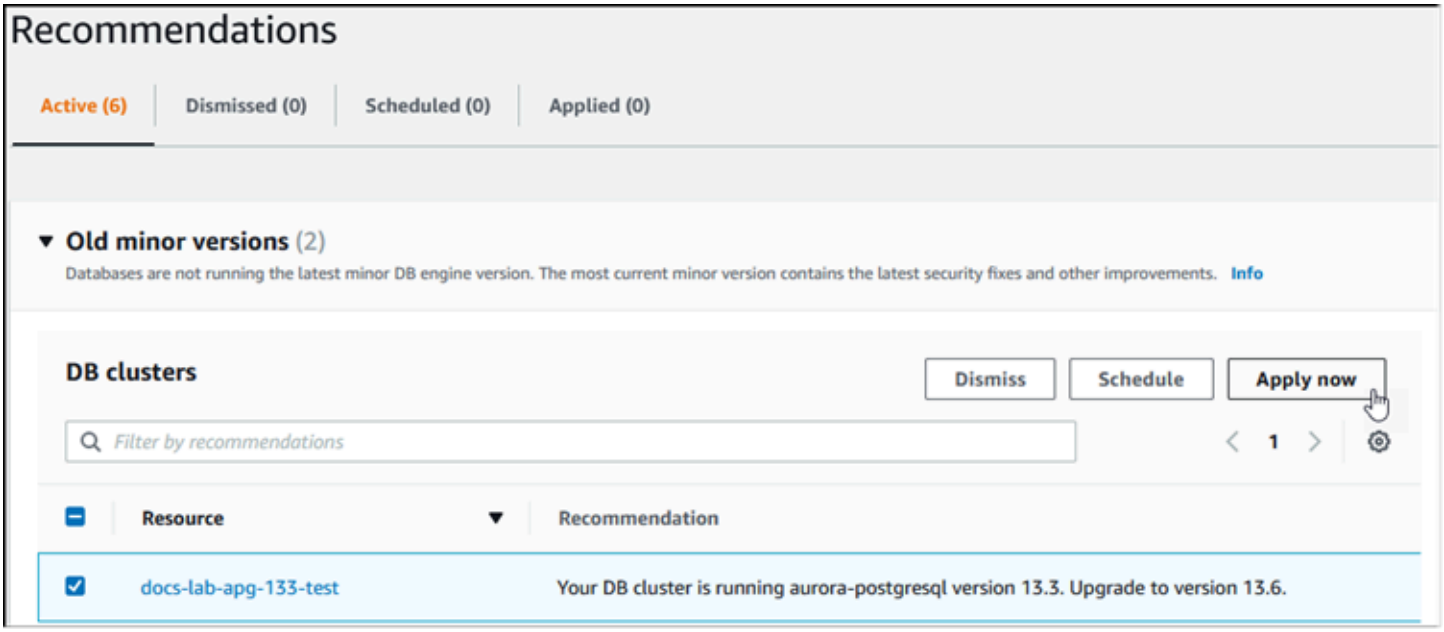
```
aws rds describe-db-instances \
  --query '*[].
{DBClusterIdentifier:DBClusterIdentifier,DBInstanceIdentifier:DBInstanceIdentifier,AutoMinorVer
```

Questa query restituisce un elenco di tutti i cluster database Aurora e delle relative istanze con un valore `true` o `false` per lo stato dell'impostazione `AutoMinorVersionUpgrade`. Il comando presuppone che la AWS CLI sia configurata per indirizzare la Regione AWS predefinita.

Per ulteriori informazioni sull'opzione Aggiornamento automatico della versione secondaria (AmVU) e su come modificare il cluster database Aurora per utilizzarla, consulta Aggiornamenti automatici delle versioni secondarie per i cluster di database Aurora.

Puoi aggiornare i cluster database Aurora PostgreSQL alle nuove versioni secondarie rispondendo alle attività di manutenzione o modificando il cluster per utilizzare la nuova versione.

Puoi identificare eventuali aggiornamenti o patch disponibili per i cluster database Aurora PostgreSQL utilizzando la console RDS e aprendo il menu Recommendations (Raccomandazioni). Qui è disponibile un elenco dei diversi problemi di manutenzione come Old minor versions (Versioni secondarie precedenti). A seconda dell'ambiente di produzione, puoi scegliere di pianificare l'aggiornamento o eseguire un'azione immediata, scegliendo Apply now (Applica ora) come mostrato di seguito.



Per ulteriori informazioni su come gestire un cluster database Aurora, incluso come applicare manualmente le patch e gli aggiornamenti della versione secondaria, consulta Manutenzione di un cluster di database Amazon Aurora.

Aggiornamenti della versione secondaria e applicazione di patch senza tempi di inattività

L'aggiornamento di un cluster database Aurora PostgreSQL comporta la possibilità di un'interruzione. Durante il processo di aggiornamento, il database viene arrestato. Se si avvia l'aggiornamento mentre il database è occupato, si perdono tutte le connessioni e le transazioni elaborate dal cluster database. Se si aspetta che il database sia inattivo per eseguire l'aggiornamento, potrebbe essere necessario attendere a lungo.

La funzione ZDP (applicazione delle patch senza tempi di inattività) migliora il processo di aggiornamento. Grazie a ZDP, è possibile applicare aggiornamenti della versione secondaria e patch con un impatto minimo sul cluster database Aurora PostgreSQL. ZDP viene utilizzato quando si applicano patch o aggiornamenti di versioni secondarie più recenti a versioni di Aurora PostgreSQL e altre versioni successive di queste versioni secondarie e le versioni principali più recenti. Ovvero, l'aggiornamento a nuove versioni secondarie da una qualsiasi di queste versioni in poi utilizza ZDP.

La tabella seguente mostra le versioni di Aurora PostgreSQL e le classi di istanze database in cui è disponibile ZDP:

Versione	Classi di istanza db.r*	Classi di istanza db.t*	Classi di istanza db.x*	Classe di istanza db.serverless
10.21 e versioni successive	Sì	Sì	Sì	N/D
11.16 e versioni successive	Sì	Sì	Sì	N/D
11.17 e versioni successive	Sì	Sì	Sì	N/D
12.11 e versioni successive	Sì	Sì	Sì	N/D
12.12 e versioni successive	Sì	Sì	Sì	N/D
13.7 e versioni successive	Sì	Sì	Sì	N/D
13.8 e versioni successive	Sì	Sì	Sì	Sì
14.3 e versioni successive	Sì	Sì	Sì	N/D
14.4 e versioni successive	Sì	Sì	Sì	N/D
14.5 e versioni successive	Sì	Sì	Sì	Sì
15.3 e versioni successive	Sì	Sì	Sì	Sì
16.1 e versioni successive	Sì	Sì	Sì	Sì

Durante il processo di aggiornamento tramite l'applicazione di patch senza tempi di inattività, il motore di database cerca un punto idoneo per sospendere tutte le nuove transazioni. Questa azione protegge il database durante le patch e gli aggiornamenti. Per assicurarsi che le applicazioni funzionino senza problemi con transazioni sospese, è consigliabile integrare la logica dell'esecuzione di nuovi tentativi nel codice. Questo approccio garantisce che il sistema sia in grado di gestire qualsiasi breve periodo di inattività senza errori e di riprovare le nuove transazioni dopo l'aggiornamento.

Quando l'applicazione di patch senza tempi di inattività è completata, le sessioni dell'applicazione vengono conservate, ad eccezione di quelle con connessioni eliminate; il motore di database viene riavviato mentre l'aggiornamento è ancora in corso. Il riavvio del motore di database può causare una riduzione temporanea della velocità di trasmissione effettiva, che dura in genere alcuni secondi o al massimo circa un minuto.

In alcuni casi, l'applicazione di patch senza tempi di inattività (ZDP) potrebbe non avere esito positivo. Ad esempio, le modifiche ai parametri che si trovano nello stato `pending` sul cluster di database Aurora PostgreSQL o le sue istanze possono interferire con l'applicazione di patch senza tempi di inattività.

Puoi trovare parametri ed eventi per le operazioni ZDP nella pagina Events (Eventi) nella console. Gli eventi includono l'avvio dell'aggiornamento ZDP e il completamento dell'aggiornamento. In questo caso puoi individuare il tempo richiesto dal processo e il numero di connessioni conservate e interrotte che si sono verificate durante il riavvio. Puoi individuare i dettagli nel registro degli errori del database.

Limitazioni dell’applicazione di patch senza tempo di inattività

Le seguenti limitazioni si applicano all’applicazione di patch senza tempi di inattività:

- ZDP cerca di preservare le connessioni client correnti all’istanza database di scrittura Aurora PostgreSQL durante il processo di aggiornamento di Aurora PostgreSQL. Tuttavia, nei seguenti casi, le connessioni verranno interrotte per l'applicazione di patch senza tempi di inattività:
  - Sono in esecuzione query o transazioni di lunga durata.
  - Sono in esecuzione istruzioni DDL (Data Definition Language).
  - Si utilizzando blocchi di tabelle o tabelle temporanee.
  - Vengono ascoltate tutte le sessioni sui canali di notifica.
  - È in uso un cursore nello stato "WITH HOLD".

  - Si utilizzano le connessioni TLSv1.1. A partire dalle versioni 16.1, 15.3, 14.8, 13.11, 12.15 e 11.20 di Aurora PostgreSQL, ZDP è supportato con connessioni TLSv1.3.
- ZDP non è supportato nei seguenti casi:
  - Quando i cluster database Aurora PostgreSQL sono configurati come Aurora Serverless v1.
  - Durante l'aggiornamento di qualsiasi istanza di lettura Aurora.
  - Durante l'aggiornamento di qualsiasi istanza di lettura Aurora che fa parte di un cluster di database globale Aurora in una Regione secondaria.
  - Durante i patch e gli aggiornamenti del sistema operativo.

Aggiornamento del motore Aurora PostgreSQL a una nuova versione secondaria

Puoi aggiornare il cluster database Aurora PostgreSQL a una nuova versione secondaria utilizzando la console, la AWS CLI o l'API RDS. Prima di eseguire l'aggiornamento, si consiglia di seguire le stesse best practice consigliate per gli aggiornamenti della versione principale. Come per le nuove versioni principali, anche le nuove versioni secondarie possono contenere miglioramenti all'ottimizzatore, ad esempio correzioni, che possono causare regressioni del piano di query. Per garantire la stabilità del piano, si consiglia di utilizzare l'estensione Query Plan Management (QPM) come descritto in Garantire la stabilità del piano dopo un aggiornamento di versione principale.

Console

Aggiornare la versione del motore del cluster database Aurora PostgreSQL

1. Accedi alla Console di gestione AWS e apri la console Amazon RDS all'indirizzo https://console.aws.amazon.com/rds/.
2. Nel riquadro di navigazione, scegliere Databases (Database) quindi selezionare il cluster di database da aggiornare.
3. Scegliere Modify (Modifica). Viene visualizzata la pagina Modify DB cluster (Modifica cluster di database).
4. In Engine version (Versione motore) scegliere la nuova versione.
5. Scegliere Continue (Continua) e controllare il riepilogo delle modifiche.
6. Per applicare immediatamente le modifiche, scegliere Apply immediately (Applica immediatamente). In alcuni casi, la chiusura di questa opzione può causare un'interruzione. Per ulteriori informazioni, consulta Modifica di un cluster database Amazon Aurora.

7. Nella pagina di conferma esaminare le modifiche. Se sono corrette, selezionare Modify Cluster (Modifica cluster) per salvare le modifiche.

   Oppure scegliere Back (Indietro) per cambiare le modifiche o Cancel (Annulla) per annullare le modifiche.

AWS CLI

Per aggiornare la versione del motore di un cluster database, utilizza il comando modify-db-cluster della AWS CLI con i seguenti parametri:

- `--db-cluster-identifier` – Nome del cluster database Aurora PostgreSQL.
- `--engine-version` – Numero di versione del motore di database a cui effettuare l'aggiornamento. Per informazioni sulle versioni motore valide, utilizza il comando AWS CLI describe-db-engine-versions.
- `--no-apply-immediately` – Applica le modifiche durante la finestra di manutenzione successiva. Per applicare immediatamente le modifiche, utilizza invece `--apply-immediately`.

Per Linux, macOS o Unix:

```
aws rds modify-db-cluster \
    --db-cluster-identifier mydbcluster \
    --engine-version new_version \
    --no-apply-immediately
```

Per Windows:

```
aws rds modify-db-cluster ^
    --db-cluster-identifier mydbcluster ^
    --engine-version new_version ^
    --no-apply-immediately
```

API RDS

Per aggiornare la versione del motore di un cluster di database, utilizza l'operazione ModifyDBCluster. Specifica i seguenti parametri:

- `DBClusterIdentifier` – Nome del cluster database, ad esempio *mydbcluster*.

- `EngineVersion` – Numero di versione del motore di database a cui effettuare l'aggiornamento. Per informazioni sulle versioni motore valide, utilizza l'operazione DescribeDBEngineVersions.
- `ApplyImmediately` – Indica se applicare le modifiche immediatamente o durante la finestra di manutenzione successiva. Per applicare le modifiche immediatamente, imposta il valore su `true`. Per applicare le modifiche durante la finestra di manutenzione successiva imposta il valore su `false`.

Aggiornamento estensioni PostgreSQL

L'aggiornamento del cluster di database Aurora PostgreSQL a una nuova versione principale o secondaria non aggiorna contemporaneamente le estensioni PostgreSQL. Per la maggior parte delle estensioni, l'estensione viene aggiornata dopo il completamento dell'aggiornamento della versione principale o secondaria. Tuttavia, in alcuni casi, l'estensione viene aggiornata prima di aggiornare il motore di database Aurora PostgreSQL. Per ulteriori informazioni, consulta list of extensions to update in Test di un aggiornamento del cluster database di produzione a una nuova versione principale.

L'installazione delle estensioni PostgreSQL richiede privilegi `rds_superuser`. In genere, un `rds_superuser` delega le autorizzazioni su estensioni specifiche agli utenti (ruoli) pertinenti, per facilitare la gestione di una determinata estensione. Ciò significa che il compito di aggiornare tutte le estensioni nel cluster database Aurora PostgreSQL potrebbe coinvolgere molti utenti (ruoli) diversi. Tenerlo presente se si desidera automatizzare il processo di aggiornamento utilizzando gli script. Per ulteriori informazioni sui privilegi e sui ruoli PostgreSQL, consulta Sicurezza con Amazon Aurora PostgreSQL.

> **Note**
>
> Per informazioni sull'aggiornamento dell'estensione PostGIS, consulta Gestione dei dati spaziali con estensione PostGIS (Passaggio 6: Aggiornamento dell'estensione PostGIS). Per aggiornare l'estensione `pg_repack`, rimuovi l'estensione e quindi crea la nuova versione nell'istanza database aggiornata. Per ulteriori informazioni, consulta pg_repack installation nella documentazione `pg_repack`.

Per aggiornare un'estensione dopo un aggiornamento del motore, utilizza il comando `ALTER EXTENSION UPDATE`.

```
ALTER EXTENSION extension_name UPDATE TO 'new_version';
```

Per elencare le estensioni attualmente installate, usa il catalogo PostgreSQL pg_extension nel seguente comando.

```
SELECT * FROM pg_extension;
```

Per visualizzare l'elenco delle versioni delle estensioni specifiche disponibili per l'installazione, utilizza la vista PostgreSQL pg_available_extension_versions nel seguente comando.

```
SELECT * FROM pg_available_extension_versions;
```

Tecnica alternativa di aggiornamento blu/verde

In alcune situazioni, la priorità principale è eseguire il passaggio immediato dal cluster precedente a quello aggiornato. In tali situazioni, è possibile utilizzare un processo in più fasi che esegue i cluster vecchi e nuovi. side-by-side Qui, i dati vengono replicati dal cluster precedente a quello nuovo fino a quando il nuovo cluster non prende il controllo. Per informazioni dettagliate, vedi Utilizzo di Deployments Blue/Green per gli aggiornamenti del database.

Utilizzo del rilascio con supporto a lungo termine (LTS) di Aurora PostgreSQL

Ogni nuova versione di Aurora PostgreSQL resta disponibile per l'utilizzo per un certo periodo di tempo quando crei o aggiorni un cluster di database. Trascorso questo periodo, devi aggiornare i cluster che utilizzano questa versione. Puoi aggiornare manualmente il cluster prima della scadenza del periodo di supporto oppure lasciare che Aurora lo faccia al posto tuo quando la versione di Aurora PostgreSQL non è più supportata.

Aurora indica alcune versioni di Aurora PostgreSQL come versioni con supporto a lungo termine (LTS). I cluster di database che utilizzano le versioni LTS possono restare più a lungo con la stessa versione ed essere sottoposti a un numero minore di cicli di aggiornamento rispetto ai cluster che utilizzano versioni non LTS. Le versioni minori LTS includono solo correzioni di bug (tramite versioni patch) per problemi critici di stabilità e sicurezza; una versione LTS non include nuove funzionalità rilasciate dopo la sua introduzione.

Una volta all'anno, ai cluster di database in esecuzione su una versione secondaria LTS viene assegnata una patch con versione più recente della versione LTS. Questa applicazione della patch

viene eseguita per garantire che si tragga vantaggio dalle correzioni cumulative per la sicurezza e la stabilità. Potremmo applicare patch a una versione secondaria di LTS più frequentemente nel caso in cui fossero presenti correzioni critiche, ad esempio per la sicurezza, che devono essere applicate.

> **Note**
>
> Per rimanere su una versione secondaria LTS per tutta la durata del suo ciclo di vita, assicurarsi di disabilitare l'aggiornamento automatico della versione secondaria per le istanze database. Per evitare di aggiornare automaticamente il cluster di database dalla versione secondaria LTS, deselezionare la casella di controllo Abilita aggiornamento automatico versione secondaria per tutte le istanze database nel cluster Aurora.

Per la maggior parte dei cluster Aurora PostgreSQL, consigliamo di eseguire l'aggiornamento all'ultima versione invece di utilizzare la versione LTS. In questo modo, puoi sfruttare Aurora anche come servizio gestito e puoi accedere alle ultime funzionalità e correzioni di bug. Le versioni LTS sono destinate a cluster con le seguenti caratteristiche:

- L'applicazione Aurora PostgreSQL non può avere tempi di inattività dovuti agli aggiornamenti, ad eccezione di rari casi per la distribuzione di patch critiche.
- Il ciclo di test per il cluster e le applicazioni associate richiede molto tempo per ogni aggiornamento al motore del database di Aurora PostgreSQL.
- La versione del database per il cluster Aurora PostgreSQL ha tutte le funzionalità del motore del database e le correzioni di bug necessarie per l'applicazione.

Le versioni LTS correnti per Aurora PostgreSQL sono le seguenti:

- PostgreSQL 16.8. È stato rilasciato il 07 aprile 2025. Per ulteriori informazioni, vedere PostgreSQL 16.8 nelle Note di rilascio per Aurora PostgreSQL.
- PostgreSQL 15.10. È stato rilasciato il 27 dicembre 2024. Per ulteriori informazioni, consultare PostgreSQL 15.10 nelle Note di rilascio di Aurora PostgreSQL.
- PostgreSQL 14.6. È stato rilasciato il 20 gennaio 2023. Per ulteriori informazioni, consulta PostgreSQL 14.6 nelle Note di rilascio di Aurora PostgreSQL.
- PostgreSQL 13.9. È stato rilasciato il 20 gennaio 2023. Per ulteriori informazioni, consulta PostgreSQL 13.9 nelle Note di rilascio di Aurora PostgreSQL.

- PostgreSQL 12.9. È stato rilasciato il 25 febbraio 2022. Per ulteriori informazioni, consultare PostgreSQL 12.9 nelle Notedi rilascio di Aurora PostgreSQL.
- PostgreSQL 11.9 (Aurora PostgreSQL versione 3.4). È stato rilasciato l'11 dicembre 2020. Per ulteriori informazioni su questa versione, consultare PostgreSQL 11.9, Aurora PostgreSQL versione 3.4 nelle Note di rilascio di Aurora PostgreSQL.

Per i dettagli sui tempi di assistenza e sui cicli di rilascio per le versioni LTS, consulta Release calendars for Aurora PostgreSQL.

Per informazioni su come identificare le versioni di Aurora e del motore del database, consulta Identificazione delle versioni di Amazon Aurora PostgreSQL.

Utilizzo di Amazon Aurora PostgreSQL Limitless Database

Amazon Aurora PostgreSQL Limitless Database offre un dimensionamento orizzontale automatizzato per elaborare milioni di transazioni di scrittura al secondo e gestisce petabyte di dati mantenendo la semplicità di funzionamento all'interno di un singolo database. Con Aurora PostgreSQL Limitless Database, è possibile dedicarsi alla creazione di applicazioni su larga scala senza dover creare e mantenere soluzioni complesse per scalare i dati su più istanze database per supportare i carichi di lavoro.

Argomenti

- Architettura di Aurora PostgreSQL Limitless Database
- Nozioni di base su Aurora PostgreSQL Limitless Database
- Requisiti e considerazioni su Aurora PostgreSQL Limitless Database
- Prerequisiti per l'utilizzo di Aurora PostgreSQL Limitless Database
- Creazione di un cluster di database che utilizza Aurora PostgreSQL Limitless Database
- Utilizzo di gruppi di shard di database
- Creazione di tabelle di Aurora PostgreSQL Limitless Database
- Caricamento dei dati in Aurora PostgreSQL Limitless Database
- Esecuzione di query su Aurora PostgreSQL Limitless Database
- Gestione di Aurora PostgreSQL Limitless Database
- Monitoraggio di Aurora PostgreSQL Limitless Database
- Backup e ripristino di Aurora PostgreSQL Limitless Database
- Aggiornamento di Amazon Aurora PostgreSQL Limitless Database
- Riferimento ad Aurora PostgreSQL Limitless Database

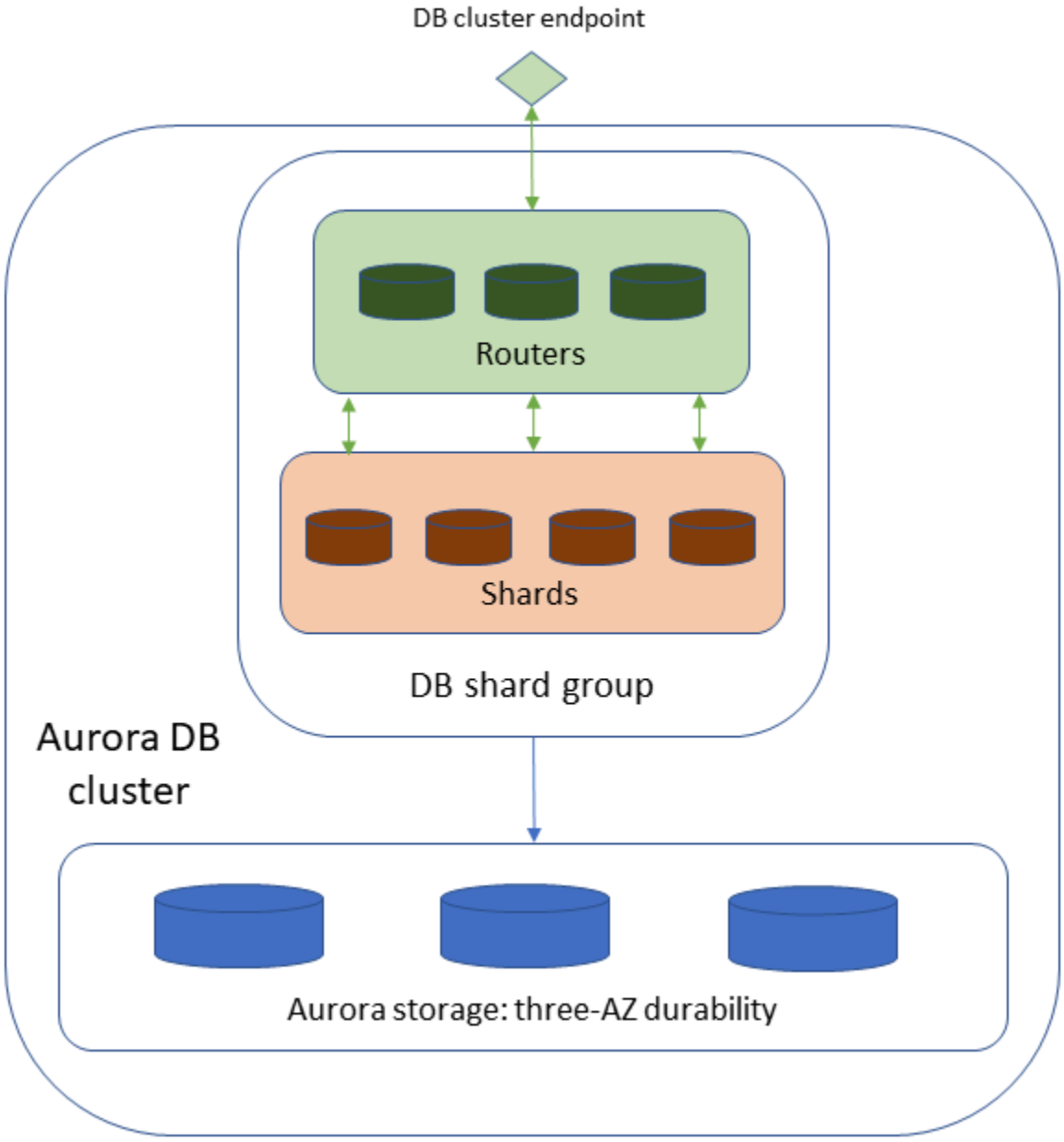
Architettura di Aurora PostgreSQL Limitless Database

Limitless Database raggiunge la scalabilità con un'architettura a due livelli composta da più nodi di database. I nodi sono router o shard.

- Gli shard sono istanze database Aurora PostgreSQL, ognuno dei quali archivia un sottoinsieme di dati per il database, consentendo l'elaborazione simultanea per ottenere un throughput di scrittura più elevato.
- I router gestiscono la natura distribuita del database e presentano un'unica immagine del database ai client del database. I router conservano i metadati relativi alla posizione in cui sono archiviati i dati, analizzano i comandi SQL in entrata e li inviano agli shard. Successivamente aggregano i dati dagli shard per restituire un singolo risultato al cliente e gestiscono le transazioni distribuite per mantenere la coerenza nell'intero database distribuito.

Aurora PostgreSQL Limitless Database differisce dai Cluster di database Aurora standard perché dispone di un gruppo di shard del database invece di un'istanza database di scrittura e istanze database di lettura. Tutti i nodi che compongono l'architettura di Limitless Database sono contenuti nel gruppo di shard del database. Gli shard e i router individuali nel gruppo di shard del database non sono visibili nell'Account AWS. Per accedere a Limitless Database, utilizzi l'endpoint del cluster di database.

La figura seguente mostra l'architettura di alto livello di Aurora PostgreSQL Limitless Database.



Per ulteriori informazioni sull'architettura di Aurora PostgreSQL Limitless Database e su come utilizzarla, consulta Raggiungere la scalabilità con Amazon Aurora PostgreSQL Limitless Database sul canale Eventi AWS su YouTube.

Per ulteriori informazioni sull'architettura di un cluster di database Aurora standard, consulta Cluster database Amazon Aurora.

Termini chiave per Aurora PostgreSQL Limitless Database

Gruppo di shard del database

Un container per i nodi di Limitless Database (shard e router).

Router

Un nodo che accetta connessioni SQL dai client, invia comandi SQL agli shard, mantiene la coerenza a livello di sistema e restituisce i risultati ai client.

Shard

Un nodo che archivia un sottoinsieme di tabelle sottoposte a sharding, copie complete di tabelle di riferimento e tabelle standard. Accetta query dai router, ma non può essere connesso direttamente dai client.

Tabella sottoposta a sharding

Una tabella i cui dati sono partizionati tra gli shard.

Chiave di shard

Una colonna o un insieme di colonne in una tabella sottoposta a sharding utilizzata per determinare la partizione tra gli shard.

Tabelle co-localizzate

Due tabelle sottoposte a sharding che condividono la stessa chiave di shard e sono dichiarate esplicitamente come co-localizzate. Tutti i dati con lo stesso valore della chiave di shard vengono inviati allo stesso shard.

Tabella di riferimento

Una tabella i cui dati sono copiati completamente su ogni shard.

Tabella standard

Il tipo di tabella predefinito in Limitless Database. Puoi convertire le tabelle standard in tabelle sottoposte a sharding e tabelle di riferimento.

Tutte le tabelle standard sono archiviate sullo stesso shard selezionato dal sistema, consentendo i collegamenti tra tabelle standard all'interno di un singolo shard. Tuttavia, le tabelle standard sono limitate dalla capacità massima dello shard (128 TiB). Questo shard archivia anche dati di tabelle sottoposte a sharding e di riferimento, quindi il limite effettivo per le tabelle standard è inferiore a 128 TiB.

Tipi di tabelle per Aurora PostgreSQL Limitless Database

Aurora PostgreSQL Limitless Database supporta tre tipi di tabella: sottoposta a sharding, di riferimento e standard.

Le tabelle sottoposte a sharding distribuiscono i dati tra tutti gli shard nel gruppo di shard del database. Limitless Database esegue questa operazione automaticamente utilizzando una chiave di shard, che è una colonna specificata o un set di colonne specificato durante il partizionamento della tabella. Tutti i dati con lo stesso valore per la chiave di shard vengono inviati allo stesso shard. Lo sharding è basato su hash, non su intervalli o elenchi.

I seguenti sono casi d'uso consigliati per le tabelle sottoposte a sharding:

- L'applicazione lavora con un sottoinsieme distinto di dati.
- La tabella è molto grande.
- La tabella cresce potenzialmente più velocemente rispetto ad altre tabelle.

Le tabelle sottoposte a sharding possono essere co-localizzate, vale a dire che condividono la stessa chiave di shard, in modo che tutti i dati di entrambe le tabelle con lo stesso valore della chiave di shard siano inviati allo stesso shard. Se co-localizzi le tabelle e le unisci utilizzando la chiave di shard, il collegamento può essere eseguito su un singolo shard perché tutti i dati necessari sono presenti su quello shard.

Le tabelle di riferimento hanno una copia completa di tutti i relativi dati su ogni shard nel gruppo di shard del database. Le tabelle di riferimento sono comunemente utilizzate per tabelle più piccole con basso volume di scrittura, ma che devono essere frequentemente collegate e non si prestano allo sharding. Esempi di tabelle di riferimento includono tabelle di date e tabelle di dati geografici come stato, città e codice postale.

Le tabelle standard sono il tipo di tabella predefinito in Aurora PostgreSQL Limitless Database. Non sono tabelle distribuite. Aurora PostgreSQL Limitless Database supporta i collegamenti tra tabelle standard e tabelle standard, sottoposte a sharding e di riferimento.

Fatturazione per Aurora PostgreSQL Limitless Database

Per informazioni su come vengono addebitati i costi per Aurora PostgreSQL Limitless Database, consulta Fatturazione delle istanze database per Aurora.

Per informazioni sui prezzi di Aurora, consulta la pagina dei prezzi di Aurora.

Nozioni di base su Aurora PostgreSQL Limitless Database

Esegui le seguenti operazioni per iniziare a usare Aurora PostgreSQL Limitless Database:

1. Crea un cluster di database Aurora PostgreSQL e un gruppo di shard del database per Limitless Database. Per ulteriori informazioni, consulta Creazione di un cluster di database che utilizza Aurora PostgreSQL Limitless Database.
2. Crea tabelle sottoposte a sharding e di riferimento nel gruppo di shard del database. Per ulteriori informazioni, consulta Creazione di tabelle di Aurora PostgreSQL Limitless Database.
3. Modifica la capacità del gruppo di shard del database, dividi gli shard e aggiungi router. Per ulteriori informazioni, consulta Utilizzo di gruppi di shard di database.
4. Carica i dati nel gruppo di shard del database. Per ulteriori informazioni, consulta Caricamento dei dati in Aurora PostgreSQL Limitless Database.
5. Esegui query e altre istruzioni SQL sul gruppo di shard del database. Per ulteriori informazioni, consulta Esecuzione di query su Aurora PostgreSQL Limitless Database.
6. Monitora le prestazioni di Limitless Database. Per ulteriori informazioni, consulta Monitoraggio di Aurora PostgreSQL Limitless Database.

Requisiti e considerazioni su Aurora PostgreSQL Limitless Database

Ad Aurora PostgreSQL Limitless Database sono associati i requisiti e le considerazioni seguenti.

Argomenti

- Requisiti per Aurora PostgreSQL Limitless Database:
- Considerazioni per Aurora PostgreSQL Limitless Database
- Funzionalità non supportate in Aurora PostgreSQL Limitless Database

Requisiti per Aurora PostgreSQL Limitless Database:

Assicurati di soddisfare questi requisiti per Aurora PostgreSQL Limitless Database.

- Il database Aurora PostgreSQL Limitless è disponibile in tutti i paesi ad eccezione dell'Asia Pacifico (Taipei). Regioni AWS

  > **Note**
  >
  > Se crei il cluster di database Aurora PostgreSQL Limitless Database nella Regione Stati Uniti orientali (Virginia settentrionale), non includere la zona di disponibilità (AZ) `us-east-1e` nel gruppo di sottoreti di database. A causa delle limitazioni delle risorse, Aurora Serverless v2 (e quindi Aurora PostgreSQL Limitless Database) non è supportato nella AZ `us-east-1e`.

- Aurora PostgreSQL Limitless Database supporta solo la configurazione di archiviazione del cluster di database Aurora I/O-Optimized. Per ulteriori informazioni, consulta Configurazioni dell'archiviazione per i cluster database Amazon Aurora.
- Aurora PostgreSQL Limitless Database utilizza versioni speciali del motore di database Aurora PostgreSQL per Aurora PostgreSQL Limitless Database:
  - `16.6.8-limitless`
  - `16.6-limitless`
  - `16.4-limitless`
- Il cluster di database non può avere istanze database di tipo scrittura o lettura.

- È necessario utilizzare il monitoraggio avanzato e Approfondimenti sulle prestazioni. Il periodo di conservazione di Approfondimenti sulle prestazioni deve essere di almeno 1 mese (31 giorni).
- Devi esportare il log PostgreSQL in Amazon Logs. CloudWatch

> **Note**
>
> Alcune funzionalità richieste, come Enhanced Monitoring, Performance Insights e CloudWatch Logs, sono soggette a costi aggiuntivi. Per informazioni sui prezzi di Aurora, consulta la pagina dei prezzi di Aurora.

Considerazioni per Aurora PostgreSQL Limitless Database

Le seguenti considerazioni si applicano ai gruppi di shard di database in Aurora PostgreSQL Limitless Database:

- È consentito un solo gruppo di shard di database per cluster di database.
- È possibile avere fino a cinque gruppi di shard DB per gruppo. Regione AWS

  Pertanto, puoi avere fino a cinque cluster Aurora PostgreSQL Limitless Database DB per ogni cluster. Regione AWS Per ulteriori informazioni, consulta Quote in Amazon Aurora.
- È possibile impostare la capacità massima di un gruppo di shard DB su 16-6144. ACUs Per limiti di capacità superiori a 6144, contattare. ACUs AWS

  Il numero iniziale di router e shard è determinato dalla capacità massima impostata al momento della creazione del gruppo di shard di database. Per ulteriori informazioni, consulta Correlazione della capacità massima del gruppo di shard del database con il numero di router e shard creati.
- Se modifichi la capacità massima di un gruppo di shard di database, il numero di router e shard non cambia.
- Assicurati che la sottorete del database in cui crei il gruppo di shard di database disponga di un numero sufficiente di indirizzi IP liberi per la connessione con il gruppo stesso. È necessario un indirizzo IP per ogni router e fino a tre indirizzi IP per ogni shard del gruppo di shard di database.

  Per ulteriori informazioni sul numero di router creati durante la creazione di un gruppo di shard di database, consulta Correlazione della capacità massima del gruppo di shard del database con il numero di router e shard creati.

- Se rendi il tuo gruppo di shard di database accessibile al pubblico, assicurati di configurare un gateway Internet nel tuo VPC.
- Utilizza le funzioni SQL per dividere gli shard e aggiungere router.
- L'unione degli shard non è supportata.
- Non è possibile eliminare router e shard singoli.
- Non è possibile modificare (eseguire operazioni `UPDATE` su) le chiavi di shard in alcun modo, inclusa la modifica dei relativi valori nelle righe delle tabelle.

  Per modificare una chiave di shard, eliminala e quindi ricreala.
- Sono supportati i livelli di isolamento Repeatable Read, Read Committed e Read Uncommitted. Non è possibile impostare il livello di isolamento su Serializzabile.
- Alcuni comandi SQL non sono supportati. Per ulteriori informazioni, consulta Riferimento ad Aurora PostgreSQL Limitless Database.
- Non tutte le estensioni PostgreSQL sono supportate. Per ulteriori informazioni, consulta Estensioni.
- Quando si crea un gruppo di shard o quando si aggiungono nuovi nodi di un gruppo di shard (shard o router), tali nodi vengono creati in una delle zone di disponibilità (AZs) disponibili per il cluster DB. Non puoi scegliere una AZ specifica per i singoli nodi.
- Se utilizzi una ridondanza di elaborazione pari a 2 (due standby di calcolo per il gruppo di shard DB), assicurati che il tuo gruppo di sottoreti DB ne abbia almeno tre. AZs
- Aurora PostgreSQL Limitless Database supporta fino a 54 caratteri per i nomi di tabelle sottoposti a sharding.

Le seguenti considerazioni si applicano al cluster di database Aurora PostgreSQL Limitless Database:

- Ti consigliamo di utilizzare policy AWS gestite per limitare le autorizzazioni per il database e le applicazioni a quelle di cui i clienti hanno bisogno per i loro casi d'uso. Per ulteriori informazioni, consulta Best practice delle policy.
- Quando crei il cluster di database Aurora PostgreSQL Limitless Database, imposti solo i parametri di dimensionamento per il gruppo di shard di database.
- Per eliminare il tuo cluster di database, devi prima eliminare il gruppo di shard di database.
- Aurora PostgreSQL Limitless Database non può essere un'origine di replica.

Funzionalità non supportate in Aurora PostgreSQL Limitless Database

Le seguenti funzionalità di Aurora PostgreSQL non sono supportate in Aurora PostgreSQL Limitless Database:

- Autenticazione Active Directory (Kerberos)
- Amazon DevOps Guru
- Amazon ElastiCache
- Implementazioni Amazon RDS Blue/Green
- Server proxy per Amazon RDS
- Aurora Auto Scaling (aggiunta automatica di istanze di lettura al cluster di database)
- Aurora Global Database
- Aurora machine learning
- Consigli su Aurora
- Aurora Serverless v1
- Integrazioni Zero-ETL di Aurora
- AWS Backup
- AWS Lambda integrazione
- Gestione dei segreti AWS
- Babelfish per Aurora PostgreSQL
- Clonazione di cluster di database
- Endpoint personalizzati
- Flussi di attività di database
- Repliche di lettura
- API dati in RDS

Prerequisiti per l’utilizzo di Aurora PostgreSQL Limitless Database

Per utilizzare Aurora PostgreSQL Limitless Database, è necessario innanzitutto eseguire le seguenti attività.

Argomenti

- Abilitazione delle operazioni dei gruppi di shard del database

Abilitazione delle operazioni dei gruppi di shard del database

Prima di poter creare un gruppo di shard del database, è necessario abilitare le operazioni del gruppo di shard del database.

- Aggiungere la seguente sezione alla policy IAM del ruolo IAM dell’utente che accede ad Aurora PostgreSQL Limitless Database:

  JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Sid": "AllowDBShardGroup",
            "Effect": "Allow",
            "Action": [
                "rds:CreateDBShardGroup",
                "rds:DescribeDBShardGroups",
                "rds:DeleteDBShardGroup",
                "rds:ModifyDBShardGroup",
                "rds:RebootDBShardGroup"
            ],
            "Resource": [
                "arn:aws:rds:*:*:shard-group:*",
                "arn:aws:rds:*:*:cluster:*"
            ]
        }
    ]
}
```

Creazione di un cluster di database che utilizza Aurora PostgreSQL Limitless Database

Crea un nuovo cluster di database Aurora utilizzando la versione Limitless Database di Aurora PostgreSQL e aggiungi un gruppo di shard del database al cluster. Quando aggiungi un gruppo di shard del database, specifichi la capacità massima di calcolo per l'intero gruppo di shard del database (somma delle capacità di tutti i router e gli shard) in unità di capacità Aurora (ACU). Ogni ACU è una combinazione di circa 2 gigabyte (GiB) di memoria, CPU corrispondente e rete. Il dimensionamento aumenta o diminuisce la capacità del gruppo di shard del database, a seconda del carico di lavoro dell'applicazione, in modo analogo al funzionamento di Aurora Serverless v2.

Argomenti

- Correlazione della capacità massima del gruppo di shard del database con il numero di router e shard creati
- Creazione di un cluster di database

Correlazione della capacità massima del gruppo di shard del database con il numero di router e shard creati

Il numero iniziale di router e shard in un gruppo di shard del database è determinato dalla capacità massima impostata al momento della creazione del gruppo di shard del database. Maggiore è la capacità massima, maggiore sarà il numero di router e shard creati nel gruppo di shard del database.

Ogni nodo (shard o router) ha il proprio valore di capacità corrente, anch'esso misurato in ACU.

- Limitless Database scala la capacità di un nodo quando la sua capacità corrente è troppo bassa per gestire il carico. Tuttavia, i nodi non aumentano verticalmente quando la capacità totale è al massimo.
- Limitless Database riduce verticalmente la capacità di un nodo quando la sua capacità corrente è superiore al necessario. Tuttavia, i nodi non riducono verticalmente la capacità quando la capacità totale è al minimo.

La tabella seguente mostra la correlazione tra la capacità massima del gruppo di shard del database in unità di capacità Aurora (ACU) e il numero di nodi (router e shard) creati.

> **Note**
>
> Questi valori sono soggetti a modifiche.
> Se imposti la ridondanza di calcolo su un valore diverso da zero, il numero totale di shard verrà raddoppiato o triplicato. Ciò comporterà costi aggiuntivi.
> I nodi negli standby di calcolo vengono aumentati o ridotti verticalmente fino alla stessa capacità dell'istanza di scrittura. L'intervallo di capacità non viene impostato separatamente per gli standby.

Totale nodi	Router	Partizioni	Capacità minima predefinita (ACU)	Intervallo di capacità massima (ACU)
4	2	2	16	16-400
5	2	3	20	401-500
6	2	4	24	501-600
7	3	4	28	601-700
8	3	5	32	701-800
9	3	6	36	801-900
10	4	6	40	901-1.000
11	4	7	44	1.001-1.100
12	4	8	48	1.101-1.200
13	5	8	52	1.201-1.300
14	5	9	56	1.301-1.400
15	5	10	60	1.401-1.500
16	6	10	64	1.501-1.600

Totale nodi	Router	Partizioni	Capacità minima predefinita (ACU)	Intervallo di capacità massima (ACU)
17	6	11	68	1.601-1.700
18	6	12	72	1.701-1.800
19	7	12	76	1.801-1.900
20	7	13	80	1.901-2.000
21	7	14	84	2.001-2.100
22	8	14	88	2.101-2.200
23	8	15	92	2.201-2.300
24	8	16	96	2.301-6.144

La configurazione dinamica basata sulla capacità massima per il gruppo di shard del database è disponibile solo durante la creazione. Il numero di router e shard rimane invariato quando la capacità massima viene modificata. Per ulteriori informazioni, consulta Modifica della capacità di un gruppo di shard del database.

Puoi utilizzare comandi SQL per aggiungere shard e router a un gruppo di shard del database. Per ulteriori informazioni, consulta gli argomenti seguenti:

- Suddivisione di uno shard in un gruppo di shard di database
- Aggiunta di un router a un gruppo di shard del database

> **Note**
>
> Non puoi eliminare shard o router.

Creazione di un cluster di database

Utilizza le seguenti procedure per creare un cluster di database Aurora PostgreSQL che utilizza Aurora PostgreSQL Limitless Database.

Puoi utilizzare Console di gestione AWS o AWS CLI per creare il cluster di database che utilizza Aurora PostgreSQL Limitless Database. Crea il cluster di database primario e il gruppo di shard del database.

Console

Quando utilizzi Console di gestione AWS per creare il cluster di database primario, anche il gruppo di shard del database viene creato nella stessa procedura.

Per creare un cluster di database tramite la console

1. Accedi alla Console di gestione AWS e apri la console Amazon RDS all'indirizzo https://console.aws.amazon.com/rds/.
2. Scegliere Crea database.

   Viene visualizzata la pagina Crea database.
3. Per Tipo di motore scegli Aurora (compatibile con PostgreSQL).
4. Per Versione, scegli una delle seguenti opzioni:

   - Aurora PostgreSQL Limitless Database (compatibile con PostgreSQL 16.4)
   - Aurora PostgreSQL Limitless Database (compatibile con PostgreSQL 16.6)
5. Per Aurora PostgreSQL Limitless Database:

a. Inserisci un identificatore del gruppo di shard del database.

> **Important**
>
> Dopo aver creato il gruppo di shard del database, non è possibile modificare l'identificatore del cluster di database o quello del gruppo di shard del database.

b. Per l'intervallo di capacità del gruppo di shard del database:

   i. Inserisci la capacità minima (ACU). Utilizza un valore pari ad almeno 16 ACU.

      Per un ambiente di sviluppo, il valore predefinito è 16 ACU. Per un ambiente di produzione, il valore predefinito è 24 ACU.

   ii. Inserisci la capacità massima (ACU). Utilizza un valore compreso tra almeno 16 ACU e al massimo 6.144 ACU.

      Per un ambiente di sviluppo, il valore predefinito è 64 ACU. Per un ambiente di produzione, il valore predefinito è 384 ACU.

   Per ulteriori informazioni, consulta Correlazione della capacità massima del gruppo di shard del database con il numero di router e shard creati.

c. Per l'implementazione di gruppi di shard del database, scegli se creare standby per il gruppo di shard del database:

- Nessuna ridondanza di calcolo: crea un gruppo di shard del database senza standby per ogni shard. Si tratta del valore di default.
- Ridondanza di calcolo con una singola destinazione di failover: crea un gruppo di shard del database con uno standby di elaborazione in una zona di disponibilità (AZ) diversa.
- Ridondanza di calcolo con due destinazioni di failover: crea un gruppo di shard del database con due standby di calcolo in due AZ diverse.

> **Note**
>
> Se imposti la ridondanza di calcolo su un valore diverso da zero, il numero totale di istanze database per shard verrà raddoppiato o triplicato. Queste istanze database aggiuntive sono standby di calcolo, aumentate o ridotte verticalmente fino alla stessa capacità dell'istanza di scrittura. L'intervallo di capacità non viene impostato separatamente per gli standby. Pertanto, l'utilizzo di ACU e la relativa fatturazione raddoppiano o triplicano di conseguenza. Per conoscere l'esatto utilizzo di ACU derivante dalla ridondanza di calcolo, consulta la metrica `DBShardGroupComputeRedundancyCapacity` in Metriche DBShardGroup.

d. Scegli se rendere il gruppo di shard del database accessibile pubblicamente.

> **Note**
>
> Non è possibile modificare questa impostazione dopo la creazione del gruppo di shard del database.

6. Per la connettività:

a. (Facoltativo) Seleziona Connetti a una risorsa di calcolo EC2, quindi scegli un'istanza EC2 esistente o creane una nuova.

> **Note**
>
> Se ti connetti a un'istanza EC2, non puoi rendere il gruppo di shard del database accessibile pubblicamente.

b. Per Tipo di rete, scegli IPv4 o Modalità dual-stack.

c. Scegli il cloud privato virtuale (VPC) e il gruppo di sottoreti di database o utilizza le impostazioni predefinite.

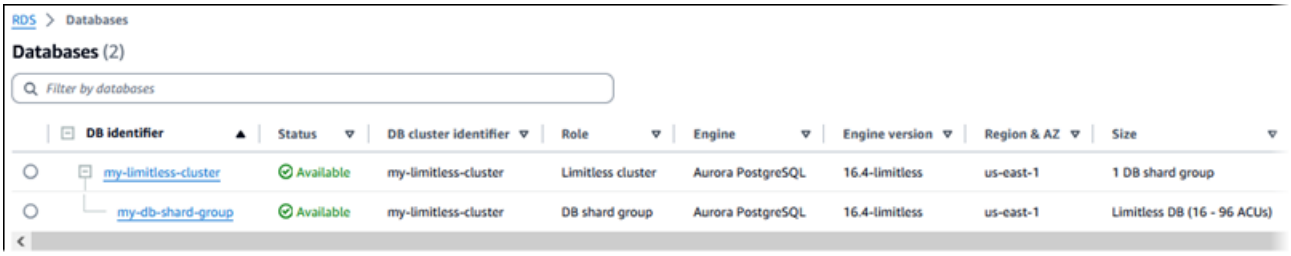
> **Note**
>
> Se crei il cluster di database Limitless Database nella Regione Stati Uniti orientali (Virginia settentrionale), non includere la zona di disponibilità (AZ) `us-east-1e` nel gruppo di sottoreti di database. A causa di limitazioni delle risorse, Aurora Serverless v2, e quindi Limitless Database, non è supportato nella AZ `us-east-1e`.

d. Scegli il gruppo di sicurezza VPC (firewall) o utilizza l'impostazione predefinita.

7. Per Autenticazione database, scegli Autenticazione password o Autenticazione password e database IAM.
8. Per Monitoraggio, assicurati che le caselle di controllo Attivazione di Performance Insights e Abilita monitoraggio avanzato siano selezionate.

   Per Approfondimenti sulle prestazioni, scegli un periodo di conservazione di almeno 1 mese.
9. Espandi Configurazione aggiuntiva nella parte inferiore della pagina.
10. Per Esportazioni log, assicurati che la casella di controllo Log di PostgreSQL sia selezionata.
11. Specifica le altre impostazioni secondo necessità. Per ulteriori informazioni, consulta Impostazioni per cluster di database Aurora.
12. Scegliere Crea database.

Dopo aver creato il cluster di database primario e il gruppo di shard del database, questi vengono visualizzati nella pagina Database.


RDS > Databases
Databases (2)
Filter by databases
DB identifier
Status
DB cluster identifier
Role
Engine
Engine version
Region & AZ
Size
my-limitless-cluster
Available
my-limitless-cluster
Limitless cluster
Aurora PostgreSQL
16.4-limitless
us-east-1
1 DB shard group
my-db-shard-group
Available
my-limitless-cluster
DB shard group
Aurora PostgreSQL
16.4-limitless
us-east-1
Limitless DB (16 - 96 ACUs)

CLI

Quando utilizzi AWS CLI per creare un cluster di database che utilizza Aurora PostgreSQL Limitless Database, devi eseguire le seguenti attività:

1. Creare il cluster di database primario.
2. Creare il gruppo di shard del database.

Creare il cluster di database primario

Per creare il cluster di database sono necessari i seguenti parametri:

- `--db-cluster-identifier`: nome del cluster di database.
- `--engine`: il cluster di database deve utilizzare il motore di database `aurora-postgresql`.
- `--engine-version`: il cluster di database deve utilizzare una delle versioni supportate del motore di database.
  - `16.4-limitless`
  - `16.6-limitless`
- `--storage-type`: il cluster di database deve utilizzare la configurazione di archiviazione del cluster di database `aurora-iopt1`.
- `--cluster-scalability-type`: specifica la modalità di scalabilità del cluster di database Aurora. Se impostato su `limitless`, il cluster funziona come Aurora PostgreSQL Limitless Database. Se impostato su `standard` (impostazione predefinita), il cluster utilizza la normale creazione di istanze database.

  > **Note**
  >
  > Non è possibile modificare questa impostazione dopo la creazione del cluster di database.

- `--master-username`: il nome dell'utente master del cluster di database.
- `--master-user-password`: la password dell'utente master.
- `--enable-performance-insights`: è necessario abilitare Approfondimenti sulle prestazioni.
- `--performance-insights-retention-period`: il periodo di conservazione di Approfondimenti sulle prestazioni deve essere di almeno 31 giorni.
- `--monitoring-interval`: l'intervallo, in secondi, tra i punti di raccolta delle metriche di monitoraggio avanzato per il cluster di database. Questo valore può essere `0`.

- `--monitoring-role-arn`: il nome della risorsa Amazon (ARN) del ruolo IAM che consente a RDS di inviare le metriche di monitoraggio avanzato ad Amazon CloudWatch Logs.
- `--enable-cloudwatch-logs-exports`: è necessario esportare i log `postgresql` in CloudWatch Logs.

I parametri seguenti sono facoltativi:

- `--db-subnet-group-name`: il gruppo di sottoreti database da associare al cluster di database. Questo determina anche il VPC associato al cluster di database.

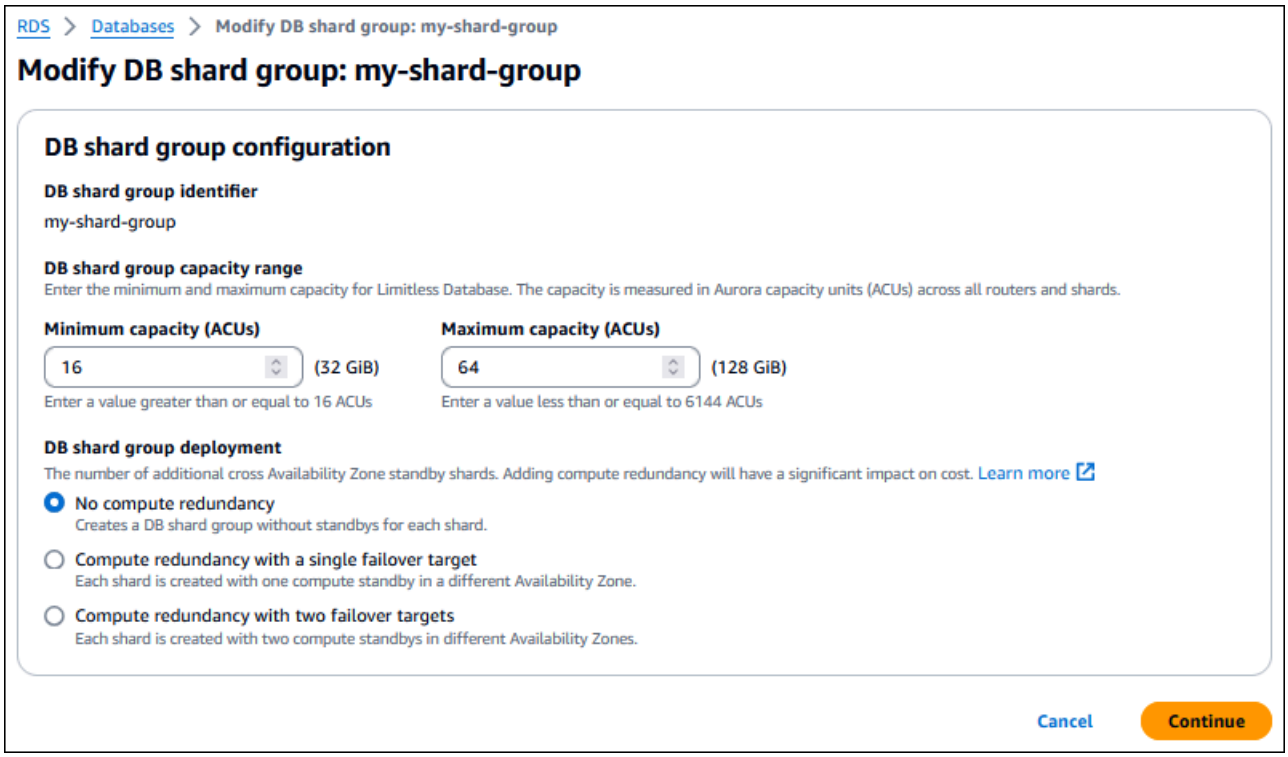
  > **Note**
  >
  > Se crei il cluster di database Limitless Database nella Regione Stati Uniti orientali (Virginia settentrionale), non includere la zona di disponibilità (AZ) `us-east-1e` nel gruppo di sottoreti di database. A causa di limitazioni delle risorse, Aurora Serverless v2, e quindi Limitless Database, non è supportato nella AZ `us-east-1e`.

- `--vpc-security-group-ids`: un elenco di gruppi di sicurezza VPC da associare al cluster di database.
- `--performance-insights-kms-key-id`: l'identificatore AWS KMS key per la crittografia dei dati di Approfondimenti sulle prestazioni. Se non specifichi una chiave KMS, viene utilizzata la chiave predefinita dell'Account AWS.
- `--region`: la Regione AWS in cui viene creato il cluster di database. Deve supportare Aurora PostgreSQL Limitless Database.

Per utilizzare il VPC e il gruppo di sicurezza VPC predefiniti, ometti le opzioni `--db-subnet-group-name` e `--vpc-security-group-ids`.

Per creare un cluster di database primario

- ```
  aws rds create-db-cluster \
      --db-cluster-identifier my-limitless-cluster \
      --engine aurora-postgresql \
      --engine-version 16.6-limitless \
      --storage-type aurora-iopt1 \
      --cluster-scalability-type limitless \
      --master-username myuser \
  ```
  ```

```
    --master-user-password mypassword \
    --db-subnet-group-name mysubnetgroup \
    --vpc-security-group-ids sg-c7e5b0d2 \
    --enable-performance-insights \
    --performance-insights-retention-period 31 \
    --monitoring-interval 5 \
    --monitoring-role-arn arn:aws:iam::123456789012:role/EMrole \
    --enable-cloudwatch-logs-exports postgresql
```

Per ulteriori informazioni, consulta create-db-cluster.

Creare il gruppo di shard del database

Successivamente, crea il gruppo di shard del database nel cluster appena creato. I parametri seguenti sono obbligatori:

- `--db-shard-group-identifier`: il nome del gruppo di shard del database.

  L'identificatore del gruppo di shard del database presenta i seguenti vincoli:

  - Deve essere unico nell'Account AWS e nella Regione AWS in cui viene creato.
  - Deve contenere da 1 a 63 lettere, numeri o trattini.
  - Il primo carattere deve essere una lettera.
  - Non può terminare con un trattino o contenere due trattini consecutivi.
  -

    Important

    Dopo aver creato il gruppo di shard del database, non è possibile modificare l'identificatore del cluster di database o quello del gruppo di shard del database.

- `--db-cluster-identifier`: il nome del cluster di database in cui viene creato il gruppo di shard del database.
- `--max-acu`: la capacità massima del gruppo di shard del database. Deve essere compresa tra 16 e 6.144 ACU. Per limiti di capacità superiori a 6.144 ACU, contatta AWS.

  Il numero iniziale di router e shard è determinato dalla capacità massima impostata al momento della creazione del gruppo di shard del database. Maggiore è la capacità massima, maggiore sarà il numero di router e shard creati nel gruppo di shard del database. Per ulteriori informazioni, consulta Correlazione della capacità massima del gruppo di shard del database con il numero di router e shard creati.

I parametri seguenti sono facoltativi:

- `--compute-redundancy`: indica se creare standby per il gruppo di shard del database. Questo parametro può avere uno dei seguenti valori:
  - `0`: crea un gruppo di shard del database senza standby per ogni shard. Si tratta del valore di default.
  - `1`: crea un gruppo di shard del database con un'istanza di standby di calcolo in una zona di disponibilità (AZ) diversa.

- 2: crea un gruppo di shard del database con due standby di calcolo in due AZ diverse.

> **Note**
>
> Se imposti la ridondanza di calcolo su un valore diverso da zero, il numero totale di shard verrà raddoppiato o triplicato. Ciò comporterà costi aggiuntivi.
> I nodi negli standby di calcolo vengono aumentati o ridotti verticalmente fino alla stessa capacità dell'istanza di scrittura. L'intervallo di capacità non viene impostato separatamente per gli standby.

- `--min-acu`: la capacità minima del gruppo di shard del database. Deve essere pari ad almeno 16 ACU, che è il valore predefinito.
- `--publicly-accessible|--no-publicly-accessible`: indica se assegnare indirizzi IP pubblici al gruppo di shard del database. L'accesso al gruppo di shard del database è controllato dai gruppi di sicurezza utilizzati dal cluster.

  Il valore di default è `--no-publicly-accessible`.

> **Note**
>
> Non è possibile modificare questa impostazione dopo la creazione del gruppo di shard del database.

Per creare il gruppo di shard del database

-
```
aws rds create-db-shard-group \
    --db-shard-group-identifier my-db-shard-group \
    --db-cluster-identifier my-limitless-cluster \
    --max-acu 1000
```

Utilizzo di gruppi di shard di database

Per aggiungere e gestire un gruppo di shard di database per Aurora PostgreSQL Limitless Database, è possibile eseguire le attività indicate di seguito.

Argomenti

- Connessione al cluster di database Aurora PostgreSQL Limitless DB
- Individuazione del numero di router e di shard in un gruppo di shard di database
- Descrizione dei gruppi di shard di database
- Riavvio di un gruppo di shard di database
- Modifica della capacità di un gruppo di shard del database
- Suddivisione di uno shard in un gruppo di shard di database
- Aggiunta di un router a un gruppo di shard del database
- Eliminazione di un gruppo di shard del database
- Aggiunta di un gruppo di shard di database a un cluster di database Aurora PostgreSQL Limitless Database esistente

Connessione al cluster di database Aurora PostgreSQL Limitless DB

Per utilizzare Aurora PostgreSQL Limitless Database, connettiti all'endpoint di scrittura o lettura del cluster. Puoi utilizzare `psql` o qualsiasi altra utilità di connessione che funzioni con PostgreSQL:

```
$ psql -h DB_cluster_endpoint -p port_number -U database_username -d postgres_limitless
```

Nell'esempio seguente viene utilizzato l'endpoint per il cluster del database che hai creato nella CLI.

```
$ psql -h my-limitless-cluster.cluster-ckifpdyyyxxx.us-east-1.rds.amazonaws.com -p 5432
 -U postgres -d postgres_limitless
```

> **Note**
>
> Il database predefinito per il gruppo di shard di database in Aurora PostgreSQL Limitless Database è `postgres_limitless`.

Utilizzo del plug-in Limitless Connection

Per connettersi ad Aurora PostgreSQL Limitless Database, i client utilizzano l'endpoint del cluster e vengono instradati verso un router di transazione da Amazon Route 53. Tuttavia, Route 53 ha una capacità di bilanciamento del carico limitata e può consentire carichi di lavoro irregolari sui router di transazione. Il plug-in Limitless Connection per il driver AWS JDBC risolve questo problema eseguendo il bilanciamento del carico sul lato client con il riconoscimento del carico. Per ulteriori informazioni sul driver AWS JDBC, consulta Connessione ad Aurora PostgreSQL con il driver JDBC per Amazon Web Services (AWS).

Individuazione del numero di router e di shard in un gruppo di shard di database

Per individuare il numero di router e di shard puoi utilizzare la seguente query:

```
SELECT * FROM rds_aurora.limitless_subclusters;

 subcluster_id | subcluster_type
---------------+-----------------
 1             | router
 2             | router
 3             | shard
 4             | shard
 5             | shard
 6             | shard
```

Descrizione dei gruppi di shard di database

Per descrivere i tuoi gruppi di shard di database, utilizza il comando `describe-db-shard-groups` della AWS CLI. Il parametro seguente è facoltativo:

- `--db-shard-group-identifier`: il nome di un gruppo di shard di database.

Nell'esempio seguente viene descritto un gruppo di shard di database specifico.

```
aws rds describe-db-shard-groups --db-shard-group-identifier my-db-shard-group
```

L'output è simile a quello riportato nel seguente esempio.

```
{
    "DBShardGroups": [
        {
            "DBShardGroupResourceId": "shardgroup-8986d309a93c4da1b1455add17abcdef",
            "DBShardGroupIdentifier": "my-shard-group",
            "DBClusterIdentifier": "my-limitless-cluster",
            "MaxACU": 1000.0,
            "ComputeRedundancy": 0,
            "Status": "available",
            "PubliclyAccessible": false,
```

```
            "Endpoint": "my-limitless-cluster.limitless-ccetp2abcdef.us-
east-1.rds.amazonaws.com"
        }
    ]
}
```

Riavvio di un gruppo di shard di database

A volte è necessario riavviare il gruppo di shard di database, ad esempio quando il parametro `max_connections` cambia a causa di una modifica della capacità massima.

È possibile utilizzare la Console di gestione AWS o la AWS CLI per modificare la capacità di un gruppo di shard di database.

Console

Attenersi alla seguente procedura:

Accedi alla Console di gestione AWS e apri la console Amazon RDS all'indirizzo https://console.aws.amazon.com/rds/.

1. Passare alla pagina Databases (Database).
2. Seleziona il gruppo di shard di database che desideri riavviare.
3. In Actions (Operazioni), scegliere Reboot (Riavvia).
4. Scegli Conferma.

CLI

Per riavviare un gruppo di shard di database, utilizza il comando `reboot-db-shard-group` della AWS CLI con i parametri obbligatori seguenti:

- `--db-shard-group-identifier`: il nome di un gruppo di shard di database.

Nell'esempio seguente viene eseguito il riavvio di un gruppo di shard di database.

```
aws rds reboot-db-shard-group --db-shard-group-identifier my-db-shard-group
```

Modifica della capacità di un gruppo di shard del database

Puoi utilizzare la Console di gestione AWS o la AWS CLI per modificare la capacità di un gruppo di shard del database.

Console

Attenersi alla seguente procedura:

Accedi alla Console di gestione AWS e apri la console Amazon RDS all'indirizzo https://console.aws.amazon.com/rds/.

1. Passare alla pagina Databases (Database).
2. Seleziona il gruppo di shard del database che desideri modificare.
3. Per Operazioni, scegli Modifica.

   Viene visualizzata la pagina Modifica il gruppo di partizioni di database.



4. Inserisci un nuovo valore Capacità minima (ACU), ad esempio **100**.
5. Inserisci un nuovo valore Capacità massima (ACU), ad esempio **1000**.
6. Scegli Continue (Continua).

   Viene visualizzata la pagina di conferma con un riepilogo delle modifiche.

7. Controlla le modifiche, quindi scegli Modifica il gruppo di partizioni di database.

CLI

Usa il comando `modify-db-shard-group` della AWS CLI con i parametri seguenti.

- `--db-shard-group-identifier`: il nome di un gruppo di shard del database.
- `--max-acu`: la nuova capacità massima del gruppo di shard del database. Puoi impostare la capacità massima del gruppo di shard del database tra 16 e 6.144 ACU. Per limiti di capacità superiori a 6.144 ACU, contatta AWS.

  Il numero di router e shard non cambia.
- `--min-acu`: la nuova capacità minima del gruppo di shard del database. Deve essere pari ad almeno 16 ACU, che è il valore predefinito.

L'esempio della CLI seguente modifica l'intervallo di capacità di un gruppo di shard del database a 100-1.000 ACU.

```
aws rds modify-db-shard-group \
    --db-shard-group-identifier my-db-shard-group \
    --min-acu 100 \
    --max-acu 1000
```

Suddivisione di uno shard in un gruppo di shard di database

È possibile suddividere manualmente uno shard di un gruppo di shard di database in due shard più piccoli. Questa operazione è denominata suddivisione dello shard avviata dall'utente.

Aurora PostgreSQL Limitless Database può suddividere gli shard anche quando presentano grandi quantità di dati o un utilizzo molto elevato. Questa operazione è denominata suddivisione dello shard avviata dal sistema.

Argomenti

- Prerequisiti
- Suddivisione di uno shard
- Tracciamento delle suddivisioni di shard
- Finalizzazione delle suddivisioni in shard
- Annullamento di una suddivisione in shard

Prerequisiti

Le suddivisioni di shard avviate dall'utente presentano i seguenti prerequisiti:

- È necessario disporre di un gruppo di shard di database.
- Il gruppo di shard di database non può essere vuoto e deve contenere almeno una tabella sottoposta a shard.
- Un utente deve avere il privilegio `rds_aurora_limitless_cluster_admin`. Il `rds_superuser` dispone di questo privilegio, quindi lo possiede anche l'utente master. Il `rds_superuser` può concedere il privilegio ad altri utenti:

```
/* Logged in as the master user or a user with rds_superuser privileges */
CREATE USER username;
GRANT rds_aurora_limitless_cluster_admin to username;
```

- È necessario conoscere l'ID del cluster secondario (nodo) dello shard che si desidera dividere. Per ottenere l'ID, utilizza la seguente query:

```
SELECT * FROM rds_aurora.limitless_subclusters;

 subcluster_id | subcluster_type
```

```
---------------+-----------------
 1             | router
 2             | router
 3             | shard
 4             | shard
 5             | shard
 6             | shard
```

Per abilitare le suddivisioni di shard avviate dal sistema, imposta i seguenti parametri del cluster di database in un gruppo di parametri personalizzato associato al cluster di database:

Parametro	Valore
`rds_aurora.limitless_enable_auto_scale`	on
`rds_aurora.limitless_auto_scale_options`	`split_shard` o `add_router,split_shard`
`rds_aurora.limitless_finalize_split_shard_mode`	Questo parametro determina come vengono finalizzate le suddivisioni di shard avviate dal sistema. Il valore può essere uno dei seguenti: • `user_initiated` : sei tu a decidere quando finalizzare la suddivisione dello shard. Si tratta del valore di default. • `immediate` : le suddivisioni degli shard vengono finalizzate immediatamente. Per ulteriori informazioni, consulta Finalizzazione delle suddivisioni in shard. **Note** Questo parametro si applica solo alle suddivisioni di shard avviate dal sistema.

Per ulteriori informazioni, consulta Gruppi di parametri del cluster di database per i cluster di database Amazon Aurora.

Suddivisione di uno shard

Per suddividere uno shard in un gruppo di shard di database, utilizza la funzione `rds_aurora.limitless_split_shard`. Questa funzione avvia un processo di suddivisione in shard che viene eseguito in modo asincrono.

```
SELECT rds_aurora.limitless_split_shard('subcluster_id');
```

Attendi la restituzione dell'ID di un processo correttamente inviato, ad esempio:

```
SELECT rds_aurora.limitless_split_shard('3');

    job_id
---------------
 1691300000000
(1 row)
```

> **Note**
>
> Le operazioni di suddivisione degli shard simultanee non sono supportate. Esegui ogni operazione in sequenza e completala prima di avviarne una aggiuntiva.

Tracciamento delle suddivisioni di shard

L'ID del processo consente di tracciare un processo di suddivisione in shard. Per descrivere un determinato processo e ottenere maggiori dettagli su di esso, esegui la query sotto indicata:

```
SELECT * FROM rds_aurora.limitless_list_shard_scale_jobs(job_id);
```

Per esempio:

```
SELECT * FROM rds_aurora.limitless_list_shard_scale_jobs(1691300000000);

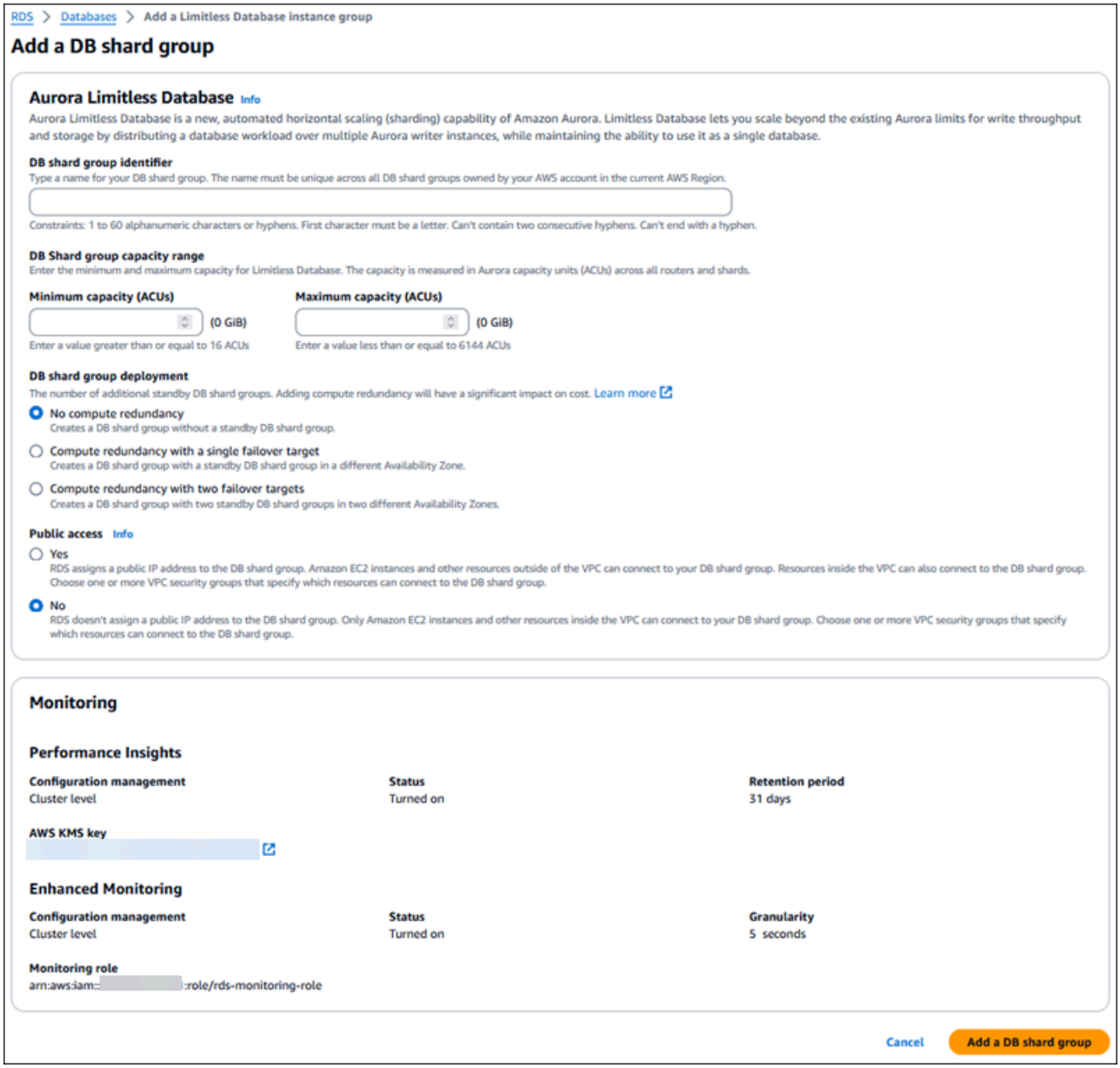
    job_id     |    action    |      job_details      | status  |    submission_time
 |                 message
```

```
--------------+------------+----------------------+---------
+------------------------+-------------------------------------------
 1691300000000 | SPLIT_SHARD | Split Shard 3 by User | SUCCESS | 2023-08-06 05:33:20+00
 | Scaling job succeeded.                +
               |            |                       |         |
 | New shard instance with ID 7 was created.
(1 row)
```

La query restituisce un errore quando si passa un processo inesistente come input.

```
SELECT * from rds_aurora.limitless_list_shard_scale_jobs(1691300000001);

ERROR:  no job found with the job ID provided
```

Puoi tenere traccia dello stato di tutti i processi di suddivisione in shard utilizzando la stessa query senza un ID del processo, ad esempio:

```
SELECT * FROM rds_aurora.limitless_list_shard_scale_jobs();

    job_id     |   action    |  job_details          |   status    |    submission_time
     |                 message
--------------+------------+----------------------+-------------
+------------------------
+-------------------------------------------------------------
 1691200000000 | SPLIT_SHARD | Split Shard 3 by User | IN_PROGRESS | 2023-08-05
 01:46:40+00 |
 1691300000000 | SPLIT_SHARD | Split Shard 4 by User | SUCCESS     | 2023-08-06
 05:33:20+00 | Scaling job succeeded. +
               |            |                       |             |
     | New shard instance with ID 7 was created.
 1691400000000 | SPLIT_SHARD | Split Shard 5 by User | FAILED      | 2023-08-07
 09:20:00+00 | Error occurred for the add shard job 1691400000000.
               |            |                       |             |
     | Retry the command. If the issue persists, contact AWS Support.
 1691500000000 | SPLIT_SHARD | Split Shard 5 by User | CANCELED    | 2023-08-07
 09:20:00+00 | Scaling job was cancelled.
(4 rows)
```

Il processo può avere uno dei seguenti stati:

- `IN_PROGRESS`: il processo di suddivisione in shard è stato inviato ed è in corso. È possibile avere un solo processo in corso per volta.

- `PENDING`: il processo di suddivisione in shard attende di essere finalizzato dall'utente. Per ulteriori informazioni, consulta Finalizzazione delle suddivisioni in shard.
- `CANCELLATION_IN_PROGRESS`: il processo di suddivisione in shard viene annullato dall'utente.
- `CANCELED`: il processo di suddivisione in shard è stato correttamente annullato dall'utente o dal sistema.
- `SUCCESS`: il processo di suddivisione in shard è stato completato correttamente. Il campo `message` contiene l'ID dell'istanza del nuovo shard.
- `FAILED`: il processo di suddivisione in shard non è riuscito. Il campo `message` contiene i dettagli dell'errore e tutte le azioni che possono essere intraprese in seguito all'esito negativo del processo.

Finalizzazione delle suddivisioni in shard

La finalizzazione è l'ultima fase del processo di suddivisione in shard e causa alcuni tempi di inattività. Se avvii un processo di suddivisione in shard, la finalizzazione avviene subito dopo il corretto completamento del processo.

Se hai abilitato la suddivisione in shard avviata dal sistema utilizzando il parametro `rds_aurora.limitless_enable_auto_scale`, è possibile che il sistema suddivida gli shard in base al carico di lavoro.

In questo caso, puoi scegliere se finalizzare il processo immediatamente o in un momento a tua scelta. Per stabilirlo, utilizza il parametro del cluster di database `rds_aurora.limitless_finalize_split_shard_mode`:

- Se imposti il valore su `immediate`, avviene immediatamente.
- Se imposti il valore su `user_initiated`, il processo di suddivisione in shard deve essere finalizzato manualmente.

  Ti viene inviato un evento RDS e lo stato del processo di suddivisione in shard viene impostato su `PENDING`.

Quando il parametro è impostato su `user_initiated`, per finalizzare il processo di suddivisione in shard utilizza la funzione `rds_aurora.limitless_finalize_split_shard`:

```
SELECT * FROM rds_aurora.limitless_finalize_split_shard(job_id);
```

> **Note**
>
> Questa funzione si applica solo alle suddivisioni in shard avviate dal sistema, non dall'utente.

Annullamento di una suddivisione in shard

È possibile annullare una suddivisione in shard avviata dall'utente o dal sistema con stato `IN_PROGRESS` o `PENDING`. Per annullare il processo, è necessario il relativo ID.

```
SELECT * from rds_aurora.limitless_cancel_shard_scale_jobs(job_id);
```

Non viene restituito alcun output. a meno che non si verifichi un errore. Puoi tracciare l'annullamento utilizzando una query di tracciamento del lavoro.

Aggiunta di un router a un gruppo di shard del database

Puoi aggiungere un router a un gruppo di shard del database.

Argomenti

- Prerequisiti
- Aggiunta di un router
- Monitoraggio delle aggiunte di router
- Annullamento dell'aggiunta di un router

Prerequisiti

L'aggiunta di un router prevede i seguenti prerequisiti:

- È necessario disporre di un gruppo di shard del database.
- Un utente deve avere il privilegio `rds_aurora_limitless_cluster_admin`. L'`rds_superuser` dispone di questo privilegio, quindi lo possiede anche l'utente master. L'`rds_superuser` può concedere il privilegio ad altri utenti:

```
/* Logged in as the master user or a user with rds_superuser privileges */
CREATE USER username;
GRANT rds_aurora_limitless_cluster_admin to username;
```

> **Note**
>
> Se modifichi il certificato CA predefinito dell'Account AWS dopo che il gruppo di shard del database è stato creato, il nuovo router utilizzerà il nuovo certificato CA, che sarà diverso dal certificato CA del router esistente. A seconda dell'archivio di trust, alcune connessioni potrebbero non riuscire.

- Per abilitare l'aggiunta di router avviata dal sistema, imposta i seguenti parametri del cluster di database in un gruppo di parametri personalizzato associato al cluster di database:

Parametro	Valore
`rds_aurora.limitless_enable_auto_scale`	`on`
`rds_aurora.limitless_auto_scale_options`	`add_router` o `add_router,split_shard` .

Per ulteriori informazioni, consulta Gruppi di parametri del cluster di database per i cluster di database Amazon Aurora.

Aggiunta di un router

Per aggiungere un router, utilizza la funzione `rds_aurora.limitless_add_router`. Questa funzione avvia un processo di aggiunta di router che viene eseguito in modo asincrono.

```
SELECT rds_aurora.limitless_add_router();
```

Attendi la restituzione di un ID di processo correttamente inviato, ad esempio:

```
    job_id
---------------
 1691300000000
(1 row)
```

> **Note**
>
> Le operazioni concorrenti di aggiunta del router non sono supportate. Esegui le operazioni in sequenza e completa ogni operazione prima di iniziarne un'altra.

Monitoraggio delle aggiunte di router

Puoi utilizzare l'ID di processo per tenere traccia di un processo di aggiunta di router. Per descrivere un determinato processo e ottenere maggiori dettagli su di esso, esegui la query sotto indicata:

```
SELECT * FROM rds_aurora.limitless_list_router_scale_jobs(job_id);
```

Per esempio:

```
SELECT * FROM rds_aurora.limitless_list_router_scale_jobs(1691300000000);

    job_id     |   action   |        job_details       | status  |    submission_time
   |                   message
---------------+------------+--------------------------+---------
+------------------------+-------------------------------------------
 1691300000000 | ADD_ROUTER | Add 1 new Router by User | SUCCESS | 2023-08-06
 05:33:20+00 | Scaling job succeeded.                   +
               |            |                          |         |
   | New router instance with ID 7 was created.
(1 row)
```

La query restituisce un errore quando si passa un processo inesistente come input.

```
SELECT * from rds_aurora.limitless_list_router_scale_jobs(1691300000001);

ERROR:  no job found with the job ID provided
```

Puoi tenere traccia dello stato di tutti i processi di aggiunta del router utilizzando la stessa query senza un ID di processo, ad esempio:

```
SELECT * FROM rds_aurora.limitless_list_router_scale_jobs();

    job_id     |   action   |        job_details       |   status    |
 submission_time     |                  message
---------------+------------+--------------------------+-------------
+------------------------+-------------------------------------------
 1691200000000 | ADD_ROUTER | Add 1 new Router by User | IN_PROGRESS | 2023-08-05
 01:46:40+00 |
 1691300000000 | ADD_ROUTER | Add 1 new Router by User | SUCCESS     | 2023-08-06
 05:33:20+00 | Scaling job succeeded.                 +
               |            |                          |             |
       | New router instance with ID 7 was created.
 1691400000000 | ADD_ROUTER | Add 1 new Router by User | FAILED      | 2023-08-07
 09:20:00+00 | Error occurred for the add router job 1691400000000.
               |            |                          |             |
       | Retry the command. If the issue persists, contact AWS Support.
```

```
 1691500000000 | ADD_ROUTER | Add 1 new Router by User | CANCELED    | 2023-08-07
 09:20:00+00 | Scaling job was cancelled.
(4 rows)
```

Lo stato del processo può essere uno dei seguenti:

- `IN_PROGRESS`: il processo di aggiunta di router è stato inviato ed è in corso. Puoi avere un solo processo in corso per volta.
- `CANCELLATION_IN_PROGRESS`: il processo di aggiunta di router è in fase di annullamento da parte dell'utente.
- `CANCELED`: il processo di aggiunta di router è stato correttamente annullato dall'utente o dal sistema.
- `SUCCESS`: il processo di aggiunta di router è stato completato correttamente. Il campo `message` contiene l'ID dell'istanza del nuovo router.
- `FAILED`: il processo di aggiunta di router non è riuscito. Il campo `message` contiene i dettagli dell'errore e tutte le azioni che possono essere intraprese in seguito alla mancata riuscita del processo.

> **Note**
>
> Non esiste uno stato `PENDING` perché le aggiunte di router non devono essere finalizzate. Non comportano tempi di inattività.

Annullamento dell'aggiunta di un router

Puoi annullare l'aggiunta di un router `IN_PROGRESS`. Per annullare il processo, è necessario il relativo ID.

```
SELECT * from rds_aurora.limitless_cancel_router_scale_jobs(job_id);
```

Non viene restituito alcun output a meno che non si verifichi un errore. Puoi tenere traccia dell'annullamento utilizzando una query di tracciamento del lavoro.

Eliminazione di un gruppo di shard del database

È possibile eliminare un gruppo di shard del database se necessario. L'eliminazione del gruppo di shard del database elimina i nodi di calcolo (shard e router), ma non l'archiviazione.

Console

Accedi alla Console di gestione AWS e apri la console Amazon RDS all'indirizzo https://console.aws.amazon.com/rds/.

1. Passare alla pagina Databases (Database).
2. Selezionare il gruppo di shard del database da eliminare.
3. In Actions (Azioni), scegliere Delete (Elimina).
4. Inserire **`delete me`** nella casella e scegliere Elimina.

Il gruppo di shard del database viene eliminato.

AWS CLI

Per eliminare il gruppo di shard del database, utilizzare il comando `delete-db-shard-group` dell'interfaccia AWS CLI con il parametro seguente:

- `--db-shard-group-identifier`: il nome del gruppo di shard del database.

L'esempio seguente elimina un gruppo di shard del database nel cluster di database Aurora PostgreSQL creato in precedenza.

```
aws rds delete-db-shard-group --db-shard-group-identifier my-db-shard-group
```

Aggiunta di un gruppo di shard di database a un cluster di database Aurora PostgreSQL Limitless Database esistente

Puoi creare un gruppo di shard di database in un cluster di database esistente, ad esempio se stai ripristinando un cluster di database o hai eliminato il gruppo di shard di database.

Per ulteriori informazioni sui requisiti del cluster di database primario e del gruppo di shard di database, consulta Requisiti e considerazioni su Aurora PostgreSQL Limitless Database.

> **Note**
>
> È consentito un solo gruppo di shard di database per cluster.
> Per poter creare un gruppo di shard di database, il cluster di database Limitless Database deve trovarsi nello stato `available`.

Console

È possibile utilizzare la Console di gestione AWS per aggiungere un gruppo di shard di database a un cluster di database esistente.

Per aggiungere un gruppo di shard di database

1. Accedi alla Console di gestione AWS e apri la console Amazon RDS all'indirizzo https://console.aws.amazon.com/rds/.
2. Passare alla pagina Databases (Database).
3. Seleziona il cluster di database Limitless Database a cui aggiungere un gruppo di shard di database.
4. Per Azioni, scegli Aggiungi un gruppo di partizioni di database.



5. Inserisci un identificatore del gruppo di shard di database.

> **Important**
>
> Dopo aver creato il gruppo di shard di database, non è possibile modificare l'identificatore del cluster di database o quello del gruppo di shard di database.

6. Inserisci la capacità minima (ACU). Utilizza un valore pari ad almeno 16 ACU.
7. Inserisci la capacità massima (ACU). Utilizza un valore compreso tra 16 e 6144 ACU.

Per ulteriori informazioni, consulta Correlazione della capacità massima del gruppo di shard del database con il numero di router e shard creati.

8. Per l'implementazione di gruppi di shard di database, scegli se creare standby per il gruppo di shard di database:

   - Nessuna ridondanza di calcolo: crea un gruppo di shard di database senza standby per ogni shard. Si tratta del valore di default.
   - Ridondanza di calcolo con una singola destinazione di failover: crea un gruppo di shard di database con uno standby di elaborazione in una zona di disponibilità (AZ) diversa.
   - Ridondanza di calcolo con due destinazioni di failover: crea un gruppo di shard di database con due istanze di standby in due AZ diverse.

9. Scegli se rendere il gruppo di shard di database accessibile pubblicamente.

   > **Note**
   >
   > Non è possibile modificare questa impostazione dopo la creazione del gruppo di shard di database.

10. Scegli Aggiungi un gruppo di partizioni di database.

AWS CLI

Utilizza il comando `create-db-shard-group` della AWS CLI per creare un gruppo di shard di database.

I parametri seguenti sono obbligatori:

- `--db-cluster-identifier`: il cluster di database a cui il gruppo di shard di database appartiene.
- `--db-shard-group-identifier`: il nome del gruppo di shard di database.

  L'identificatore del gruppo di shard di database presenta i seguenti vincoli:

  - Deve essere unico nell'Regione AWS e nella Account AWS in cui viene creato.
  - Deve contenere da 1 a 63 lettere, numeri o trattini.
  - Il primo carattere deve essere una lettera.
  - Non può terminare con un trattino o contenere due trattini consecutivi.

  > **Important**
  >
  > Dopo aver creato il gruppo di shard di database, non è possibile modificare l'identificatore del cluster di database o quello del gruppo di shard di database.

- `--max-acu`: la capacità massima del gruppo di shard di database. Utilizza un valore compreso tra 16 e 6144 ACU.

I parametri seguenti sono facoltativi:

- `--compute-redundancy`: indica se creare standby per il gruppo di shard di database. Questo parametro può avere uno dei seguenti valori:
  - `0`: crea un gruppo di shard di database senza standby per ogni shard. Si tratta del valore di default.
  - `1`: crea un gruppo di shard di database con un'istanza di standby di calcolo in una zona di disponibilità (AZ) diversa.
  - `2`: crea un gruppo di shard di database con due istanze di standby di calcolo in due AZ diverse.

> **Note**
>
> Se imposti la ridondanza di calcolo su un valore diverso da zero, il numero totale di nodi verrà raddoppiato o triplicato. Ciò comporterà costi aggiuntivi.

- `--min-acu`: la capacità minima del gruppo di shard di database. Deve essere pari ad almeno 16 ACU, che è il valore predefinito.
- `--publicly-accessible|--no-publicly-accessible`: indica se assegnare indirizzi IP pubblici al gruppo di shard di database. L'accesso al gruppo di shard di database è controllato dai gruppi di sicurezza utilizzati dal cluster.

  Il valore predefinito è `--no-publicly-accessible`.

> **Note**
>
> Non è possibile modificare questa impostazione dopo la creazione del gruppo di shard di database.

Nell'esempio seguente viene creato un gruppo di shard di database in un cluster di database Aurora PostgreSQL.

```
aws rds create-db-shard-group \
    --db-cluster-identifier my-db-cluster \
    --db-shard-group-identifier my-new-shard-group \
    --max-acu 1000
```

L'output è simile a quello riportato nel seguente esempio.

```
{
    "Status": "CREATING",
    "Endpoint": "my-db-cluster.limitless-ckifpdyyyxxx.us-east-1.rds.amazonaws.com",
    "PubliclyAccessible": false,
    "DBClusterIdentifier": "my-db-cluster",
    "MaxACU": 1000.0,
    "DBShardGroupIdentifier": "my-new-shard-group",
    "DBShardGroupResourceId": "shardgroup-8986d309a93c4da1b1455add17abcdef",
    "ComputeRedundancy": 0
```

```
}
```

Creazione di tabelle di Aurora PostgreSQL Limitless Database

In Aurora PostgreSQL Limitless Database, esistono tre tipi di tabelle che contengono i dati:

- Standard: è il tipo di tabella predefinito in Aurora PostgreSQL Limitless Database. Puoi creare tabelle standard utilizzando il comando CREATE TABLE ed eseguire su di esse operazioni DDL (Data Definition Language) e DML (Data Manipulation Language).

  Le tabelle standard non sono tabelle distribuite. Vengono archiviate su uno degli shard selezionato internamente dal sistema.
- Sottoposte a sharding: queste tabelle sono distribuite su più shard. I dati vengono suddivisi tra gli shard in base ai valori di specifiche colonne della tabella. Questo set di colonne è chiamato chiave di shard.
- Riferimento: queste tabelle vengono replicate su tutti gli shard. Vengono utilizzate per dati di riferimento modificati raramente, come cataloghi di prodotti e codici postali.

  Le query join tra tabelle di riferimento e tabelle sottoposte a sharding possono essere eseguite direttamente sugli shard, eliminando movimenti di dati non necessari tra shard e router.

Esistono due modi per creare tabelle Limitless:

- Creazione di tabelle Limitless utilizzando variabili: utilizza questo metodo quando vuoi creare nuove tabelle sottoposte a sharding e di riferimento.
- Conversione di tabelle standard in tabelle Limitless: utilizza questo metodo per convertire tabelle standard esistenti in tabelle sottoposte a sharding e di riferimento.

Sono inoltre disponibili schemi di esempio per Aurora PostgreSQL Limitless Database.

Creazione di tabelle Limitless utilizzando variabili

Puoi utilizzare variabili per creare tabelle sottoposte a sharding e tabelle di riferimento impostando la modalità di creazione delle tabelle. Le tabelle che creerai utilizzeranno questa modalità finché non imposterai una modalità diversa.

Utilizza le seguenti variabili per creare tabelle sottoposte a sharding e tabelle di riferimento:

- `rds_aurora.limitless_create_table_mode`: imposta questa variabile di sessione su `sharded` o `reference`. Il valore predefinito di questa variabile è `standard`.
- `rds_aurora.limitless_create_table_shard_key`: imposta questa variabile di sessione su un array di nomi di colonna da utilizzare come chiavi di shard. Questa variabile viene ignorata quando `rds_aurora.limitless_create_table_mode` non è `sharded`.

  Formatta il valore come `untyped array literal`, simile a quando inserisci letterali in una colonna di array. Per ulteriori informazioni, consulta Arrays nella documentazione di PostgreSQL.
- `rds_aurora.limitless_create_table_collocate_with`: imposta questa variabile di sessione su un nome di tabella specifico per co-localizzare le tabelle appena create con quella tabella.

  Se due o più tabelle sono sottoposte a sharding utilizzando la stessa chiave di shard, è possibile allinearle esplicitamente (co-localizzarle). Quando due o più tabelle sono co-localizzate, le righe con gli stessi valori di chiave di shard vengono posizionate nello stesso shard. La co-localizzazione aiuta a limitare alcune operazioni a un singolo shard, migliorando le prestazioni.

> **Note**
>
> Tutte le chiavi primarie e uniche devono includere la chiave di shard. Ciò significa che la chiave di shard è un sottoinsieme della chiave primaria o unica.
> Le tabelle Limitless presentano alcune limitazioni. Per ulteriori informazioni, consulta Limitazioni DDL e altre informazioni per Aurora PostgreSQL Limitless Database.

Argomenti

- Esempi di utilizzo di variabili per creare tabelle Limitless
- Visualizzazioni della tabella di Aurora PostgreSQL Limitless Database

Esempi di utilizzo di variabili per creare tabelle Limitless

Gli esempi seguenti mostrano come utilizzare queste variabili per creare tabelle sottoposte a sharding e tabelle di riferimento.

Crea una tabella sottoposta a sharding denominata `items`, con la chiave di shard `id`.

```
BEGIN;
SET LOCAL rds_aurora.limitless_create_table_mode='sharded';
SET LOCAL rds_aurora.limitless_create_table_shard_key='{"id"}';
CREATE TABLE items(id int, val int, item text);
COMMIT;
```

Crea una tabella sottoposta a sharding denominata `items`, con una chiave di shard composta dalle colonne `item_id` e `item_cat`.

```
BEGIN;
SET LOCAL rds_aurora.limitless_create_table_mode='sharded';
SET LOCAL rds_aurora.limitless_create_table_shard_key='{"item_id", "item_cat"}';
CREATE TABLE items(item_id int, item_cat varchar, val int, item text);
COMMIT;
```

Crea una tabella sottoposta a sharding denominata `item_description`, con una chiave di shard composta dalle colonne `item_id` e `item_cat` e co-localizzala con la tabella `items` dell'esempio precedente.

```
BEGIN;
SET LOCAL rds_aurora.limitless_create_table_mode='sharded';
SET LOCAL rds_aurora.limitless_create_table_shard_key='{"item_id", "item_cat"}';
SET LOCAL rds_aurora.limitless_create_table_collocate_with='items';
CREATE TABLE item_description(item_id int, item_cat varchar, color_id int);
COMMIT;
```

Crea una tabella di riferimento denominata `colors`.

```
BEGIN;
SET LOCAL rds_aurora.limitless_create_table_mode='reference';
CREATE TABLE colors(color_id int primary key, color varchar);
COMMIT;
```

Per reimpostare la variabile di sessione `rds_aurora.limitless_create_table_mode` su `standard`, utilizza la seguente istruzione:

```
RESET rds_aurora.limitless_create_table_mode;
```

Dopo aver reimpostato questa variabile, le tabelle vengono create come tabelle standard, che è il valore predefinito. Per ulteriori informazioni sulle tabelle standard, consulta Conversione di tabelle standard in tabelle Limitless.

Visualizzazioni della tabella di Aurora PostgreSQL Limitless Database

Puoi trovare informazioni sulle tabelle di Limitless Database utilizzando le seguenti visualizzazioni.

rds_aurora.limitless_tables

La visualizzazione `rds_aurora.limitless_tables` contiene informazioni sulle tabelle Limitless e sui loro tipi.

```
postgres_limitless=> SELECT * FROM rds_aurora.limitless_tables;

 table_gid | local_oid | schema_name | table_name  | table_status | table_type  |
 distribution_key
-----------+-----------+-------------+-------------+--------------+-------------
+------------------
         5 |     18635 | public      | standard    | active       | standard    |
         6 |     18641 | public      | ref         | active       | reference   |
         7 |     18797 | public      | orders      | active       | sharded     |
 HASH (order_id)
         2 |     18579 | public      | customer    | active       | sharded     |
 HASH (cust_id)
(4 rows)
```

rds_aurora.limitless_table_collocations

La visualizzazione `rds_aurora.limitless_table_collocations` contiene informazioni sulle tabelle sottoposte a sharding co-localizzate. Ad esempio, le tabelle `orders` e `customers` sono co-localizzate e presentano lo stesso `collocation_id`. Le tabelle `users` e `followers` sono co-localizzate e presentano lo stesso `collocation_id`.

```
postgres_limitless=> SELECT * FROM rds_aurora.limitless_table_collocations ORDER BY
 collocation_id;

 collocation_id | schema_name | table_name
----------------+-------------+------------
          16002 | public      | orders
          16002 | public      | customers
          16005 | public      | users
          16005 | public      | followers
(4 rows)
```

rds_aurora.limitless_table_collocation_distributions

`rds_aurora.limitless_table_collocation_distributions` mostra la distribuzione delle chiavi per ogni co-localizzazione.

```
postgres_limitless=> SELECT * FROM
 rds_aurora.limitless_table_collocation_distributions ORDER BY collocation_id,
 lower_bound;

 collocation_id | subcluster_id |     lower_bound      |     upper_bound
----------------+---------------+----------------------+----------------------
          16002 |             6 | -9223372036854775808 | -4611686018427387904
          16002 |             5 | -4611686018427387904 |                    0
          16002 |             4 |                    0 |  4611686018427387904
          16002 |             3 |  4611686018427387904 |  9223372036854775807
          16005 |             6 | -9223372036854775808 | -4611686018427387904
          16005 |             5 | -4611686018427387904 |                    0
          16005 |             4 |                    0 |  4611686018427387904
          16005 |             3 |  4611686018427387904 |  9223372036854775807
(8 rows)
```

Conversione di tabelle standard in tabelle Limitless

Puoi convertire tabelle standard in tabelle sottoposte a sharding o di riferimento. Durante la conversione, i dati vengono spostati dalla tabella standard alla tabella distribuita, quindi la tabella standard di origine viene eliminata. I dati vengono spostati utilizzando il comando `INSERT INTO SELECT FROM`.

Indice

- Creazione di tabelle sottoposte a sharding
- Creazione di tabelle co-localizzate
- Creazione di tabelle di riferimento

Creazione di tabelle sottoposte a sharding

Crea tabelle sottoposte a sharding eseguendo la procedura `rds_aurora.limitless_alter_table_type_sharded` su tabelle standard. Questa procedura riceve una tabella standard e un elenco di colonne, quindi distribuisce la tabella indicata utilizzando l'elenco di colonne come chiave di shard. La procedura viene eseguita in modalità sincrona e acquisisce un blocco `ACCESS EXCLUSIVE` sulla tabella.

Al termine corretto della procedura, la tabella standard di origine viene eliminata e una tabella sottoposta a sharding con lo stesso nome diventa disponibile.

La procedura `rds_aurora.limitless_alter_table_type_sharded` utilizza la seguente sintassi:

```
postgres=> CALL rds_aurora.limitless_alter_table_type_sharded('schema.table',
 ARRAY['shard_key1', 'shard_key2', ... 'shard_keyn']);
```

La procedura richiede i seguenti parametri.

- `schema`: lo schema del database che contiene la tabella da sottoporre a sharding. Se lo schema non è specificato, la procedura utilizza `search_path`.
- `table`: la tabella da sottoporre a sharding.
- `shard_key`*n*: un array di colonne della tabella da utilizzare come chiave di shard.

  I valori della chiave di shard sono stringhe letterali e quindi rispettano la distinzione tra maiuscole e minuscole. Se una chiave di shard contiene una virgoletta singola ('), usare un'altra virgoletta

singola per eseguire la procedura di escape. Ad esempio, se una colonna della tabella è denominata `customer's id`, utilizza `customer''s id` come chiave di shard. Non è necessario eseguire la procedura di escape per le barre rovesciate (\) e le virgolette doppie (").

> **Note**
>
> Tutte le chiavi primarie e uniche devono includere la chiave di shard. Ciò significa che la chiave di shard è un sottoinsieme della chiave primaria o unica.
> Nelle tabelle sottoposte a sharding, il vincolo `CHECK` non supporta le espressioni.
> Per ulteriori informazioni, consulta Vincoli.

Per creare una tabella sottoposta a sharding

L'esempio seguente mostra come creare la tabella sottoposta a sharding `customer` con la chiave di shard `customer_id`.

1. Crea la tabella standard.

```
CREATE TABLE customer (customer_id INT PRIMARY KEY NOT NULL, zipcode INT, email
 VARCHAR);
```

2. Convertire la tabella standard in tabella sottoposta a sharding.

```
postgres=> CALL rds_aurora.limitless_alter_table_type_sharded('public.customer',
 ARRAY['customer_id']);

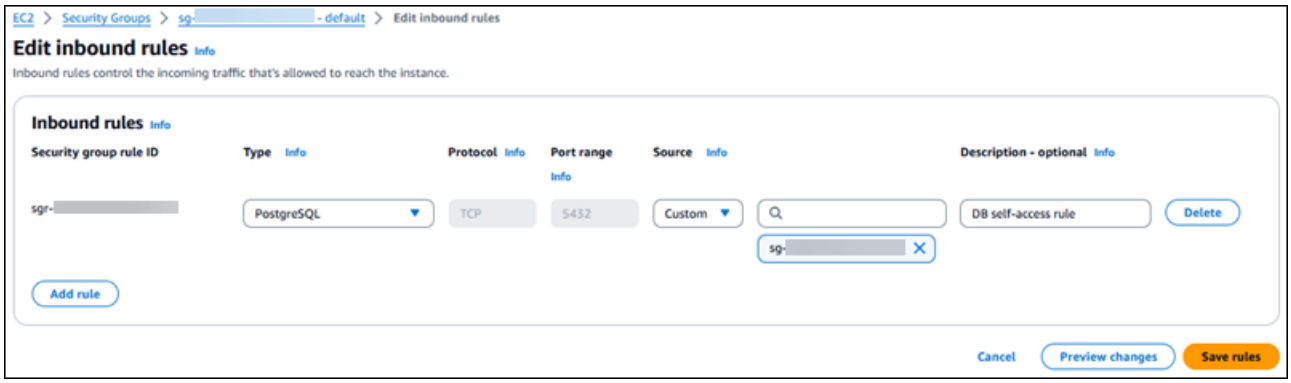
postgres=> \d

                  List of relations
 Schema |     Name     |       Type        |       Owner
--------+--------------+-------------------+--------------------
 public | customer     | partitioned table | postgres_limitless
 public | customer_fs1 | foreign table     | postgres_limitless
 public | customer_fs2 | foreign table     | postgres_limitless
 public | customer_fs3 | foreign table     | postgres_limitless
 public | customer_fs4 | foreign table     | postgres_limitless
 public | customer_fs5 | foreign table     | postgres_limitless
(6 rows)
```

Creazione di tabelle co-localizzate

Se due o più tabelle sono sottoposte a sharding utilizzando la stessa chiave di shard, è possibile allinearle esplicitamente (co-localizzarle). Quando due o più tabelle sono co-localizzate, le righe con gli stessi valori di chiave di shard vengono posizionate nello stesso shard. La co-localizzazione aiuta a limitare alcune operazioni a un singolo shard, migliorando le prestazioni.

Utilizza la procedura `rds_aurora.limitless_alter_table_type_sharded` con la seguente sintassi:

```
postgres=> CALL
 rds_aurora.limitless_alter_table_type_sharded('schema.collocated_table',
 ARRAY['shard_key1', 'shard_key2', ... 'shard_keyn'], 'schema.sharded_table');
```

La procedura richiede i seguenti parametri.

- `schema`: lo schema del database che contiene le tabelle da co-localizzare. Se lo schema non è specificato, la procedura utilizza `search_path`.
- `collocated_table`: la tabella da co-localizzare.
- `shard_key`*n*: un array di colonne della tabella da utilizzare come chiave di shard.

  Devi usare la stessa chiave di shard della tabella sottoposta a sharding originale, incluse le stesse colonne e tipi di dati.
- `sharded_table`: la tabella sottoposta a sharding con cui stai co-localizzando `collocated_table`.

Per creare una tabella co-localizzata

1. Crea la prima tabella sottoposta a sharding seguendo la procedura in Creazione di tabelle sottoposte a sharding.
2. Crea la tabella standard per la tabella co-localizzata.

   ```
   CREATE TABLE mytable2 (customer_id INT PRIMARY KEY NOT NULL, column1 INT, column2
    VARCHAR);
   ```
3. Converti la tabella standard in una tabella co-localizzata.

   ```
   postgres=> CALL rds_aurora.limitless_alter_table_type_sharded('public.mytable2',
   ```

```
ARRAY['customer_id'], 'public.customer');

postgres=> \d

                   List of relations
 Schema |     Name     |       Type        |       Owner
--------+--------------+-------------------+--------------------
 public | customer     | partitioned table | postgres_limitless
 public | customer_fs1 | foreign table     | postgres_limitless
 public | customer_fs2 | foreign table     | postgres_limitless
 public | customer_fs3 | foreign table     | postgres_limitless
 public | customer_fs4 | foreign table     | postgres_limitless
 public | customer_fs5 | foreign table     | postgres_limitless
 public | mytable2     | partitioned table | postgres_limitless
 public | mytable2_fs1 | foreign table     | postgres_limitless
 public | mytable2_fs2 | foreign table     | postgres_limitless
 public | mytable2_fs3 | foreign table     | postgres_limitless
 public | mytable2_fs4 | foreign table     | postgres_limitless
 public | mytable2_fs5 | foreign table     | postgres_limitless
(12 rows)
```

Creazione di tabelle di riferimento

Crea tabelle di riferimento eseguendo la procedura `rds_aurora.limitless_alter_table_type_reference` su tabelle standard. Questa procedura replica una determinata tabella su tutti gli shard del gruppo di shard del database e ne cambia il tipo in tabella di riferimento. La procedura viene eseguita in modalità sincrona e acquisisce un blocco `ACCESS EXCLUSIVE` sulla tabella.

Al termine corretto della procedura, la tabella standard di origine viene eliminata e una tabella di riferimento con lo stesso nome diventa disponibile.

La procedura `rds_aurora.limitless_alter_table_type_reference` utilizza la seguente sintassi:

```
postgres=> CALL rds_aurora.limitless_alter_table_type_reference('schema.table');
```

La procedura archiviata richiede i seguenti parametri:

- `schema`: lo schema del database che contiene le tabelle da replicare. Se lo schema non è specificato, la procedura utilizza `search_path`.

- `table`: la tabella da replicare.

> **Note**
>
> La tabella standard da cui crei la tabella di riferimento deve avere una chiave primaria.
> Nelle tabelle di riferimento, il vincolo CHECK non supporta le espressioni.
> La funzione precedente, `limitless_table_alter_type_reference`, è obsoleta.

Per creare una tabella di riferimento

L'esempio seguente mostra come creare la tabella di riferimento `zipcodes`.

1. Crea la tabella standard.

```
CREATE TABLE zipcodes (zipcode INT PRIMARY KEY, details VARCHAR);
```

2. Converti la tabella standard in una tabella di riferimento.

```
CALL rds_aurora.limitless_alter_table_type_reference('public.zipcodes');

postgres=> \d

                    List of relations
 Schema |     Name     |       Type        |       Owner
--------+--------------+-------------------+--------------------
 public | customer     | partitioned table | postgres_limitless
 public | customer_fs1 | foreign table     | postgres_limitless
 public | customer_fs2 | foreign table     | postgres_limitless
 public | customer_fs3 | foreign table     | postgres_limitless
 public | customer_fs4 | foreign table     | postgres_limitless
 public | customer_fs5 | foreign table     | postgres_limitless
 public | zipcodes     | foreign table     | postgres_limitless
(7 rows)
```

   L'output mostra la tabella sottoposta a sharding `customer` e la tabella di riferimento `zipcodes`.

Schemi di esempio di Aurora PostgreSQL Limitless Database.

Forniamo i seguenti schemi di esempio per Aurora PostgreSQL Limitless Database:

- Limitless E-Commerce sample schema
- Limitless pgbench

È possibile utilizzare questi schemi per creare rapidamente un database di esempio e caricare i dati nelle tabelle di Aurora PostgreSQL Limitless Database. Per ulteriori informazioni, consulta il repository GitHub.

Caricamento dei dati in Aurora PostgreSQL Limitless Database

Puoi caricare i dati nelle tabelle di Aurora PostgreSQL Limitless Database utilizzando il comando `COPY` o l'utilità di caricamento dei dati.

> **Note**
>
> Puoi caricare dati in tabelle standard, sottoposte a sharding e di riferimento.

Indice

- Utilizzo del comando COPY con Aurora PostgreSQL Limitless Database
  - Utilizzo del comando COPY per caricare dati in Aurora PostgreSQL Limitless Database
    - Divisione dei dati in più file
  - Utilizzo del comando COPY per copiare dati di Limitless Database in un file
- Utilizzo dell'utility di caricamento dei dati di Aurora PostgreSQL Limitless Database
  - Limitazioni
  - Prerequisiti
  - Preparazione del database di origine
  - Preparazione del database di destinazione
  - Creazione delle credenziali di database
    - Creazione delle credenziali del database di origine
    - Creazione delle credenziali del database di destinazione
  - Configurazione dell'autenticazione del database e dell'accesso alle risorse tramite script
    - Script di configurazione per l'utilità di caricamento dei dati
    - Output dello script di configurazione dell'utilità di caricamento dei dati
    - Pulizia delle risorse non riuscite
  - Configurazione manuale dell'autenticazione del database e dell'accesso alle risorse
    - Creazione della soluzione gestita dal cliente AWS KMS key
    - Creazione dei segreti del database
    - Creazione del ruolo IAM
    - Aggiornamento della versione gestita dal cliente AWS KMS key
    - Aggiunta delle policy di autorizzazione del ruolo IAM

- Caricamento dei dati da un cluster di database Aurora PostgreSQL o da un'istanza database RDS per PostgreSQL
- Monitoraggio del caricamento dei dati
  - Elenco dei processi di caricamento dei dati
  - Visualizzazione dei dettagli dei processi di caricamento dei dati tramite l'ID di processo
  - Monitoraggio del gruppo di log di Amazon CloudWatch
  - Monitoraggio degli eventi RDS
- Annullamento del caricamento dei dati

Utilizzo del comando COPY con Aurora PostgreSQL Limitless Database

Puoi utilizzare la funzionalità\copy nell'utilità `psql` per importare ed esportare dati in Aurora PostgreSQL Limitless Database.

Utilizzo del comando COPY per caricare dati in Aurora PostgreSQL Limitless Database

Aurora PostgreSQL Limitless Database è compatibile con la funzionalità \copy nell'utilità `psql` per l'importazione dei dati.

In Limitless Database come in Aurora PostgreSQL, quanto segue non è supportato:

- Accesso SSH diretto alle istanze database: non puoi copiare un file di dati (ad esempio in formato .csv) sull'host dell'istanza database ed eseguire `COPY` dal file.
- Utilizzo di file locali sull'istanza database: utilizza `COPY ... FROM STDIN` e `COPY ... TO STDOUT`.

Il comando `COPY` in PostgreSQL offre opzioni per lavorare con file locali (`FROM/TO`) e trasmettere dati utilizzando una connessione tra il client e il server (`STDIN/STDOUT`). Per ulteriori informazioni, consulta COPY nella documentazione di PostgreSQL.

Il comando `\copy` nell'utilità `psql` PostgreSQL funziona con i file locali sul computer in cui viene eseguito il client `psql`. Invoca il rispettivo comando `COPY ... FROM STDIN` o `COPY ... FROM STDOUT` sul server remoto (ad esempio, Limitless Database) a cui effettui la connessione. Legge i dati dal file locale verso `STDIN` o li scrive da `STDOUT`.

Divisione dei dati in più file

I dati vengono archiviati su più shard in Aurora PostgreSQL Limitless Database. Per velocizzare il caricamento dei dati utilizzando `\copy`, puoi suddividere i dati in più file. Quindi importa in modo indipendente ogni file indipendentemente eseguendo comandi `\copy` separati in parallelo.

Ad esempio, hai un file di dati di input in formato CSV con 3 milioni di righe da importare. Puoi dividere il file in blocchi contenenti ciascuno 200.000 righe (15 blocchi):

```
split -l200000 data.csv data_ --additional-suffix=.csv -d
```

Questo genera i file da `data_00.csv` fino a `data_14.csv`. Quindi, puoi importare i dati utilizzando 15 comandi `\copy` paralleli, ad esempio:

```
psql -h dbcluster.limitless-111122223333.aws-region.rds.amazonaws.com -U username -c
 "\copy test_table from '/tmp/data_00.csv';" postgres_limitless &
psql -h dbcluster.limitless-111122223333.aws-region.rds.amazonaws.com -U username -c
 "\copy test_table FROM '/tmp/data_01.csv';" postgres_limitless &
...
psql -h dbcluster.limitless-111122223333.aws-region.rds.amazonaws.com -U username -c
 "\copy test_table FROM '/tmp/data_13.csv';" postgres_limitless &
psql -h dbcluster.limitless-111122223333.aws-region.rds.amazonaws.com -U username -c
 "\copy test_table FROM '/tmp/data_14.csv';" postgres_limitless
```

Utilizzando questa tecnica, la stessa quantità di dati viene importata circa 10 volte più velocemente rispetto all'utilizzo di un singolo comando `\copy`.

Utilizzo del comando COPY per copiare dati di Limitless Database in un file

Puoi utilizzare il comando\copy per copiare dati da una tabella di Limitless in un file, come mostrato nell'esempio seguente:

```
postgres_limitless=> \copy test_table TO '/tmp/test_table.csv' DELIMITER ',' CSV
 HEADER;
```

Utilizzo dell'utility di caricamento dei dati di Aurora PostgreSQL Limitless Database

Aurora fornisce un'utilità per caricare direttamente i dati in Limitless Database da un cluster di database Aurora PostgreSQL o da un'istanza database RDS per PostgreSQL.

Per utilizzare l'utilità di caricamento dati, esegui i seguenti passaggi:

1. Prerequisiti
2. Preparazione del database di origine
3. Preparazione del database di destinazione
4. Creazione delle credenziali di database
5. Una delle seguenti:
   - Configurazione dell'autenticazione del database e dell'accesso alle risorse tramite script (consigliato)
   - Configurazione manuale dell'autenticazione del database e dell'accesso alle risorse
6. Caricamento dei dati da un cluster di database Aurora PostgreSQL o da un'istanza database RDS per PostgreSQL

Limitazioni

L'utilità di caricamento dei dati presenta le seguenti limitazioni:

- I seguenti tipi di dati non sono supportati: `enum`, `ARRAY`, `BOX`, `CIRCLE`, `LINE`, `LSEG`, `PATH`, `PG_LSN`, `PG_SNAPSHOT`, `POLYGON`, `TSQUERY`, `TSVECTOR` e `TXID_SNAPSHOT`.
- Gli zeri iniziali (`0`) vengono rimossi dal tipo di dati `VARBIT` durante il caricamento.
- La migrazione dei dati non riesce se nelle tabelle di destinazione sono presenti chiavi esterne.
- Il caricamento dei dati dai cluster di database Multi-AZ RDS per PostgreSQL non è supportato.

Prerequisiti

L'utilità di caricamento dei dati presenta i seguenti prerequisiti:

- Il database di origine utilizza Aurora PostgreSQL o RDS per PostgreSQL versione 11.x o superiore.

- Il database di origine si trova nello stesso Account AWS e nella stessa Regione AWS del gruppo di shard del database di destinazione.
- Il cluster di database o l'istanza database di origine si trova nello stato `available`.
- Le tabelle nel database di origine e nel database di Limitless devono avere gli stessi nomi di tabella, nomi di colonna e tipi di dati di colonna.
- Le tabelle di origine e di destinazione devono avere chiavi primarie che utilizzano le stesse colonne e lo stesso ordine delle colonne.
- Devi disporre di un ambiente per la connessione a Limitless Database per eseguire i comandi di caricamento dei dati. I comandi disponibili sono i seguenti:
  - `rds_aurora.limitless_data_load_start`
  - `rds_aurora.limitless_data_load_cancel`
- Per CDC:
  - Sia il database di origine che il gruppo di shard del database di destinazione devono utilizzare lo stesso gruppo di sottoreti di database, lo stesso gruppo di sicurezza VPC e la stessa porta di database. Queste configurazioni servono per stabilire connessioni di rete sia con il database di origine che con i router nel gruppo di shard del database.
  - Devi abilitare la replica logica nel database di origine. L'utente del database di origine deve disporre dei privilegi per leggere la replica logica.

Preparazione del database di origine

Per accedere al database di origine per il caricamento dei dati, devi permettere il traffico di rete in entrata. Esegui questa procedura.

Per consentire il traffico di rete verso il database di origine

1. Accedi a Console di gestione AWS e apri la console Amazon EC2 all'indirizzo https://console.aws.amazon.com/ec2/.
2. Vai alla pagina Gruppi di sicurezza.
3. Scegli l'ID gruppo di sicurezza per il gruppo di sicurezza utilizzato dall'istanza database e dal cluster di database di origine.

   Ad esempio, l'ID gruppo di sicurezza è `sg-056a84f1712b77926`.
4. Nella scheda Regole in entrata:

a. Sceglere Edit inbound rules (Modifica regole in entrata).

b. Aggiungi una nuova regola in entrata per l'istanza database e il cluster di database di origine:

- Intervallo porte: porta del database per il database di origine, in genere `5432`
- ID gruppo di sicurezza: `sg-056a84f1712b77926` in questo esempio



5. Nella scheda Regole in uscita:

a. Scegli Edit outbound rules (Modifica regole in uscita).

b. Aggiungi una nuova regola in uscita per l'istanza database o il cluster di database di origine:

- Porta del database: `All traffic` (include le porte `0-65535`)
- ID gruppo di sicurezza: `sg-056a84f1712b77926` in questo esempio

6. Accedi alla Console di gestione AWS e apri la console Amazon VPC all'indirizzo https://console.aws.amazon.com/vpc/.

7. Vai alla pagina Liste di controllo degli accessi di rete.

8. Aggiungi la configurazione ACL di rete predefinita come indicato in ACL di rete predefinito.

Preparazione del database di destinazione

Segui le procedure descritte in Creazione di tabelle di Aurora PostgreSQL Limitless Database per creare le tabelle di destinazione nel gruppo di shard del database.

Le tabelle di destinazione devono avere gli stessi schemi, gli stessi nomi di tabella e le stesse chiavi primarie delle tabelle di origine.

Creazione delle credenziali di database

Devi creare utenti di database nei database di origine e di destinazione e concedere i privilegi necessari a tali utenti. Per ulteriori informazioni, consulta CREATE USER e GRANT nella documentazione di PostgreSQL.

Creazione delle credenziali del database di origine

L'utente del database di origine viene passato nel comando di avvio del caricamento. Questo utente deve disporre dei privilegi necessari per eseguire la replica dal database di origine.

1. Utilizza l'utente master del database (o un altro utente con il ruolo `rds_superuser`) per creare un utente del database di origine con privilegi `LOGIN`.

   ```
   CREATE USER source_db_username WITH PASSWORD 'source_db_user_password';
   ```

2. Autorizza il ruolo `rds_superuser` per l'utente del database di origine.

   ```
   GRANT rds_superuser to source_db_username;
   ```

3. Se utilizzi la modalità `full_load_and_cdc`, concedi il ruolo `rds_replication` all'utente del database di origine. Il ruolo `rds_replication` fornisce le autorizzazioni per gestire gli slot logici e per eseguire lo streaming dei dati utilizzando gli slot logici.

   ```
   GRANT rds_replication to source_db_username;
   ```

Creazione delle credenziali del database di destinazione

L'utente del database di destinazione deve disporre dell'autorizzazione per scrivere nelle tabelle di destinazione del gruppo di shard del database.

1. Utilizza l'utente master del database (o un altro utente con il ruolo `rds_superuser`) per creare un utente del database di destinazione con privilegi `LOGIN`.

```
CREATE USER destination_db_username WITH PASSWORD 'destination_db_user_password';
```

2. Autorizza il ruolo `rds_superuser` per l'utente del database di destinazione.

```
GRANT rds_superuser to destination_db_username;
```

Configurazione dell’autenticazione del database e dell’accesso alle risorse tramite script

Lo script di configurazione crea un ruolo gestito dal cliente AWS KMS key, un ruolo AWS Identity and Access Management (IAM) e due Gestione dei segreti AWS segreti.

Per utilizzare lo script di configurazione, attieniti ai passaggi seguenti:

1. Assicurati di averlo AWS CLI installato e configurato con le tue Account AWS credenziali.
2. Installa il processore JSON a riga di comando `jq`. Per ulteriori informazioni, consulta jqlang/jq.
3. Copia il file data_loading_script.zip sul computer ed estrai il file `data_load_aws_setup_script.sh` in esso contenuto.
4. Modifica lo script per sostituire le variabili segnaposto con i valori appropriati per quanto segue:
   - Le tue Account AWS
   - La Regione AWS
   - Credenziali del database di origine
   - Credenziali del database di destinazione
5. Apri un nuovo terminale sul computer ed esegui il comando seguente:

```
bash ./data_load_aws_setup_script.sh
```

Script di configurazione per l’utilità di caricamento dei dati

Forniamo qui il testo del file `data_load_aws_setup_script.sh` a titolo di riferimento.

```
#!/bin/bash
# Aurora Limitless data loading - AWS resources setup script #
# Set up the account credentials in advance. #
# Update the following script variables. #

###################################
#### Start of variable section ####

ACCOUNT_ID="12-digit_AWS_account_ID"
REGION="AWS_Region"
DATE=$(date +'%m%d%H%M%S')
RANDOM_SUFFIX="${DATE}"
SOURCE_SECRET_NAME="secret-source-${DATE}"
```

```
SOURCE_USERNAME="source_db_username"
SOURCE_PASSWORD="source_db_password"
DESTINATION_SECRET_NAME="secret-destination-${DATE}"
DESTINATION_USERNAME="destination_db_username"
DESTINATION_PASSWORD="destination_db_password"
DATA_LOAD_IAM_ROLE_NAME="aurora-data-loader-${RANDOM_SUFFIX}"
TMP_WORK_DIR="./tmp_data_load_aws_resource_setup/"

#### End of variable section ####
################################

# Main logic start
echo "DATE - [${DATE}]"
echo "RANDOM_SUFFIX - [${RANDOM_SUFFIX}]"
echo 'START!'

mkdir -p $TMP_WORK_DIR

# Create the symmetric KMS key for encryption and decryption.
TMP_FILE_PATH="${TMP_WORK_DIR}tmp_create_key_response.txt"
aws kms create-key --region $REGION | tee $TMP_FILE_PATH
KMS_KEY_ARN=$(cat $TMP_FILE_PATH | jq -r '.KeyMetadata.Arn')
aws kms create-alias \
    --alias-name alias/"${DATA_LOAD_IAM_ROLE_NAME}-key" \
    --target-key-id $KMS_KEY_ARN \
    --region $REGION

# Create the source secret.
TMP_FILE_PATH="${TMP_WORK_DIR}tmp_create_source_secret_response.txt"
aws secretsmanager create-secret \
    --name $SOURCE_SECRET_NAME \
    --kms-key-id $KMS_KEY_ARN \
    --secret-string "{\"username\":\"$SOURCE_USERNAME\",\"password\":\"$SOURCE_PASSWORD
\"}" \
    --region $REGION \
    | tee $TMP_FILE_PATH
SOURCE_SECRET_ARN=$(cat $TMP_FILE_PATH | jq -r '.ARN')

# Create the destination secret.
TMP_FILE_PATH="${TMP_WORK_DIR}tmp_create_destination_secret_response.txt"
aws secretsmanager create-secret \
    --name $DESTINATION_SECRET_NAME \
    --kms-key-id $KMS_KEY_ARN \
```

```
    --secret-string "{\"username\":\"$DESTINATION_USERNAME\",\"password\":
\"$DESTINATION_PASSWORD\"}" \
    --region $REGION \
    | tee $TMP_FILE_PATH
DESTINATION_SECRET_ARN=$(cat $TMP_FILE_PATH | jq -r '.ARN')

# Create the RDS trust policy JSON file.
# Use only rds.amazonaws.com for RDS PROD use cases.
TRUST_POLICY_PATH="${TMP_WORK_DIR}rds_trust_policy.json"
echo '{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Principal": {
                "Service": [
                    "rds.amazonaws.com"
                ]
            },
            "Action": "sts:AssumeRole"
        }
    ]
}' > $TRUST_POLICY_PATH

# Create the IAM role.
TMP_FILE_PATH="${TMP_WORK_DIR}tmp_create_iam_role_response.txt"
aws iam create-role \
    --role-name $DATA_LOAD_IAM_ROLE_NAME \
    --assume-role-policy-document "file://${TRUST_POLICY_PATH}" \
    --tags Key=assumer,Value=aurora_limitless_table_data_load \
    --region $REGION \
    | tee $TMP_FILE_PATH
IAM_ROLE_ARN=$(cat $TMP_FILE_PATH | jq -r '.Role.Arn')

# Create the permission policy JSON file.
PERMISSION_POLICY_PATH="${TMP_WORK_DIR}data_load_permission_policy.json"
permission_json_policy=$(cat &lt;&lt;EOF
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "Ec2Permission",
            "Effect": "Allow",
            "Action": [
```

```
                "ec2:DescribeNetworkInterfaces",
                "ec2:CreateNetworkInterface",
                "ec2:DeleteNetworkInterface",
                "ec2:CreateNetworkInterfacePermission",
                "ec2:DeleteNetworkInterfacePermission",
                "ec2:DescribeNetworkInterfacePermissions",
                "ec2:ModifyNetworkInterfaceAttribute",
                "ec2:DescribeNetworkInterfaceAttribute",
                "ec2:DescribeAvailabilityZones",
                "ec2:DescribeRegions",
                "ec2:DescribeVpcs",
                "ec2:DescribeSubnets",
                "ec2:DescribeSecurityGroups",
                "ec2:DescribeNetworkAcls"
            ],
            "Resource": "*"
        },
        {
            "Sid": "SecretsManagerPermissions",
            "Effect": "Allow",
            "Action": [
                "secretsmanager:GetSecretValue",
                "secretsmanager:DescribeSecret"
            ],
            "Resource": [
                "$SOURCE_SECRET_ARN",
                "$DESTINATION_SECRET_ARN"
            ]
        },
        {
            "Sid": "KmsPermissions",
            "Effect": "Allow",
            "Action": [
                "kms:Decrypt",
                "kms:DescribeKey",
                "kms:GenerateDataKey"
            ],
            "Resource": "$KMS_KEY_ARN"
        },
        {
            "Sid": "RdsPermissions",
            "Effect": "Allow",
            "Action": [
                "rds:DescribeDBClusters",
```

```
                "rds:DescribeDBInstances"
            ],
            "Resource": "*"
        }
    ]
}
EOF
)
echo $permission_json_policy > $PERMISSION_POLICY_PATH

# Add the inline policy.
aws iam put-role-policy \
    --role-name $DATA_LOAD_IAM_ROLE_NAME \
    --policy-name aurora-limitless-data-load-policy \
    --policy-document "file://${PERMISSION_POLICY_PATH}" \
    --region $REGION

# Create the key policy JSON file.
KEY_POLICY_PATH="${TMP_WORK_DIR}data_load_key_policy.json"
key_json_policy=$(cat &lt;&lt;EOF
{
    "Id": "key-aurora-limitless-data-load-$RANDOM_SUFFIX",
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "Enable IAM User Permissions",
            "Effect": "Allow",
            "Principal": {
                "AWS": "arn:aws:iam::$ACCOUNT_ID:root"
            },
            "Action": "kms:*",
            "Resource": "*"
        },
        {
            "Sid": "Allow use of the key",
            "Effect": "Allow",
            "Principal": {
                "AWS": "$IAM_ROLE_ARN"
            },
            "Action": [
                "kms:Decrypt",
                "kms:DescribeKey",
                "kms:GenerateDataKey"
            ],
```

```
            "Resource": "*"
        }
    ]
}
EOF
)
echo $key_json_policy > $KEY_POLICY_PATH

# Add the key policy.
TMP_FILE_PATH="${TMP_WORK_DIR}tmp_put_key_policy_response.txt"
sleep 10 # sleep 10 sec for IAM role ready
aws kms put-key-policy \
    --key-id $KMS_KEY_ARN \
    --policy-name default \
    --policy "file://${KEY_POLICY_PATH}" \
    --region $REGION \
    | tee $TMP_FILE_PATH

echo 'DONE!'

echo "ACCOUNT_ID : [${ACCOUNT_ID}]"
echo "REGION : [${REGION}]"
echo "RANDOM_SUFFIX : [${RANDOM_SUFFIX}]"
echo "IAM_ROLE_ARN : [${IAM_ROLE_ARN}]"
echo "SOURCE_SECRET_ARN : [${SOURCE_SECRET_ARN}]"
echo "DESTINATION_SECRET_ARN : [${DESTINATION_SECRET_ARN}]"

# Example of a successful run:
# ACCOUNT_ID : [012345678912]
# REGION : [ap-northeast-1]
# RANDOM_SUFFIX : [0305000703]
# IAM_ROLE_ARN : [arn:aws:iam::012345678912:role/aurora-data-loader-0305000703]
# SOURCE_SECRET_ARN : [arn:aws:secretsmanager:ap-
northeast-1:012345678912:secret:secret-source-0305000703-yQDtow]
# DESTINATION_SECRET_ARN : [arn:aws:secretsmanager:ap-
northeast-1:012345678912:secret:secret-destination-0305000703-5d5Jy8]

# If you want to manually clean up failed resource,
# please remove them in the following order:
# 1. IAM role.
 # aws iam delete-role-policy --role-name Test-Role --policy-name ExamplePolicy --
region us-east-1
# aws iam delete-role --role-name Test-Role --region us-east-1
# 2. Source and destination secrets.
```

```
# aws secretsmanager delete-secret --secret-id MyTestSecret --force-delete-without-
recovery --region us-east-1
# 3. KDM key.
# aws kms schedule-key-deletion --key-id arn:aws:kms:us-
east-1:123456789012:key/1234abcd-12ab-34cd-56ef-1234567890ab --pending-window-in-days 7
 --region us-east-1
```

Output dello script di configurazione dell'utilità di caricamento dei dati

L'esempio seguente mostra l'output di un'esecuzione riuscita dello script.

```
% bash ./data_load_aws_setup_script.sh
DATE - [0305000703]
RANDOM_SUFFIX - [0305000703]
START!
{
    "KeyMetadata": {
        "AWSAccountId": "123456789012",
        "KeyId": "1234abcd-12ab-34cd-56ef-1234567890ab",
        "Arn": "arn:aws:kms:ap-
northeast-1:123456789012:key/1234abcd-12ab-34cd-56ef-1234567890ab",
        "CreationDate": "2024-03-05T00:07:49.852000+00:00",
        "Enabled": true,
        "Description": "",
        "KeyUsage": "ENCRYPT_DECRYPT",
        "KeyState": "Enabled",
        "Origin": "AWS_KMS",
        "KeyManager": "CUSTOMER",
        "CustomerMasterKeySpec": "SYMMETRIC_DEFAULT",
        "KeySpec": "SYMMETRIC_DEFAULT",
        "EncryptionAlgorithms": [
            "SYMMETRIC_DEFAULT"
        ],
        "MultiRegion": false
    }
}
{
    "ARN": "arn:aws:secretsmanager:ap-northeast-1:123456789012:secret:secret-
source-0305000703-yQDtow",
    "Name": "secret-source-0305000703",
    "VersionId": "a017bebe-a71b-4220-b923-6850c2599c26"
}
{
```

```
    "ARN": "arn:aws:secretsmanager:ap-northeast-1:123456789012:secret:secret-
destination-0305000703-5d5Jy8",
    "Name": "secret-destination-0305000703",
    "VersionId": "32a1f989-6391-46b1-9182-f65d242f5eb6"
}
{
    "Role": {
        "Path": "/",
        "RoleName": "aurora-data-loader-0305000703",
        "RoleId": "AROAYPX63ITQOYORQSC6U",
        "Arn": "arn:aws:iam::123456789012:role/aurora-data-loader-0305000703",
        "CreateDate": "2024-03-05T00:07:54+00:00",
        "AssumeRolePolicyDocument": {
           "Version": "2012-10-17",
            "Statement": [
                {
                    "Effect": "Allow",
                    "Principal": {
                        "Service": [
                            "rds.amazonaws.com"
                        ]
                    },
                    "Action": "sts:AssumeRole"
                }
            ]
        },
        "Tags": [
            {
                "Key": "assumer",
                "Value": "aurora_limitless_table_data_load"
            }
        ]
    }
}
DONE!
ACCOUNT_ID : [123456789012]
REGION : [ap-northeast-1]
RANDOM_SUFFIX : [0305000703]
IAM_ROLE_ARN : [arn:aws:iam::123456789012:role/aurora-data-loader-0305000703]
SOURCE_SECRET_ARN : [arn:aws:secretsmanager:ap-northeast-1:123456789012:secret:secret-
source-0305000703-yQDtow]
DESTINATION_SECRET_ARN : [arn:aws:secretsmanager:ap-
northeast-1:123456789012:secret:secret-destination-0305000703-5d5Jy8]
```

Pulizia delle risorse non riuscite

Se desideri pulire manualmente le risorse non riuscite, rimuovile nell'ordine seguente:

1. Ruolo IAM, ad esempio:

```
aws iam delete-role-policy \
--role-name Test-Role \
--policy-name ExamplePolicy

aws iam delete-role \
--role-name Test-Role
```

2. Segreti di origine e destinazione, ad esempio:

```
aws secretsmanager delete-secret \
--secret-id MyTestSecret \
--force-delete-without-recovery
```

3. Chiave KMS, ad esempio:

```
aws kms schedule-key-deletion \
--key-id arn:aws:kms:us-west-2:123456789012:key/1234abcd-12ab-34cd-56ef-1234567890ab
 \
--pending-window-in-days 7
```

Quindi puoi riprovare a eseguire lo script.

Configurazione manuale dell’autenticazione del database e dell’accesso alle risorse

Il processo manuale per configurare l’autenticazione del database e l’accesso alle risorse prevede i seguenti passaggi:

1. Creazione della soluzione gestita dal cliente AWS KMS key
2. Aggiunta delle policy di autorizzazione del ruolo IAM
3. Creazione dei segreti del database
4. Creazione del ruolo IAM
5. Aggiornamento della versione gestita dal cliente AWS KMS key

Questo processo è facoltativo ed esegue le stesse attività descritte in Configurazione dell’autenticazione del database e dell’accesso alle risorse tramite script. Consigliamo di utilizzare lo script.

Creazione della soluzione gestita dal cliente AWS KMS key

Segui le procedure descritte in Creazione di chiavi di crittografia simmetriche per creare una chiave KMS gestita dal cliente. Puoi anche utilizzare una chiave esistente, se soddisfa questi requisiti.

Per creare una chiave KMS gestita dal cliente

1. Accedi a Console di gestione AWS e apri la AWS KMS console su https://console.aws.amazon.com /kms.
2. Vai alla pagina Chiavi gestite dal cliente.
3. Scegli Crea chiave.
4. Nella pagina Configura chiave:
    a. Per Tipo di chiave, seleziona Simmetrica.
    b. In Utilizzo delle chiavi, scegli Crittografa e decrittografa.
    c. Scegli Next (Successivo).
5. Nella pagina Aggiungi etichette, inserisci un alias, ad esempio **`limitless`**, quindi scegli Successivo.
6. Nella pagina Definisci le autorizzazioni per gestire la chiave, assicurati che la casella di controllo Consenti agli amministratori delle chiavi di eliminare questa chiave sia selezionata, quindi scegli Successivo.

7. Nella pagina Definisci le autorizzazioni per utilizzare le chiavi scegli Avanti.
8. Nella pagina Review (Rivedi), scegliere Finish (Fine).

   Aggiornerai la policy della chiave in un secondo momento.

Annota il nome della risorsa Amazon (ARN) della chiave KMS da utilizzare in Aggiunta delle policy di autorizzazione del ruolo IAM.

Per informazioni sull'utilizzo di per AWS CLI creare la chiave KMS gestita dal cliente, consulta create-key e create-alias.

Creazione dei segreti del database

Per consentire all'utilità di caricamento dei dati di accedere alle tabelle del database di origine e di destinazione, è necessario creare due segreti inGestione dei segreti AWS: uno per il database di origine e uno per il database di destinazione. Questi segreti archiviano i nomi utente e le password per accedere ai database di origine e di destinazione.

Segui le procedure descritte in Creazione di un segreto Gestione dei segreti AWS per creare i segreti della coppia chiave-valore.

Per creare i segreti del database

1. Apri la console Secrets Manager all'indirizzo https://console.aws.amazon.com/secretsmanager/.
2. Scegli Archivia un nuovo segreto.
3. Nella pagina Scegli il tipo di segreto:
   a. Per Tipo di segreto, scegli Altro tipo di segreto.
   b. Per Coppie chiave/valore, scegli la scheda Testo in chiaro.
   c. Inserisci il seguente codice JSON, dove *sourcedbreader* e *sourcedbpassword* sono le credenziali dell'utente del database di origine da Creazione delle credenziali del database di origine.

```
{
    "username":"sourcedbreader",
    "password":"sourcedbpassword"
}
```

   d. Per Chiave di crittografia, scegli la chiave KMS in cui hai creato Creazione della soluzione gestita dal cliente AWS KMS key, ad esempio `limitless`.
   e. Scegli Next (Successivo).
4. Nella pagina Configura del segreto, inserisci un nome del segreto, ad esempio **`source_DB_secret`** e scegli Successivo.
5. Nella pagina Configura rotazione – facoltativo, scegli Successivo.
6. Nella pagina Review (Rivedi), scegli Store (Archivia).
7. Ripeti la procedura per il segreto del database di destinazione:

   a. Inserisci il seguente codice JSON, dove *`destinationdbwriter`* e *`destinationdbpassword`* sono le credenziali dell'utente del database di destinazione da Creazione delle credenziali del database di destinazione.

```
{
    "username":"destinationdbwriter",
    "password":"destinationdbpassword"
}
```

   b. Inserisci un nome del segreto, ad esempio **`destination_DB_secret`**.

Registra ARNs i segreti da utilizzareAggiunta delle policy di autorizzazione del ruolo IAM.

Creazione del ruolo IAM

Il caricamento dei dati richiede l'accesso alle AWS risorse. Per fornire l'accesso, crea il ruolo IAM `aurora-data-loader` seguendo le procedure descritte in Creazione di un ruolo per delegare le autorizzazioni a un utente IAM.

Per creare il ruolo IAM

1. Accedi Console di gestione AWS e apri la console IAM all'indirizzo https://console.aws.amazon.com/iam/.
2. Vai alla pagina Ruoli.
3. Scegli Crea ruolo.
4. Nella pagina Seleziona un'entità attendibile:

   a. Per Tipo di entità attendibile, seleziona Policy di attendibilità personalizzata.

b. Inserisci il seguente codice JSON per la policy di attendibilità personalizzata:

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Principal": {
                "Service": [
                    "rds.amazonaws.com"
                ]
            },
            "Action": "sts:AssumeRole"
        }
    ]
}
```

c. Scegli Next (Successivo).

5. Nella pagina Add permissions (Aggiungi autorizzazioni), scegli Next (Successivo).
6. Nella pagina Nomina, verifica e crea:

a. Per Nome del ruolo, inserisci **`aurora-data-loader`** o un altro nome a scelta.

b. Scegli Aggiungi tage inserisci il seguente tag:

- Chiave: **`assumer`**
- Value (Valore): **`aurora_limitless_table_data_load`**

> **Important**
>
> Aurora PostgreSQL Limitless Database può assumere solo un ruolo IAM che contiene questo tag.

c. Scegli Crea ruolo.

Aggiornamento della versione gestita dal cliente AWS KMS key

Segui le procedure descritte in Modifica di una policy della chiave per aggiungere il ruolo IAM `aurora-data-loader` alla policy della chiave predefinita.

Per aggiungere il ruolo IAM alla policy della chiave

1. Accedi a Console di gestione AWS e apri la AWS KMS console su https://console.aws.amazon.com /kms.
2. Vai alla pagina Chiavi gestite dal cliente.
3. Scegli la chiave KMS che hai creato in Creazione della soluzione gestita dal cliente AWS KMS key, ad esempio `limitless`.
4. Nella scheda Policy della chiave, per Utenti delle chiavi, scegli Aggiungi.
5. Nella finestra Aggiungi utenti chiave, seleziona il nome del ruolo IAM in cui hai creatoCreazione del ruolo IAM, ad esempio. aurora-data-loader
6. Scegliere Aggiungi.

Aggiunta delle policy di autorizzazione del ruolo IAM

Devi aggiungere policy di autorizzazione del ruolo IAM creato. In questo modo, l'utilità di caricamento dei dati di Aurora PostgreSQL Limitless Database può accedere alle risorse AWS correlate per creare connessioni di rete e recuperare i segreti delle credenziali dei database di origine e di destinazione.

Per ulteriori informazioni, consulta Modifica un ruolo.

Per aggiungere le policy di autorizzazione

1. Accedi Console di gestione AWS e apri la console IAM all'indirizzo https://console.aws.amazon.com/iam/.
2. Vai alla pagina Ruoli.
3. Scegli, ad esempioCreazione del ruolo IAM, il ruolo IAM in cui hai creato aurora-data-loader.
4. Nella scheda Autorizzazioni, per Policy di autorizzazione scegli Aggiungi autorizzazioni, quindi Crea policy inline.
5. Nella pagina Specifica autorizzazioni, seleziona l'editor JSON.
6. Copia e incolla il seguente modello nell'editor JSON, sostituendo i segnaposto con i segreti del database e ARNs la chiave KMS.

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Sid": "Ec2Permission",
            "Effect": "Allow",
            "Action": [
                "ec2:DescribeNetworkInterfaces",
                "ec2:CreateNetworkInterface",
                "ec2:DeleteNetworkInterface",
                "ec2:CreateNetworkInterfacePermission",
                "ec2:DeleteNetworkInterfacePermission",
                "ec2:DescribeNetworkInterfacePermissions",
                "ec2:ModifyNetworkInterfaceAttribute",
                "ec2:DescribeNetworkInterfaceAttribute",
                "ec2:DescribeAvailabilityZones",
                "ec2:DescribeRegions",
                "ec2:DescribeVpcs",
                "ec2:DescribeSubnets",
                "ec2:DescribeSecurityGroups",
                "ec2:DescribeNetworkAcls"
            ],
            "Resource": "*"
        },
        {
            "Sid": "SecretsManagerPermissions",
            "Effect": "Allow",
            "Action": [
                "secretsmanager:GetSecretValue",
                "secretsmanager:DescribeSecret"
            ],
            "Resource": [
                "arn:aws:secretsmanager:us-
east-1:123456789012:secret:source_DB_secret-ABC123",
                "arn:aws:secretsmanager:us-
east-1:123456789012:secret:destination_DB_secret-456DEF"
            ]
        },        {
            "Sid": "KmsPermissions",
            "Effect": "Allow",
```

```
            "Action": [
                "kms:Decrypt",
                "kms:DescribeKey",
                "kms:GenerateDataKey"
            ],
            "Resource": "arn:aws:kms:us-east-1:123456789012:key/aa11bb22-
####-####-####-fedcba123456"
        },
        {
            "Sid": "RdsPermissions",
            "Effect": "Allow",
            "Action": [
                "rds:DescribeDBClusters",
                "rds:DescribeDBInstances"
            ],
            "Resource": "*"
        }
    ]
}
```

7. Verifica la presenza di errori e correggili.
8. Scegli Next (Successivo).
9. Nella pagina Rivedi e crea, inserisci un nome della policy ad esempio **`data_loading_policy`**, quindi scegli Crea policy.

Caricamento dei dati da un cluster di database Aurora PostgreSQL o da un'istanza database RDS per PostgreSQL

Dopo aver completato la configurazione delle risorse e dell'autenticazione, connettiti all'endpoint del cluster e chiama la stored procedure `rds_aurora.limitless_data_load_start` da Limitless Database, ad esempio `postgres_limitless`. Limitless Database è un database nel gruppo di shard del database in cui desideri migrare i dati.

Questa funzione si connette in modo asincrono in background al database di origine specificato nel comando, legge i dati dall'origine e li carica sugli shard. Per prestazioni migliori, i dati vengono caricati utilizzando thread paralleli. La funzione recupera uno snapshot della tabella point-in-time eseguendo un comando `SELECT` per leggere i dati delle tabelle fornite nel comando.

Puoi caricare in tabelle sottoposte a sharding, di riferimento e standard.

Puoi caricare dati a livello di database, schema o tabella nelle chiamate `rds_aurora.limitless_data_load_start`.

- Database: puoi caricare un database alla volta in ogni chiamata, senza alcun limite per il numero di schemi o tabelle all'interno del database.
- Schema: puoi caricare un massimo di 15 schemi in ogni chiamata, senza limite per il numero di tabelle in ogni schema.
- Tabella: puoi caricare un massimo di 15 tabelle in ogni chiamata.

> **Note**
>
> Questa funzionalità non utilizza snapshot Amazon RDS né l'isolamento point-in-time del database. Per garantire la coerenza tra le tabelle, consigliamo di clonare il database di origine e utilizzare il database clonato come origine.

La stored procedure utilizza la seguente sintassi.

```
CALL rds_aurora.limitless_data_load_start('source_type',
    'source_DB_cluster_or_instance_ID',
    'source_database_name',
    'streaming_mode',
    'data_loading_IAM_role_arn',
    'source_DB_secret_arn',
    'destination_DB_secret_arn',
    'ignore_primary_key_conflict_boolean_flag',
    'is_dry_run',
    (optional parameter) schemas/tables => ARRAY['name1', 'name2', ...]);
```

I parametri di input sono i seguenti:

- `source_type`: il tipo di origine: `aurora_postgresql` o `rds_postgresql`
- `source_DB_cluster_or_instance_ID`: l'identificatore del cluster di database Aurora PostgreSQL di origine o dell'istanza database RDS per PostgreSQL di origine
- `source_database_name`: il nome del database di origine, ad esempio *`postgres`*
- `streaming_mode`: indica se includere l'acquisizione dei dati di modifica (CDC): `full_load` o `full_load_and_cdc`
- `data_loading_IAM_role_arn`: il nome della risorsa Amazon (ARN) del ruolo IAM per `aurora-data-loader`
- `source_DB_secret_arn`: l'ARN segreto del database di origine
- `destination_DB_secret_arn`: l'ARN segreto del database di destinazione
- `ignore_primary_key_conflict_boolean_flag`: indica se continuare in caso di conflitto di chiave primaria:
  - Se impostato su `true`, il caricamento dei dati ignora le nuove modifiche per le righe con conflitto di chiave primaria.
  - Se impostato su `false`, il caricamento dei dati sovrascrive le righe esistenti nelle tabelle di destinazione quando si verifica un conflitto di chiave primaria.
- `is_dry_run`: indica se verificare che il processo di caricamento dei dati possa connettersi ai database di origine e di destinazione:
  - Se impostato su `true`, verifica le connessioni senza caricare dati
  - Se impostato su `false`, carica i dati

- (Facoltativo) `schemas` o `tables`: un array di schemi o tabelle da caricare. Puoi specificare uno dei seguenti:
  - Un elenco di tabelle nel formato `tables => ARRAY['schema1.table1', 'schema1.table2', 'schema2.table1', ...]`
  - Un elenco di schemi nel formato `schemas => ARRAY['schema1', 'schema2', ...]`

  Se non includi questo parametro, l'intero database di origine specificato viene migrato.

Il parametro di output è l'ID di processo con un messaggio.

L'esempio seguente mostra come utilizzare la stored procedure `rds_aurora.limitless_data_load_start` per caricare i dati da un cluster di database Aurora PostgreSQL.

```
CALL rds_aurora.limitless_data_load_start('aurora_postgresql',
    'my-db-cluster',
    'postgres',
    'full_load_and_cdc',
    'arn:aws:iam::123456789012:role/aurora-data-loader-8f2c66',
    'arn:aws:secretsmanager:us-east-1:123456789012:secret:secret-source-8f2c66-EWrr0V',
    'arn:aws:secretsmanager:us-east-1:123456789012:secret:secret-destination-8f2c66-
d04fbD',
    'true',
    'false',
    tables => ARRAY['public.customer', 'public.order', 'public.orderdetails']);

INFO: limitless data load job id 1688761223647 is starting.
```

Monitoraggio del caricamento dei dati

Aurora PostgreSQL Limitless Database offre diversi modi per monitorare i processi di caricamento dei dati:

- Elenco dei processi di caricamento dei dati
- Visualizzazione dei dettagli dei processi di caricamento dei dati tramite l'ID di processo
- Monitoraggio del gruppo di log di Amazon CloudWatch
- Monitoraggio degli eventi RDS

Elenco dei processi di caricamento dei dati

Puoi connetterti all'endpoint del cluster e usare la visualizzazione `rds_aurora.limitless_data_load_jobs` per elencare i processi di caricamento dei dati.

```
postgres_limitless=> SELECT * FROM rds_aurora.limitless_data_load_jobs LIMIT 6;

    job_id     |  status   | message |     source_db_identifier      | source_db_name |
 full_load_complete_time |
   progress_details                                                                  |
     start_time       |   last_updated_time    |  streaming_mode   | source_engine_type
 | ignore_primary_key_conflict | is_dryrun
--------------+-----------+---------+-------------------------------+---------------
+-------------------------
+-------------------------------------------------------------------------------------------------------------------------------
+-----------------------+-----------------------+------------------
+--------------------+-----------------------------+-----------
 1725697520693 | COMPLETED |         | persistent-kdm-auto-source-01 | postgres
    | 2024-09-07 08:48:15+00  | {"FULL_LOAD": {"STATUS": "COMPLETED", "DETAILS": "9
 of 9 tables loaded", "COMPLETED_AT": "2024/09/07 08:48:15+00", "RECORDS_MIGRATED":
 600003}} | 2024-09-07 08:47:13+00 | 2024-09-07 08:48:15+00 | full_load         |
 aurora_postgresql  | t                           | f
 1725696114225 | COMPLETED |         | persistent-kdm-auto-source-01 | postgres
    | 2024-09-07 08:24:20+00  | {"FULL_LOAD": {"STATUS": "COMPLETED", "DETAILS": "3
 of 3 tables loaded", "COMPLETED_AT": "2024/09/07 08:24:20+00", "RECORDS_MIGRATED":
 200001}} | 2024-09-07 08:23:56+00 | 2024-09-07 08:24:20+00 | full_load         |
 aurora_postgresql  | t                           | f
 1725696067630 | COMPLETED |         | persistent-kdm-auto-source-01 | postgres
    | 2024-09-07 08:23:45+00  | {"FULL_LOAD": {"STATUS": "COMPLETED", "DETAILS": "6
 of 6 tables loaded", "COMPLETED_AT": "2024/09/07 08:23:45+00", "RECORDS_MIGRATED":
```

```
 400002}} | 2024-09-07 08:23:10+00 | 2024-09-07 08:23:45+00 | full_load         |
 aurora_postgresql  | t                           | f
 1725694221753 | CANCELED  |         | persistent-kdm-auto-source-01 | postgres
    |                         | {}

           | 2024-09-07 07:31:18+00 | 2024-09-07 07:51:49+00 | full_load_and_cdc |
 aurora_postgresql  | t                           | f
 1725691698210 | COMPLETED |         | persistent-kdm-auto-source-01 | postgres
    | 2024-09-07 07:10:51+00  | {"FULL_LOAD": {"STATUS": "COMPLETED", "DETAILS": "1
 of 1 tables loaded", "COMPLETED_AT": "2024/09/07 07:10:51+00", "RECORDS_MIGRATED":
 100000}} | 2024-09-07 07:10:42+00 | 2024-09-07 07:10:52+00 | full_load         |
 aurora_postgresql  | t                           | f
 1725691695049 | COMPLETED |         | persistent-kdm-auto-source-01 | postgres
    | 2024-09-07 07:10:48+00  | {"FULL_LOAD": {"STATUS": "COMPLETED", "DETAILS": "1
 of 1 tables loaded", "COMPLETED_AT": "2024/09/07 07:10:48+00", "RECORDS_MIGRATED":
 100000}} | 2024-09-07 07:10:41+00 | 2024-09-07 07:10:48+00 | full_load         |
 aurora_postgresql  | t                           | f
(6 rows)
```

I record dei processi vengono eliminati dopo 90 giorni.

Visualizzazione dei dettagli dei processi di caricamento dei dati tramite l'ID di processo

Se conosci un ID di processo, puoi connetterti all'endpoint del cluster e usare la visualizzazione `rds_aurora.limitless_data_load_job_details` per visualizzare i dettagli di tale processo, inclusi il nome della tabella, lo stato del processo e il numero di righe caricate. Puoi ottenere l'ID di processo nelle risposte alle funzioni di avvio del caricamento dei dati o dalla visualizzazione `rds_aurora.limitless_data_load_jobs`.

```
postgres_limitless=> SELECT * FROM rds_aurora.limitless_data_load_job_details WHERE
 job_id='1725696114225';

job_id        | destination_table_name | destination_schema_name | start_time
      | status    | full_load_rows | full_load_total_rows | full_load_complete_time |
 cdc_insert | cdc_update | cdc_delete
--------------+------------------------+-------------------------
+------------------------+-----------+----------------+---------------------
+-------------------------+------------+------------+------------
1725696114225 | standard_1             | public                  | 2024-09-07
 08:23:57+00 | COMPLETED | 100000         | 100000              | 2024-09-07
 08:24:08+00  | 0          | 0          | 0
```

```
1725696114225 | standard_2             | public                  | 2024-09-07
 08:24:08+00 | COMPLETED | 100000         | 100000               | 2024-09-07
 08:24:17+00  | 0          | 0          | 0
1725696114225 | standard_3             | public                  | 2024-09-07
 08:24:18+00 | COMPLETED | 1              | 1                    | 2024-09-07
 08:24:20+00  | 0          | 0          | 0
1725696114225 | standard_4             | public                  | 2024-09-07
 08:23:58+00 | PENDING   | 0              | 0                    |
    | 0          | 0          | 0
(4 rows)
```

I record dei processi vengono eliminati dopo 90 giorni.

Monitoraggio del gruppo di log di Amazon CloudWatch

Dopo che lo stato del processo di caricamento dati passa a `RUNNING`, puoi controllare l'avanzamento in tempo reale utilizzando Amazon CloudWatch Logs.

Per monitorare i flussi di log di CloudWatch

Accedi al Console di gestione AWS e apri la console CloudWatch all'indirizzo https://console.aws.amazon.com/cloudwatch/.

1. Vai a Log, quindi a Gruppi di log.
2. Scegli il gruppo di log /aws/rds/aurora-limitless-database.
3. Cerca il flusso di log del processo di caricamento dei dati tramite job_id.

   Il flusso di log ha il modello Data-Load-Job-***job_id***.
4. Scegli il flusso di log per visualizzare gli eventi di log.

Ogni flusso di log mostra eventi contenenti lo stato del processo e il numero di righe caricate nelle tabelle di destinazione di Aurora PostgreSQL Limitless Database. Se un processo di caricamento dei dati non riesce, viene creato anche un log di errore che mostra lo stato dell'errore e il motivo.

I record dei processi vengono eliminati dopo 90 giorni.

Monitoraggio degli eventi RDS

Il processo di caricamento dei dati pubblica anche eventi RDS, ad esempio quando un processo ha esito positivo, negativo o viene annullato. Puoi visualizzare gli eventi dal database di destinazione.

Per ulteriori informazioni, consulta Eventi gruppo shard database.

Annullamento del caricamento dei dati

Per annullare un processo di caricamento dei dati, chiama la stored procedure `rds_aurora.limitless_data_load_cancel`, con l'ID di processo come parametro di input. Questa stored procedure viene richiamata dallo stesso database nel gruppo di shard del database in cui è stato avviato lo specifico processo di caricamento dei dati. Per esempio:

```
CALL rds_aurora.limitless_data_load_cancel(12345);

INFO: limitless data load job with id 12345 is canceling without rollback.
```

Non puoi annullare un processo di caricamento dati che non esiste o che non è in esecuzione nello stesso gruppo di shard del database.

L'utilità di caricamento dati di Aurora PostgreSQL Limitless Database lascia i dati caricati nelle tabelle di destinazione senza eseguire il rollback, come indicato nella risposta. Se non desideri conservare i dati caricati, ti consigliamo di troncare le tabelle di destinazione.

Esecuzione di query su Aurora PostgreSQL Limitless Database

Aurora PostgreSQL Limitless Database è compatibile con la sintassi PostgreSQL per le query. È possibile eseguire query su Limitless Database utilizzando `psql` o qualsiasi altra utilità di connessione che funzioni con PostgreSQL. Per eseguire le query, connettersi all'endpoint illimitato come mostrato in Connessione al cluster di database Aurora PostgreSQL Limitless DB.

Tutte le query `SELECT` PostgreSQL sono supportate in Aurora PostgreSQL Limitless Database. Tuttavia, le query vengono eseguite su due livelli:

1. Router a cui il client invia la query
2. Shard dove si trovano i dati

Le prestazioni dipendono dall'esecuzione delle query sul database in modo da consentirgli di ottenere un elevato grado di elaborazione simultanea di diverse query su shard diversi. Le query vengono prima analizzate nel livello di transazione distribuito (router). Prima di pianificare l'esecuzione della query, è prevista una fase di analisi per identificare la posizione di tutte le relazioni che partecipano alla query. Se tutte le relazioni sono tabelle sottoposte a sharding con una chiave shard filtrata sullo stesso shard o tabelle di riferimento, la pianificazione delle query viene ignorata sul livello del router e trasmessa completamente allo shard per la pianificazione e l'esecuzione. Questo processo riduce il numero di round trip tra diversi nodi (router e shard) e si traduce in prestazioni migliori nella maggior parte dei casi. Per ulteriori informazioni, consulta Query a shard singolo in Aurora PostgreSQL Limitless Database..

> **Note**
>
> In alcuni casi specifici, ad esempio in un prodotto cartesiano (cross join), la query offre prestazioni migliori recuperando i dati separatamente dallo shard.

Per ulteriori informazioni sui piani di esecuzione delle query, consulta EXPLAIN in Riferimento ad Aurora PostgreSQL Limitless Database. Per ulteriori informazioni sulle query, consulta Query nella documentazione di PostgreSQL.

Argomenti

- Query a shard singolo in Aurora PostgreSQL Limitless Database.
- Query distribuite in Aurora PostgreSQL Limitless Database

- Tracciamento distribuito delle query nei log di PostgreSQL in Aurora PostgreSQL Limitless Database
- Deadlock distribuiti in Aurora PostgreSQL Limitless Database

Query a shard singolo in Aurora PostgreSQL Limitless Database.

Una query a shard singolo è una query che può essere eseguita direttamente su uno shard mantenendo la semantica SQL ACID. Quando rileva una query di questo tipo sul router, il pianificatore di query procede a inviare l'intera query SQL allo shard corrispondente.

Questa ottimizzazione riduce il numero di andate e ritorni di rete dal router allo shard, migliorando le prestazioni. Attualmente questa ottimizzazione viene eseguita per `INSERT`, `SELECT`, `UPDATE` e per le query `DELETE`.

Argomenti

- Esempi di query a shard singolo
- Restrizioni per le query a shard singolo
- Join completamente qualificati (espliciti)
- Impostazione di una chiave di shard attiva

Esempi di query a shard singolo

Negli esempi seguenti, viene mostrata la tabella condivisa `customers`, con la chiave di shard `customer_id`, e la tabella di riferimento `zipcodes`.

SELECT

```
postgres_limitless=> EXPLAIN (VERBOSE, COSTS OFF) SELECT * FROM customers WHERE
 customer_id = 100;

                        QUERY PLAN
-----------------------------------------------------------
 Foreign Scan
   Output: customer_id, other_id, customer_name, balance
   Remote SQL:  SELECT customer_id,
     other_id,
     customer_name,
     balance
```

```
    FROM public.customers
   WHERE (customer_id = 100)
 Single Shard Optimized
(9 rows)
```

```
postgres_limitless=> EXPLAIN (VERBOSE, COSTS OFF) SELECT * FROM orders
    LEFT JOIN zipcodes ON orders.zipcode_id = zipcodes.zipcode_id
    WHERE customer_id = 11;

                                                QUERY PLAN

-----------------------------------------------------------------------------------------------------
 Foreign Scan
   Output: customer_id, order_id, zipcode_id, customer_name, balance,
 zipcodes.zipcode_id, zipcodes.city
   Remote SQL:  SELECT orders.customer_id,
     orders.order_id,
     orders.zipcode_id,
     orders.customer_name,
     orders.balance,
     zipcodes.zipcode_id,
     zipcodes.city
    FROM (public.orders
      LEFT JOIN public.zipcodes ON ((orders.zipcode_id = zipcodes.zipcode_id)))
   WHERE (orders.customer_id = 11)
 Single Shard Optimized
(13 rows)
```

INSERT

```
postgres_limitless=> EXPLAIN (VERBOSE, COSTS OFF) INSERT INTO customers
    (customer_id, other_id, customer_name, balance)
    VALUES (1, 10, 'saikiran', 1000);

                      QUERY PLAN
------------------------------------------------------
 Insert on public.customers
   ->  Result
         Output: 1, 10, 'saikiran'::text, '1000'::real
 Single Shard Optimized
(4 rows)
```

UPDATE

```
postgres_limitless=> EXPLAIN (VERBOSE, COSTS OFF) UPDATE orders SET balance = balance +
 100
    WHERE customer_id = 100;

                                        QUERY PLAN

-------------------------------------------------------------------------------------------
 Update on public.orders
   Foreign Update on public.orders_fs00002 orders_1
   ->  Foreign Update
         Remote SQL:  UPDATE public.orders SET balance = (balance + (100)::double
 precision)
   WHERE (customer_id = 100)
 Single Shard Optimized
(6 rows)
```

DELETE

```
postgres_limitless=> EXPLAIN (VERBOSE, COSTS OFF) DELETE FROM orders
    WHERE customer_id = 100 and balance = 0;

                             QUERY PLAN
---------------------------------------------------------------------
 Delete on public.orders
   Foreign Delete on public.orders_fs00002 orders_1
   ->  Foreign Delete
         Remote SQL:  DELETE FROM public.orders
   WHERE ((customer_id = 100) AND (balance = (0)::double precision))
 Single Shard Optimized
(6 rows)
```

Restrizioni per le query a shard singolo

Le query a shard singolo prevedono le seguenti restrizioni:

Funzioni

Se una query a shard singolo contiene una funzione, la query è idonea per l'ottimizzazione a shard singolo solo se si verifica una delle seguenti condizioni:

- La funzione è immutabile. Per ulteriori informazioni, consulta Volatilità della funzione.
- La funzione è mutabile, ma è registrata nella visualizzazione `rds_aurora.limitless_distributed_functions`. Per ulteriori informazioni, consulta Distribuzione delle funzioni.

Visualizzazioni

Se una query contiene una o più visualizzazioni, l'ottimizzazione a shard singolo per la query è disabilitata se si verifica una delle seguenti condizioni:

- Ogni visualizzazione possiede l'attributo `security_barrier`.
- Gli oggetti utilizzati nella query richiedono più privilegi utente. Ad esempio, una query contiene due visualizzazioni che vengono eseguite da due utenti diversi.

```
CREATE VIEW v1 AS SELECT customer_name FROM customers c WHERE c.customer_id =  1;
CREATE VIEW v2 WITH (security_barrier) AS SELECT customer_name FROM customers c
 WHERE c.customer_id =  1;

postgres_limitless=> EXPLAIN VERBOSE SELECT * FROM v1;
                                    QUERY PLAN
-------------------------------------------------------------------------------
 Foreign Scan  (cost=100.00..101.00 rows=100 width=0)
   Output: customer_name
   Remote Plans from Shard postgres_s3:
         Seq Scan on public.customers_ts00001 c  (cost=0.00..24.12 rows=6 width=32)
           Output: c.customer_name
           Filter: (c.customer_id = 1)
         Query Identifier: -6005737533846718506
   Remote SQL:  SELECT customer_name
    FROM ( SELECT c.customer_name
             FROM public.customers c
            WHERE (c.customer_id = 1)) v1
 Query Identifier: -5754424854414896228
(12 rows)


postgres_limitless=> EXPLAIN VERBOSE SELECT * FROM v2;
                                        QUERY PLAN
-------------------------------------------------------------------------------------------
 Foreign Scan on public.customers_fs00001 c  (cost=100.00..128.41 rows=7 width=32)
   Output: c.customer_name
   Remote Plans from Shard postgres_s3:
```

```
         Seq Scan on public.customers_ts00001 customers  (cost=0.00..24.12 rows=6
 width=32)
           Output: customers.customer_name
           Filter: (customers.customer_id = 1)
         Query Identifier: 4136563775490008117
   Remote SQL: SELECT customer_name FROM public.customers WHERE ((customer_id = 1))
 Query Identifier: 5056054318010163757
(9 rows)
```

Istruzioni PREPARE ed EXECUTE

Aurora PostgreSQL Limitless Database supporta l'ottimizzazione a shard singolo per le istruzioni preparate `SELECT`, `UPDATE` e `DELETE`.

Tuttavia, se si utilizzano istruzioni preparate per `PREPARE` e `EXECUTE` con `plan_cache_mode` impostata su `'force_generic_plan'`, il pianificatore di query rifiuta l'ottimizzazione a shard singolo per quella query.

PL/pgSQL

Le query con variabili PL/pgSQL vengono eseguite come istruzioni preparate in modo implicito. Se una query contiene variabili PL/pgSQL, il pianificatore di query rifiuta l'ottimizzazione a shard singolo.

L'ottimizzazione è supportata nel blocco PL/pgSQL se l'istruzione non contiene alcuna variabile PL/pgSQL.

Join completamente qualificati (espliciti)

L'ottimizzazione a shard singolo si basa sull'eliminazione delle partizioni. Lo strumento di ottimizzazione per PostgreSQL elimina le partizioni in base a condizioni costanti. Se Aurora PostgreSQL Limitless Database rileva che tutte le partizioni e le tabelle rimanenti si trovano sullo stesso shard, contrassegna la query come idonea per l'ottimizzazione a shard singolo. Tutte le condizioni di filtro devono essere esplicite affinché l'eliminazione delle partizioni funzioni. Aurora PostgreSQL Limitless Database non è in grado di eliminare le partizioni senza uno o più predicati di join o predicati di filtro sulle chiavi di shard di ogni tabella sottoposta a sharding nell'istruzione.

Supponi di aver partizionato le tabelle `customers`, `orders` e `order_details` in base alla colonna `customer_id`. In questo schema, l'applicazione cerca di conservare tutti i dati di un cliente su un singolo shard.

Considera la query seguente:

```
SELECT * FROM
    customers c, orders o, order_details od
WHERE c.customer_id = o.customer_id
    AND od.order_id = o.order_id
    AND c.customer_id = 1;
```

Questa query recupera tutti i dati di un cliente (`c.customer_id = 1`). I dati di questo cliente si trovano su uno shard singolo, ma Aurora PostgreSQL Limitless Database non considera questa query come query a shard singolo. Lo strumento di ottimizzazione per la query è il seguente:

1. Lo strumento di ottimizzazione può eliminare le partizioni per i `customers` e gli `orders` in base alla seguente condizione:

   ```
   c.customer_id = 1
   c.customer_id = o.customer_id
   o.customer_id =  1 (transitive implicit condition)
   ```

2. Lo strumento di ottimizzazione non può eliminare alcuna partizione per i `order_details`, perché non esiste una condizione costante sulla tabella.
3. Lo strumento di ottimizzazione conclude di aver letto tutte le partizioni dai `order_details`. Pertanto, la query non può essere idonea per l'ottimizzazione a shard singolo.

Per rendere la query a shard singolo, aggiungiamo la seguente condizione di join esplicito:

```
o.customer_id = od.customer_id
```

La query modificata è simile alla seguente:

```
SELECT * FROM
    customers c, orders o,  order_details od
WHERE c.customer_id = o.customer_id
     AND o.customer_id = od.customer_id
     AND od. order_id = o. order_id
 AND c.customer_id =  1;
```

Ora lo strumento di ottimizzazione può eliminare le partizioni per i `order_details`. La nuova query diventa una query a shard singolo e quindi idonea per l'ottimizzazione.

Impostazione di una chiave di shard attiva

Questa funzionalità consente di impostare una chiave di shard singola durante l'esecuzione di query sul database, facendo sì che tutte le query SELECT e DML vengano aggiunte con la chiave di shard come predicato costante. Questa funzionalità è utile se hai effettuato la migrazione ad Aurora PostgreSQL Limitless Database e hai denormalizzato lo schema aggiungendo chiavi di shard alle tabelle.

È possibile aggiungere automaticamente un predicato sulla chiave di shard alla logica SQL esistente, senza modificare la semantica delle query. L'aggiunta di un predicato sulla chiave di shard attiva viene eseguita solo per le tabelle compatibili.

La funzionalità chiave di shard attiva utilizza la variabile `rds_aurora.limitless_active_shard_key`, che ha la seguente sintassi:

```
SET [session | local] rds_aurora.limitless_active_shard_key = '{"col1_value",
 "col2_value", ...}';
```

Alcune considerazioni sulle chiavi di shard attive e sulle chiavi esterne:

- Una tabella sottoposta a sharding può avere un vincolo di chiave esterna se le tabelle principale e secondaria sono co-localizzate e la chiave esterna è un superset della chiave di shard.
- Una tabella sottoposta a sharding può avere un vincolo di chiave esterna rispetto a una tabella di riferimento.
- Una tabella di riferimento può avere un vincolo di chiave esterna rispetto a un'altra tabella di riferimento.

Supponi di avere una tabella `customers` sottoposta a sharding sulla colonna `customer_id`.

```
BEGIN;
SET local rds_aurora.limitless_create_table_mode='sharded';
SET local rds_aurora.limitless_create_table_shard_key='{"customer_id"}';
CREATE TABLE customers(customer_id int PRIMARY KEY, name text , email text);
COMMIT;
```

Con un set di chiavi di shard attive, le query presentano le seguenti trasformazioni.

SELECT

```
SET rds_aurora.limitless_active_shard_key = '{"123"}';
SELECT * FROM customers;

-- This statement is changed to:
SELECT * FROM customers WHERE customer_id = '123'::int;
```

INSERT

```
SET rds_aurora.limitless_active_shard_key = '{"123"}';
INSERT INTO customers(name, email) VALUES('Alex', 'alex@example.com');

-- This statement is changed to:
INSERT INTO customers(customer_id, name, email) VALUES('123'::int, 'Alex',
 'alex@example.com');
```

UPDATE

```
SET rds_aurora.limitless_active_shard_key = '{"123"}';
UPDATE customers SET email = 'alex_new_email@example.com';

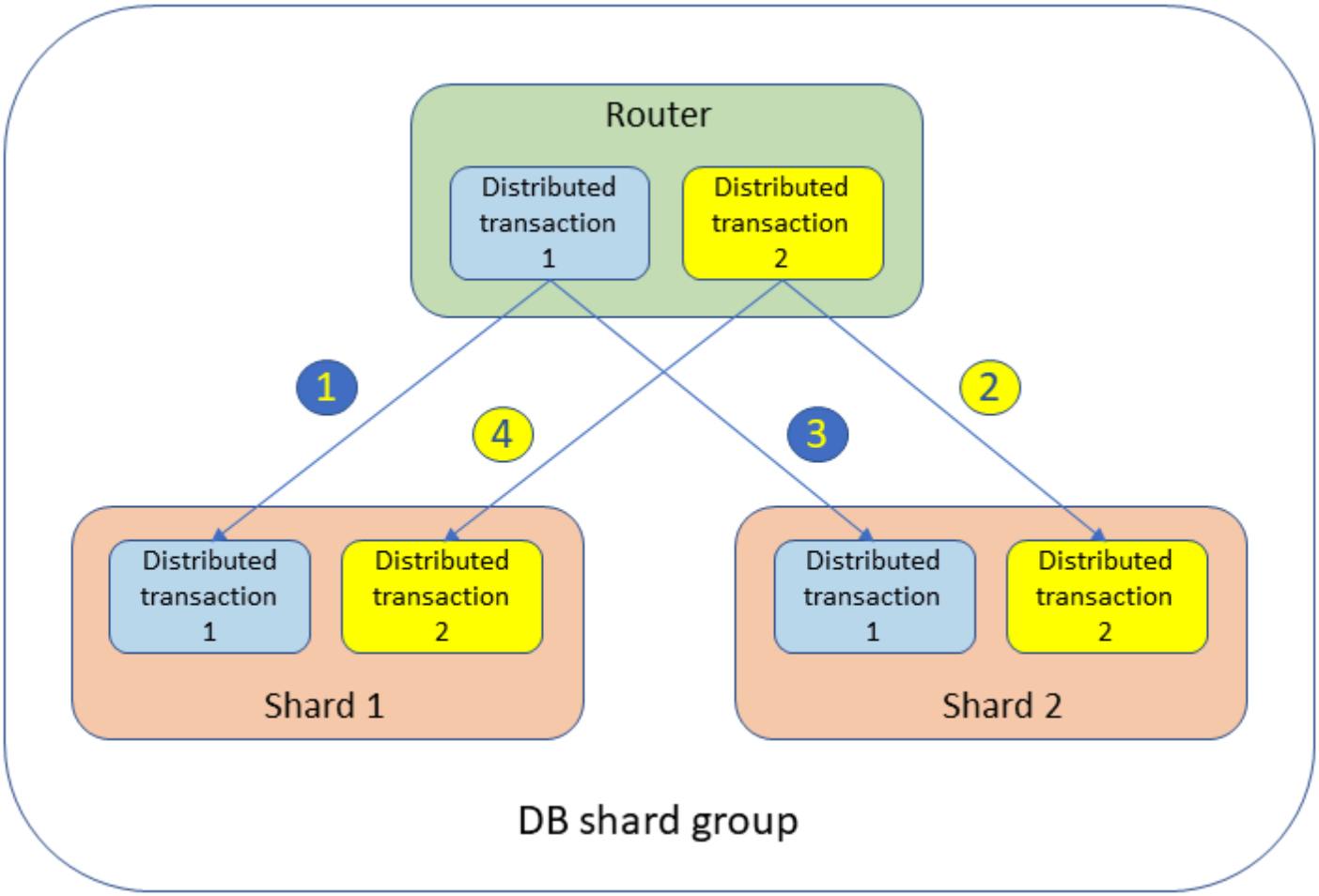
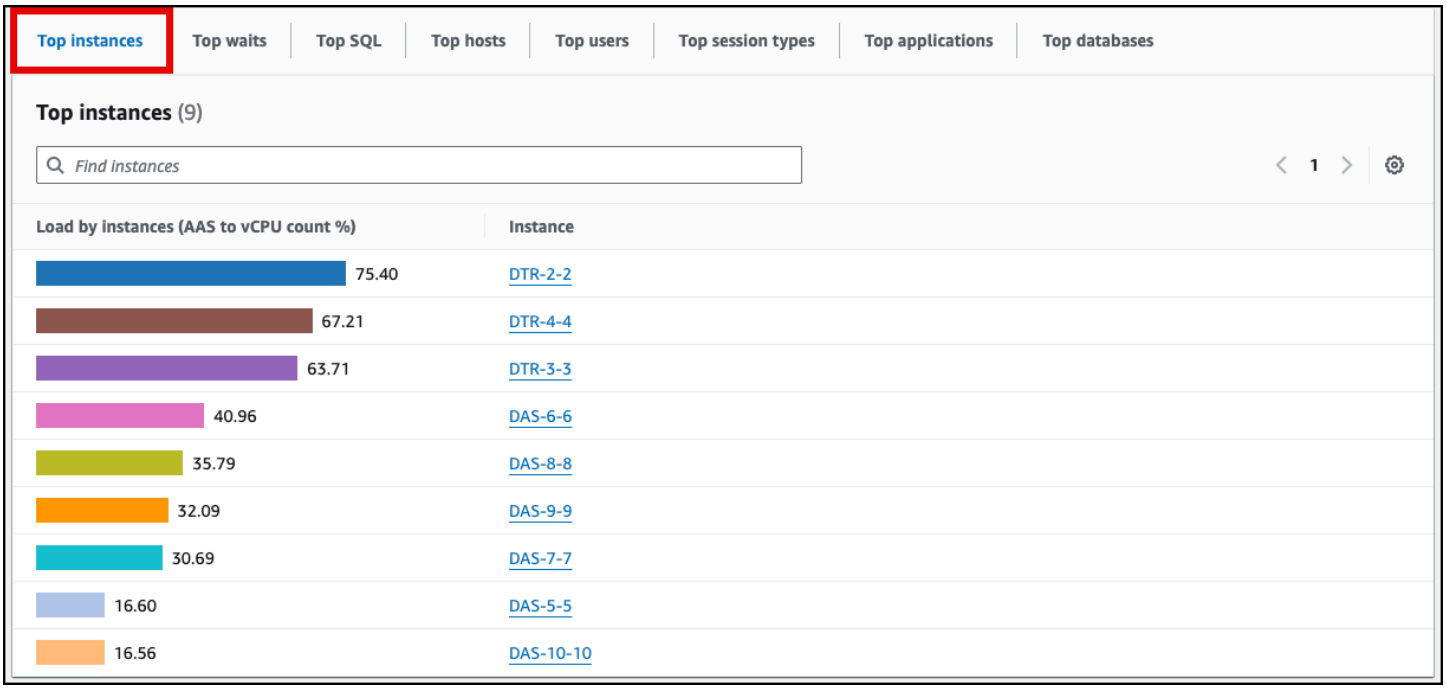
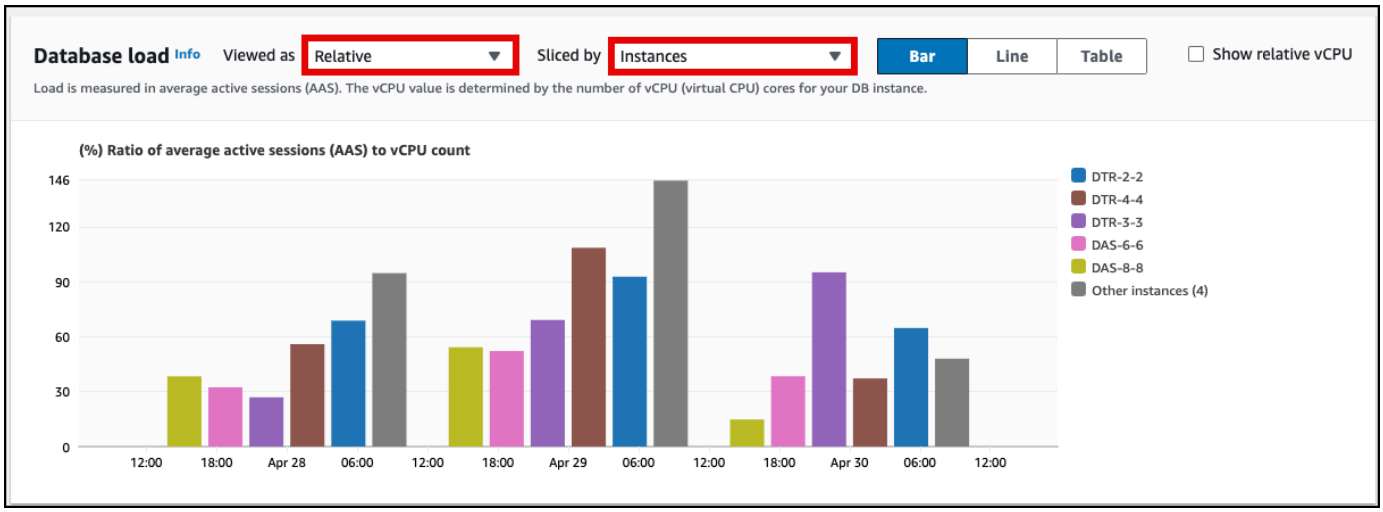
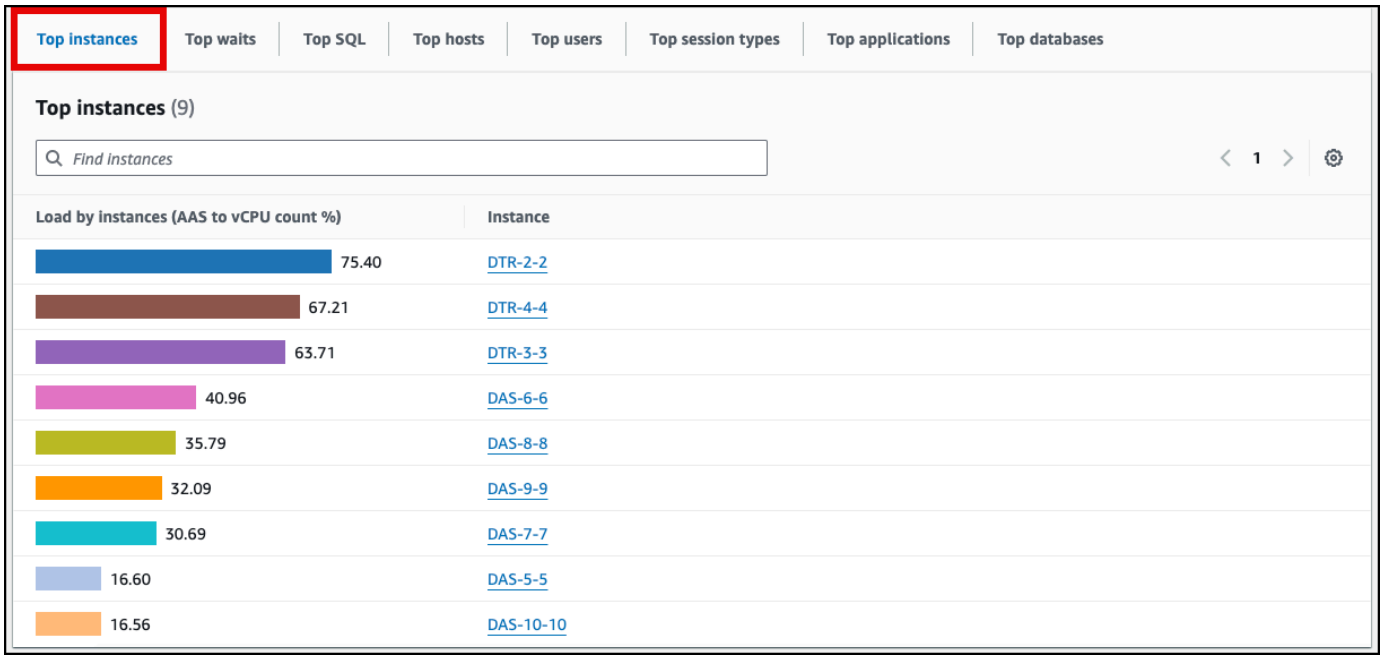
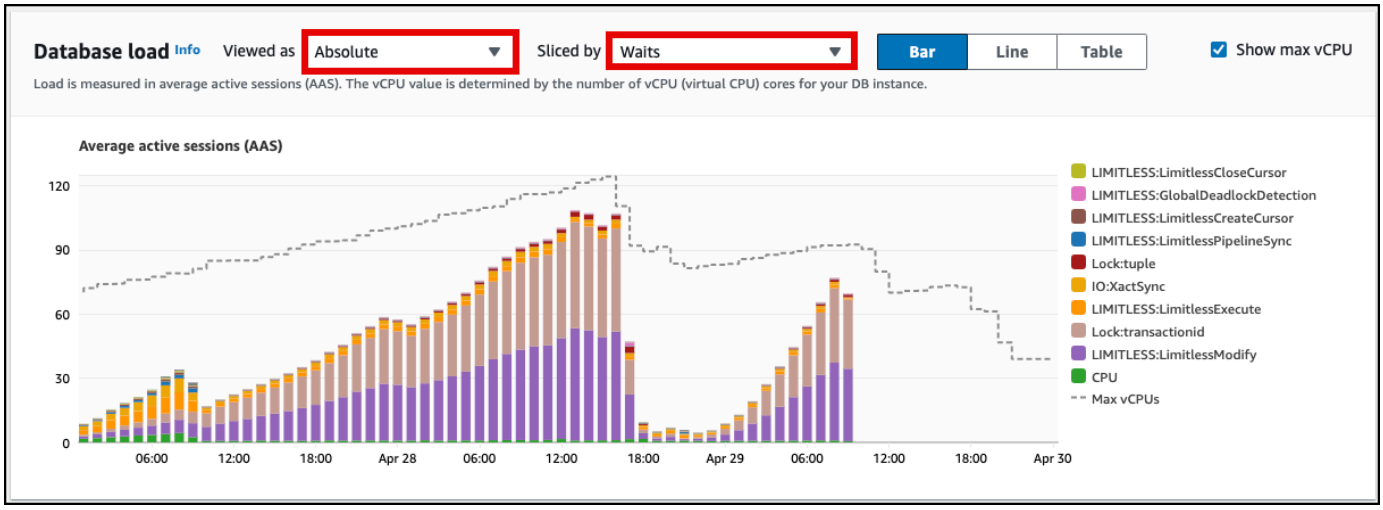
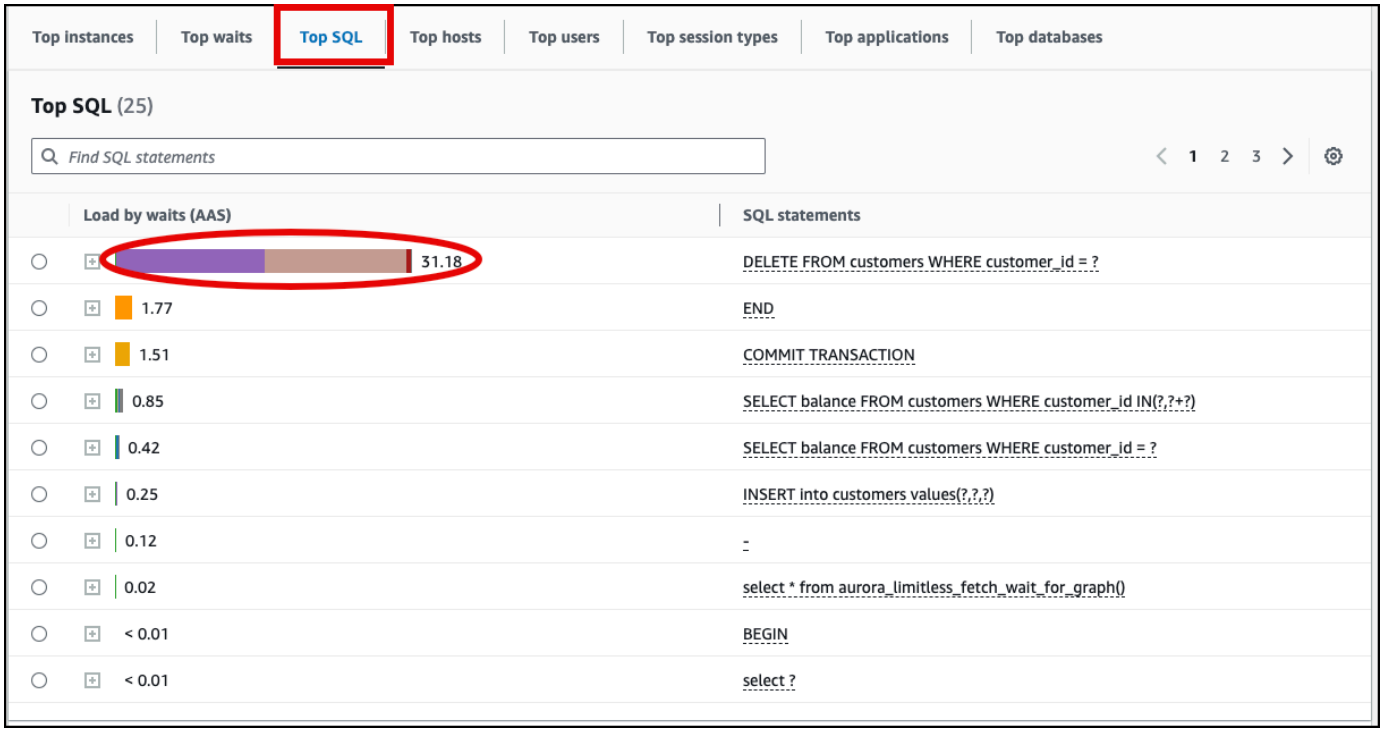
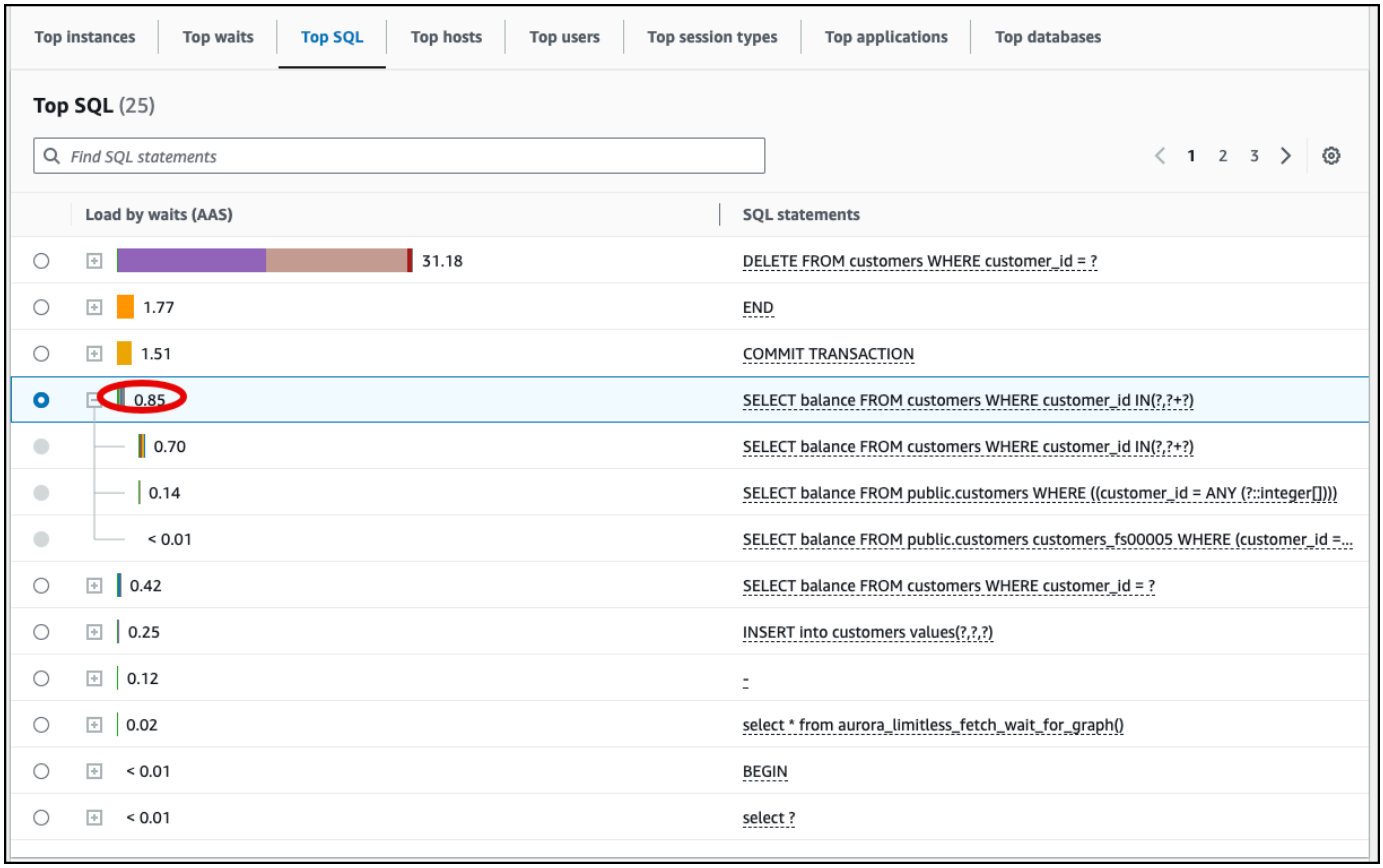
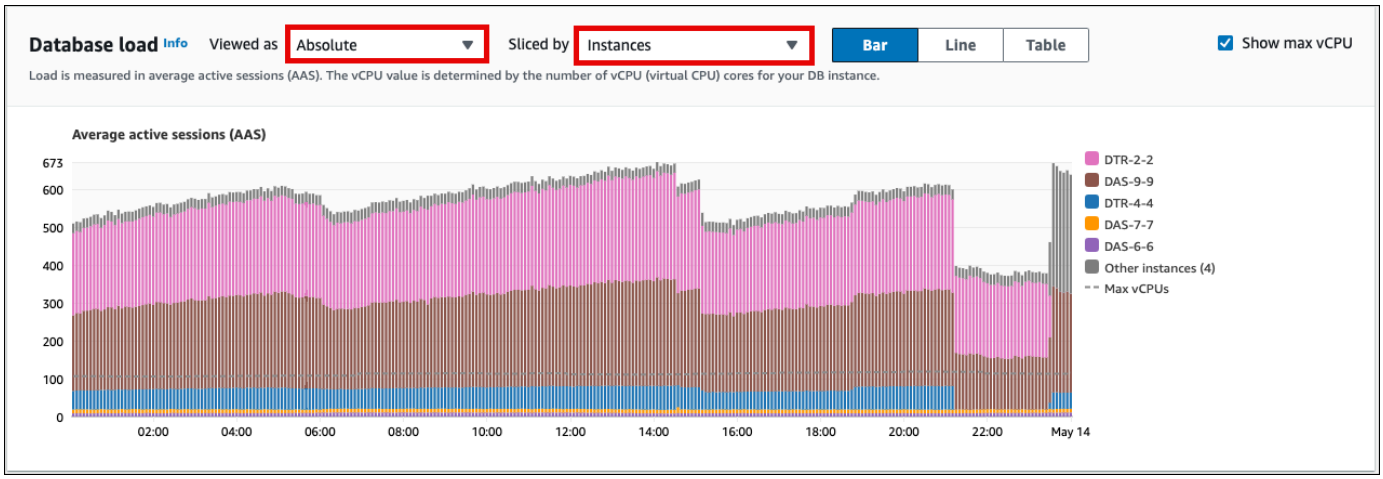
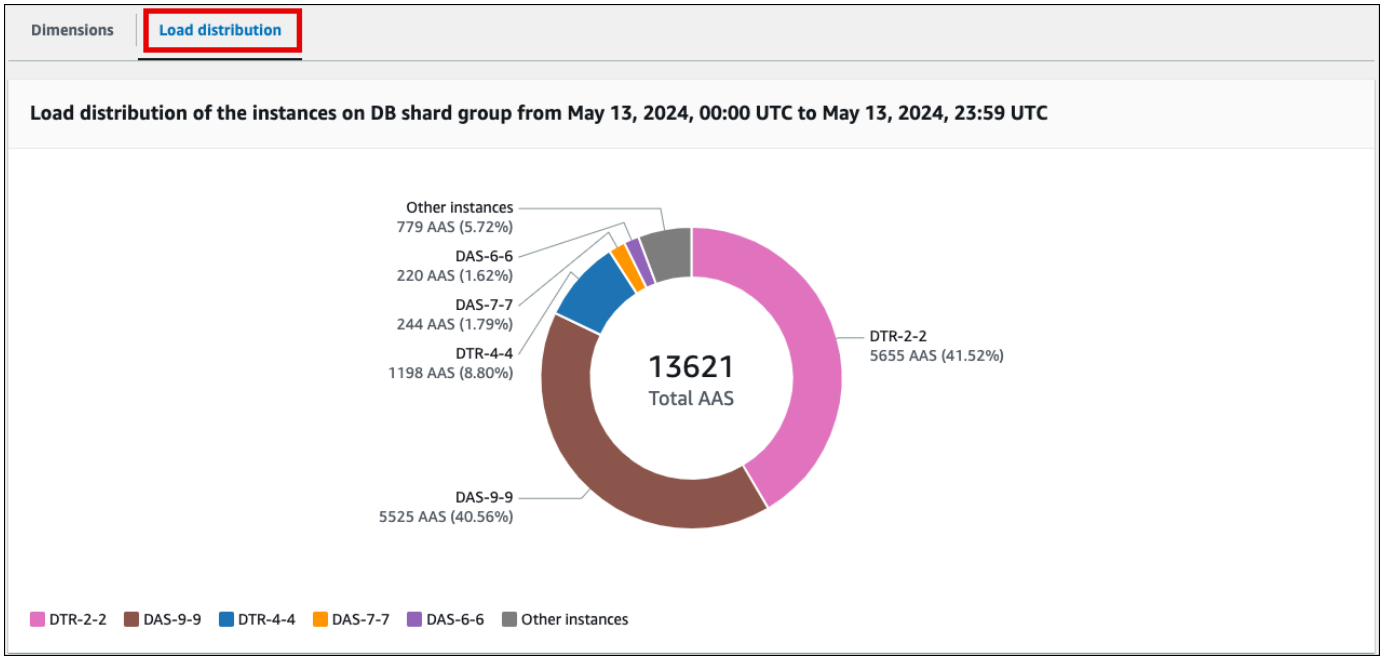
-- This statement is changed to:
UPDATE customers SET email = 'alex_new_email@example.com' WHERE customer_id =
 '123'::int;
```

DELETE

```
SET rds_aurora.limitless_active_shard_key = '{"123"}';
DELETE FROM customers;

-- This statement is changed to:
DELETE FROM customers WHERE customer_id = '123'::int;
```

Join

Quando si eseguono operazioni di join su tabelle con una chiave di shard attiva, il predicato della chiave di shard viene aggiunto automaticamente a tutte le tabelle coinvolte nel join. Questa aggiunta automatica del predicato sulla chiave di shard si verifica solo quando tutte le tabelle della query appartengono allo stesso gruppo di co-localizzazione. Se la query riguarda tabelle appartenenti a gruppi di co-localizzazione diversi, viene invece generato un errore.

Supponi di avere anche tabelle di `orders` e `order_details` co-localizzate con la tabella `customers`.

```
SET local rds_aurora.limitless_create_table_mode='sharded';
SET local rds_aurora.limitless_create_table_collocate_with='customers';
SET local rds_aurora.limitless_create_table_shard_key='{"customer_id"}';
CREATE TABLE orders (id int , customer_id int, total_amount int, date date);
CREATE TABLE order_details (id int , order_id int, customer_id int, product_name
 VARCHAR(100), price int);
COMMIT;
```

Recupera le ultime 10 fatture d'ordine per un cliente il cui ID è 10.

```
SET rds_aurora.limitless_active_shard_key = '{"10"}';
SELECT * FROM customers, orders, order_details WHERE
    orders.customer_id = customers.customer_id AND
    order_details.order_id = orders.order_id AND
    customers.customer_id = 10
    order by order_date limit 10;
```

Questa query viene trasformata come segue:

```
SELECT * FROM customers, orders, order_details WHERE
    orders.customer_id = customers.customer_id AND
    orders.order_id = order_details.order_id AND
    customers.customer_id = 10 AND
    order_details.customer_id = 10 AND
    orders.customer_id = 10 AND
    ORDER BY "order_date" LIMIT 10;
```

Tabelle compatibili con la chiave di shard attiva

Il predicato della chiave di shard viene aggiunto solo alle tabelle compatibili con la chiave di shard attiva. Una tabella è considerata compatibile se ha lo stesso numero di colonne nella relativa chiave di shard, come specificato nella variabile `rds_aurora.limitless_active_shard_key`. Se la query riguarda tabelle incompatibili con la chiave di shard attiva, il sistema non procede con la query e genera un errore.

Per esempio:

```
-- Compatible table
```

```
SET rds_aurora.limitless_active_shard_key = '{"10"}';

-- The following query works because the customers table is sharded on one column.
SELECT * FROM customers;

-- Incompatible table
SET rds_aurora.limitless_active_shard_key = '{"10","20"}';

-- The following query raises a error because the customers table isn't sharded on
 two columns.
 SELECT * FROM customers;
```

Query distribuite in Aurora PostgreSQL Limitless Database

Le query distribuite vengono eseguite su un router e su più di uno shard. La query viene ricevuta da uno dei router. Il router crea e gestisce la transazione distribuita, che viene inviata agli shard partecipanti. Gli shard creano una transazione locale con il contesto fornito dal router e la query viene eseguita.

Quando la transazione viene sottoposta al commit, il router utilizza un protocollo di commit a due fasi ottimizzato, se necessario, e il controllo della concorrenza multiversione (MVCC) basato sul tempo per fornire la semantica ACID in un sistema di database distribuito.

MVCC basato sul tempo registra l'ora di commit per ogni transazione e utilizza l'ora di inizio della transazione per generare l'ora dello snapshot dei dati. Per identificare se una transazione è stata sottoposta al commit (visibile) in base allo snapshot di un'istanza di lettura, il database confronta il tempo di commit con il tempo dello snapshot. Se il tempo di commit è inferiore al tempo di snapshot dell'istanza di lettura, è visibile; in caso contrario, è invisibile. Con questo protocollo, ci si aspetterà sempre di vedere dati fortemente coerenti su Aurora PostgreSQL Limitless Database.

Tracciamento distribuito delle query nei log di PostgreSQL in Aurora PostgreSQL Limitless Database

Il tracciamento distribuito delle query è uno strumento per tracciare e correlare le query nei log di PostgreSQL su Aurora PostgreSQL Limitless Database. In Aurora PostgreSQL, le transazioni vengono identificate mediante il relativo ID. Tuttavia, in Aurora PostgreSQL Limitless Database, l'ID della transazione può essere ripetuto su vari router. È quindi consigliabile ricorrere piuttosto all'ID di tracciamento in Limitless Database.

Di seguito sono riportati i principali casi d'uso:

- Utilizza la funzione `rds_aurora.limitless_get_last_trace_id()` per trovare l'ID di tracciamento univoco dell'ultima query eseguita nella sessione corrente. Quindi cerca nel gruppo di log del cluster di database in Amazon CloudWatch Logs utilizzando quell'ID di tracciamento per trovare tutti i log correlati.

  Puoi utilizzare i parametri `log_min_error_statement` e `log_min_messages` insieme per controllare il volume dei log stampati e stampare un'istruzione che contiene l'ID di tracciamento.
- Utilizza il parametro `log_min_duration_statement` per determinare il tempo di esecuzione trascorso il quale tutte le query stampano la durata dell'esecuzione e l'ID di tracciamento. Puoi cercare questo tempo di esecuzione nel gruppo di log del cluster di database in CloudWatch Logs e utilizzarlo per determinare i nodi dei colli di bottiglia e gli sforzi di ottimizzazione del pianificatore.

  Il parametro `log_min_duration_statement` abilita l'ID di tracciamento per tutti i nodi, indipendentemente dai valori di `log_min_messages` e `log_min_error_statement`.

Argomenti

- ID di tracciamento
- Utilizzo del tracciamento delle query
- Esempi di log

ID di tracciamento

L'elemento fondamentale di questa funzionalità è un identificatore univoco noto come ID di tracciamento. L'ID di tracciamento è una stringa di 31 cifre aggiunta alle righe di log STATEMENT dei log di PostgreSQL, che funge da identificatore esatto per correlare i log relativi all'esecuzione

di una query specifica. Alcuni esempi sono `1126253375719408504000000000011` e `1126253375719408495000000000090`.

L'ID di tracciamento è costituito dai seguenti elementi:

- Identificatore della transazione: le prime 20 cifre, che identificano in modo univoco la transazione.
- Identificatore del comando: le prime 30 cifre, che indicano una singola query all'interno di una transazione.

  Se all'interno di un blocco di transazione esplicito vengono eseguite più di 4.294.967.294 query, l'identificatore del comando torna a 1. Quando questo accade, ti viene notificato dal seguente messaggio di `LOG` nel log di PostgreSQL:

  ```
  wrapping around the tracing ID back to 1 after running 4294967294 (4.2 billion or
   2^32-2) queries inside of an explicit transaction block
  ```
- Identificatore del tipo di nodo: l'ultima cifra, che indica se il nodo funziona come router coordinatore (`1`) o nodo partecipante (`0`).

Gli esempi seguenti illustrano i componenti dell'ID di tracciamento:

- `1126253375719408504000000000011`:
  - Identificatore della transazione: `1126253375719408504`
  - Identificatore del comando: `1126253375719408504000000000001` indica il primo comando nel blocco di transazione
  - Identificatore del tipo di nodo: 1 indica un router coordinatore
- `1126253375719408495000000000090`:
  - Identificatore della transazione: `1126253375719408495`
  - Identificatore del comando: `1126253375719408495000000000009` indica il nono comando nel blocco di transazione
  - Identificatore del tipo di nodo: `0` indica un nodo partecipante

Utilizzo del tracciamento delle query

Per utilizzare il tracciamento delle query, esegui queste attività:

1. Assicurati che il tracciamento sia abilitato.

Per verificare, utilizza il comando seguente:

```
SHOW rds_aurora.limitless_log_distributed_trace_id;
```

È abilitato per impostazione predefinita (on). Se non è abilitato, impostalo utilizzando il comando seguente:

```
SET rds_aurora.limitless_log_distributed_trace_id = on;
```

2. Controlla il volume dei log stampati configurando il livello di gravità del log.

   Il volume dei log è controllato dal parametro `log_min_messages`. Il parametro `log_min_error_statement` viene utilizzato per stampare la riga `STATEMENT` con l'ID del tracciamento. L'impostazione predefinita di entrambi è `ERROR`. Per verificare, utilizza i comandi seguenti:

   ```
   SHOW log_min_messages;
   SHOW log_min_error_statement;
   ```

   Per aggiornare il livello di gravità e stampare la riga `STATEMENT` per la sessione corrente, utilizza i seguenti comandi con uno di questi livelli di gravità:

```
SET log_min_messages = 'DEBUG5 | DEBUG4 | DEBUG3 | DEBUG2 | DEBUG1 | INFO | NOTICE
 | WARNING | ERROR | LOG | FATAL | PANIC';
SET log_min_error_statement = 'DEBUG5 | DEBUG4 | DEBUG3 | DEBUG2 | DEBUG1 | INFO
 | NOTICE | WARNING | ERROR | LOG | FATAL | PANIC';
```

   Per esempio:

   ```
   SET log_min_messages = 'WARNING';
   SET log_min_error_statement = 'WARNING';
   ```

3. Abilita la stampa dell'ID di tracciamento nei log una volta superato un tempo di esecuzione specifico.

   Il parametro `log_min_duration_statement` può essere modificato impostando il tempo minimo di esecuzione della query oltre il quale la query stampa una riga del log con la durata dell'esecuzione insieme agli ID di tracciamento nel cluster di database. L'impostazione predefinita

di questo parametro è `-1`, il che significa che è disabilitato. Per verificare, utilizza il comando seguente:

```
SHOW log_min_duration_statement;
```

Se si imposta il parametro su `0`, vengono stampati la durata e l'ID di tracciamento nei log per ogni query nel cluster di database. È possibile impostare il parametro su `0` per la sessione corrente utilizzando il comando seguente:

```
SET log_min_duration_statement = 0;
```

4. Ottieni l'ID di tracciamento.

   Dopo aver eseguito una query (anche all'interno di un blocco di transazione esplicito), chiama la funzione `rds_aurora.limitless_get_last_trace_id` per ottenere l'ID di tracciamento dell'ultima esecuzione della query:

   ```
   SELECT * FROM rds_aurora.limitless_get_last_trace_id();
   ```

   Questa funzione restituisce l'identificatore della transazione e l'identificatore del comando. Non restituisce l'identificatore del tipo di nodo.

   ```
   => SELECT * FROM customers;
    customer_id | fname | lname
   -------------+-------+-------
   (0 rows)

   => SELECT * FROM rds_aurora.limitless_get_last_trace_id();
    transaction_identifier |       command_identifier
   ------------------------+--------------------------------
    10104661421959001813   | 1010466142195900181300000000001
   (1 row)
   ```

   La funzione restituisce una riga vuota per le query non distribuite, in quanto non dispongono di un ID di tracciamento.

   ```
   => SET search_path = public;
   SET
   ```

```
=> SELECT * FROM rds_aurora.limitless_get_last_trace_id();
 transaction_identifier | command_identifier
------------------------+--------------------
                        |
(1 row)
```

> **Note**
>
> Per le query VACUUM e ANALYZE, l'istruzione di durata non viene registrata con l'ID di tracciamento. Pertanto, `limitless_get_last_trace_id()` non restituisce l'ID di tracciamento. Se VACUUM o ANALYZE è un'operazione di lunga durata, puoi ottenere il relativo ID di tracciamento utilizzando la seguente query:
>
> ```
> SELECT * FROM rds_aurora.limitless_stat_activity
> WHERE distributed_tracing_id IS NOT NULL;
> ```
>
> Se il server si ferma prima che sia possibile trovare l'ultimo ID di tracciamento, sarà necessario eseguire una ricerca manuale nei log di PostgreSQL per individuare gli ID di tracciamento nei log precedenti all'errore.

5. Cerca l'ID di tracciamento nei log del cluster di database utilizzando CloudWatch.

   Utilizza CloudWatch Insights per interrogare il gruppo di log del cluster di database, come illustrato negli esempi seguenti.

   - Esegui una query per un particolare identificatore della transazione e tutti i comandi che vengono eseguiti al suo interno:

     ```
     fields @timestamp, @message
     | filter @message like /10104661421959001813/
     | sort @timestamp desc
     ```

   - Query per un particolare identificatore del comando:

     ```
     fields @timestamp, @message
     | filter @message like /10104661421959001813000000001/
     | sort @timestamp desc
     ```

6. Esamina tutti i log del cluster di database generati dalla query distribuita.

Esempi di log

I seguenti esempi mostrano l'utilizzo del tracciamento delle query.

Correlazione dei log per le query soggette a errori

In questo esempio, il comando `TRUNCATE` viene eseguito sulla tabella `customers` quando questa non esiste.

Senza tracciamento delle query

File di log PostgreSQL sul router coordinatore:

```
2023-09-26 04:03:19 UTC:[local]: postgres@postgres_limitless:[27503]:ERROR: failed to
 execute remote query
2023-09-26 04:03:19 UTC:[local]: postgres@postgres_limitless:[27503]:DETAIL: relation
 "public.customers" does not exist
2023-09-26 04:03:19 UTC:[local]: postgres@postgres_limitless:[27503]:CONTEXT: remote
 SQL command: truncate public.customers;
2023-09-26 04:03:19 UTC:[local]: postgres@postgres_limitless:[27503]:STATEMENT:
 truncate customers;
```

File di log PostgreSQL su uno shard partecipante:

```
2023-09-26 04:03:19 UTC:[local]: postgres@postgres_limitless:[ 27503]:ERROR: failed to
 execute remote query
2023-09-26 04:03:19 UTC:[local]: postgres@postgres_limitless:[ 27503]:STATEMENT:
 truncate customers;
```

Questi log sono tipici. Mancano degli identificatori precisi necessari per correlare facilmente le query all'interno del cluster di database.

Con tracciamento delle query

File di log PostgreSQL sul router coordinatore:

```
2023-09-26 04:03:19 UTC:[local]:postgres@postgres_limitless:[27503]:ERROR: failed to
 execute remote query
2023-09-26 04:03:19 UTC:[local]:postgres@postgres_limitless:[27503]:DETAIL: relation
 "public.customers" does not exist
2023-09-26 04:03:19 UTC:[local]:postgres@postgres_limitless:[27503]:CONTEXT: remote SQL
 command: truncate public.customers;
```

```
2023-09-26 04:03:19 UTC:[local]:postgres@postgres_limitless:[27503]:STATEMENT: /* tid =
 11262533757194085027000000000011 */ truncate customers;
```

File di log PostgreSQL su uno shard partecipante:

```
2023-09-26 04:03:19 UTC:[local]:postgres@postgres_limitless:[27503]:ERROR:  failed to
 execute remote query
2023-09-26 04:03:19 UTC:[local]:postgres@postgres_limitless:[27503]:STATEMENT:  /* tid
 = 11262533757194085027000000000010 */ truncate customers;
```

Se si esegue il tracciamento delle query, a ogni riga di log viene aggiunto un identificatore univoco di 31 cifre. Qui, `11262533757194085027000000000011` e `11262533757194085027000000000010` rappresentano gli ID di tracciamento per i nodi coordinatore e partecipante, rispettivamente.

- Identificatore della transazione: `11262533757194085027`
- Identificatore del comando: `1126253375719408502700000000001`
- Identificatore del tipo di nodo: l'ultima cifra, `1` o `0`, indica un router coordinatore e un nodo partecipante, rispettivamente.

Correlazione di log per individuare il tempo di esecuzione della query su vari nodi

In questo esempio, il parametro `log_min_duration_statement` è stato aggiornato a `0` per stampare la durata di tutte le query.

Senza tracciamento delle query

```
2024-01-15 07:28:46 UTC:[local]:postgres@postgres_limitless:[178322]:LOG: duration:
 12.779 ms statement: select * from customers;
```

Con tracciamento delle query

File di log PostgreSQL sul router coordinatore:

```
2024-01-15 07:32:08 UTC:[local]:postgres@postgres_limitless:[183877]:LOG: duration:
 12.618 ms statement: /* tid = 045766956624049708840000000011 */ select * from
 customers;
```

File di log PostgreSQL su uno shard partecipante:

```
2024-01-15 07:32:08 UTC:localhost(46358):postgres@postgres_limitless:[183944]:LOG:
 duration: 0.279 ms statement: /* tid = 045766956624049708840000000010 */ START
 TRANSACTION ISOLATION LEVEL READ COMMITTED
2024-01-15 07:32:08 UTC:localhost(46358):postgres@postgres_limitless:[183944]:LOG:
 duration: 0.249 ms parse <unnamed>: SELECT customer_id, fname, lname FROM
 public.customers
2024-01-15 07:32:08 UTC:localhost(46358):postgres@postgres_limitless:[183944]:LOG:
 duration: 0.398 ms bind <unnamed>/c1: SELECT customer_id, fname, lname FROM
 public.customers
2024-01-15 07:32:08 UTC:localhost(46358):postgres@postgres_limitless:[183944]:LOG:
 duration: 0.019 ms execute <unnamed>/c1: SELECT customer_id, fname, lname FROM
 public.customers
2024-01-15 07:32:08 UTC:localhost(46358):postgres@postgres_limitless:[183944]:LOG:
 duration: 0.073 ms statement: /* tid = 045766956624049708840000000010 */ COMMIT
 TRANSACTION
```

Deadlock distribuiti in Aurora PostgreSQL Limitless Database

In un gruppo di shard del database, possono verificarsi dei deadlock tra le transazioni distribuite tra router e shard diversi. Ad esempio, vengono eseguite due transazioni distribuite simultanee che si estendono su due shard, come illustrato nella figura seguente.



Le transazioni bloccano le tabelle e creano eventi di attesa nei due shard come segue:

1. Transazione distribuita 1:

   ```
   UPDATE table SET value = 1 WHERE key = 'shard1_key';
   ```

   Questo contiene un blocco sullo shard 1.

2. Transazione distribuita 2:

   ```
   UPDATE table SET value = 2 WHERE key = 'shard2_key';
   ```

Questo contiene un blocco sullo shard 2.

3. Transazione distribuita 1:

```
UPDATE table SET value = 3 WHERE key = 'shard2_key';
```

La transazione distribuita 1 è in attesa sullo shard 2.

4. Transazione distribuita 2:

```
UPDATE table SET value = 4 WHERE key = 'shard1_key';
```

La transazione distribuita 2 è in attesa sullo shard 1.

In questo scenario, né lo shard 1 né lo shard 2 vedono il problema: la transazione 1 è in attesa della transazione 2 sullo shard 2 e la transazione 2 è in attesa della transazione 1 sullo shard 1. Da un punto di vista globale, la transazione 1 è in attesa della transazione 2 e la transazione 2 è in attesa della transazione 1. Questa situazione in cui due transazioni su due shard diversi sono in attesa l'una dell'altra viene chiamata deadlock distribuito.

Aurora PostgreSQL Limitless Database è in grado di rilevare e risolvere automaticamente i deadlock distribuiti. Un router del gruppo di shard del database riceve una notifica quando una transazione attende troppo a lungo per acquisire una risorsa. Il router che riceve la notifica inizia a raccogliere le informazioni necessarie da tutti i router e gli shard all'interno del gruppo di shard del database. Il router procede quindi a terminare le transazioni che partecipano a un deadlock distribuito, finché le altre transazioni nel gruppo di shard del database non possono procedere senza essere bloccate l'una dall'altra.

Viene visualizzato il seguente errore quando la transazione fa parte di un deadlock distribuito e viene quindi interrotta dal router:

```
ERROR: aborting transaction participating in a distributed deadlock
```

Il parametro del cluster di database `rds_aurora.limitless_distributed_deadlock_timeout` imposta il tempo di attesa di ogni transazione su una risorsa prima di notificare al router di verificare la presenza di un deadlock distribuito. È possibile aumentare il valore del parametro se il carico di lavoro è meno soggetto a situazioni di deadlock. Il valore predefinito è `1000` millisecondi (1 secondo).

Il ciclo di deadlock distribuito viene pubblicato nei log di PostgreSQL quando viene rilevato e risolto un deadlock tra nodi. Le informazioni su ogni processo che fa parte del deadlock includono quanto segue:

- Nodo coordinatore che ha avviato la transazione
- ID della transazione virtuale (xid) della transazione sul nodo coordinatore, nel formato *backend_id*/*backend_local_xid*
- L'ID della sessione distribuita della transazione

Gestione di Aurora PostgreSQL Limitless Database

Gli argomenti seguenti descrivono come gestire i cluster di database Aurora PostgreSQL Limitless Database.

Argomenti

- Considerazioni sulle dimensioni di database e tabelle
- Recupero dello spazio di archiviazione tramite il processo di vacuum

Considerazioni sulle dimensioni di database e tabelle

In Aurora PostgreSQL Limitless Database, in ogni shard una tabella sottoposta a sharding è suddivisa in un numero di sezioni di tabella variabile in base al numero di shard disponibili nel gruppo di shard del database. Ogni sezione di tabella può raggiungere una dimensione massima di 32 TiB, ma ogni shard ha una capacità massima di 128 TiB. Le tabelle di riferimento hanno un limite di dimensione di 32 TiB per l'intero gruppo di shard del database.

> **Note**
>
> La capacità massima di ogni nodo (router o shard) è pari a 128 TiB, che rappresenta la capacità massima di un cluster di database Aurora PostgreSQL.

Il numero massimo di relazioni per database (incluse tabelle, visualizzazioni e indici), sia in Aurora PostgreSQL che in Aurora PostgreSQL Limitless Database, è pari a 1.431.650.303.

Per ulteriori informazioni, consulta Appendix K. PostgreSQL limits nella documentazione di PostgreSQL e Limiti di dimensione Amazon Aurora.

Recupero dello spazio di archiviazione tramite il processo di vacuum

PostgreSQL Multiversion Concurrency Control (MVCC) aiuta a preservare l'integrità dei dati salvando una copia interna delle righe aggiornate o eliminate fino al commit o al rollback di una transazione. Queste copie, chiamate anche tuple, possono causare l'aumento delle dimensioni della tabella se non vengono pulite regolarmente. Le istanze PostgreSQL ordinano le transazioni in base ai relativi ID di transazione e PostgreSQL utilizza l'MVCC basato su ID di transazione per controllare la visibilità delle tuple e fornire l'isolamento delle transazioni. Ogni transazione stabilisce uno snapshot dei dati e ogni tupla possiede una versione. Sia lo snapshot che la versione sono basati sull'ID della transazione.

Per pulire i dati, l'utility `VACUUM` esegue in PostgreSQL quattro funzioni chiave:

- `VACUUM`: rimuove le versioni di riga scadute, rendendo lo spazio disponibile per il riutilizzo.
- `VACUUM FULL`: fornisce una deframmentazione completa rimuovendo le versioni delle tuple morte e compattando le tabelle, in modo da ridurre le dimensioni e aumentare l'efficienza.
- `VACUUM FREEZE`: protegge dai problemi di wraparound degli ID delle transazioni contrassegnando le versioni di riga precedenti come bloccate.
- `VACUUM ANALYZE` rimuove le versioni delle tuple morte e aggiorna le statistiche di pianificazione delle query del database. Si tratta di una combinazione delle funzioni `VACUUM` e `ANALYZE`. Per ulteriori informazioni sul funzionamento di `ANALYZE` in Aurora PostgreSQL Limitless Database, consulta ANALYZE.

Come in MVCC, anche in Aurora PostgreSQL il processo di vacuum è basato sull'ID delle transazioni. Se al momento dell'avvio del processo di vacuum è in corso una transazione, le righe ancora visibili in quella transazione non vengono rimosse.

Per ulteriori informazioni sull'utility `VACUUM`, consulta VACUUM nella documentazione di PostgreSQL. Per ulteriori informazioni sul supporto di `VACUUM` in Aurora PostgreSQL Limitless Database, consulta VACUUM.

Argomenti

- AUTOVACUUM
- Processo di vacuum basato sul tempo in Aurora PostgreSQL Limitless Database
- Utilizzo delle statistiche del database per il processo di vacuum

- Differenze nel comportamento di vacuum tra Aurora PostgreSQL e Aurora PostgreSQL Limitless Database

AUTOVACUUM

Per rimuovere le tuple non necessarie, Aurora PostgreSQL ricorre alle utility `VACUUM` e `AUTOVACUUM`. Il meccanismo di base di `AUTOVACUUM` e del `VACUUM` manuale è lo stesso, l'unica differenza è l'automazione.

In Aurora PostgreSQL e Aurora PostgreSQL Limitless Database, `AUTOVACUUM` è una combinazione delle utility `VACUUM` e `ANALYZE`. `AUTOVACUUM` determina quali database e tabelle pulire in base a una regola predefinita, come la percentuale di tuple morte e il numero di inserti.

Ad esempio, `AUTOVACUUM` "si riattiva" periodicamente per eseguire la pulizia. L'intervallo è controllato dal parametro `autovacuum_naptime`. Il valore predefinito è di 1 minuto. I valori predefiniti per i parametri di configurazione `AUTOVACUUM` e `VACUUM` sono gli stessi sia per Aurora PostgreSQL Limitless Database che per Aurora PostgreSQL.

Se abilitato, il daemon `AUTOVACUUM` invia automaticamente i comandi `ANALYZE` ogni volta che il contenuto di una tabella risulta sufficientemente modificato. In Aurora PostgreSQL Limitless Database, `AUTOVACUUM` invia `ANALYZE` sia sui router che sugli shard.

Per ulteriori informazioni sul daemon `AUTOVACUUM` e sui parametri di archiviazione della tabella associati ad `AUTOVACUUM`, consulta The autovacuum daemon e Storage parameters nella documentazione di PostgreSQL.

Processo di vacuum basato sul tempo in Aurora PostgreSQL Limitless Database

Aurora PostgreSQL Limitless Database è un sistema distribuito. Questo significa che in una transazione possono essere coinvolte più istanze. Pertanto, la visibilità basata sull'ID della transazione non si applica. Aurora PostgreSQL Limitless Database utilizza la visibilità basata sul tempo, poiché se gli ID delle transazioni non sono "unificati" tra le istanze, è invece possibile "unificare" il tempo. Ogni snapshot delle transazioni e ogni versione delle tuple rispondono al tempo anziché all'ID delle transazioni. Più specificamente, agli snapshot delle transazioni è associata un'ora di inizio e alle tuple è associata un'ora di creazione (in presenza di una richiesta `INSERT` o un `UPDATE`) e un'ora di eliminazione (in presenza di una richiesta `DELETE`).

Per mantenere la coerenza dei dati tra le istanze del gruppo di shard di database, Aurora PostgreSQL Limitless Database deve assicurarsi che il processo di vacuum non rimuova le tuple

ancora visibili su qualsiasi transazione attiva nel gruppo di shard di database. Pertanto, anche in Aurora PostgreSQL Limitless Database il processo di vacuum è basato sul tempo. Altri aspetti di `VACUUM` rimangono invariati, incluso il fatto che per eseguire `VACUUM` su una particolare tabella, un utente deve avere accesso a tale tabella.

> **Note**
>
> È fortemente consigliato non lasciare le transazioni aperte per lunghi periodi di tempo. Il processo di vacuum basato sul tempo consuma più memoria rispetto al vacuum basato sull'ID delle transazioni.

Nell'esempio seguente viene illustrato come funziona il processo di vacuum basato sul tempo.

1. Una tabella dei clienti viene distribuita su quattro shard.
2. La transazione 1 inizia con un livello di isolamento Repeatable Read e riguarda un solo shard (shard 1). Questa transazione rimane aperta.

   La transazione 1 è più datata di qualsiasi altra transazione iniziata dopo di essa.
3. La transazione 2 inizia successivamente e, dopo aver eliminato tutte le tuple da una tabella, esegue il commit.
4. Se `AUTOVACUUM` o il `VACUUM` manuale tenta di pulire le tuple morte (morte a causa della transazione 2), non rimuove nulla.

   Questo vale non solo per gli shard 1, ma anche per gli shard da 2 a 4, perché la transazione 1 potrebbe avere ancora bisogno di accedere a tali tuple. Sono ancora visibili nella transazione 1 grazie a MVCC.

L'ultimo passaggio viene eseguito tramite la sincronizzazione, in modo che tutti gli shard siano consapevoli della transazione 1, anche se questa non li riguarda tutti.

Utilizzo delle statistiche del database per il processo di vacuum

Per ottenere informazioni sulle tuple che potresti dover pulire, utilizza la vista limitless_stat_all_tables, che funziona in modo analogo a pg_stat_all_tables. Nell'esempio seguente viene eseguita una query sulla vista.

```
SELECT * FROM rds_aurora.limitless_stat_all_tables WHERE relname LIKE '%customer%';
```

Allo stesso modo, per le statistiche del database utilizza limitless_stat_database anziché pg_stat_database e limitless_stat_activity anziché pg_stat_activity.

Per controllare l'utilizzo del disco della tabella, ricorri alla funzione limitless_stat_relation_sizes, che funziona in modo analogo a pg_relation_size. Negli esempi seguenti vengono inviate query alla funzione.

```
SELECT * FROM rds_aurora.limitless_stat_relation_sizes('public','customer');
```

Per tracciare l'avanzamento di un'operazione VACUUM su Aurora PostgreSQL Limitless Database, utilizza la vista limitless_stat_progress_vacuum anziché pg_stat_progress_vacuum. Nell'esempio seguente viene eseguita una query sulla vista.

```
SELECT * FROM rds_aurora.limitless_stat_progress_vacuum;
```

Per ulteriori informazioni, consulta Visualizzazioni di Aurora PostgreSQL Limitless Database e Funzioni di Aurora PostgreSQL Limitless Database.

Differenze nel comportamento di vacuum tra Aurora PostgreSQL e Aurora PostgreSQL Limitless Database

Il funzionamento del processo di vacuum in Aurora PostgreSQL e in Aurora PostgreSQL Limitless Database presenta altre differenze, tra cui:

- Aurora PostgreSQL esegue operazioni VACUUM sugli ID delle transazioni fino alla transazione in corso più datata. Se nel database non è presente alcuna transazione in corso, VACUUM esegue l'operazione fino all'ultima transazione.
- Aurora PostgreSQL Limitless Database esegue la sincronizzazione dello snapshot più datato ogni 10 secondi. Pertanto, VACUUM potrebbe non eseguire l'operazione su alcuna transazione eseguita negli ultimi 10 secondi.

Per informazioni sul supporto per VACUUM in Aurora PostgreSQL Limitless Database, consulta VACUUM in Riferimento ad Aurora PostgreSQL Limitless Database.

Monitoraggio di Aurora PostgreSQL Limitless Database

Puoi utilizzare Amazon CloudWatch, il monitoraggio avanzato e Approfondimenti sulle prestazioni per monitorare Aurora PostgreSQL Limitless Database. Sono inoltre disponibili nuove visualizzazioni e funzioni di statistiche, oltre a eventi di attesa, per Aurora PostgreSQL Limitless Database, che puoi utilizzare per attività di monitoraggio e diagnostica.

Argomenti

- Monitoraggio di Aurora PostgreSQL Limitless Database con Amazon CloudWatch
- Monitoraggio di Aurora PostgreSQL Limitless Database con CloudWatch Database Insights
- Monitoraggio di Aurora PostgreSQL Limitless Database con Amazon CloudWatch Logs
- Monitoraggio di Aurora PostgreSQL Limitless Database con Monitoraggio avanzato
- Monitoraggio di Aurora PostgreSQL Limitless Database con Approfondimenti sulle prestazioni
- Monitoraggio di Aurora PostgreSQL Limitless Database con la protezione RDS Amazon GuardDuty
- Funzioni e visualizzazioni per Aurora PostgreSQL Limitless Database
- Eventi di attesa per Aurora PostgreSQL Limitless Database

Monitoraggio di Aurora PostgreSQL Limitless Database con Amazon CloudWatch

Le metriche CloudWatch per Aurora PostgreSQL Limitless Database vengono riportate nelle seguenti dimensioni:

- DBShardGroup
- DBShardGroupRouterAggregation
- DBShardGroupInstance
- DBClusterIdentifier

Per ulteriori informazioni sui parametri di CloudWatch, consulta Monitoraggio dei parametri di Amazon Aurora con Amazon CloudWatch.

Metriche DBShardGroup

Per visualizzare le metriche `DBShardGroup` per Aurora PostgreSQL Limitless Database nella console CloudWatch, scegli RDS e poi DBShardGroup.

Puoi tenere traccia delle seguenti metriche di CloudWatch:

- `DBShardGroupACUUtilization`: l'utilizzo dell'unità di capacità Aurora (ACU) come percentuale calcolata dividendo `DBShardGroupCapacity` per `DBShardGroupMaxACU`.
- `DBShardGroupCapacity`: il numero di ACU utilizzate dalle istanze di scrittura del gruppo di shard del database.
- `DBShardGroupComputeRedundancyCapacity`: il numero di ACU utilizzate dalle istanze di standby del gruppo di shard del database.
- `DBShardGroupMaxACU`: il numero massimo di ACU configurate per il gruppo di shard del database.
- `DBShardGroupMinACU`: il numero minimo di ACU richiesto dal gruppo di shard del database.

La chiave di dimensione `DBShardGroupIdentifier` è disponibile per l'aggregazione delle metriche `DBShardGroup`.

Metriche di DBShardGroupRouterAggregation

Per visualizzare le metriche `DBShardGroupRouterAggregation` per Aurora PostgreSQL Limitless Database nella console CloudWatch, scegli RDS e poi DBShardGroupRouterAggregation.

Puoi tenere traccia delle seguenti metriche di CloudWatch:

- `CommitThroughput`: il numero medio di operazioni di commit al secondo su tutti i nodi del router del gruppo di shard del database.
- `DatabaseConnections`: la somma di tutte le connessioni su tutti i nodi del router nel gruppo di shard del database.

Metriche di DBShardGroupInstance

Un'istanza DBShardGroupInstance rappresenta una singola istanza database all'interno di ogni shard o cluster secondario del router.

Per visualizzare le metriche `DBShardGroupInstance` per Aurora PostgreSQL Limitless Database nella console CloudWatch, scegli RDS e poi DBShardGroupInstance.

Puoi tenere traccia delle seguenti metriche di CloudWatch:

- `ACUUtilization`: la percentuale calcolata dividendo la metrica `ServerlessDatabaseCapacity` per il valore massimo di ACU assegnato al cluster secondario.
- `AuroraReplicaLag`: per i cluster Limitless con ridondanza di calcolo abilitata, il ritardo durante la replica degli aggiornamenti dall'istanza primaria del cluster secondario.
- `AuroraReplicaLagMaximum`: per i cluster Limitless con ridondanza di calcolo abilitata, il ritardo massimo durante la replica degli aggiornamenti dall'istanza primaria del cluster secondario. Quando le repliche di lettura vengono eliminate o rinominate, potrebbe verificarsi un picco temporaneo del ritardo di replica dovuto al processo di riciclaggio a cui è sottoposta la vecchia risorsa. Utilizza questa metrica per determinare se si è verificato un failover dovuto all'elevato ritardo della replica su un'istanza di lettura.
- `AuroraReplicaLagMinimum`: per i cluster Limitless con ridondanza di calcolo abilitata, il ritardo minimo durante la replica degli aggiornamenti dall'istanza primaria del cluster secondario.
- `BufferCacheHitRatio`: la percentuale di dati e indici serviti dalla cache di memoria dell'istanza, invece che dal volume di archiviazione.
- `CommitLatency`: la durata media necessaria al motore e all'archiviazione per completare le operazioni di commit in uno specifico nodo (router o shard).

- `CommitThroughput`: il numero medio di operazioni di commit al secondo.
- `CPUUtilization`: l'utilizzo della CPU come percentuale del valore ACU massimo assegnato al cluster secondario.
- `FreeableMemory`: la quantità di memoria inutilizzata disponibile quando il gruppo di shard viene scalato alla sua capacità massima. Questo valore è determinato in base alle ACU assegnate al gruppo di shard. Per ogni ACU la cui capacità corrente è inferiore alla capacità massima, questo valore aumenta di circa 2 GiB. Di conseguenza, questa metrica non si avvicina a zero finché il gruppo di shard del database non viene aumentato verticalmente fino al limite massimo.
- `MaximumUsedTransactionIDs`: l'età dell'ID transazione non sottoposto al comando VACUUM meno recente, nelle transazioni. Se questo valore raggiunge 2.146.483.648 (2^31 - 1.000.000), il database viene forzato in modalità di sola lettura, per evitare il wraparound dell'ID transazione. Per ulteriori informazioni, consulta Impedire gli errori wraparound dell'ID transazione nella documentazione di PostgreSQL.
- `NetworkReceiveThroughput`: la quantità di throughput della rete ricevuto dai client da ogni istanza del gruppo di shard del database. Questo throughput non include il traffico di rete tra le istanze nel gruppo di shard del database e il volume del cluster.
- `NetworkThroughput`: il throughput della rete aggregato (sia trasmesso che ricevuto) tra client e router, e tra router e shard all'interno del gruppo di shard del database. Questo throughput non include il traffico di rete tra le istanze nel gruppo di shard del database e il volume del cluster.
- `NetworkTransmitThroughput`: la quantità di throughput della rete inviato ai client da ogni istanza del gruppo di shard del database. Questo throughput non include il traffico di rete tra le istanze nel gruppo di shard del database e il volume del cluster.
- `ReadIOPS`: il numero medio di operazioni di input/output al secondo (IOPS) di lettura su disco.
- `ReadLatency`: la quantità di tempo medio impiegato per ogni operazione di input/output (I/O) di lettura su disco.
- `ReadThroughput`: il numero medio di byte letti dal disco al secondo.
- `ServerlessDatabaseCapacity`: la capacità attuale dello shard del database o del cluster secondario del router all'interno del gruppo di shard.
- `StorageNetworkReceiveThroughput`: la quantità di throughput della rete ricevuto dal sottosistema di archiviazione Aurora da ogni istanza nel gruppo di shard del database.
- `StorageNetworkThroughput`: il throughput della rete aggregato, sia trasmesso che ricevuto, tra ogni istanza del gruppo di shard del database e il sottosistema di archiviazione Aurora.
- `StorageNetworkTransmitThroughput`: la quantità di throughput della rete inviato al sottosistema di archiviazione Aurora da ogni istanza del gruppo di shard del database.

- `SwapUsage`: la quantità di spazio di swap utilizzato dal gruppo di shard del database.
- `TempStorageIOPS`: il numero medio di operazioni di I/O eseguite sull'archiviazione locale collegata all'istanza database. Include operazioni di I/O di lettura e scrittura.

  È possibile utilizzare `TempStorageIOPS` e `TempStorageThroughput` per diagnosticare i rari casi in cui l'attività di rete relativa ai trasferimenti tra le istanze database e i dispositivi di archiviazione locale è responsabile di aumenti imprevisti della capacità.
- `TempStorageThroughput`: la quantità di dati trasferiti da e verso l'archiviazione locale associata a un router o a uno shard.
- `WriteIOPS`: il numero medio di IOPS di scrittura su disco.
- `WriteLatency`: la quantità di tempo media impiegato per ogni operazione di I/O di scrittura su disco.
- `WriteThroughput`: il numero medio di byte scritti su disco al secondo.

Le seguenti chiavi di dimensione sono disponibili per l'aggregazione delle metriche `DBShardGroupInstance`.

- `DBClusterIdentifier`: il cluster di database Aurora PostgreSQL.
- `DBShardGroupIdentifier`: il gruppo di shard del database a cui appartiene l'istanza.
- `DBShardGroupSubClusterType`: il tipo di nodo, `Distributed Transaction Router` (router) o `Data Access Shard` (shard).
- `DBShardGroupSubClusterIdentifier`: il nome del router o dello shard a cui appartiene l'istanza.

Di seguito sono riportati alcuni esempi di aggregazione delle metriche di CloudWatch:

- Totale `CPUUtilization` di tutte le istanze che appartengono a un particolare shard o router in un gruppo di shard del database.
- Totale `CPUUtilization` di tutte le istanze in un gruppo di shard del database.

Metriche di DBClusterIdentifier

Per visualizzare le metriche `DBClusterIdentifier` per Aurora PostgreSQL Limitless Database nella console CloudWatch, scegli RDS e poi DBClusterIdentifier.

Quando utilizzi Aurora PostgreSQL Limitless Database, potresti riscontrare un numero maggiore di operazioni di input/output (I/O) rispetto a un cluster di database Aurora. Puoi tenere traccia delle seguenti metriche di CloudWatch per il cluster Limitless Database:

- `VolumeReadIops`: il numero di operazioni I/O di lettura fatturate da un volume del cluster, indicato a intervalli di 5 minuti.
- `VolumeWriteIops`: il numero di operazioni I/O di scrittura su disco verso il volume del cluster, indicato a intervalli di 5 minuti.

Aurora PostgreSQL Limitless Database utilizza la configurazione di archiviazione del cluster Aurora I/O-Optimized. Con Aurora I/O-Optimized, paghi un unico prezzo mensile per tutte le operazioni di I/O, invece di un costo per ogni milione di richieste I/O. Per ulteriori informazioni, consulta Configurazioni dell'archiviazione per i cluster database Amazon Aurora.

Potresti anche utilizzare una quantità di archiviazione maggiore rispetto a un cluster di database Aurora. Puoi tenere traccia delle seguenti metriche di CloudWatch relative all'archiviazione:

- `BackupRetentionPeriodStorageUsed`: il totale dell'utilizzo fatturato dell'archiviazione di backup continuo del cluster Aurora PostgreSQL Limitless Database.
- `SnapshotStorageUsed`: il totale dell'utilizzo fatturato dell'archiviazione degli snapshot del cluster Aurora PostgreSQL Limitless Database.
- `TotalBackupStorageBilled`: la somma dei costi per la conservazione dei backup automatici e per gli snapshot del cluster di database.

  Per ulteriori informazioni sui costi dell'archiviazione di backup, consulta Informazioni sull'utilizzo dello storage di backup Amazon Aurora.
- `VolumeBytesUsed`: la quantità di archiviazione utilizzata dal cluster Aurora PostgreSQL Limitless Database, indicata a intervalli di 5 minuti.

Monitoraggio di Aurora PostgreSQL Limitless Database con CloudWatch Database Insights

La modalità standard di Database Insights è richiesta come parte dell'abilitazione di Aurora PostgreSQL Limitless Database. Puoi utilizzarla per monitorare il carico del database (carico del DB) dell'istanza database Limitless Database in tempo reale. L'opzione Carico del DB misura il livello di attività della sessione in un database. Puoi utilizzare Database Insights per analizzare e risolvere i problemi di prestazioni delle istanze database Aurora PostgreSQL Limitless Database su larga scala.

Per ulteriori informazioni su CloudWatch Database Insights, consulta quanto segue.

- Monitoraggio dei database Amazon Aurora con CloudWatch Database Insights
- CloudWatch Database Insights nella Guida per l'utente di Amazon CloudWatch
- Nozioni di base di CloudWatch Database Insights nella Guida per l'utente di Amazon CloudWatch
- Configurazione del database per monitorare le query SQL lente con Database Insights per Amazon Aurora

Per informazioni sull'attivazione della modalità avanzata o della modalità standard di Database Insights, consulta i seguenti argomenti.

Argomenti

- Attivazione della modalità avanzata di Database Insights per Aurora PostgreSQL Limitless Database
- Attivazione della modalità standard di Database Insights per Aurora PostgreSQL Limitless Database

Attivazione della modalità avanzata di Database Insights per Aurora PostgreSQL Limitless Database

Per attivare la modalità avanzata di Database Insights per il database Aurora PostgreSQL Limitless, utilizzare le seguenti procedure.

Attivazione della modalità avanzata di Database Insights durante la creazione di un cluster di database per Aurora PostgreSQL Limitless Database

Attiva la modalità avanzata di Database Insights durante la creazione di un database per Aurora PostgreSQL Limitless Database.

Console

Nella console è possibile attivare la modalità avanzata di Database Insights quando si crea un cluster di database. Le impostazioni per Database Insights si applicano a tutte le istanze database nel cluster di database.

Per attivare la modalità avanzata di Database Insights durante la creazione di un cluster di database utilizzando la console

1. Accedi alla Console di gestione AWS e apri la console Amazon RDS all'indirizzo https://console.aws.amazon.com/rds/.
2. Scegli Database.
3. Scegliere Crea database.
4. Nella sezione Database Insights, seleziona Modalità avanzata. Quindi, scegli le seguenti opzioni:
   - Retention (Conservazione) – Quantità di tempo per cui conservare i dati di Performance Insights. Il periodo di conservazione deve essere di 15-24 mesi per la modalità avanzata di Database Insights.
   - AWS KMS key: specificare la chiave KMS. Performance Insights crittografa tutti i dati potenzialmente sensibili con la chiave KMS. I dati vengono crittografati mentre sono in transito o inattivi. Per ulteriori informazioni, consulta Crittografia delle risorse Amazon Aurora.
5. Scegliere Crea database.

AWS CLI

Per attivare la modalità avanzata di Database Insights durante la creazione di un cluster di database, chiama il comando AWS CLI create-db-cluster e fornisci i seguenti valori:

- `--db-cluster-identifier`: l'identificatore del cluster di database.
- `--database-insights-mode advanced` per attivare la modalità avanzata di Database Insights.
- `--engine`: il cluster di database deve utilizzare il motore di database `aurora-postgresql`.
- `--engine-version`: il cluster di database deve utilizzare una delle versioni supportate del motore di database.
  - `16.4-limitless`

  - `16.6-limitless`
- `--storage-type`: il cluster di database deve utilizzare la configurazione di archiviazione del cluster di database `aurora-iopt1`.
- `--cluster-scalability-type`: specifica la modalità di scalabilità del cluster di database Aurora. Se impostato su `limitless`, il cluster funziona come Aurora PostgreSQL Limitless Database. Se impostato su `standard` (impostazione predefinita), il cluster utilizza la normale creazione di istanze database.

  > **Note**
  >
  > Non è possibile modificare questa impostazione dopo la creazione del cluster di database.

- `--master-username`: il nome dell'utente master del cluster di database.
- `--master-user-password`: la password dell'utente master.
- `--enable-performance-insights` per attivare Approfondimenti sulle prestazioni per Database Insights.
- `--performance-insights-retention-period`: il periodo di conservazione dei dati per il cluster di database. Per attivare Database Insights, il periodo di conservazione deve essere di almeno 465 giorni.
- `--monitoring-interval`: l'intervallo, in secondi, tra i punti di raccolta delle metriche di monitoraggio avanzato per il cluster di database. Questo valore può essere `0`.
- `--monitoring-role-arn`: il nome della risorsa Amazon (ARN) del ruolo IAM che consente a RDS di inviare le metriche di monitoraggio avanzato ad Amazon CloudWatch Logs.
- `--enable-cloudwatch-logs-exports`: è necessario esportare i log `postgresql` in CloudWatch Logs.

L'esempio seguente abilita la modalità avanzata di Database Insights durante la creazione di un cluster di database.

Per Linux, macOS o Unix:

```
aws rds create-db-cluster \
--db-cluster-identifier my-limitless-cluster \
--database-insights-mode advanced \
--engine aurora-postgresql \
```

```
--engine-version 16.6-limitless \
--storage-type aurora-iopt1 \
--cluster-scalability-type limitless \
--master-username myuser \
--master-user-password mypassword \
--enable-performance-insights \
--performance-insights-retention-period 465 \
--monitoring-interval 5 \
--monitoring-role-arn arn:aws:iam::123456789012:role/EMrole \
--enable-cloudwatch-logs-exports postgresql
```

Per Windows:

```
aws rds create-db-cluster ^
--db-cluster-identifier my-limitless-cluster ^
--database-insights-mode advanced ^
--engine aurora-postgresql ^
--engine-version 16.6-limitless ^
--storage-type aurora-iopt1 ^
--cluster-scalability-type limitless ^
--master-username myuser ^
--master-user-password mypassword ^
--enable-performance-insights ^
--performance-insights-retention-period 465 ^
--monitoring-interval 5 ^
--monitoring-role-arn arn:aws:iam::123456789012:role/EMrole ^
--enable-cloudwatch-logs-exports postgresql
```

RDS API

Per attivare la modalità avanzata di Database Insights quando crei , specifica i seguenti parametri per l'operazione API CreateDBCluster di Amazon RDS.

- `DatabaseInsightsMode` Da a `advanced`
- `Engine` Da a `aurora-postgresql`
- Da `EngineVersion` a una versione del motore disponibile per Limitless Database
- `StorageType` Da a `aurora-iopt1`
- `ClusterScalabilityType` Da a `limitless`
- `MasterUsername`
- `MasterUserPassword`

- `EnablePerformanceInsights` Da a `True`
- Da `PerformanceInsightsRetentionPeriod` ad almeno `465` giorni
- Da `MonitoringInterval` a un valore diverso da `0`
- Da `MonitoringRoleArn` al nome della risorsa Amazon (ARN) del ruolo IAM che consente a RDS di inviare le metriche di monitoraggio avanzato ad Amazon CloudWatch Logs.

Attivazione della modalità avanzata di Database Insights durante la modifica di un cluster di database per Aurora PostgreSQL Limitless Database

Attiva Database Insights durante la modifica di un database per Aurora PostgreSQL Limitless Database.

> **Note**
>
> Per abilitare Database Insights, ogni istanza database in un cluster di database deve avere le stesse impostazioni di Approfondimenti sulle prestazioni e monitoraggio avanzato.

Console

Nella console è possibile attivare la modalità avanzata di Database Insights durante la modifica di un cluster di database. Le impostazioni per Database Insights si applicano a tutte le istanze database nel cluster di database.

Per attivare la modalità avanzata di Database Insights durante la modifica di un cluster di database utilizzando la console

1. Accedi alla Console di gestione AWS e apri la console Amazon RDS all'indirizzo https://console.aws.amazon.com/rds/.
2. Scegli Database.
3. Scegli un cluster di database e scegli Modifica.
4. Nella sezione Database Insights, seleziona Modalità avanzata. Quindi, scegli le seguenti opzioni:
    - Retention (Conservazione) – Quantità di tempo per cui conservare i dati di Performance Insights. Il periodo di conservazione deve essere di 15-24 mesi per la modalità avanzata di Database Insights.

- AWS KMS key: specificare la chiave KMS. Performance Insights crittografa tutti i dati potenzialmente sensibili con la chiave KMS. I dati vengono crittografati mentre sono in transito o inattivi. Per ulteriori informazioni, consulta Crittografia delle risorse Amazon Aurora.

5. Scegli Continue (Continua).
6. In Scheduling of Modifications (Pianificazione delle modifiche), scegliere Apply immediately (Applica immediatamente). Se scegli Applica durante la prossima finestra di manutenzione pianificata, il database ignora questa impostazione e attiva immediatamente la modalità avanzata di Database Insights.
7. Scegli Modifica cluster.

AWS CLI

Per attivare la modalità avanzata di Database Insights durante la modifica di un cluster di database, chiama il comando AWS CLI modify-db-cluster e fornisci i seguenti valori:

- `--database-insights-mode advanced` per attivare la modalità avanzata di Database Insights.
- `--db-cluster-identifier`: l'identificatore del cluster di database.
- `--enable-performance-insights` per attivare Approfondimenti sulle prestazioni per Database Insights.
- `--performance-insights-retention-period`: il periodo di conservazione dei dati per il cluster di database. Per attivare la modalità avanzata di Database Insights, il periodo di conservazione deve essere di almeno 465 giorni.

L'esempio seguente abilita la modalità avanzata di Database Insights quando si modifica un cluster di database.

Per Linux, macOS o Unix:

```
aws rds modify-db-cluster \
    --database-insights-mode advanced \
    --db-cluster-identifier sample-db-identifier \
    --enable-performance-insights \
    --performance-insights-retention-period 465
```

Per Windows:

```
aws rds modify-db-cluster ^
    --database-insights-mode advanced ^
    --db-cluster-identifier sample-db-identifier ^
    --enable-performance-insights ^
    --performance-insights-retention-period 465
```

RDS API

Per attivare la modalità avanzata di Database Insights durante la modifica di un cluster di database, specifica i seguenti parametri per l'operazione API ModifyDBCluster di Amazon RDS.

- `DatabaseInsightsMode` Da a `advanced`
- `EnablePerformanceInsights` Da a `True`
- Da `PerformanceInsightsRetentionPeriod` ad almeno `465` giorni

Attivazione della modalità standard di Database Insights per Aurora PostgreSQL Limitless Database

Per attivare la modalità standard di Database Insights per Aurora PostgreSQL Limitless Database, utilizza le procedure seguenti.

Attivazione della modalità standard di Database Insights durante la creazione di un cluster di database per Aurora PostgreSQL Limitless Database

Attiva la modalità standard di Database Insights durante la creazione di un database per Aurora PostgreSQL Limitless Database.

Console

Nella console, puoi attivare la modalità standard di Database Insights quando crei un cluster di database. Le impostazioni per Database Insights si applicano a tutte le istanze database nel cluster di database.

Per attivare la modalità standard di Database Insights durante la creazione di un cluster di database utilizzando la console

1. Accedi alla Console di gestione AWS e apri la console Amazon RDS all'indirizzo https://console.aws.amazon.com/rds/.
2. Scegli Database.

3. Scegliere Crea database.
4. Nella sezione Database Insights, seleziona Modalità standard. Quindi, scegli le seguenti opzioni:
   - Retention (Conservazione) – Quantità di tempo per cui conservare i dati di Performance Insights. Per creare un cluster di database per Aurora PostgreSQL Limitless Database, il periodo di conservazione deve essere di almeno 31 giorni.
   - AWS KMS key: specificare la chiave KMS. Performance Insights crittografa tutti i dati potenzialmente sensibili con la chiave KMS. I dati vengono crittografati mentre sono in transito o inattivi. Per ulteriori informazioni, consulta Crittografia delle risorse Amazon Aurora.
5. Scegliere Crea database.

AWS CLI

Per attivare la modalità standard di Database Insights durante la creazione di un cluster di database, chiama il comando AWS CLI create-db-cluster e fornisci i seguenti valori:

- `--db-cluster-identifier`: l'identificatore del cluster di database.
- `--database-insights-mode standard` per attivare la modalità standard di Database Insights.
- `--engine`: il cluster di database deve utilizzare il motore di database `aurora-postgresql`.
- `--engine-version`: il cluster di database deve utilizzare una delle versioni supportate del motore di database.
  - `16.4-limitless`
  - `16.6-limitless`
- `--storage-type`: il cluster di database deve utilizzare la configurazione di archiviazione del cluster di database `aurora-iopt1`.
- `--cluster-scalability-type`: specifica la modalità di scalabilità del cluster di database Aurora. Se impostato su `limitless`, il cluster funziona come Aurora PostgreSQL Limitless Database. Se impostato su `standard` (impostazione predefinita), il cluster utilizza la normale creazione di istanze database.

> **Note**
>
> Non è possibile modificare questa impostazione dopo la creazione del cluster di database.

- `--master-username`: il nome dell'utente master del cluster di database.
- `--master-user-password`: la password dell'utente master.
- `--enable-performance-insights` per attivare Approfondimenti sulle prestazioni per Database Insights.
- `--performance-insights-retention-period`: il periodo di conservazione dei dati per il cluster di database. Per creare un cluster di database per Aurora PostgreSQL Limitless Database, il periodo di conservazione deve essere di almeno 31 giorni.
- `--monitoring-interval`: l'intervallo, in secondi, tra i punti di raccolta delle metriche di monitoraggio avanzato per il cluster di database. Questo valore può essere `0`.
- `--monitoring-role-arn`: il nome della risorsa Amazon (ARN) del ruolo IAM che consente a RDS di inviare le metriche di monitoraggio avanzato ad Amazon CloudWatch Logs.
- `--enable-cloudwatch-logs-exports`: è necessario esportare i log `postgresql` in CloudWatch Logs.

L'esempio seguente abilita la modalità standard di Database Insights durante la creazione di un cluster di database.

Per Linux, macOS o Unix:

```
aws rds create-db-cluster \
--db-cluster-identifier my-limitless-cluster \
--database-insights-mode standard \
--engine aurora-postgresql \
--engine-version 16.6-limitless \
--storage-type aurora-iopt1 \
--cluster-scalability-type limitless \
--master-username myuser \
--master-user-password mypassword \
--enable-performance-insights \
--performance-insights-retention-period 31 \
--monitoring-interval 5 \
```

```
--monitoring-role-arn arn:aws:iam::123456789012:role/EMrole \
--enable-cloudwatch-logs-exports postgresql
```

Per Windows:

```
aws rds create-db-cluster ^
--db-cluster-identifier my-limitless-cluster ^
--database-insights-mode standard ^
--engine aurora-postgresql ^
--engine-version 16.6-limitless ^
--storage-type aurora-iopt1 ^
--cluster-scalability-type limitless ^
--master-username myuser ^
--master-user-password mypassword ^
--enable-performance-insights ^
--performance-insights-retention-period 31 ^
--monitoring-interval 5 ^
--monitoring-role-arn arn:aws:iam::123456789012:role/EMrole ^
--enable-cloudwatch-logs-exports postgresql
```

RDS API

Per attivare la modalità standard di Database Insights quando crei , specifica i seguenti parametri per l'operazione API CreateDBCluster di Amazon RDS.

- `DatabaseInsightsMode` Da a `standard`
- `Engine` Da a `aurora-postgresql`
- Da `EngineVersion` a una versione del motore disponibile per Limitless Database
- `StorageType` Da a `aurora-iopt1`
- `ClusterScalabilityType` Da a `limitless`
- `MasterUsername`
- `MasterUserPassword`
- `EnablePerformanceInsights` Da a `True`
- Da `PerformanceInsightsRetentionPeriod` ad almeno `31` giorni
- Da `MonitoringInterval` a un valore diverso da `0`
- Da `MonitoringRoleArn` al nome della risorsa Amazon (ARN) del ruolo IAM che consente a RDS di inviare le metriche di monitoraggio avanzato ad Amazon CloudWatch Logs.

Attivazione della modalità standard di Database Insights durante la modifica di un cluster di database per Aurora PostgreSQL Limitless Database

Attiva Database Insights durante la modifica di un database per Aurora PostgreSQL Limitless Database.

> **Note**
>
> Per abilitare Database Insights, ogni istanza database in un cluster di database deve avere le stesse impostazioni di Approfondimenti sulle prestazioni e Monitoraggio avanzato.

Console

Nella console, puoi attivare la modalità standard di Database Insights quando crei un cluster di database. Le impostazioni per Database Insights si applicano a tutte le istanze database nel cluster di database.

Per attivare la modalità standard di Database Insights durante la modifica di un cluster di database utilizzando la console

1. Accedi alla Console di gestione AWS e apri la console Amazon RDS all'indirizzo https://console.aws.amazon.com/rds/.
2. Scegli Database.
3. Scegli un cluster di database e scegli Modifica.
4. Nella sezione Database Insights, seleziona Modalità standard. Quindi, scegli le seguenti opzioni:
   - Retention (Conservazione) – Quantità di tempo per cui conservare i dati di Performance Insights. Per creare un cluster di database per Aurora PostgreSQL Limitless Database, il periodo di conservazione deve essere di almeno 31 giorni.
   - AWS KMS key: specificare la chiave KMS. Performance Insights crittografa tutti i dati potenzialmente sensibili con la chiave KMS. I dati vengono crittografati mentre sono in transito o inattivi. Per ulteriori informazioni, consulta Crittografia delle risorse Amazon Aurora.
5. Scegli Continue (Continua).
6. In Scheduling of Modifications (Pianificazione delle modifiche), scegliere Apply immediately (Applica immediatamente). Se scegli Applica durante la prossima finestra di manutenzione

pianificata, il database ignora questa impostazione e attiva immediatamente la modalità standard di Database Insights.

7. Scegli Modifica cluster.

AWS CLI

Per attivare la modalità standard di Database Insights durante la modifica di un cluster di database, chiama il comando AWS CLI modify-db-cluster e fornisci i seguenti valori:

- `--db-cluster-identifier`: l'identificatore del cluster di database.
- `--database-insights-mode standard` per attivare la modalità standard di Database Insights.
- `--enable-performance-insights` per attivare Approfondimenti sulle prestazioni per Database Insights.
- `--performance-insights-retention-period`: il periodo di conservazione dei dati per il cluster di database. Per attivare la modalità standard di Database Insights, il periodo di conservazione deve essere di almeno 31 giorni.

L'esempio seguente abilita la modalità standard di Database Insights durante la modifica di un cluster di database.

Per Linux, macOS o Unix:

```
aws rds modify-db-cluster \
    --database-insights-mode standard \
    --db-cluster-identifier sample-db-identifier \
    --enable-performance-insights \
    --performance-insights-retention-period 31
```

Per Windows:

```
aws rds modify-db-cluster ^
    --database-insights-mode standard ^
    --db-cluster-identifier sample-db-identifier ^
    --enable-performance-insights ^
    --performance-insights-retention-period 31
```

RDS API

Per attivare la modalità standard di Database Insights durante la modifica di un cluster di database, specifica i seguenti parametri per l'operazione API ModifyDBCluster di Amazon RDS.

- `DatabaseInsightsMode` Da a `standard`
- `EnablePerformanceInsights` Da a `True`
- Da `PerformanceInsightsRetentionPeriod` ad almeno `31` giorni

Monitoraggio di Aurora PostgreSQL Limitless Database con Amazon CloudWatch Logs

L'esportazione dei log di PostgreSQL in CloudWatch Logs è obbligatoria come parte del processo di abilitazione di Aurora PostgreSQL Limitless Database. Puoi accedere e analizzare questi log in CloudWatch Logs Insights, in modo analogo a come accedi ai log di PostgreSQL per un cluster di database Aurora PostgreSQL standard. Per ulteriori informazioni, consulta Analisi dei registri Aurora PostgreSQL utilizzando approfondimenti sui file di log Amazon CloudWatch.

Il nome del gruppo di log per il cluster di database è lo stesso di Aurora PostgreSQL:

```
/aws/rds/cluster/DB_cluster_ID/postgresql
```

Il nome del gruppo di log per il gruppo di shard del database assume il seguente formato:

```
/aws/rds/cluster/DB_cluster_ID/DB_shard_group_ID/postgresql
```

Esistono flussi di log per ogni nodo (router o shard). Queste espressioni hanno il seguente formato:

```
[DistributedTransactionRouter|
DataAccessShard]/node_cluster_serial_ID-node_instance_serial_ID/n
```

Per esempio:

- Router: `DistributedTransactionRouter/6-6.2`
- Shard: `DataAccessShard/22-22.0`

> **Note**
>
> Non puoi visualizzare direttamente i file di log di PostgreSQL per il gruppo di shard del database nella console RDS, nella AWS CLI o tramite nell'API RDS, come invece avviene per un cluster di database. Devi utilizzare CloudWatch Logs Insights per visualizzarli.

Monitoraggio di Aurora PostgreSQL Limitless Database con Monitoraggio avanzato

Monitoraggio avanzato è necessario come parte dell'abilitazione di Aurora PostgreSQL Limitless Database. Puoi usarlo per monitorare il sistema operativo dell'istanza database Limitless Database in tempo reale.

Aurora pubblica le metriche di Monitoraggio avanzato in CloudWatch Logs. Alcune delle metriche principali disponibili includono le connessioni al database, l'utilizzo dell'archiviazione e la latenza delle query. Queste possono aiutare a identificare i colli di bottiglia delle prestazioni.

Per ulteriori informazioni sulle metriche di Monitoraggio avanzato, consulta Parametri del sistema operativo per Aurora.

Monitoraggio di Aurora PostgreSQL Limitless Database con Approfondimenti sulle prestazioni

Utilizza Approfondimenti sulle prestazioni per monitorare il cluster Aurora PostgreSQL Limitless Database. Approfondimenti sulle prestazioni funziona in modo simile per Aurora PostgreSQL Limitless Database e per i cluster di database Aurora standard. Tuttavia, monitora le metriche a livello di gruppo di shard per Aurora PostgreSQL Limitless Database.

Le due metriche principali di Approfondimenti sulle prestazioni da monitorare sono le seguenti:

- Il carico del database: misura il livello di attività nel database. Il parametro chiave in Performance Insights è `DBLoad`, che viene raccolto ogni secondo.

  L'unità della metrica `DBLoad` in Approfondimenti sulle prestazioni sono le sessioni attive medie (AAS). Per ottenere le sessioni attive medie (AAS), Performance Insights esegue il campionamento del numero di sessioni che eseguono contemporaneamente una query. L'AAS è il numero totale di sessioni diviso per il numero totale di campioni per un periodo di tempo specifico. Per ulteriori informazioni su `DBLoad` e AAS, consulta Carico del database.
- CPU massima: la massima potenza di calcolo disponibile per il database. Per verificare se le sessioni attive superano la CPU massima, esamina la loro relazione con la linea `Max vCPU` massima. Il valore `Max vCPU` massima è determinato dal numero di core vCPU (CPU virtuale) per l'istanza database. Per ulteriori informazioni su `Max vCPU`, consulta CPU massima.

Inoltre, puoi "sezionare" la metrica `DBLoad` in dimensioni, che sono sottocategorie della metrica. Le dimensioni più utili sono le seguenti:

- Istanze principali: mostra il carico del database relativo per le istanze (shard e router) in ordine decrescente.
- Eventi di attesa: indicano eventi per cui le istruzioni SQL devono attendere prima di poter continuare l'esecuzione. Gli eventi di attesa indicano dove il processo è ostacolato.
- SQL principale: mostra quali query contribuiscono maggiormente al carico del database.

Per ulteriori informazioni su Approfondimenti sulle prestazioni, consulta Dimensioni.

La figura seguente mostra la dimensione Istanze principali per un gruppo di shard del database.



Argomenti

- [Analisi del carico del database per Aurora PostgreSQL Limitless Database tramite la dashboard Approfondimenti sulle prestazioni]

Analisi del carico del database per Aurora PostgreSQL Limitless Database tramite la dashboard Approfondimenti sulle prestazioni

Con Approfondimenti sulle prestazioni, puoi tenere traccia delle metriche a livello di gruppo di shard e a livello di istanza per un database Aurora PostgreSQL Limitless. Quando si analizza il carico di database per un database Aurora PostgreSQL Limitless, è possibile confrontare il carico del DB per ogni shard e router con la vCPU massima.

> **Note**
>
> Aurora PostgreSQL Limitless Database ha sempre abilitato Approfondimenti sulle prestazioni e Monitoraggio avanzato. Il periodo minimo di conservazione dei dati di Approfondimenti sulle prestazioni per Limitless Database è di 31 giorni (1 mese).

La visualizzazione Assoluto mostra il numero di sessioni attive medie (AAS) e la vCPU stimata. La visualizzazione Relativo mostra il rapporto tra AAS e la vCPU stimata.

Argomenti

- Analisi del carico di database relativo per Aurora PostgreSQL Limitless Database tramite la dashboard di Approfondimenti sulle prestazioni
- Analisi del carico del database in base alle attese per Aurora PostgreSQL Limitless Database tramite la dashboard Approfondimenti sulle prestazioni
- Analisi della distribuzione del carico per Aurora PostgreSQL Limitless Database tramite la dashboard Approfondimenti sulle prestazioni

Analisi del carico di database relativo per Aurora PostgreSQL Limitless Database tramite la dashboard di Approfondimenti sulle prestazioni

Potresti voler migliorare le prestazioni del tuo database Aurora PostgreSQL Limitless monitorando il carico relativo del database. Per analizzare il carico di database relativo per istanza per il database Aurora PostgreSQL Limitless, utilizzare la procedura seguente.

Per analizzare il carico relativo del database per Aurora PostgreSQL Limitless Database utilizzando la console

1. Apri la console Amazon RDS all'indirizzo https://console.aws.amazon.com/rds/.

2. Nel pannello di navigazione scegli Performance Insights.
3. Scegli un database Aurora PostgreSQL Limitless Viene visualizzata la dashboard di Approfondimenti sulle prestazioni per il database Aurora PostgreSQL Limitless.
4. Nella sezione Carico del database (Carico DB), scegli Istanze per Suddivisione per. Per visualizzare il rapporto tra sessioni attive medie (AAS) e core vCPU per tutte le istanze database Aurora PostgreSQL Limitless, scegli Relativo per Visualizzato come.

   Il grafico Average active sessions (Media delle sessioni attive) mostra il carico di database per le istanze nel database Aurora PostgreSQL Limitless.



5. Per visualizzare le istanze principali, scegli la scheda Istanze principali.

   Nell'esempio seguente, l'istanza con il carico di database più elevato è `DTR-2-2`.



6. (Facoltativo) Per analizzare il carico di database per un'istanza nel tuo database Aurora PostgreSQL Limitless, scegli il nome dell'istanza nella colonna Istanze. Per visualizzare il carico di database per`DTR-2-2`, scegli `DTR-2-2` nella colonna Istanze.

   > **Note**
   >
   > È possibile visualizzare le metriche di Approfondimenti sulle prestazioni solo per le istanze in un database Aurora PostgreSQL Limitless.

Analisi del carico del database in base alle attese per Aurora PostgreSQL Limitless Database tramite la dashboard Approfondimenti sulle prestazioni

Potresti voler migliorare le prestazioni del tuo database Aurora PostgreSQL Limitless monitorando gli eventi di attesa. Per analizzare il carico di database in base agli eventi di attesa per il database Aurora PostgreSQL Limitless, utilizzare la procedura seguente.

Per analizzare il carico di database in base alle attese di Aurora per il database PostgreSQL Limitless utilizzando la console

1. Apri la console Amazon RDS all'indirizzo https://console.aws.amazon.com/rds/.
2. Nel pannello di navigazione scegli Performance Insights.
3. Scegli un database Aurora PostgreSQL Limitless Viene visualizzata la dashboard di Approfondimenti sulle prestazioni per il database Aurora PostgreSQL Limitless.
4. Nella sezione Carico del database (Carico DB), scegli Attese per Suddivisione per. Per visualizzare il numero di AAS e la vCPU stimata, scegli Assoluto per Visualizzato come.

   Il grafico Average active sessions (Media delle sessioni attive) mostra il carico di database per le istanze nel database Aurora PostgreSQL Limitless.



5. Scorri verso il basso fino alla scheda Top SQL (Prime istruzioni SQL).

   Nell'esempio seguente, l'istruzione SQL con il carico per attese più elevato è l'istruzione `DELETE`.



6. Scegli l'istruzione SQL per espanderla nelle istruzioni componenti.

   Nell'esempio seguente, l'istruzione `SELECT` ha 3 istruzioni componenti.



Analisi della distribuzione del carico per Aurora PostgreSQL Limitless Database tramite la dashboard Approfondimenti sulle prestazioni

Potresti voler bilanciare la distribuzione del carico per le istanze sul tuo database Aurora PostgreSQL Limitless. Per analizzare la distribuzione del carico delle istanze su un database Aurora PostgreSQL Limitless, utilizzare la procedura seguente.

Per analizzare la distribuzione del carico delle istanze su un database Aurora PostgreSQL Limitless utilizzando la console

1. Apri la console Amazon RDS all'indirizzo https://console.aws.amazon.com/rds/.
2. Nel pannello di navigazione scegli Performance Insights.
3. Scegli un database Aurora PostgreSQL Limitless Viene visualizzata la dashboard di Approfondimenti sulle prestazioni per il database Aurora PostgreSQL Limitless.
4. Nella sezione Carico del database (Carico DB), scegli Istanze per Suddivisione per. Per visualizzare il numero di AAS e la vCPU stimata per tutte le istanze database Aurora PostgreSQL Limitless, scegli Assoluto per Visualizzato come.

Il grafico Average active sessions (Media delle sessioni attive) mostra il carico di database per le istanze nel database Aurora PostgreSQL Limitless.



5. Per visualizzare un grafico della distribuzione del carico delle istanze nel database Aurora PostgreSQL Limitless, scegli la scheda Distribuzione del carico.

   Nell'esempio seguente, l'istanza con il carico di database più elevato è `DTR-2-2`.



Monitoraggio di Aurora PostgreSQL Limitless Database con la protezione RDS Amazon GuardDuty

Amazon GuardDuty è un servizio di rilevamento delle minacce che aiuta a proteggere gli account, i container, i carichi di lavoro e i dati nell'ambiente AWS. Grazie a modelli di machine learning (ML) e funzionalità di rilevamento di anomalie e minacce, GuardDuty monitora continuamente origini di log e attività di runtime diverse per identificare e dare priorità ai potenziali rischi per la sicurezza e alle attività dannose nell'ambiente.

La protezione RDS GuardDuty analizza e profila gli eventi di accesso per individuare potenziali minacce di accesso ai database Amazon Aurora. Quando attivi la protezione RDS, GuardDuty utilizza gli eventi di accesso RDS dei database Aurora. La protezione RDS monitora questi eventi e li profila per potenziali minacce interne o attori esterni.

Per ulteriori informazioni sulla protezione RDS GuardDuty, consulta Monitoraggio delle minacce con la protezione RDS Amazon GuardDuty per Amazon Aurora.

Per maggiori informazioni sull'abilitazione della protezione RDS GuardDuty, consulta Protezione GuardDuty RDS nella Guida per l'utente di Amazon GuardDuty.

Funzioni e visualizzazioni per Aurora PostgreSQL Limitless Database

In Aurora PostgreSQL Limitless Database sono state aggiunte funzioni e visualizzazioni. Si basano sulle funzioni e sulle visualizzazioni corrispondenti di Aurora PostgreSQL.

> **Note**
>
> Alcune statistiche possono restituire risultati incoerenti se sono presenti transazioni in corso.

Argomenti

- Funzioni di Aurora PostgreSQL Limitless Database
- Visualizzazioni di Aurora PostgreSQL Limitless Database

Funzioni di Aurora PostgreSQL Limitless Database

La tabella seguente mostra le nuove funzioni per Aurora PostgreSQL Limitless Database.

> **Note**
>
> Le funzioni elencate in questa tabella si trovano nello schema `rds_aurora`. Quando utilizzi una funzione di Limitless Database, assicurati di includere il nome completo dell'oggetto: `rds_aurora.`*`object_name`*.

Funzione di Aurora PostgreSQL Limitless Database	Funzione di Aurora PostgreSQL corrispondente
limitless_backend_dsid	pg_backend_pid
limitless_cancel_session	pg_cancel_backend
limitless_stat_clear_snapshot	pg_stat_clear_snapshot
limitless_stat_database_size	pg_database_size
limitless_stat_get_snapshot_timestamp	pg_stat_get_snapshot_timestamp

Funzione di Aurora PostgreSQL Limitless Database	Funzione di Aurora PostgreSQL corrispondente
limitless_stat_prepared_xacts	pg_prepared_xacts
limitless_stat_relation_sizes	pg_indexes_size, pg_relation_size, pg_table_size, pg_total_relation_size
limitless_stat_reset	pg_stat_reset
limitless_stat_statements_reset	pg_stat_statements_reset
limitless_stat_system_waits	aurora_stat_system_waits
limitless_terminate_session	pg_terminate_backend
limitless_wait_report	aurora_wait_report

Gli esempi seguenti forniscono dettagli sulle funzioni di Aurora PostgreSQL Limitless Database. Per ulteriori informazioni sulle funzioni di PostgreSQL, consulta Functions and operators nella documentazione di PostgreSQL.

limitless_backend_dsid

La funzione `limitless_backend_dsid` restituisce l'ID di sessione distribuita per la sessione corrente. Una sessione distribuita viene eseguita su un router in un gruppo di shard del database e coinvolge processi di backend su uno o più shard del gruppo di shard del database.

Il seguente esempio illustra come utilizzare la funzione `limitless_backend_dsid`.

```
SELECT rds_aurora.limitless_backend_dsid();

limitless_backend_dsid
------------------------
8CACD7B04D0FC2A5
(1 row)
```

limitless_cancel_session

La funzione `limitless_cancel_session` funziona in modo simile a `pg_cancel_backend`, ma tenta di annullare tutti i processi di backend associati all'ID di sessione distribuita fornito, inviando un `SIGINT` (segnale di interruzione).

Il parametro di input è il seguente:

- `distributed_session_id` (testo): l'ID della sessione distribuita da annullare.

I parametri di output sono i seguenti:

- `subcluster_id` (testo): l'ID del cluster secondario a cui appartiene il processo.
- `pid` (testo): l'ID del processo di backend.
- `success` (booleano): indica se l'annullamento è riuscito.

Il seguente esempio illustra come utilizzare la funzione `limitless_cancel_session`.

```
SELECT * FROM rds_aurora.limitless_cancel_session('940CD5C81E3C796B');

 subcluster_id |  pid  | success
---------------+-------+---------
             1 | 26920 | t
(1 row)
```

limitless_stat_clear_snapshot

La funzione `limitless_stat_clear_snapshot` elimina lo snapshot delle statistiche corrente o le informazioni memorizzate nella cache su tutti i nodi.

Il seguente esempio illustra come utilizzare la funzione `limitless_stat_clear_snapshot`.

```
SELECT rds_aurora.limitless_stat_clear_snapshot();
```

limitless_stat_database_size

La funzione `limitless_stat_database_size` restituisce le dimensioni di un database nel gruppo di shard del database.

Il parametro di input è il seguente:

- `dbname` (nome): il database per il quale ottenere le dimensioni.

I parametri di output sono i seguenti:

- `subcluster_id` (testo): l'ID del cluster secondario a cui appartiene il processo.
- `subcluster_type` (testo): il tipo di cluster secondario a cui appartiene questo processo (`router` o `shard`).
- `db_size`: la dimensione del database in questo cluster secondario in byte.

Il seguente esempio illustra come utilizzare la funzione `limitless_stat_database_size`.

```
SELECT * FROM rds_aurora.limitless_stat_database_size('postgres_limitless');

 subcluster_id | subcluster_type | db_size
---------------+-----------------+----------
             1 | router          |  8895919
             2 | router          |  8904111
             3 | shard           | 21929391
             4 | shard           | 21913007
             5 | shard           | 21831087
(5 rows)
```

limitless_stat_get_snapshot_timestamp

La funzione `limitless_stat_get_snapshot_timestamp` restituisce il timestamp dello snapshot delle statistiche corrente oppure `NULL` se non è stato acquisito alcuno snapshot. Uno snapshot viene acquisito al primo accesso alle statistiche cumulative in una transazione, se `stats_fetch_consistency` è impostato su `snapshot`. Restituisce una visualizzazione consolidata dei timestamp degli snapshot da tutti i nodi. Le colonne `subcluster_id` e `subcluster_type` mostrano da quale nodo provengono i dati.

Il seguente esempio illustra come utilizzare la funzione `limitless_stat_get_snapshot_timestamp`.

```
SELECT * FROM rds_aurora.limitless_stat_get_snapshot_timestamp();

 subcluster_id | subcluster_type | snapshot_timestamp
---------------+-----------------+--------------------
             1 | router          |
             2 | router          |
             3 | shard           |
             4 | shard           |
             5 | shard           |
```

```
(5 rows)
```

limitless_stat_prepared_xacts

La funzione `limitless_stat_prepared_xacts` restituisce informazioni sulle transazioni su tutti i nodi attualmente preparati per il commit in due fasi. Per ulteriori informazioni, consulta pg_prepared_xacts nella documentazione di PostgreSQL.

Il seguente esempio illustra come utilizzare la funzione `limitless_stat_prepared_xacts`.

```
postgres_limitless=> SELECT * FROM rds_aurora.limitless_stat_prepared_xacts;

 subcluster_id | subcluster_type | transaction_id |             gid              |
         prepared            |  owner_id  |    database_id
---------------+-----------------+----------------+------------------------------
+-------------------------------+------------+--------------------
 8             | shard           |        5815978 | 7_4599899_postgres_limitless |
 2024-09-03 15:51:17.659603+00 | auroraperf | postgres_limitless
 12            | shard           |        4599138 | 7_4599899_postgres_limitless |
 2024-09-03 15:51:17.659637+00 | auroraperf | postgres_limitless
(2 rows)
```

limitless_stat_relation_sizes

La funzione `limitless_stat_relation_sizes` restituisce le diverse dimensioni di una tabella nel gruppo di shard del database.

I parametri di input sono i seguenti:

- `relnspname` (nome): il nome dello schema contenente la tabella.
- `relname` (nome): il nome della tabella.

I parametri di output sono i seguenti:

- `subcluster_id` (testo): l'ID del cluster secondario a cui appartiene il processo.
- `subcluster_type` (testo): il tipo di cluster secondario a cui appartiene questo processo (`router` o `shard`).
- `main_size`: la dimensione in byte della fork dei dati principale in questo nodo.
- `fsm_size`: la dimensione in byte della mappa dello spazio libero per la tabella in questo nodo.
- `vm_size`: la dimensione in byte della mappa di visibilità per la tabella in questo nodo.

- `init_size`: la dimensione in byte dell'inizializzazione della tabella in questo nodo.
- `toast_size`: la dimensione in byte della tabella TOAST associata alla tabella in questa forcella.
- `index_size`: la dimensione in byte di tutti gli indici della tabella in questo nodo.
- `total_size`: la dimensione in byte di tutti i segmenti della tabella in questo nodo.

Il seguente esempio illustra come utilizzare la funzione `limitless_stat_relation_sizes` (alcune colonne sono omesse).

```
SELECT * FROM rds_aurora.limitless_stat_relation_sizes('public','customers');

 subcluster_id | subcluster_type | main_size | fsm_size | vm_size | toast_size |
 table_size | total_size
---------------+-----------------+-----------+----------+---------+------------
+------------+------------
             1 | router          |         0 |        0 |       0 |          0 |
   0 |          0
             2 | router          |         0 |        0 |       0 |          0 |
   0 |          0
             3 | shard           |   4169728 |  4177920 | 1392640 |    1392640 |
 11132928 |   11132928
             4 | shard           |   4169728 |  4177920 | 1392640 |    1392640 |
 11132928 |   11132928
             5 | shard           |   3981312 |  4227072 | 1409024 |    1409024 |
 11026432 |   11026432
(5 rows)
```

limitless_stat_reset

La funzione `limitless_stat_reset` azzera tutti i contatori delle statistiche per il database corrente impostandoli su zero (0). Se `track_functions` è abilitato, la colonna `stats_reset` in `limitless_stat_database` mostra l'ultima volta in cui le statistiche sono state reimpostate per il database. Per impostazione predefinita, `limitless_stat_reset` può essere eseguito solo da un utente con privilegi avanzati. È possibile concedere l'autorizzazione ad altri utenti utilizzando il privilegio EXECUTE.

Il seguente esempio illustra come utilizzare la funzione `limitless_stat_reset`.

```
SELECT tup_inserted, tup_deleted FROM pg_stat_database
WHERE datname = 'postgres_limitless';
```

```
 tup_inserted | tup_deleted
--------------+-------------
          896 |           0
(1 row)

SELECT rds_aurora.limitless_stat_reset();

limitless_stat_reset
---------------------
(1 row)

SELECT tup_inserted, tup_deleted FROM pg_stat_database
WHERE datname = 'postgres_limitless';

tup_inserted | tup_deleted
-------------+-------------
           0 |           0
(1 row)
```

limitless_stat_statements_reset

La funzione `limitless_stat_statements_reset` elimina le statistiche raccolte finora da `limitless_stat_statements` corrispondenti ai parametri `username`, `dbname`, `distributed_query_id` e `queryid` specificati. Se uno qualsiasi dei parametri non è specificato, viene utilizzato il valore predefinito `""` o `0` (non valido) per ciascuno di essi e vengono reimpostate le statistiche che corrispondono agli altri parametri. Se nessun parametro viene specificato o tutti i parametri specificati sono `""` o `0` (non valido), la funzione elimina tutte le statistiche. Se tutte le statistiche nella visualizzazione `limitless_stat_statements` vengono eliminate, la funzione reimposta anche le statistiche nella visualizzazione `limitless_stat_statements_info`.

I parametri di input sono i seguenti:

- `username` (nome): l'utente che ha eseguito la query sull'istruzione.
- `dbname` (nome): il database su cui è stata eseguita la query.
- `distributed_query_id` (bigint): l'ID della query principale proveniente dal nodo coordinatore. Questa colonna è `NULL` se è la query principale. Il nodo coordinatore esegue il push down dell'ID della query distribuita ai nodi partecipanti. Pertanto, per i nodi partecipanti, i valori dell'ID della query distribuita e dell'ID della query principale sono diversi.
- `queryid` (bigint): l'ID della query dell'istruzione.

Il seguente esempio illustra come utilizzare la funzione `limitless_stat_statements_reset` per reimpostare tutte le statistiche raccolte da `limitless_stat_statements`.

```
SELECT rds_aurora.limitless_stat_statements_reset();
```

limitless_stat_system_waits

La funzione `limitless_stat_system_waits` restituisce una visualizzazione consolidata dei dati degli eventi di attesa da `aurora_stat_system_waits`, che riporta l'attività di attesa a livello di sistema in un'istanza, da tutti i nodi. Le colonne `subcluster_id` e `subcluster_type` mostrano da quale nodo provengono i dati.

Il seguente esempio illustra come utilizzare la funzione `limitless_stat_system_waits`.

```
postgres_limitless=> SELECT *
FROM rds_aurora.limitless_stat_system_waits() lssw, pg_catalog.aurora_stat_wait_event()
 aswe
WHERE lssw.event_id=aswe.event_id and aswe.event_name='LimitlessTaskScheduler';

 subcluster_id | subcluster_type | type_id | event_id  | waits  |  wait_time   |
 event_name
---------------+-----------------+---------+-----------+--------+--------------
+------------------------
             1 | router          |      12 | 201326607 | 677068 | 616942216307 |
 LimitlessTaskScheduler
             2 | router          |      12 | 201326607 | 678586 | 616939897111 |
 LimitlessTaskScheduler
             3 | shard           |      12 | 201326607 | 756640 | 616965545172 |
 LimitlessTaskScheduler
             4 | shard           |      12 | 201326607 | 755184 | 616958057620 |
 LimitlessTaskScheduler
             5 | shard           |      12 | 201326607 | 757522 | 616963183539 |
 LimitlessTaskScheduler
(5 rows)
```

limitless_terminate_session

La funzione `limitless_terminate_session` funziona in modo simile a `pg_terminate_backend`, ma tenta di terminare tutti i processi di backend associati all'ID di sessione distribuita fornito, inviando un `SIGTERM` (segnale di termine).

Il parametro di input è il seguente:

- `distributed_session_id` (testo): l'ID della sessione distribuita da terminare.

I parametri di output sono i seguenti:

- `subcluster_id` (testo): l'ID del cluster secondario a cui appartiene il processo.
- `pid` (testo): l'ID del processo di backend.
- `success` (booleano): indica se il processo è stato terminato correttamente.

Il seguente esempio illustra come utilizzare la funzione `limitless_terminate_session`.

```
SELECT * FROM rds_aurora.limitless_terminate_session('940CD5C81E3C796B');

 subcluster_id |  pid  | success
---------------+-------+---------
             1 | 26920 | t
(1 row)
```

limitless_wait_report

La funzione `limitless_wait_report` restituisce l'attività degli eventi di attesa di tutti i nodi in un periodo di tempo. Le colonne `subcluster_id` e `subcluster_type` mostrano da quale nodo provengono i dati.

I parametri di output sono i seguenti:

- `subcluster_id` (testo): l'ID del cluster secondario a cui appartiene il processo.
- `subcluster_type` (testo): il tipo di cluster secondario a cui appartiene questo processo (`router` o `shard`).

Le restanti colonne sono le stesse di `aurora_wait_report`.

Il seguente esempio illustra come utilizzare la funzione `limitless_wait_report`.

```
postgres_limitless=> select * from rds_aurora.limitless_wait_report();

 subcluster_id | subcluster_type | type_name | event_name | waits | wait_time |
 ms_per_wait | waits_per_xact | ms_per_xact
---------------+-----------------+-----------+------------+-------+-----------
+-------------+--------------- +-------------
             1 | router          | Client    | ClientRead |    57 | 741550.14 |
 13009.652 |           0.19 |    2505.237
             5 | shard           | Client    | ClientRead |    54 | 738897.68 |
 13683.290 |           0.18 |    2496.276
```

```
            4 | shard           | Client    | ClientRead |    54 | 738859.53 |
13682.584 |           0.18 |    2496.147
            2 | router          | Client    | ClientRead |    53 | 719223.64 |
13570.257 |           0.18 |    2429.810
            3 | shard           | Client    | ClientRead |    54 | 461720.40 |
8550.378 |           0.18 |    1559.86
```

Visualizzazioni di Aurora PostgreSQL Limitless Database

La tabella seguente mostra le nuove visualizzazioni per Aurora PostgreSQL Limitless Database.

> **Note**
>
> Le visualizzazioni elencate in questa tabella si trovano nello schema `rds_aurora`. Quando utilizzi una visualizzazione di Limitless Database, assicurati di includere il nome completo dell'oggetto: `rds_aurora.object_name`.

Visualizzazione di Aurora PostgreSQL Limitless Database	Visualizzazione di Aurora PostgreSQL corrispondente
limitless_database	pg_database
limitless_locks	pg_locks
limitless_stat_activity	pg_stat_activity
limitless_stat_all_indexes	pg_stat_all_indexes
limitless_stat_all_tables	pg_stat_all_tables
limitless_stat_database	pg_stat_database
limitless_stat_progress_vacuum	pg_stat_progress_vacuum
limitless_stat_statements	pg_stat_statements
limitless_stat_subclusters	Nessuno
limitless_stat_statements_info	pg_stat_statements_info
limitless_statio_all_indexes	pg_statio_all_indexes
limitless_statio_all_tables	pg_statio_all_tables
limitless_tables	pg_tables
limitless_table_collocations	Nessuno

Visualizzazione di Aurora PostgreSQL Limitless Database	Visualizzazione di Aurora PostgreSQL corrispondente
limitless_table_collocation_distributions	Nessuno

Gli esempi seguenti forniscono dettagli sulle visualizzazioni di Aurora PostgreSQL Limitless Database. Per ulteriori informazioni sulle visualizzazioni di PostgreSQL, consulta Viewing statistics nella documentazione di PostgreSQL.

> **Note**
>
> Alcune visualizzazioni statistiche possono restituire risultati incoerenti se sono presenti transazioni in corso.

limitless_database

Questa visualizzazione contiene informazioni sui database disponibili nel gruppo di shard del database. Per esempio:

```
postgres_limitless=> SELECT subcluster_id, subcluster_type, oid, datname, datacl
 FROM rds_aurora.limitless_database;

 subcluster_id | subcluster_type |  oid  |      datname       |
                                 datacl

---------------+-----------------+-------+--------------------
+-------------------------------------------------------------------------------------------
 2             | router          |     4 | template0          | {=c/
rdsadmin,rdsadmin=CTc/rdsadmin}
 2             | router          |     5 | postgres           |
 2             | router          | 16384 | rdsadmin           |
 {rdsadmin=CTc/rdsadmin,rds_aurora_limitless_metadata_admin=c/
rdsadmin,rds_aurora_limitless_heat_mgmt_admin=c/rdsadmin}
 2             | router          | 16477 | postgres_limitless |
 2             | router          |     1 | template1          | {=c/
rdsadmin,rdsadmin=CTc/rdsadmin}
 6             | shard           |     4 | template0          | {=c/
rdsadmin,rdsadmin=CTc/rdsadmin}
```

I parametri di output sono i seguenti:

- `subcluster_id` (testo): l'ID del cluster secondario (nodo)
- `subcluster_type` (testo): il tipo di cluster secondario (nodo), router o shard

Le restanti colonne sono le stesse di `pg_database`.

limitless_locks

Questa visualizzazione contiene una riga per processo per nodo. Fornisce accesso alle informazioni sui blocchi dei processi attivi nel server del database.

Example di creazione di un blocco con due transazioni

In questo esempio, eseguiamo due transazioni contemporaneamente su due router.

```
# Transaction 1 (run on router 1)
BEGIN;
SET search_path = public;
SELECT * FROM customers;
INSERT INTO customers VALUES (400,'foo','bar');

# Transaction 2 (run on router 2)
BEGIN;
SET search_path = public;
ALTER TABLE customers ADD COLUMN phone VARCHAR;
```

Viene eseguita la prima transazione. Le transazioni successive devono attendere il completamento della prima transazione. Pertanto la seconda transazione viene bloccata con un blocco. Per verificarne la causa principale, eseguiamo un comando unendo `limitless_locks` e `limitless_stat_activity`.

```
# Run on router 2
SELECT distributed_session_id, state, usename, query, query_start
FROM rds_aurora.limitless_stat_activity
WHERE distributed_session_id in (
SELECT distributed_session_id
FROM rds_aurora.limitless_locks
WHERE relname = 'customers'
);

 distributed_session_id | state               | usename                | query
                                  | query_start
```

```
----------------------+--------------------+-------------------------
+-------------------------------- -------------+-----------------------------
 47BDE66E9A5E8477       | idle in transaction | limitless_metadata_admin | INSERT INTO
 customers VALUES (400,'foo','bar'); | 2023-04-13 17:44:45.152244+00
 2AD7F370202D0FA9       | active              | limitless_metadata_admin | ALTER TABLE
 customers ADD COLUMN phone VARCHAR; | 2023-04-13 17:44:55.113388+00
 47BDE66E9A5E8477       |                     | limitless_auth_admin     |
 <insufficient privilege>                       |
 2AD7F370202D0FA9       |                     | limitless_auth_admin     |
 <insufficient privilege>                       |
 47BDE66E9A5E8477       |                     | limitless_auth_admin     |
 <insufficient privilege>                       |
 2AD7F370202D0FA9       |                     | limitless_auth_admin     |
 <insufficient privilege>                       |
(6 rows)
```

Example di creazione di un blocco in modo esplicito

In questo esempio, creiamo un blocco in modo esplicito, quindi utilizziamo la visualizzazione `limitless_locks` per vedere i blocchi (alcune colonne vengono omesse).

```
BEGIN;
SET search_path = public;
LOCK TABLE customers IN ACCESS SHARE MODE;
SELECT * FROM rds_aurora.limitless_locks WHERE relname = 'customers';

 subcluster_id | subcluster_type | distributed_session_id | locktype |      datname
   | relnspname |  relname  | virtualtransaction |  pid  |      mode
---------------+-----------------+------------------------+----------
+--------------------+------------+ ----------+--------------------+-------
+-----------------
             1 | router          | 7207702F862FC937       | relation |
 postgres_limitless | public     | customers | 28/600787          | 59564 |
 AccessShareLock
             2 | router          | 7207702F862FC937       | relation |
 postgres_limitless | public     | customers | 28/600405          | 67130 |
 AccessShareLock
             3 | shard           | 7207702F862FC937       | relation |
 postgres_limitless | public     | customers | 15/473401          | 27735 |
 AccessShareLock
             4 | shard           | 7207702F862FC937       | relation |
 postgres_limitless | public     | customers | 13/473524          | 27734 |
 AccessShareLock
```

```
             5 | shard           | 7207702F862FC937       | relation |
 postgres_limitless | public     | customers | 13/472935          | 27737 |
 AccessShareLock
             6 | shard           | 7207702F862FC937       | relation |
 postgres_limitless | public     | customers | 13/473015          | 48660 |
 AccessShareLock
(6 rows)
```

limitless_stat_activity

Questa visualizzazione contiene una riga per processo per nodo. Può essere utilizzata per tenere traccia dell'integrità generale del sistema e identificare processi che richiedono tempi di esecuzione elevati. Per esempio:

```
postgres=# SELECT
    subcluster_id,
    subcluster_type,
    distributed_session_id,
    distributed_session_state,
    datname,
    distributed_query_id,
    is_sso_query
FROM
    rds_aurora.limitless_stat_activity
WHERE
    distributed_session_id in ('D2470C97E3D07E06', '5A3CD7B8E5FD13FF')
    order by  distributed_session_id;

 subcluster_id | subcluster_type | distributed_session_id | distributed_session_state |
      datname       | distributed_query_id | is_sso_query
---------------+-----------------+------------------------+---------------------------
+--------------------+----------------------+-------------
 2             | router          | 5A3CD7B8E5FD13FF       | coordinator               |
 postgres_limitless |                      | f
 3             | shard           | 5A3CD7B8E5FD13FF       | participant               |
 postgres_limitless |  6808291725541680947 |
 4             | shard           | 5A3CD7B8E5FD13FF       | participant               |
 postgres_limitless |  6808291725541680947 |
 2             | router          | D2470C97E3D07E06       | coordinator               |
 postgres_limitless |                      | t
 3             | shard           | D2470C97E3D07E06       | participant               |
 postgres_limitless |  4058400544464210222 |
(5 rows)
```

I parametri di output sono i seguenti:

- `subcluster_id` (testo): l'ID del cluster secondario a cui appartiene il processo.
- `subcluster_type` (testo): il tipo di cluster secondario a cui appartiene questo processo (`router` o `shard`).
- `distributed_session_id` (testo): l'ID della sessione distribuita a cui appartiene questo processo.
- `distributed_session_state` (testo): indica se si tratta di processo coordinatore, partecipante o autonomo/non distribuito (mostrato come `NULL`).
- `datname` (testo): il database a cui è connesso questo processo.
- `distributed_query_id` (bigint): l'ID della query principale proveniente dal nodo coordinatore. Questa colonna è `NULL` se è la query principale. Il nodo coordinatore esegue il push down dell'ID della query distribuita ai nodi partecipanti. Pertanto, per i nodi partecipanti, i valori dell'ID della query distribuita e dell'ID della query principale sono diversi.
- `is_sso_query` (testo): indica se la query è ottimizzata per un singolo shard o meno.

Le restanti colonne sono le stesse di `pg_stat_activity`.

limitless_stat_all_indexes

Questa visualizzazione contiene statistiche di utilizzo sugli indici nel gruppo di shard del database. Per esempio:

```
postgres_limitless=> SELECT schemaname, relname, indexrelname, idx_scan
  FROM rds_aurora.limitless_stat_all_indexes
  WHERE relname LIKE 'orders_ts%' ORDER BY indexrelname LIMIT 10;

 schemaname |    relname     |    indexrelname     | idx_scan
------------+----------------+---------------------+----------
 ec_sample  | orders_ts00001 | orders_ts00001_pkey |   196801
 ec_sample  | orders_ts00002 | orders_ts00002_pkey |   196703
 ec_sample  | orders_ts00003 | orders_ts00003_pkey |   196376
 ec_sample  | orders_ts00004 | orders_ts00004_pkey |   197966
 ec_sample  | orders_ts00005 | orders_ts00005_pkey |   195301
 ec_sample  | orders_ts00006 | orders_ts00006_pkey |   195673
 ec_sample  | orders_ts00007 | orders_ts00007_pkey |   194475
 ec_sample  | orders_ts00008 | orders_ts00008_pkey |   191694
 ec_sample  | orders_ts00009 | orders_ts00009_pkey |   193744
 ec_sample  | orders_ts00010 | orders_ts00010_pkey |   195421
(10 rows)
```

limitless_stat_all_tables

Questa visualizzazione contiene statistiche su tutte le tabelle nel database corrente del gruppo di shard del database. È utile per monitorare le operazioni di vacuum e le operazioni DML. Per esempio:

```
postgres_limitless=> SELECT subcluster_id, subcluster_type, relname,
 n_ins_since_vacuum, n_tup_ins, last_vacuum
  FROM rds_aurora.limitless_stat_all_tables
  WHERE relname LIKE 'orders_ts%' ORDER BY relname LIMIT 10;

 subcluster_id | subcluster_type |    relname     | n_ins_since_vacuum | n_tup_ins |
 last_vacuum
---------------+-----------------+----------------+--------------------+-----------
+-------------
 5             | shard           | orders_ts00001 |              34779 |    196083 |
 5             | shard           | orders_ts00002 |              34632 |    194721 |
 5             | shard           | orders_ts00003 |              34950 |    195965 |
 5             | shard           | orders_ts00004 |              34745 |    197283 |
 5             | shard           | orders_ts00005 |              34879 |    195754 |
 5             | shard           | orders_ts00006 |              34340 |    194605 |
 5             | shard           | orders_ts00007 |              33779 |    192203 |
 5             | shard           | orders_ts00008 |              33826 |    191293 |
 5             | shard           | orders_ts00009 |              34660 |    194117 |
 5             | shard           | orders_ts00010 |              34569 |    195560 |
(10 rows)
```

I parametri di output sono i seguenti:

- `subcluster_id` (testo): l'ID del cluster secondario a cui appartiene il processo.
- `subcluster_type` (testo): il tipo di cluster secondario a cui appartiene questo processo (`router` o `shard`).
- `relname` (nome): il nome della tabella.

Le restanti colonne sono le stesse di `pg_stat_all_tables`.

limitless_stat_database

Questa visualizzazione contiene statistiche su tutti i database nel gruppo di shard del database. Restituisce una riga per database per nodo. Per esempio:

```
postgres_limitless=> SELECT
    subcluster_id,
```

```
    subcluster_type,
    datname,
    blks_read,
    blks_hit
FROM
    rds_aurora.limitless_stat_database
WHERE
    datname='postgres_limitless';
 subcluster_id | subcluster_type |      datname       | blks_read | blks_hit
---------------+-----------------+--------------------+-----------+----------
             1 | router          | postgres_limitless |       484 | 34371314
             2 | router          | postgres_limitless |       673 | 33859317
             3 | shard           | postgres_limitless |      1299 | 17749550
             4 | shard           | postgres_limitless |      1094 | 17492849
             5 | shard           | postgres_limitless |      1036 | 17485098
             6 | shard           | postgres_limitless |      1040 | 17437257
(6 rows)
```

I parametri di output sono i seguenti:

- `subcluster_id` (testo): l'ID del cluster secondario a cui appartiene il processo.
- `subcluster_type` (testo): il tipo di cluster secondario a cui appartiene questo processo (`router` o `shard`).
- `datname` (nome): il nome del database.

Le restanti colonne sono le stesse di `pg_stat_database`.

limitless_stat_progress_vacuum

Questa visualizzazione contiene informazioni sulle operazioni di vacuum in corso. Per esempio:

```
postgres_limitless=> SELECT * FROM rds_aurora.limitless_stat_progress_vacuum;

-[ RECORD 1 ]----------+------------------
subcluster_id          | 3
subcluster_type        | shard
distributed_session_id | A56D96E2A5C9F426
pid                    | 5270
datname                | postgres
nspname                | public
relname                | customer_ts2
phase                  | vacuuming heap
heap_blks_total        | 130500
```

```
heap_blks_scanned      | 100036
heap_blks_vacuumed     | 0
index_vacuum_count     | 0
max_dead_tuples        | 11184810
num_dead_tuples        | 0

-[ RECORD 2 ]----------+------------------
subcluster_id          | 3
subcluster_type        | shard
distributed_session_id | 56DF26A89EC23AB5
pid                    | 6854
datname                | postgres
nspname                | public
relname                | sales_ts1
phase                  | vacuuming heap
heap_blks_total        | 43058
heap_blks_scanned      | 24868
heap_blks_vacuumed     | 0
index_vacuum_count     | 0
max_dead_tuples        | 8569523
num_dead_tuples        | 0
```

I parametri di output sono i seguenti:

- `subcluster_id` (testo): l'ID del cluster secondario a cui appartiene il processo.
- `subcluster_type` (testo): il tipo di cluster secondario a cui appartiene questo processo (`router` o `shard`).
- `distributed_session_id` (testo): l'identificatore della sessione che ha avviato l'operazione di vacuum.
- `datname` (nome): il database in cui viene eseguita l'operazione di vacuum.
- `nspname` (nome): il nome dello schema della tabella in cui viene eseguita l'operazione di vacuum. È `null` se la tabella non si trova nello stesso database a cui l'utente è connesso.
- `relname` (nome): il nome della tabella in cui viene eseguita l'operazione di vacuum. È `null` se la tabella non si trova nello stesso database a cui l'utente è connesso.

Le restanti colonne sono le stesse di `pg_stat_progress_vacuum`.

limitless_stat_statements

Questa visualizzazione consente di monitorare le statistiche di pianificazione ed esecuzione di tutte le istruzioni SQL eseguite su tutti i nodi.

> **Note**
>
> Devi installare l'estensione pg_stat_statements per utilizzare la visualizzazione `limitless_stat_statements`.
>
> ```
> -- CREATE EXTENSION must be run by a superuser
> CREATE EXTENSION pg_stat_statements;
>
> -- Verify that the extension is created on all nodes in the DB shard group
> SELECT distinct node_id
>     FROM rds_aurora.limitless_stat_statements
>     LIMIT 10;
> ```

L'esempio seguente illustra l'utilizzo della visualizzazione `limitless_stat_statements`.

```
postgres_limitless=> SELECT
 subcluster_id,
 subcluster_type,
 distributedqueryid,
 username,
 dbname,
 sso_calls
FROM
 rds_aurora.limitless_stat_statements;

 subcluster_id | subcluster_type |  distributedqueryid  |             username
       |       dbname       | sso_calls
---------------+-----------------+----------------------
+--------------------------------------+---------------------+-----------
 2             | router          |                      | postgres
       | postgres_limitless |         0
 2             | router          |                      | postgres
       | postgres_limitless |         0
 2             | router          |                      | postgres
       | postgres_limitless |         0
 2             | router          |                      | postgres
       | postgres_limitless |         0
 2             | router          |                      | postgres
       | postgres_limitless |         0
 2             | router          |                      | postgres
       | postgres_limitless |         1
```

```
 3             | shard           | -7975178695405682176 | postgres
       | postgres_limitless |
[...]
```

I parametri di output sono i seguenti:

- `subcluster_id` (testo): l'ID del cluster secondario a cui appartiene il processo.
- `subcluster_type` (testo): il tipo di cluster secondario a cui appartiene questo processo (`router` o `shard`).
- `distributedqueryid` (bigint): l'ID della query principale proveniente dal nodo coordinatore. Questa colonna è `NULL` se è la query principale. Il nodo coordinatore esegue il push down dell'ID della query distribuita ai nodi partecipanti. Pertanto, per i nodi partecipanti, i valori dell'ID della query distribuita e dell'ID della query principale sono diversi.
- `username` (nome): l'utente che ha eseguito la query sull'istruzione.
- `dbname` (nome): il database su cui è stata eseguita la query.
- `sso_calls` (nome): il numero di volte in cui la dichiarazione è stata ottimizzata per singolo shard.

Le restanti colonne sono le stesse di pg_stat_statements.

limitless_stat_statements_info

Questa visualizzazione contiene le statistiche relative alla visualizzazione `limitless_stat_statements`. Ogni riga contiene i dati per la visualizzazione pg_stat_statements_info di ciascun nodo. La colonna `subcluster_id` identifica ogni nodo.

```
postgres_limitless=> SELECT * FROM rds_aurora.limitless_stat_statements_info;

 subcluster_id | subcluster_type | dealloc |          stats_reset
---------------+-----------------+---------+-------------------------------
             1 | router          |       0 | 2023-06-30 21:22:09.524781+00
             2 | router          |       0 | 2023-06-30 21:21:40.834111+00
             3 | shard           |       0 | 2023-06-30 21:22:10.709942+00
             4 | shard           |       0 | 2023-06-30 21:22:10.740179+00
             5 | shard           |       0 | 2023-06-30 21:22:10.774282+00
             6 | shard           |       0 | 2023-06-30 21:22:10.808267+00
(6 rows)
```

Il parametro di output è il seguente:

- `subcluster_id` (testo): l'ID del cluster secondario a cui appartiene il processo.

Le restanti colonne sono le stesse di pg_stat_statements_info.

limitless_stat_subclusters

Questa visualizzazione contiene statistiche di rete tra router e altri nodi. Contiene una riga per coppia di router e altro nodo, ad esempio:

```
postgres_limitless=> SELECT * FROM rds_aurora.limitless_stat_subclusters;

 orig_subcluster | orig_instance_az | dest_subcluster | dest_instance_az |
 latency_us |       latest_collection       | failed_requests | received_bytes |
 sent_bytes | same_az_requests | cross_az_requests |     stat_reset_timestamp
-----------------+------------------+-----------------+------------------
+------------+-------------------------------+-----------------+---------------
+------------+------------------+-------------------+------------------------------
 3               | us-west-2b       | 2               | us-west-2a       |
 847 | 2024-10-07 17:25:39.518617+00 |               0 |       35668633 |   92090171
 |                0 |           302787 | 2024-10-05 12:39:55.239675+00
 3               | us-west-2b       | 4               | us-west-2b       |
 419 | 2024-10-07 17:25:39.546376+00 |               0 |      101190464 |  248795719
 |           883478 |                 0 | 2024-10-05 12:39:55.231218+00
 3               | us-west-2b       | 5               | us-west-2c       |
    1396 | 2024-10-07 17:25:39.52122+00  |               0 |       72864849 |
 172086292 |                0 |           557726 | 2024-10-05 12:39:55.196412+00
 3               | us-west-2b       | 6               | us-west-2c       |
 729 | 2024-10-07 17:25:39.54828+00  |               0 |       35668584 |   92090171
 |                0 |           302787 | 2024-10-05 12:39:55.247334+00
 3               | us-west-2b       | 7               | us-west-2a       |
    1702 | 2024-10-07 17:25:39.545307+00 |               0 |       71699576 |
 171634844 |                0 |           556278 | 2024-10-05 12:39:52.715168+00
 2               | us-west-2a       | 3               | us-west-2b       |
 868 | 2024-10-07 17:25:40.293927+00 |               0 |       35659611 |   92011872
 |                0 |           302817 | 2024-10-05 12:39:54.420758+00
 2               | us-west-2a       | 4               | us-west-2b       |
 786 | 2024-10-07 17:25:40.296863+00 |               0 |      102437253 |  251838024
 |                0 |           895060 | 2024-10-05 12:39:54.404081+00
 2               | us-west-2a       | 5               | us-west-2c       |
    1232 | 2024-10-07 17:25:40.292021+00 |               0 |       71990027 |
 168828110 |                0 |           545453 | 2024-10-05 12:39:36.769549+00
```

I parametri di output sono i seguenti:

- `orig_subcluster` (testo): l'ID del router da cui originano le comunicazioni.
- `orig_subcluster_az` (testo): la zona di disponibilità (AZ) del router di origine.

- `dest_subcluster` (testo): l'ID del nodo di destinazione.
- `dest_subcluster_az` (testo): l'ultima AZ raccolta del nodo di destinazione.
- `latency_us` (bigint): l'ultima latenza di rete raccolta tra i nodi, in microsecondi. Il valore è `0` se il nodo non è raggiungibile.
- `latest_collection` (timestamp): il timestamp dell'ultima raccolta di AZ e latenza per il nodo di destinazione.
- `failed_requests` (bigint): il conteggio cumulativo delle richieste interne non riuscite.
- `received_bytes` (bigint): il numero cumulativo stimato di byte ricevuti da questo nodo.
- `sent_bytes` (bigint): il numero cumulativo stimato di byte inviati a questo nodo.
- `same_az_requests` (bigint): il numero cumulativo di richieste database interne verso questo nodo quando è nella stessa AZ del router di origine.
- `cross_az_requests` (bigint): il numero cumulativo di richieste database interne verso questo nodo quando è in una AZ diversa dal router di origine.
- `stat_reset_timestamp` (timestamp): il timestamp dell'ultimo ripristino delle statistiche cumulative per questa visualizzazione.

limitless_statio_all_indexes

Questa visualizzazione contiene statistiche di input/output (I/O) per tutti gli indici nel gruppo di shard del database. Per esempio:

```
postgres_limitless=> SELECT * FROM rds_aurora.limitless_statio_all_indexes WHERE
 relname like'customers_ts%';

 subcluster_id | subcluster_type | schemaname |      relname      |
 indexrelname             | idx_blks_read | idx_blks_hit
---------------+-----------------+------------+-------------------
+-------------------------------------+ --------------+--------------
             3 | shard           | public     | customers_ts00002 |
 customers_ts00002_customer_name_idx |             1 |           0
             3 | shard           | public     | customers_ts00001 |
 customers_ts00001_customer_name_idx |             1 |           0
             4 | shard           | public     | customers_ts00003 |
 customers_ts00003_customer_name_idx |             1 |           0
             4 | shard           | public     | customers_ts00004 |
 customers_ts00004_customer_name_idx |             1 |           0
             5 | shard           | public     | customers_ts00005 |
 customers_ts00005_customer_name_idx |             1 |           0
```

```
             5 | shard           | public     | customers_ts00006 |
 customers_ts00006_customer_name_idx |             1 |           0
             6 | shard           | public     | customers_ts00007 |
 customers_ts00007_customer_name_idx |             1 |           0
             6 | shard           | public     | customers_ts00008 |
 customers_ts00008_customer_name_idx |             1 |           0
(8 rows)
```

limitless_statio_all_tables

Questa visualizzazione contiene statistiche di input/output (I/O) per tutte le tabelle nel gruppo di shard del database. Per esempio:

```
postgres_limitless=> SELECT
    subcluster_id,
    subcluster_type,
    schemaname,
    relname,
    heap_blks_read,
    heap_blks_hit
FROM
    rds_aurora.limitless_statio_all_tables
WHERE
    relname LIKE 'customers_ts%';

 subcluster_id | subcluster_type | schemaname |      relname      | heap_blks_read |
 heap_blks_hit
---------------+-----------------+------------+-------------------+----------------
+---------------
             3 | shard           | public     | customers_ts00002 |            305 |
      57780
             3 | shard           | public     | customers_ts00001 |            300 |
      56972
             4 | shard           | public     | customers_ts00004 |            302 |
      57291
             4 | shard           | public     | customers_ts00003 |            302 |
      57178
             5 | shard           | public     | customers_ts00006 |            300 |
      56932
             5 | shard           | public     | customers_ts00005 |            302 |
      57386
             6 | shard           | public     | customers_ts00008 |            300 |
      56881
```

```
             6 | shard           | public     | customers_ts00007 |            304 |
      57635
(8 rows)
```

limitless_tables

Questa visualizzazione contiene informazioni sulle tabelle in Aurora PostgreSQL Limitless Database.

```
postgres_limitless=> SELECT * FROM rds_aurora.limitless_tables;

 table_gid | local_oid | schema_name | table_name  | table_status | table_type  |
 distribution_key
-----------+-----------+-------------+-------------+--------------+-------------
+------------------
         5 |     18635 | public      | placeholder | active       | placeholder |
         6 |     18641 | public      | ref         | active       | reference   |
         7 |     18797 | public      | orders      | active       | sharded     | HASH
 (order_id)
         2 |     18579 | public      | customer    | active       | sharded     | HASH
 (cust_id)
(4 rows)
```

limitless_table_collocations

Questa visualizzazione contiene informazioni sulle tabelle sottoposte a sharding co-localizzate.

Nell'esempio seguente, le tabelle `orders` e `customers` sono collocate congiuntamente e le tabelle `users` e `followers` sono co-localizzate. Le tabelle co-localizzate hanno lo stesso `collocation_id`.

```
postgres_limitless=> SELECT * FROM rds_aurora.limitless_table_collocations ORDER BY
 collocation_id;

 collocation_id | schema_name | table_name
----------------+-------------+------------
              2 | public      | orders
              2 | public      | customers
              5 | public      | users
              5 | public      | followers
(4 rows)
```

limitless_table_collocation_distributions

Questa visualizzazione mostra la distribuzione delle chiavi per ogni co-localizzazione.

```
postgres_limitless=> SELECT * FROM
 rds_aurora.limitless_table_collocation_distributions ORDER BY collocation_id,
 lower_bound;

 collocation_id | subcluster_id |     lower_bound      |     upper_bound
----------------+---------------+----------------------+----------------------
              2 |             6 | -9223372036854775808 | -4611686018427387904
              2 |             5 | -4611686018427387904 |                    0
              2 |             4 |                    0 |  4611686018427387904
              2 |             3 |  4611686018427387904 |  9223372036854775807
              5 |             6 | -9223372036854775808 | -4611686018427387904
              5 |             5 | -4611686018427387904 |                    0
              5 |             4 |                    0 |  4611686018427387904
              5 |             3 |  4611686018427387904 |  9223372036854775807
(8 rows)
```

Eventi di attesa per Aurora PostgreSQL Limitless Database

In Aurora PostgreSQL, un evento di attesa indica una risorsa per la quale una sessione è in attesa, ad esempio input/output (I/O) e blocchi. Gli eventi di attesa sono utili per scoprire perché le sessioni restano in attesa di risorse e per individuare i colli di bottiglia. Per ulteriori informazioni, consulta Eventi di attesa Aurora PostgreSQL.

Aurora PostgreSQL Limitless Database include eventi di attesa specifici relativi a router e shard. Molti di questi riguardano router in attesa che gli shard completino determinate attività. Gli eventi di attesa degli shard contengono dettagli sulle attività in corso.

Argomenti

- Esecuzione di query sugli eventi di attesa
- Eventi di attesa Limitless Database

Esecuzione di query sugli eventi di attesa

Puoi utilizzare la visualizzazione limitless_stat_activity per eseguire query sugli eventi di attesa, come mostrato nell'esempio seguente.

```
SELECT wait_event FROM rds_aurora.limitless_stat_activity WHERE
 wait_event_type='AuroraLimitless';

      wait_event
----------------------
 RemoteStatementSetup
 RemoteStatementSetup
(2 rows)
```

Puoi inoltre utilizzare la funzione `aurora_stat_system_waits` per elencare il numero di attese e il tempo totale trascorso per ogni evento di attesa, come mostrato nell'esempio seguente.

```
postgres_limitless=> SELECT type_name,event_name,waits,wait_time
    FROM aurora_stat_system_waits()
    NATURAL JOIN aurora_stat_wait_event()
    NATURAL JOIN aurora_stat_wait_type()
    WHERE type_name='AuroraLimitless'
    ORDER BY wait_time DESC;
```

```
    type_name    |       event_name          |  waits  |  wait_time
-----------------+---------------------------+---------+-------------
 AuroraLimitless | RemoteStatementSetup      |    7518 | 75236507897
 AuroraLimitless | RemoteStatementExecution  |      40 |      132986
 AuroraLimitless | Connect                   |       5 |        1453
(3 rows)
```

Eventi di attesa Limitless Database

I seguenti eventi di attesa si applicano ad Aurora PostgreSQL Limitless Database. È possibile monitorare questi eventi di attesa per identificare i colli di bottiglia nell'elaborazione del database Aurora PostgreSQL Limitless.

Argomenti

- IO:TwophaseFilePoolWrite, evento di attesa
- IO:TwophaseFilePoolRead, evento di attesa
- Evento di attesa AuroraLimitless:Connect
- Evento di attesa AuroraLimitless:AsyncConnect
- Evento di attesa AuroraLimitless:RemoteStatementSetup
- Evento di attesa AuroraLimitless:RemoteDDLExecution
- Evento di attesa AuroraLimitless:RemoteStatementExecution
- AuroraLimitless:FetchRemoteResults, evento di attesa
- Evento di attesa AuroraLimitless:AsyncGetInitialResponse
- Evento di attesa AuroraLimitless:AsyncGetNextResponse
- Evento di attesa AuroraLimitless:AbortedCommandCleanup
- Evento di attesa AuroraLimitless:DistributedCommitPrepare
- Evento di attesa AuroraLimitless:DistributedCommit
- Evento di attesa AuroraLimitless:DistributedCommitPrepareThrottle
- Evento di attesa AuroraLimitless:PreparedTransactionResolution
- Evento di attesa AuroraLimitless:SendPreparedTransactionOutcome
- Evento di attesa AuroraLimitless:CommitClockBarrier
- Evento di attesa AuroraLimitless:SnapshotClockBarrier
- Evento di attesa AuroraLimitless:ReaderSnapshotClockBarrier
- Evento di attesa AuroraLimitless:GatherDistributedDeadlockGraph

- Evento di attesa AuroraLimitless:DistributedDeadlockDetection
- Evento di attesa AuroraLimitless:DistributedDeadlockAbort
- Evento di attesa AuroraLimitless:GatherRemoteStats
- Evento di attesa AuroraLimitless:GlobalSequenceRefresh
- Evento di attesa AuroraLimitless:GlobalVacuumTimeExchange
- Evento di attesa AuroraLimitless:DistributedTransactionMonitorGather
- Evento di attesa AuroraLimitlessActivity:AdminTaskSchedulerMain
- Evento di attesa AuroraLimitlessActivity:AdminTaskExecutorMain
- Evento di attesa AuroraLimitlessActivity:AdminTaskMonitorMain
- Evento di attesa AuroraLimitlessActivity:DatabaseCleanupMonitorMain
- Evento di attesa AuroraLimitlessActivity:TopologyCleanupMonitorMain
- Evento di attesa AuroraLimitlessActivity:ToplogyChangeMonitorMain
- Evento di attesa AuroraLimitlessActivity:DistributedTransactionMonitorMain
- Evento di attesa AuroraLimitlessActivity:GlobalVacuumMonitorMain

IO:TwophaseFilePoolWrite, evento di attesa

In attesa della scrittura di un file di stato in due fasi all'interno del pool di file di stato in due fasi. Questo è un evento specifico di Aurora.

Cause

I processi che eseguono un `PREPARED TRANSACTION` comando, inclusi i partecipanti a una transazione distribuita con Limitless Database, devono mantenere lo stato della transazione in un file a due fasi. Aurora utilizza un pool di file per migliorare le prestazioni di questa operazione.

Azione

Si tratta di un'operazione di I/O di scrittura sincrona e pertanto un'elevata latenza in questo evento ha cause simili `IO:XactSync` e può essere esaminata allo stesso modo. Se si utilizza Limitless Database, potrebbe essere necessario ridurre il numero di transazioni distribuite eseguite.

IO:TwophaseFilePoolRead, evento di attesa

In attesa di una lettura di un file di stato in due fasi all'interno del pool di file di stato in due fasi.

Cause

I processi che eseguono un `COMMIT PREPARED` comando su una transazione preparata in precedenza, inclusi i partecipanti a una transazione distribuita da Limitless Database, potrebbero dover leggere lo stato della transazione precedentemente persistente da un file a due fasi. Aurora utilizza un pool di file per migliorare le prestazioni di questa operazione.

Azione

Questa è un'operazione di I/O di lettura. Pertanto, un'elevata latenza in questo evento ha cause simili `IO:DataFileRead` e può essere studiata allo stesso modo. Se si utilizza Limitless Database, potrebbe essere necessario ridurre il numero di transazioni distribuite eseguite.

Evento di attesa AuroraLimitless:Connect

Il processo è in attesa che venga stabilita una connessione a un altro nodo del cluster.

Cause

Vengono stabilite connessioni tra processi e nodi remoti per eseguire query, transazioni distribuite ed eseguire DDL.

Azione

Riduci il numero di connessioni simultanee al cluster o ottimizza l'uso delle query cross-shard.

Evento di attesa AuroraLimitless:AsyncConnect

Questo evento è simile a`Connect`, ma rappresenta un processo in attesa che vengano stabilite connessioni parallele a un insieme di nodi.

Cause

La creazione di connessioni parallele viene eseguita più comunemente durante l'esecuzione di istruzioni DDL.

Azione

Riduci il numero di istruzioni DDL o combina più DDL nella stessa sessione per migliorare il riutilizzo della connessione.

Evento di attesa AuroraLimitless:RemoteStatementSetup

Il processo è in attesa della configurazione dell'esecuzione remota delle query, ad esempio l'apertura, la chiusura del cursore o la creazione di istruzioni preparate.

Cause

Questo evento di attesa aumenta con il numero di scansioni su tabelle frammentate in cui l'istruzione non può essere ottimizzata per un singolo shard.

Azione

Ottimizza le query per ridurre il numero di operazioni di scansione o aumentare l'idoneità all'ottimizzazione a shard singolo.

Evento di attesa AuroraLimitless:RemoteDDLExecution

Il processo è in attesa del completamento di un comando DDL remoto.

Cause

Quando si esegue un comando DDL su un gruppo di shard DB, è necessario distribuirlo ad altri router e nodi shard prima di confermare l'operazione. Alcune operazioni DDL possono essere eseguite a lungo, poiché i dati devono essere adattati alle modifiche dello schema.

Azione

Identifica i comandi DDL a esecuzione prolungata in modo da poterli ottimizzare.

Evento di attesa AuroraLimitless:RemoteStatementExecution

Un processo è in attesa del completamento di un comando remoto.

Cause

Un comando SQL è in esecuzione su un nodo remoto. Questo evento verrà visualizzato frequentemente per le comunicazioni interne, ad esempio per i `auto_analyze` controlli del battito cardiaco.

Azione

Identifica i comandi a esecuzione prolungata utilizzando la vista limitless_stat_statements. In molti casi si tratta di un evento previsto, specialmente per i lavoratori in background o i processi interni, e non è necessaria alcuna azione.

AuroraLimitless:FetchRemoteResults, evento di attesa

Un processo è in attesa di recuperare le righe da un nodo remoto.

Cause

Questo evento di attesa può aumentare quando si recupera un numero elevato di righe da una tabella remota, ad esempio una tabella frammentata o di riferimento.

Azione

Identifica le `SELECT` query non ottimizzate utilizzando la vista. `limitless_stat_statements` Ottimizza le query per recuperare solo i dati necessari. Non è possibile specificare anche il parametro `rds_aurora.limitless_maximum_adaptive_fetch_size`.

Evento di attesa AuroraLimitless:AsyncGetInitialResponse

Il processo è in attesa di una risposta iniziale quando viene utilizzata la modalità pipeline nell'esecuzione delle query.

Cause

Ciò si verifica in genere durante l'esecuzione di router to shard per le query con posizionamento di dati a frammento singolo ed è una parte prevista della normale esecuzione.

Azione

Non è richiesta alcuna operazione aggiuntiva.

Evento di attesa AuroraLimitless:AsyncGetNextResponse

Il processo è in attesa di risposte aggiuntive quando si utilizza la modalità pipeline nell'esecuzione delle query.

Cause

Ciò si verifica in genere durante l'esecuzione di router to shard per le query con posizionamento di dati a frammento singolo ed è una parte prevista della normale esecuzione.

Azione

Non è richiesta alcuna operazione aggiuntiva.

Evento di attesa AuroraLimitless:AbortedCommandCleanup

Il processo è in attesa del risultato di una query di pulizia remota. Le query di pulizia vengono inviate ai nodi shard per riportarli allo stato appropriato al termine di una transazione distribuita.

Cause

La pulizia delle transazioni viene eseguita quando una transazione viene interrotta perché è stato rilevato un errore o perché un utente ha chiamato un comando ABORT esplicito o ha annullato la query in esecuzione.

Azione

Indagate sulla causa dell'annullamento della transazione.

Evento di attesa AuroraLimitless:DistributedCommitPrepare

Il processo sta eseguendo una transazione distribuita ed è in attesa che tutti i partecipanti confermino il comando prepare.

Cause

Le transazioni che modificano più nodi devono eseguire un commit distribuito. Una lunga attesa `DistributedCommitPrepare` potrebbe essere causata da lunghe attese nell'`IO:TwophaseFilePoolWrite`evento sui nodi partecipanti.

Azione

Riduci il numero di transazioni che modificano i dati su più nodi. Analizza `IO:TwophaseFilePoolWrite` gli eventi in altri nodi del cluster.

Evento di attesa AuroraLimitless:DistributedCommit

Il processo sta commettendo una transazione distribuita ed è in attesa che il partecipante principale confermi il comando di commit.

Cause

Le transazioni che modificano più nodi devono eseguire un commit distribuito. Una lunga attesa `DistributedCommit` potrebbe essere causata da lunghe attese nell'`IO:XactSync`evento da parte del partecipante principale.

Azione

Riduci il numero di transazioni che modificano i dati su più nodi. Analizza `IO:XactSync` gli eventi in altri nodi del cluster.

Evento di attesa AuroraLimitless:DistributedCommitPrepareThrottle

Il processo sta tentando di preparare una transazione distribuita e viene rallentato a causa delle transazioni preparate esistenti.

Cause

Le transazioni che modificano più nodi devono eseguire un commit distribuito. I partecipanti a queste transazioni devono eseguire un'operazione di preparazione come parte del protocollo di commit. Aurora limita il numero di preparazioni simultanee e, se questo limite viene superato, il processo attenderà l'evento. `DistributedCommitPrepareThrottle`

Azione

Riduci il numero di transazioni che modificano i dati su più nodi. Esamina `IO:TwophaseFilePoolWrite` gli eventi, poiché il prolungamento del tempo trascorso in tali eventi potrebbe causare l'accumulo di transazioni preparate esistenti, con conseguente limitazione (della larghezza di banda della rete) per i nuovi tentativi di preparazione.

Evento di attesa AuroraLimitless:PreparedTransactionResolution

Il processo ha rilevato una tupla modificata da una transazione distribuita che si trova nello stato preparato. Il processo deve determinare se la transazione distribuita diventerà visibile nella relativa istantanea.

Cause

Le transazioni che modificano più nodi devono eseguire un commit distribuito che include una fase di preparazione. Un numero elevato di transazioni distribuite o una maggiore latenza sui commit distribuiti possono causare l'evento di `PreparedTransactionResolution` attesa in altri processi.

Azione

Riduci il numero di transazioni che modificano i dati su più nodi. Analizzate gli eventi relativi al commit distribuito, in quanto l'aumento del tempo trascorso in tali eventi potrebbe aumentare la latenza nel

percorso di commit delle transazioni distribuite. Potresti anche voler esaminare i carichi di rete e della CPU.

Evento di attesa AuroraLimitless:SendPreparedTransactionOutcome

Il processo viene eseguito su un nodo che coordina una transazione distribuita e un altro processo ha richiesto informazioni sullo stato di tale transazione oppure il processo ha eseguito una transazione distribuita e invia il risultato ai partecipanti.

Cause

I processi che riscontrano l'evento di `PreparedTransactionResolution` attesa interrogheranno il coordinatore della transazione. La risposta sul coordinatore della transazione incontrerà SendPreparedTransactionOutcome.

Azione

Riduci il numero di transazioni che modificano i dati su più nodi. Analizza gli eventi `IO:TwophaseFilePoolWrite` e `IO:TwophaseFilePoolRead` gli eventi relativi al commit distribuito, poiché tali eventi potrebbero aumentare la latenza nel percorso di commit delle transazioni distribuite. Potresti anche voler esaminare i carichi di rete e della CPU.

Evento di attesa AuroraLimitless:CommitClockBarrier

Il processo sta eseguendo una transazione e deve attendere per garantire che il tempo di commit assegnato sia garantito come passato per tutti i nodi del cluster.

Cause

La saturazione della CPU o della rete potrebbe causare un aumento della deriva del clock, con conseguente tempo impiegato in questo evento di attesa.

Azione

Analizza la saturazione della CPU o della rete nel cluster.

Evento di attesa AuroraLimitless:SnapshotClockBarrier

Il processo ha ricevuto un'istantanea dell'ora da un altro nodo con un orologio del futuro ed è in attesa che il proprio orologio raggiunga quell'ora.

Cause

Ciò si verifica in genere dopo che il processo ha ricevuto i risultati da una funzione che è stata trasferita su uno shard e si è verificata una variazione dell'orologio tra i nodi. La saturazione della CPU o della rete potrebbe causare un aumento della deriva del clock, con conseguente tempo impiegato in questo evento di attesa.

Azione

Analizza la saturazione della CPU o della rete nel cluster.

Evento di attesa AuroraLimitless:ReaderSnapshotClockBarrier

Questo evento si verifica sui nodi di lettura. Il processo è in attesa che il nodo di lettura riproduca il flusso di scrittura in modo che tutte le scritture avvenute prima dell'ora dell'istantanea del processo siano state applicate.

Cause

Un aumento del ritardo della replica di Aurora può causare un aumento del tempo di attesa in questo evento.

Azione

Indaga sul ritardo della replica di Aurora.

Evento di attesa AuroraLimitless:GatherDistributedDeadlockGraph

Il processo sta comunicando con altri nodi per raccogliere grafici di blocco come parte del rilevamento distribuito dei deadlock.

Cause

Quando un processo è in attesa di un blocco, eseguirà un controllo di deadlock distribuito dopo aver atteso più di. `rds_aurora.limitless_distributed_deadlock_timeout`

Azione

Analizzate le cause della contrapposizione dei blocchi nella vostra applicazione e valutate la possibilità di ottimizzarla. `rds_aurora.limitless_distributed_deadlock_timeout`

Evento di attesa AuroraLimitless:DistributedDeadlockDetection

Il processo sta comunicando con altri nodi per rilevare un deadlock distribuito.

Cause

Quando un processo è in attesa di un blocco, eseguirà un controllo di deadlock distribuito dopo aver atteso più di. `rds_aurora.limitless_distributed_deadlock_timeout`

Azione

Analizzate le cause della contrapposizione dei blocchi nella vostra applicazione e valutate la possibilità di ottimizzarla. `rds_aurora.limitless_distributed_deadlock_timeout`

Evento di attesa AuroraLimitless:DistributedDeadlockAbort

Il processo sta comunicando con un altro nodo per interrompere una sessione scelta come vittima in un deadlock distribuito.

Cause

I modelli di applicazione danno luogo a situazioni di stallo distribuite.

Azione

Analizzate i modelli di applicazione che determinano situazioni di stallo distribuite.

Evento di attesa AuroraLimitless:GatherRemoteStats

Il processo sta raccogliendo statistiche da altri nodi del cluster.

Cause

Il monitoraggio delle nostre attività, le query e le visualizzazioni, ad esempio`limitless_stat_activity`, recupereranno le statistiche da altri nodi.

Azione

Non è richiesta alcuna operazione aggiuntiva.

Evento di attesa AuroraLimitless:GlobalSequenceRefresh

Il processo sta generando un nuovo valore di sequenza e deve richiedere un nuovo blocco dalla sequenza globale.

Cause

In questo caso, un'elevata velocità di generazione del valore di sequenza può causare uno stallo se `rds_aurora.limitless_sequence_chunk_size` è insufficiente.

Azione

Si tratta di un evento normale. Se in questo caso il tempo a disposizione è eccessivo, valuta la possibilità di effettuare la sintonizzazione. `rds_aurora.limitless_sequence_chunk_size` Consulta la documentazione sulle sequenze in Limitless Database.

Evento di attesa AuroraLimitless:GlobalVacuumTimeExchange

Il processo consiste nello scambio di dati istantanei per supportare il vuoto.

Cause

I nodi di Limitless Database scambiano i dati temporali più vecchi relativi alle istantanee attive con altri nodi per calcolare l'ora limite corretta per l'esecuzione sotto vuoto.

Azione

Non è richiesta alcuna operazione aggiuntiva.

Evento di attesa AuroraLimitless:DistributedTransactionMonitorGather

Il processo sta raccogliendo i metadati delle transazioni da altri nodi per supportare la pulizia distribuita delle transazioni.

Cause

I nodi di Limitless Database scambiano i metadati delle transazioni con altri nodi per determinare quando è possibile eliminare lo stato della transazione distribuita.

Azione

Non è richiesta alcuna operazione aggiuntiva.

Evento di attesa AuroraLimitlessActivity:AdminTaskSchedulerMain

Attesa nel ciclo principale del processo di pianificazione delle attività.

Evento di attesa AuroraLimitlessActivity:AdminTaskExecutorMain

Attesa nel ciclo principale del processo di esecuzione delle attività.

Evento di attesa AuroraLimitlessActivity:AdminTaskMonitorMain

Attesa nel ciclo principale del processo di monitoraggio delle attività.

Evento di attesa AuroraLimitlessActivity:DatabaseCleanupMonitorMain

Attesa nel ciclo principale del processo di monitoraggio della pulizia del database.

Evento di attesa AuroraLimitlessActivity:TopologyCleanupMonitorMain

Attesa nel ciclo principale del processo di monitoraggio della pulizia della topologia.

Evento di attesa AuroraLimitlessActivity:ToplogyChangeMonitorMain

Attesa nel ciclo principale del processo di monitoraggio delle modifiche alla topologia.

Evento di attesa AuroraLimitlessActivity:DistributedTransactionMonitorMain

Attesa nel ciclo principale del processo di monitoraggio distribuito delle transazioni.

Evento di attesa AuroraLimitlessActivity:GlobalVacuumMonitorMain

Attesa nel ciclo principale del processo globale di monitoraggio del vuoto.

Backup e ripristino di Aurora PostgreSQL Limitless Database

Puoi eseguire il backup e il ripristino di un cluster di database che utilizza Aurora PostgreSQL Limitless Database.

Indice

- Backup di un cluster di database che utilizza Aurora PostgreSQL Limitless Database
  - Creazione di uno snapshot del cluster database
- Ripristino di un cluster di database che utilizza Aurora PostgreSQL Limitless Database
  - Ripristino di un cluster di database da uno snapshot
  - Ripristino di un cluster di database utilizzando il recupero point-in-time
- Le utilità di backup e ripristino di PostgreSQL non sono supportate

Backup di un cluster di database che utilizza Aurora PostgreSQL Limitless Database

Il backup di un cluster di database con Aurora PostgreSQL Limitless Database presenta analogie e differenze funzionali rispetto al backup di un cluster di database Aurora standard.

- Quando acquisisci uno snapshot di un cluster di database Aurora manuale che utilizza Limitless Database, lo snapshot include i dati del gruppo di shard del database.
- I backup continui includono i dati del gruppo di shard del database.
- Gli snapshot giornalieri automatizzati includono i dati del gruppo di shard del database.
- È supportata la copia degli snapshot del cluster di database. Per ulteriori informazioni, consulta Copia di uno snapshot del cluster di database.
- È supportata la condivisione degli snapshot del cluster di database. Per ulteriori informazioni, consulta Condivisione di uno snapshot cluster database.
- Non puoi utilizzare le utilità `pg_dump` o `pg_dumpall` per eseguire backup dei database nel gruppo di shard del database.
- È supportata l'acquisizione di snapshot finali durante l'eliminazione di cluster di database in Aurora PostgreSQL Limitless Database.
- Non è supportata la conservazione dei backup automatizzati durante l'eliminazione di cluster di database in Aurora PostgreSQL Limitless Database.

Creazione di uno snapshot del cluster database

Puoi creare uno snapshot di un cluster di database Aurora PostgreSQL Limitless Database nello stesso modo previsto per un cluster di database Aurora standard, come mostrato nell'esempio AWS CLI seguente:

```
aws rds create-db-cluster-snapshot \
    --db-cluster-identifier my-db-cluster \
    --db-cluster-snapshot-identifier my-db-cluster-snapshot
```

Per ulteriori informazioni sui backup dei cluster di database, consulta Panoramica di backup e ripristino di un cluster di database Aurora.

Ripristino di un cluster di database che utilizza Aurora PostgreSQL Limitless Database

Il ripristino di un cluster di database con Aurora PostgreSQL Limitless Database presenta analogie e differenze funzionali rispetto al ripristino di un cluster di database Aurora standard.

- Puoi ripristinare un cluster di database Limitless Database solo da un cluster di origine che utilizza una versione del motore di database compatibile con Limitless Database, come `16.4-limitless`.
- Quando ripristini un cluster di database da uno snapshot manuale di un cluster di database che utilizza Limitless Database, viene ripristinata l'intera archiviazione del cluster. Ciò include l'archiviazione del gruppo di shard del database.

  Devi creare un gruppo di shard del database per accedere all'archiviazione di Limitless Database.
- Puoi ripristinare un cluster di database utilizzando il recupero point-in-time (PITR) in qualsiasi momento all'interno del periodo di conservazione. Il cluster di database ripristinato include l'archiviazione del gruppo di shard del database.

  Devi creare un gruppo di shard del database per accedere all'archiviazione di Limitless Database.
- Il PITR non è supportato per i cluster di database Aurora PostgreSQL Limitless Database eliminati.
- Quando ripristini un cluster di database da uno snapshot giornaliero automatizzato, viene ripristinata anche l'archiviazione del gruppo di shard del database.
- Quando ripristini un cluster di database Aurora PostgreSQL Limitless Database, devi abilitare il monitoraggio avanzato e Approfondimenti sulle prestazioni. Assicurati di includere l'ID della chiave KMS di Approfondimenti sulle prestazioni.

Dopo aver ripristinato un cluster di database Aurora PostgreSQL Limitless Database, verificane il corretto funzionamento eseguendo le query.

Ripristino di un cluster di database da uno snapshot

Gli esempi AWS CLI seguenti mostrano come ripristinare un cluster di database Aurora PostgreSQL Limitless Database da uno snapshot del cluster di database.

Devi utilizzare la versione del motore di database `16.4-limitless`.

Per ripristinare un cluster di database Limitless Database da uno snapshot del cluster di database

1. Ripristina il cluster di database:

```
aws rds restore-db-cluster-from-snapshot \
    --db-cluster-identifier my-new-db-cluster \
    --snapshot-identifier my-db-cluster-snapshot \
    --engine aurora-postgresql \
    --engine-version 16.4-limitless \
    --enable-performance-insights \
    --performance-insights-retention-period 31 \
    --performance-insights-kms-key-id arn:aws:kms:us-
east-1:123456789012:key/1234abcd-12ab-34cd-56ef-1234567890ab \
    --monitoring-interval 5 \
    --monitoring-role-arn arn:aws:iam::123456789012:role/EMrole
```

2. Crea il gruppo di shard del database:

```
aws rds create-db-shard-group \
    --db-cluster-identifier my-new-db-cluster \
    --db-shard-group-identifier my-new-DB-shard-group \
    --max-acu 1000
```

   Per ulteriori informazioni, consulta Aggiunta di un gruppo di shard di database a un cluster di database Aurora PostgreSQL Limitless Database esistente.

Per ulteriori informazioni sul ripristino di cluster di database Aurora da snapshot del cluster di database, consulta Ripristino da uno snapshot cluster database.

Ripristino di un cluster di database utilizzando il recupero point-in-time

Gli esempi AWS CLI seguenti mostrano come ripristinare un cluster di database Aurora PostgreSQL Limitless Database utilizzando il recupero point-in-time (PITR).

Per ripristinare un cluster di database Limitless Database utilizzando PITR

1. Ripristina il cluster di database:

```
aws rds restore-db-cluster-to-point-in-time \
    --source-db-cluster-identifier my-db-cluster \
    --db-cluster-identifier my-new-db-cluster \
    --use-latest-restorable-time \
    --enable-performance-insights \
    --performance-insights-retention-period 31 \
    --performance-insights-kms-key-id arn:aws:kms:us-
east-1:123456789012:key/1234abcd-12ab-34cd-56ef-1234567890ab \
    --monitoring-interval 5 \
    --monitoring-role-arn arn:aws:iam::123456789012:role/EMrole
```

2. Crea il gruppo di shard del database:

```
aws rds create-db-shard-group \
    --db-cluster-identifier my-new-db-cluster \
    --db-shard-group-identifier my-new-DB-shard-group \
    --max-acu 1000
```

   Per ulteriori informazioni, consulta Aggiunta di un gruppo di shard di database a un cluster di database Aurora PostgreSQL Limitless Database esistente.

Per ulteriori informazioni su PITR, consulta Ripristino di un cluster di database a un determinato momento.

Le utilità di backup e ripristino di PostgreSQL non sono supportate

Le seguenti utilità PostgreSQL non sono supportate né per il cluster di database primario né per il gruppo di shard del database:

- `pg_dump`
- `pg_dumpall`
- `pg_restore`

Sebbene sia possibile utilizzarle con file binari open source o metodi alternativi, ciò potrebbe produrre risultati incoerenti.

Aggiornamento di Amazon Aurora PostgreSQL Limitless Database

Quanto segue si applica all'aggiornamento di Aurora PostgreSQL Limitless Database:

- Sono supportati gli aggiornamenti a versioni secondarie.
- È supportata l'applicazione di patch ad Aurora PostgreSQL Limitless Database. Le patch vengono visualizzate come operazioni di manutenzione in sospeso da applicare durante la finestra di manutenzione.

Aggiornamento di cluster di database che utilizzano Amazon Aurora PostgreSQL Limitless Database

È possibile aggiornare un cluster di database modificandolo e scegliendo una nuova versione del motore di database. Puoi utilizzare la Console di gestione AWS o l'AWS CLI.

Console

Per aggiornare il cluster di database Limitless Database

1. Accedi alla Console di gestione AWS e apri la console Amazon RDS all'indirizzo https://console.aws.amazon.com/rds/.
2. Passare alla pagina Databases (Database).
3. Selezionare il cluster di database Limitless Database da aggiornare.
4. Scegliere Modify (Modifica).

   Viene visualizzata la pagina Modifica il cluster DB.
5. Per la Versione del motore DB, scegliere la nuova versione del motore di database, ad esempio Aurora PostgreSQL with Limitless Database (compatibile con PostgreSQL 16.6).
6. Scegli Continue (Continua).
7. Nella pagina di riepilogo, selezionare se applicare le modifiche immediatamente o durante la successiva finestra di manutenzione pianificata, quindi scegliere Modifica cluster.

CLI

Utilizzare il comando AWS CLI modify-db-cluster come mostrato nell'esempio seguente.

```
aws rds modify-db-cluster \
    --db-cluster-identifier my-sv2-cluster \
    --engine-version 16.6-limitless \
    --apply-immediately
```

Riferimento ad Aurora PostgreSQL Limitless Database

Forniamo i seguenti argomenti di riferimento per Aurora PostgreSQL Limitless Database.

Argomenti

- Comandi SQL DDL (Data Definition Language) supportati e non supportati
- Limitazioni DDL e altre informazioni per Aurora PostgreSQL Limitless Database
- Comandi SQL supportati e non supportati per Data Manipulation Language (DML) e per l'elaborazione delle query
- Limitazioni DML e altre informazioni per Aurora PostgreSQL Limitless Database
- Variabili in Aurora PostgreSQL Limitless Database
- Parametri del cluster di database in Aurora PostgreSQL Limitless Database

Comandi SQL DDL (Data Definition Language) supportati e non supportati

La tabella seguente elenca i comandi DDL supportati e non supportati da Aurora PostgreSQL Limitless Database, con riferimenti a limitazioni o ulteriori informazioni.

Comando	Supportato?	Limitazioni o ulteriori informazioni
ALTER AGGREGATE	No	Non applicabile
ALTER COLLATION	Sì	Nessuno
ALTER CONVERSION	Sì	Nessuno
ALTER DATABASE	No	Non applicabile
ALTER DEFAULT PRIVILEGES	No	Non applicabile
ALTER DOMAIN	No	Non applicabile
ALTER EVENT TRIGGER	No	Non applicabile
ALTER EXTENSION	Sì	Estensioni

Comando	Supportato?	Limitazioni o ulteriori informazioni
ALTER FOREIGN DATA WRAPPER	No	Non applicabile
ALTER FOREIGN TABLE	No	Non applicabile
ALTER FUNCTION	Sì	Funzioni
ALTER GROUP	Sì	Nessuno
ALTER INDEX	Sì	Nessuno
ALTER LANGUAGE	No	Non applicabile
ALTER LARGE OBJECT	No	Non applicabile
ALTER MATERIALIZED VIEW	No	Non applicabile
ALTER OPERATOR	Sì	Nessuno
ALTER OPERATOR CLASS	Sì	Nessuno
ALTER OPERATOR FAMILY	Sì	Nessuno
ALTER POLICY	No	Non applicabile
ALTER PROCEDURE	Sì	Nessuno
ALTER PUBLICATION	No	Non applicabile
ALTER ROLE	Sì	Nessuno
ALTER ROUTINE	No	Non applicabile
ALTER RULE	No	Non applicabile
ALTER SCHEMA	Sì	Nessuno
ALTER SEQUENCE	Sì	Sequenze

Comando	Supportato?	Limitazioni o ulteriori informazioni
ALTER SERVER	No	Non applicabile
ALTER STATISTICS	No	Non applicabile
ALTER SUBSCRIPTION	No	Non applicabile
ALTER SYSTEM	No	Non applicabile
ALTER TABLE	Sì	ALTER TABLE
ALTER TABLESPACE	No	Non applicabile
ALTER TEXT SEARCH CONFIGURATION	No	Non applicabile
ALTER TEXT SEARCH DICTIONARY	No	Non applicabile
ALTER TEXT SEARCH PARSER	No	Non applicabile
ALTER TEXT SEARCH TEMPLATE	No	Non applicabile
ALTER TRIGGER	No	Non applicabile
ALTER TYPE	Sì	Nessuno
ALTER USER	Sì	Nessuno
ALTER USER MAPPING	No	Non applicabile
ALTER VIEW	Sì	Nessuno
COMMENT	No	Non applicabile
CREATE ACCESS METHOD	No	Non applicabile

Comando	Supportato?	Limitazioni o ulteriori informazioni
CREATE AGGREGATE	No	Non applicabile
CREATE CAST	Sì	Nessuno
CREATE COLLATION	Sì	Nessuno
CREATE CONVERSION	Sì	Nessuno
CREATE DATABASE	Sì	CREATE DATABASE
CREATE DOMAIN	No	Non applicabile
CREATE EVENT TRIGGER	No	Non applicabile
CREATE EXTENSION	Sì	Estensioni
CREATE FOREIGN DATA WRAPPER	No	Non applicabile
CREATE FOREIGN TABLE	No	Non applicabile
CREATE FUNCTION	Sì	Funzioni
CREATE GROUP	Sì	Nessuno
CREATE INDEX	Sì	CREATE INDEX
CREATE LANGUAGE	No	Non applicabile
CREATE MATERIALIZED VIEW	No	Non applicabile
CREATE OPERATOR	Sì	Nessuno
CREATE OPERATOR CLASS	Sì	Nessuno
CREATE OPERATOR FAMILY	Sì	Nessuno

Comando	Supportato?	Limitazioni o ulteriori informazioni
CREATE POLICY	Sì	Nessuno
CREATE PROCEDURE	Sì	Nessuno
CREATE PUBLICATION	No	Non applicabile
CREATE ROLE	Sì	Nessuno
CREATE RULE	No	Non applicabile
CREATE SCHEMA	Sì	CREATE SCHEMA
CREATE SEQUENCE	Sì	Sequenze
CREATE SERVER	No	Non applicabile
CREATE STATISTICS	No	Non applicabile
CREATE SUBSCRIPTION	No	Non applicabile
CREATE TABLE	Sì	CREATE TABLE
CREATE TABLE AS	Sì	CREATE TABLE AS
CREATE TABLESPACE	No	Non applicabile
CREATE TEMPORARY TABLE	No	Non applicabile
CREATE TEMPORARY TABLE AS	No	Non applicabile
CREATE TEXT SEARCH CONFIGURATION	No	Non applicabile
CREATE TEXT SEARCH DICTIONARY	No	Non applicabile

Comando	Supportato?	Limitazioni o ulteriori informazioni
CREATE TEXT SEARCH PARSER	No	Non applicabile
CREATE TEXT SEARCH TEMPLATE	No	Non applicabile
CREATE TRANSFORM	No	Non applicabile
CREATE TRIGGER	No	Non applicabile
CREATE TYPE	Sì	Nessuno
CREA UTENTE	Sì	Nessuno
CREATE USER MAPPING	No	Non applicabile
CREATE VIEW	Sì	Nessuno
DROP ACCESS METHOD	No	Non applicabile
DROP AGGREGATE	Sì	Nessuno
DROP CAST	Sì	Nessuno
DROP COLLATION	Sì	Nessuno
DROP CONVERSION	Sì	Nessuno
DROP DATABASE	Sì	DROP DATABASE
DROP DOMAIN	No	Non applicabile
DROP EVENT TRIGGER	No	Non applicabile
DROP EXTENSION	Sì	Estensioni
DROP FOREIGN DATA WRAPPER	No	Non applicabile

Comando	Supportato?	Limitazioni o ulteriori informazioni
DROP FOREIGN TABLE	No	Non applicabile
DROP FUNCTION	Sì	Funzioni
DROP GROUP	Sì	Nessuno
DROP INDEX	Sì	Nessuno
DROP LANGUAGE	No	Non applicabile
DROP MATERIALIZED VIEW	No	Non applicabile
DROP OPERATOR	Sì	Nessuno
DROP OPERATOR CLASS	Sì	Nessuno
DROP OPERATOR FAMILY	Sì	Nessuno
DROP OWNED	No	Non applicabile
DROP POLICY	No	Non applicabile
DROP PROCEDURE	Sì	Nessuno
DROP PUBLICATION	No	Non applicabile
DROP ROLE	Sì	Nessuno
DROP ROUTINE	No	Non applicabile
DROP RULE	No	Non applicabile
DROP SCHEMA	Sì	Nessuno
DROP SEQUENCE	Sì	Nessuno
DROP SERVER	No	Non applicabile
DROP STATISTICS	No	Non applicabile

Comando	Supportato?	Limitazioni o ulteriori informazioni
DROP SUBSCRIPTION	No	Nessuno
DROP TABLE	Sì	Nessuno
DROP TABLESPACE	No	Non applicabile
DROP TEXT SEARCH CONFIGURATION	No	Non applicabile
DROP TEXT SEARCH DICTIONARY	No	Non applicabile
DROP TEXT SEARCH PARSER	No	Non applicabile
DROP TEXT SEARCH TEMPLATE	No	Non applicabile
DROP TRANSFORM	No	Non applicabile
DROP TRIGGER	No	Non applicabile
DROP TYPE	Sì	Nessuno
DROP USER	Sì	Nessuno
DROP USER MAPPING	No	Non applicabile
DROP VIEW	Sì	Nessuno
GRANT	Sì	Nessuno
REASSIGN OWNED	No	Non applicabile
REVOKE	Sì	Nessuno
SECURITY LABEL	No	Non applicabile

Comando	Supportato?	Limitazioni o ulteriori informazioni
SELECT INTO	Sì	SELECT INTO
SET	Sì	Nessuno
SET CONSTRAINTS	No	Non applicabile
SET ROLE	Sì	Nessuno
SET SESSION AUTHORIZATION	Sì	Nessuno
SET TRANSACTION	Sì	Nessuno
TRUNCATE	Sì	Nessuno

Limitazioni DDL e altre informazioni per Aurora PostgreSQL Limitless Database

I seguenti argomenti descrivono le limitazioni o forniscono ulteriori informazioni per i comandi SQL DDL in Aurora PostgreSQL Limitless Database.

Argomenti

- ALTER TABLE
- CREATE DATABASE
- CREATE INDEX
- CREATE SCHEMA
- CREATE TABLE
- CREATE TABLE AS
- DROP DATABASE
- SELECT INTO
- Vincoli
- Valori predefiniti
- Estensioni
- Chiavi esterne
- Funzioni
- Sequenze

ALTER TABLE

Il comando `ALTER TABLE` è generalmente supportato in Aurora PostgreSQL Limitless Database. Per ulteriori informazioni, consulta ALTER TABLE nella documentazione di PostgreSQL.

Limitazioni

`ALTER TABLE` presenta le seguenti limitazioni per le opzioni supportate.

Rimozione di una colonna

- Nelle tabelle sottoposte a sharding, non puoi rimuovere le colonne che fanno parte della chiave shard.
- Nelle tabelle di riferimento, non puoi rimuovere le colonne che fanno parte della chiave primaria.

Modifica del tipo di dati di una colonna

- L'espressione `USING` non è supportata.
- Nelle tabelle sottoposte a sharding, non puoi modificare il tipo delle colonne che fanno parte della chiave shard.

Aggiunta o rimozione di un vincolo

Per i dettagli sulle opzioni non supportate, consulta Vincoli.

Modifica del valore predefinito di una colonna

I valori predefiniti sono supportati. Per ulteriori informazioni, consulta Valori predefiniti.

Opzioni non supportate

Alcune opzioni non sono supportate perché dipendono da funzionalità non disponibili, come i trigger.

Le seguenti opzioni a livello di tabella per `ALTER TABLE` non sono supportate:

- `ALL IN TABLESPACE`
- `ATTACH PARTITION`
- `DETACH PARTITION`
- Flag `ONLY`
- `RENAME CONSTRAINT`

Le seguenti opzioni a livello di colonna per `ALTER TABLE` non sono supportate:

- ADD GENERATED
- DROP EXPRESSION [IF EXISTS]
- DROP IDENTITY [IF EXISTS]
- RESET
- RESTART
- SET
- SET COMPRESSION
- SET STATISTICS

CREATE DATABASE

In Aurora PostgreSQL Limitless Database sono supportati solo Limitless Database.

Durante l'esecuzione di `CREATE DATABASE`, i database creati correttamente in uno o più nodi potrebbero non riuscire a essere creati negli altri nodi, poiché la creazione del database è un'operazione non transazionale. In questo caso, gli oggetti del database creati correttamente vengono rimossi automaticamente da tutti i nodi entro un periodo di tempo predeterminato per mantenere la coerenza nel gruppo shard database. Durante questo intervallo di tempo, un nuovo tentativo di creazione di un database con lo stesso nome potrebbe generare un errore che indica che il database esiste già.

Sono supportate le seguenti opzioni:

- Regola di confronto:

```
CREATE DATABASE name WITH
    [LOCALE = locale]
    [LC_COLLATE = lc_collate]
    [LC_CTYPE = lc_ctype]
    [ICU_LOCALE = icu_locale]
    [ICU_RULES = icu_rules]
    [LOCALE_PROVIDER = locale_provider]
    [COLLATION_VERSION = collation_version];
```

- `CREATE DATABASE WITH OWNER`:

```
CREATE DATABASE name WITH OWNER = user_name;
```

Le seguenti opzioni non sono supportate:

- `CREATE DATABASE WITH TABLESPACE`:

```
CREATE DATABASE name WITH TABLESPACE = tablespace_name;
```

- `CREATE DATABASE WITH TEMPLATE`:

```
CREATE DATABASE name WITH TEMPLATE = template;
```

CREATE INDEX

`CREATE INDEX CONCURRENTLY` è supportata per le tabelle sottoposte a sharding:

```
CREATE INDEX CONCURRENTLY index_name ON table_name(column_name);
```

`CREATE UNIQUE INDEX` è supportata per tutti i tipi di tabella:

```
CREATE UNIQUE INDEX index_name ON table_name(column_name);
```

`CREATE UNIQUE INDEX CONCURRENTLY` non è supportata:

```
CREATE UNIQUE INDEX CONCURRENTLY index_name ON table_name(column_name);
```

Per ulteriori informazioni, consulta UNIQUE. Per informazioni generali sulla creazione di indici, consulta CREATE INDEX nella documentazione di PostgreSQL.

Visualizzazione degli indici

> Non tutti gli indici sono visibili sui router quando utilizzi `\d table_name` o comandi simili. In alternativa, utilizza la vista `pg_catalog.pg_indexes` per ottenere gli indici, come mostrato nell'esempio seguente.

```
SET rds_aurora.limitless_create_table_mode='sharded';
SET rds_aurora.limitless_create_table_shard_key='{"id"}';
CREATE TABLE items (id int PRIMARY KEY, val int);
CREATE INDEX items_my_index on items (id, val);

postgres_limitless=> SELECT * FROM pg_catalog.pg_indexes WHERE tablename='items';
```

```
 schemaname | tablename |   indexname    | tablespace |
   indexdef
------------+-----------+----------------+------------
+-------------------------------------------------------------------------
 public     | items     | items_my_index |            | CREATE INDEX items_my_index
 ON ONLY public.items USING btree (id, val)
 public     | items     | items_pkey     |            | CREATE UNIQUE INDEX
 items_pkey ON ONLY public.items USING btree (id)
(2 rows)
```

CREATE SCHEMA

`CREATE SCHEMA` con un elemento dello schema non è supportato:

```
CREATE SCHEMA my_schema CREATE TABLE (column_name INT);
```

Questa opzione genera un errore simile al seguente:

```
ERROR: CREATE SCHEMA with schema elements is not supported
```

CREATE TABLE

Le relazioni nelle istruzioni `CREATE TABLE` non sono supportate, ad esempio:

```
CREATE TABLE orders (orderid int, customerId int, orderDate date) WITH
 (autovacuum_enabled = false);
```

Le colonne `IDENTITY` non sono supportate, ad esempio:

```
CREATE TABLE orders (orderid INT GENERATED ALWAYS AS IDENTITY);
```

Aurora PostgreSQL Limitless Database supporta fino a 54 caratteri per i nomi di tabelle sottoposte a sharding.

CREATE TABLE AS

Per creare una tabella utilizzando `CREATE TABLE AS`, è necessario utilizzare la variabile `rds_aurora.limitless_create_table_mode`. Per le tabelle sottoposte a sharding, è

necessario utilizzare anche la variabile `rds_aurora.limitless_create_table_shard_key`. Per ulteriori informazioni, consulta Creazione di tabelle Limitless utilizzando variabili.

```
-- Set the variables.
SET rds_aurora.limitless_create_table_mode='sharded';
SET rds_aurora.limitless_create_table_shard_key='{"a"}';

CREATE TABLE ctas_table AS SELECT 1 a;

-- "source" is the source table whose columns and data types are used to create the new
 "ctas_table2" table.
CREATE TABLE ctas_table2 AS SELECT a,b FROM source;
```

Non puoi utilizzare `CREATE TABLE AS` per creare tabelle di riferimento, poiché richiedono vincoli di chiave primaria. `CREATE TABLE AS` non propaga le chiavi primarie alle nuove tabelle.

Per ulteriori informazioni, consulta CREATE DATABASE nella documentazione di PostgreSQL.

DROP DATABASE

Puoi eliminare i database creati.

Il comando `DROP DATABASE` viene eseguito in background in modo asincrono. Durante l'esecuzione, riceverai un errore se provi a creare un nuovo database con lo stesso nome.

SELECT INTO

`SELECT INTO` è funzionalmente simile a CREATE TABLE AS. Devi utilizzare la variabile `rds_aurora.limitless_create_table_mode`. Per le tabelle sottoposte a sharding, è necessario utilizzare anche la variabile `rds_aurora.limitless_create_table_shard_key`. Per ulteriori informazioni, consulta Creazione di tabelle Limitless utilizzando variabili.

```
-- Set the variables.
SET rds_aurora.limitless_create_table_mode='sharded';
SET rds_aurora.limitless_create_table_shard_key='{"a"}';

-- "source" is the source table whose columns and data types are used to create the new
 "destination" table.
SELECT * INTO destination FROM source;
```

Attualmente, l'operazione `SELECT INTO` viene eseguita tramite il router, non direttamente tramite gli shard. Di conseguenza, le prestazioni possono essere lente.

Per ulteriori informazioni, consulta SELECT INTO nella documentazione di PostgreSQL.

Vincoli

Le seguenti limitazioni si applicano ai vincoli in Aurora PostgreSQL Limitless Database.

CHECK

Sono supportati vincoli semplici che coinvolgono operatori di confronto con valori letterali. Le espressioni e i vincoli più complessi che richiedono convalide delle funzioni non sono supportati, come illustrato negli esempi seguenti.

```
CREATE TABLE my_table (
    id  INT CHECK (id > 0)                                        -- supported
  , val INT CHECK (val > 0 AND val < 1000)                        -- supported
  , tag TEXT CHECK (length(tag) > 0)                              -- not supported:
 throws "Expression inside CHECK constraint is not supported"
  , op_date TIMESTAMP WITH TIME ZONE CHECK (op_date <= now())  -- not supported:
 throws "Expression inside CHECK constraint is not supported"
);
```

Puoi assegnare nomi espliciti ai vincoli, come illustrato nell'esempio seguente.

```
CREATE TABLE my_table (
    id  INT CONSTRAINT positive_id  CHECK (id > 0)
  , val INT CONSTRAINT val_in_range CHECK (val > 0 AND val < 1000)
);
```

Puoi utilizzare la sintassi dei vincoli a livello di tabella con il vincolo `CHECK`, come illustrato nell'esempio seguente.

```
CREATE TABLE my_table (
    id INT CONSTRAINT positive_id  CHECK (id > 0)
  , min_val INT CONSTRAINT min_val_in_range CHECK (min_val > 0 AND min_val < 1000)
  , max_val INT
  , CONSTRAINT max_val_in_range CHECK (max_val > 0 AND max_val < 1000 AND max_val >
 min_val)
);
```

EXCLUDE

I vincoli di esclusione non sono supportati in Aurora PostgreSQL Limitless Database.

FOREIGN KEY

Per ulteriori informazioni, consulta Chiavi esterne.

NOT NULL

I vincoli NOT NULL non sono supportati senza restrizioni.

PRIMARY KEY

La chiave primaria implica vincoli univoci e pertanto si applicano le stesse restrizioni sui vincoli unici alla chiave primaria. Ciò significa che:

- Se una tabella viene convertita in una tabella sottoposta a sharding, la chiave shard deve essere un sottoinsieme della chiave primaria. Vale a dire, la chiave primaria contiene tutte le colonne della chiave shard.
- Se una tabella viene convertita in una tabella di riferimento, deve avere una chiave primaria.

Negli esempi seguenti viene illustrato l'uso delle chiavi primarie.

```
-- Create a standard table.
CREATE TABLE public.my_table (
    item_id INT
  , location_code INT
  , val INT
  , comment text
);

-- Change the table to a sharded table using the 'item_id' and 'location_code'
 columns as shard keys.
CALL rds_aurora.limitless_alter_table_type_sharded('public.my_table',
 ARRAY['item_id', 'location_code']);
```

Tentativo di aggiungere una chiave primaria che non contiene una chiave shard:

```
-- Add column 'item_id' as the primary key.
-- Invalid because the primary key doesnt include all columns from the shard key:
-- 'location_code' is part of the shard key but not part of the primary key
ALTER TABLE public.my_table ADD PRIMARY KEY (item_id); -- ERROR

-- add column "val" as primary key
-- Invalid because primary key does not include all columns from shard key:
--  item_id and location_code iare part of shard key but not part of the primary key
ALTER TABLE public.my_table ADD PRIMARY KEY (item_id); -- ERROR
```

Tentativo di aggiungere una chiave primaria che contiene una chiave shard:

```
-- Add the 'item_id' and 'location_code' columns as the primary key.
-- Valid because the primary key contains the shard key.
ALTER TABLE public.my_table ADD PRIMARY KEY (item_id, location_code); -- OK

-- Add the 'item_id', 'location_code', and 'val' columns as the primary key.
-- Valid because the primary key contains the shard key.
ALTER TABLE public.my_table ADD PRIMARY KEY (item_id, location_code, val); -- OK
```

Conversione di una tabella standard in tabella di riferimento.

```
-- Create a standard table.
CREATE TABLE zipcodes (zipcode INT PRIMARY KEY, details VARCHAR);

-- Convert the table to a reference table.
CALL rds_aurora.limitless_alter_table_type_reference('public.zipcode');
```

Per ulteriori informazioni sulla creazione di tabelle sottoposte a sharding e di riferimento, consulta Creazione di tabelle di Aurora PostgreSQL Limitless Database.

UNIQUE

Nelle tabelle sottoposte a sharding, la chiave univoca deve contenere la chiave shard, ovvero la chiave shard deve essere un sottoinsieme della chiave univoca. Questa condizione viene verificata durante la conversione del tipo di tabella in sottoposta a sharding. Nelle tabelle di riferimento non sono presenti restrizioni.

```
CREATE TABLE customer (
    customer_id INT NOT NULL
  , zipcode INT
  , email TEXT UNIQUE
);
```

I vincoli UNIQUE a livello di tabella sono supportati, come illustrato nell'esempio seguente.

```
CREATE TABLE customer (
    customer_id INT NOT NULL
  , zipcode INT
  , email TEXT
  , CONSTRAINT zipcode_and_email UNIQUE (zipcode, email)
```

```
);
```

L'esempio seguente mostra l'uso combinato di una chiave primaria e una chiave univoca. Entrambe le chiavi devono includere la chiave shard.

```
SET rds_aurora.limitless_create_table_mode='sharded';
SET rds_aurora.limitless_create_table_shard_key='{"p_id"}';
CREATE TABLE t1 (
p_id BIGINT NOT NULL,
c_id BIGINT NOT NULL,
PRIMARY KEY (p_id),
UNIQUE (p_id, c_id)
);
```

Per ulteriori informazioni, consulta Vincoli nella documentazione di PostgreSQL.

Valori predefiniti

Aurora PostgreSQL Limitless Database supporta espressioni nei valori predefiniti.

L'esempio seguente mostra l'uso dei valori predefiniti.

```
CREATE TABLE t (
    a INT DEFAULT 5,
    b TEXT DEFAULT 'NAN',
    c NUMERIC
);

CALL rds_aurora.limitless_alter_table_type_sharded('t', ARRAY['a']);
INSERT INTO t DEFAULT VALUES;
SELECT * FROM t;

 a |  b  | c
---+-----+---
 5 | NAN |
(1 row)
```

Le espressioni sono supportate, come illustrato nell'esempio seguente.

```
CREATE TABLE t1 (a NUMERIC DEFAULT random());
```

Nell'esempio seguente viene aggiunta una nuova colonna che è `NOT NULL` e ha un valore predefinito.

```
ALTER TABLE t ADD COLUMN d BOOLEAN NOT NULL DEFAULT FALSE;
SELECT * FROM t;

 a |  b  | c | d
---+-----+---+---
 5 | NAN |   | f
(1 row)
```

L'esempio seguente modifica una colonna esistente con un valore predefinito.

```
ALTER TABLE t ALTER COLUMN c SET DEFAULT 0.0;
INSERT INTO t DEFAULT VALUES;
SELECT * FROM t;

 a |  b  | c   |  d
---+-----+-----+-----
 5 | NAN |     | f
 5 | NAN | 0.0 | f
(2 rows)
```

L'esempio seguente rimuove un valore predefinito.

```
ALTER TABLE t ALTER COLUMN a DROP DEFAULT;
INSERT INTO t DEFAULT VALUES;
SELECT * FROM t;

 a |  b  | c   |  d
---+-----+-----+-----
 5 | NAN |     | f
 5 | NAN | 0.0 | f
   | NAN | 0.0 | f
(3 rows)
```

Per ulteriori informazioni, consulta Valori predefiniti nella documentazione di PostgreSQL.

Estensioni

Le seguenti estensioni PostgreSQL sono supportate in Aurora PostgreSQL Limitless Database:

- `aurora_limitless_fdw`: questa estensione è preinstallata. e non può essere rimossa.
- `aws_s3`: questa estensione funziona in Aurora PostgreSQL Limitless Database in modo simile ad Aurora PostgreSQL.

  Puoi importare dati da un bucket Amazon S3 in un cluster di database Aurora PostgreSQL Limitless Database oppure esportare dati da un cluster di database Aurora PostgreSQL Limitless Database in un bucket Amazon S3. Per ulteriori informazioni, consulta Importazione di dati da Amazon S3 in un cluster database Aurora PostgreSQL e Esportazione di dati da del cluster di database Aurora PostgreSQLRDS per PostgreSQL a Amazon S3.
- `btree_gin`
- `citext`
- `ip4r`
- `pg_buffercache`: questa estensione si comporta in modo diverso in Aurora PostgreSQL Limitless Database rispetto alla versione Community PostgreSQL. Per ulteriori informazioni, consulta Differenze di pg_buffercache in Aurora PostgreSQL Limitless Database.
- `pg_stat_statements`
- `pg_trgm`
- `pgcrypto`
- `pgstattuple`: questa estensione si comporta in modo diverso in Aurora PostgreSQL Limitless Database rispetto alla versione Community PostgreSQL. Per ulteriori informazioni, consulta Differenze di pgstattuple in Aurora PostgreSQL Limitless Database.
- `pgvector`
- `plpgsql`: questa estensione è preinstallata, ma puoi rimuoverla.
- `PostGIS`: le transazioni lunghe e le funzioni di gestione delle tabelle non sono supportate. La modifica della tabella di riferimento spaziale non è supportata.
- `unaccent`
- `uuid`

La maggior parte delle estensioni PostgreSQL non è attualmente supportata in Aurora PostgreSQL Limitless Database. Tuttavia, puoi comunque utilizzare l'impostazione della configurazione

shared_preload_libraries (SPL) per caricare le estensioni nel cluster di database primario Aurora PostgreSQL. Le estensioni vengono caricate anche in Aurora PostgreSQL Limitless Database, ma potrebbero non funzionare correttamente.

Ad esempio, puoi caricare l'estensione `pg_hint_plan`, ma il suo caricamento non garantisce che vengano utilizzati i suggerimenti passati nei commenti delle query.

> **Note**
>
> Non puoi modificare gli oggetti associati all'estensione pg_stat_statements. Per informazioni sull'installazione di `pg_stat_statements`, consulta limitless_stat_statements.

Puoi utilizzare le funzioni `pg_available_extensions` e `pg_available_extension_versions` per trovare le estensioni supportate in Aurora PostgreSQL Limitless Database.

Le seguenti istruzioni DDL sono supportate per le estensioni:

CREATE EXTENSION

Puoi creare estensioni, come in PostgreSQL.

```
CREATE EXTENSION [ IF NOT EXISTS ] extension_name
    [ WITH ] [ SCHEMA schema_name ]
             [ VERSION version ]
             [ CASCADE ]
```

Per ulteriori informazioni, consulta CREATE EXTENSION nella documentazione di PostgreSQL.

ALTER EXTENSION

Le seguenti istruzioni DDL sono supportate:

```
ALTER EXTENSION name UPDATE [ TO new_version ]

ALTER EXTENSION name SET SCHEMA new_schema
```

Per ulteriori informazioni, consulta ALTER EXTENSION nella documentazione di PostgreSQL.

DROP EXTENSION

Puoi rimuovere le estensioni, come in PostgreSQL.

```
DROP EXTENSION [ IF EXISTS ] name [, ...] [ CASCADE | RESTRICT ]
```

Per ulteriori informazioni, consulta DROP EXTENSION nella documentazione di PostgreSQL.

Le seguenti istruzioni DDL non sono supportate per le estensioni:

ALTER EXTENSION

Non puoi aggiungere o rimuovere oggetti membri delle estensioni.

```
ALTER EXTENSION name ADD member_object

ALTER EXTENSION name DROP member_object
```

Differenze di pg_buffercache in Aurora PostgreSQL Limitless Database

In Aurora PostgreSQL Limitless Database, quando installi l'estensione pg_buffercache e utilizzi la vista `pg_buffercache`, ricevi informazioni relative al buffer solo dal nodo a cui sei hai attualmente effettuato la connessione: il router. Allo stesso modo, l'uso della funzione `pg_buffercache_summary` o `pg_buffercache_usage_counts` fornisce informazioni solo dal nodo connesso.

Puoi disporre di numerosi nodi e potresti aver bisogno di accedere alle informazioni del buffer da qualsiasi nodo per diagnosticare i problemi in modo efficace. Pertanto, Aurora PostgreSQL Limitless Database fornisce le seguenti funzioni:

- `rds_aurora.limitless_pg_buffercache(subcluster_id)`
- `rds_aurora.limitless_pg_buffercache_summary(subcluster_id)`
- `rds_aurora.limitless_pg_buffercache_usage_counts(subcluster_id)`

Inserendo l'ID cluster secondario di qualsiasi nodo, che si tratti di un router o di uno shard, puoi accedere facilmente alle informazioni del buffer specifiche di quel nodo. Queste funzioni sono disponibili subito dopo l'installazione dell'estensione `pg_buffercache` nel Limitless Database.

> **Note**
>
> Aurora PostgreSQL Limitless Database supporta queste funzioni per la versione 1.4 e successive dell'estensione `pg_buffercache`.

Le colonne mostrate nella vista `limitless_pg_buffercache` differiscono leggermente da quelle nella vista `pg_buffercache`:

- `bufferid`: rimane invariata rispetto a `pg_buffercache`.
- `relname`: invece di mostrare il numero di nodo del file come in `pg_buffercache`, `limitless_pg_buffercache` presenta il valore `relname` associato, se disponibile, nel database attuale o nei cataloghi del sistema condiviso, altrimenti `NULL`.
- `parent_relname`: questa nuova colonna, non presente in `pg_buffercache`, mostra il valore `relname` principale se il valore nella colonna `relname` rappresenta una tabella partizionata (nel caso di tabelle sottoposte a sharding). Altrimenti, mostra `NULL`.
- `spcname`: invece di mostrare l'identificatore dell'oggetto tablespace (OID) come in `pg_buffercache`, `limitless_pg_buffercache` mostra il nome del tablespace.
- `datname`: invece di mostrare l'OID del database come in `pg_buffercache`, `limitless_pg_buffercache` mostrare il nome del database.
- `relforknumber`: rimane invariata rispetto a `pg_buffercache`.
- `relblocknumber`: rimane invariata rispetto a `pg_buffercache`.
- `isdirty`: rimane invariata rispetto a `pg_buffercache`.
- `usagecount`: rimane invariata rispetto a `pg_buffercache`.
- `pinning_backends`: rimane invariata rispetto a `pg_buffercache`.

Le colonne nelle viste `limitless_pg_buffercache_summary` e `limitless_pg_buffercache_usage_counts` sono le stesse di quelle presenti nelle normali viste `pg_buffercache_summary` e `pg_buffercache_usage_counts`, rispettivamente.

Tramite queste funzioni, puoi accedere a informazioni dettagliate sulla cache del buffer in tutti i nodi dell'ambiente Aurora PostgreSQL Limitless Database, facilitando una diagnosi e una gestione più efficaci dei sistemi di database.

Differenze di pgstattuple in Aurora PostgreSQL Limitless Database

In Aurora PostgreSQL, l'estensione pgstattuple attualmente non supporta tabelle esterne, tabelle partizionate o indici partizionati. Tuttavia, in Aurora PostgreSQL Limitless Database, gli oggetti creati dall'utente sono spesso tra questi tipi non supportati. Sebbene esistano tabelle e indici normali (ad esempio, le tabelle di catalogo e i relativi indici), la maggior parte degli oggetti risiede su nodi esterni, diventando quindi oggetti esterni per il router.

Riconosciamo l'importanza di questa estensione per ottenere statistiche a livello di tuple, fondamentali per attività come la rimozione del bloat e la raccolta di informazioni diagnostiche. Pertanto, Aurora PostgreSQL Limitless Database fornisce supporto per l'estensione `pgstattuple` nei Limitless Database.

Aurora PostgreSQL Limitless Database include le seguenti funzioni nello schema `rds_aurora`:

Funzioni statistiche a livello di tuple

`rds_aurora.limitless_pgstattuple(relation_name)`

- Scopo: estrarre statistiche a livello di tuple per le tabelle standard e i relativi indici
- Input: `relation_name` (testo): il nome della relazione
- Output: colonne coerenti con quelle restituite dalla funzione `pgstattuple` in Aurora PostgreSQL

`rds_aurora.limitless_pgstattuple(relation_name, subcluster_id)`

- Scopo: estrarre statistiche a livello di tuple per le tabelle di riferimento, le tabelle sottoposte a sharding, le tabelle di catalogo e i relativi indici
- Input:
  - `relation_name` (testo): il nome della relazione
  - `subcluster_id` (testo): l'ID cluster secondario del nodo da cui estrarre le statistiche
- Output:
  - Per le tabelle di riferimento e di catalogo (inclusi i relativi indici), le colonne sono coerenti con Aurora PostgreSQL.
  - Per le tabelle sottoposte a sharding, le statistiche rappresentano solo la partizione della tabella sottoposta a sharding che risiede nel cluster secondario specificato.

Funzioni di statistiche sugli indici

`rds_aurora.limitless_pgstatindex(relation_name)`

- Scopo: estrarre le statistiche per gli indici B-tree su tabelle standard
- Input: `relation_name` (testo): il nome dell'indice B-tree
- Output: vengono restituite tutte le colonne tranne `root_block_no`. Le colonne restituite sono coerenti con la funzione `pgstatindex` in Aurora PostgreSQL.

`rds_aurora.limitless_pgstatindex(relation_name, subcluster_id)`

- Scopo: estrarre le statistiche per gli indici B-tree su tabelle di riferimento, tabelle sottoposte a sharding e tabelle di catalogo.
- Input:
  - `relation_name` (testo): il nome dell'indice B-tree
  - `subcluster_id` (testo): l'ID cluster secondario del nodo da cui estrarre le statistiche
- Output:
  - Per gli indici delle tabelle di riferimento e di catalogo, vengono restituite tutte le colonne (tranne `root_block_no`). Le colonne restituite sono coerenti con Aurora PostgreSQL.
  - Per le tabelle sottoposte a sharding, le statistiche rappresentano solo la partizione dell'indice della tabella sottoposta a sharding che risiede nel cluster secondario specificato. La colonna `tree_level` mostra la media di tutte le sezioni di tabella del cluster secondario richiesto.

`rds_aurora.limitless_pgstatginindex(relation_name)`

- Scopo: estrarre le statistiche per gli indici invertiti generalizzati (GIN) su tabelle standard
- Input: `relation_name` (testo): il nome del GIN
- Output: colonne coerenti con quelle restituite dalla funzione `pgstatginindex` in Aurora PostgreSQL

`rds_aurora.limitless_pgstatginindex(relation_name, subcluster_id)`

- Scopo: estrarre le statistiche per gli indici GIN su tabelle di riferimento, tabelle sottoposte a sharding e tabelle di catalogo.
- Input:
  - `relation_name` (testo): il nome dell'indice
  - `subcluster_id` (testo): l'ID cluster secondario del nodo da cui estrarre le statistiche
- Output:
  - Per gli indici GIN di tabelle di riferimento e catalogo, le colonne sono coerenti con Aurora PostgreSQL.

- Per le tabelle sottoposte a sharding, le statistiche rappresentano solo la partizione dell'indice della tabella sottoposta a sharding che risiede nel cluster secondario specificato.

`rds_aurora.limitless_pgstathashindex(`*`relation_name`*`)`

- Scopo: estrarre le statistiche per gli indici hash su tabelle standard
- Input: `relation_name` (testo): il nome del indice hash
- Output: colonne coerenti con quelle restituite dalla funzione `pgstathashindex` in Aurora PostgreSQL

`rds_aurora.limitless_pgstathashindex(`*`relation_name`*`, `*`subcluster_id`*`)`

- Scopo: estrarre le statistiche per gli indici hash su tabelle di riferimento, tabelle sottoposte a sharding e tabelle di catalogo.
- Input:
  - `relation_name` (testo): il nome dell'indice
  - `subcluster_id` (testo): l'ID cluster secondario del nodo da cui estrarre le statistiche
- Output:
  - Per gli indici hash di tabelle di riferimento e catalogo, le colonne sono coerenti con Aurora PostgreSQL.
  - Per le tabelle sottoposte a sharding, le statistiche rappresentano solo la partizione dell'indice della tabella sottoposta a sharding che risiede nel cluster secondario specificato.

Funzioni di conteggio delle pagine

`rds_aurora.limitless_pg_relpages(`*`relation_name`*`)`

- Scopo: estrarre il conteggio delle pagine per le tabelle standard e i relativi indici
- Input: `relation_name` (testo): il nome della relazione
- Output: conteggio di pagine della relazione specificata

`rds_aurora.limitless_pg_relpages(`*`relation_name`*`, `*`subcluster_id`*`)`

- Scopo: estrarre il conteggio delle pagine per le tabelle di riferimento, le tabelle sottoposte a sharding e le tabelle di catalogo (inclusi i relativi indici)
- Input:
  - `relation_name` (testo): il nome della relazione
  - `subcluster_id` (testo): l'ID cluster secondario del nodo da cui estrarre il conteggio delle pagine

- Output: per le tabelle sottoposte a sharding, il conteggio delle pagine è la somma delle pagine di tutte le sezioni della tabella sul cluster secondario specificato.

Funzioni di statistiche approssimate a livello di tuple

`rds_aurora.limitless_pgstattuple_approx(`*`relation_name`*`)`

- Scopo: estrarre statistiche approssimative a livello di tuple per le tabelle standard e i relativi indici
- Input: `relation_name` (testo): il nome della relazione
- Output: colonne coerenti con quelle restituite dalla funzione pgstattuple_approx in Aurora PostgreSQL

`rds_aurora.limitless_pgstattuple_approx(`*`relation_name`*`, `*`subcluster_id`*`)`

- Scopo: estrarre statistiche approssimate a livello di tuple per le tabelle di riferimento, le tabelle sottoposte a sharding e le tabelle di catalogo (inclusi i relativi indici)
- Input:
  - `relation_name` (testo): il nome della relazione
  - `subcluster_id` (testo): l'ID cluster secondario del nodo da cui estrarre le statistiche
- Output:
  - Per le tabelle di riferimento e di catalogo (inclusi i relativi indici), le colonne sono coerenti con Aurora PostgreSQL.
  - Per le tabelle sottoposte a sharding, le statistiche rappresentano solo la partizione della tabella sottoposta a sharding che risiede nel cluster secondario specificato.

> **Note**
>
> Attualmente, Aurora PostgreSQL Limitless Database non supporta l'estensione `pgstattuple` su viste materializzate, tabelle TOAST o tabelle temporanee.
> In Aurora PostgreSQL Limitless Database, è necessario fornire l'input come testo, sebbene Aurora PostgreSQL supporti altri formati.

Chiavi esterne

I vincoli di chiave esterna (`FOREIGN KEY`) sono supportati con alcune limitazioni:

- `CREATE TABLE` con `FOREIGN KEY` è supportato solo per le tabelle standard. Per creare una tabella sottoposta a sharding o una tabella di riferimento con `FOREIGN KEY`, create innanzitutto la tabella senza un vincolo di chiave esterna. Successivamente, modificala utilizzando la seguente istruzione:

  ```
  ALTER TABLE ADD CONSTRAINT;
  ```

- La conversione di una tabella standard in una tabella sottoposta a sharding o di riferimento non è supportata se la tabella possiede un vincolo di chiave esterna. Rimuovi il vincolo, quindi aggiungilo dopo la conversione.
- Le seguenti limitazioni si applicano ai tipi di tabella per i vincoli di chiave esterna:
  - Una tabella standard può avere un vincolo di chiave esterna rispetto a un'altra tabella standard.
  - Una tabella sottoposta a sharding può avere un vincolo di chiave esterna se le tabelle principale e secondaria sono co-localizzate e la chiave esterna è un superset della chiave shard.
  - Una tabella sottoposta a sharding può avere un vincolo di chiave esterna rispetto a una tabella di riferimento.
  - Una tabella di riferimento può avere un vincolo di chiave esterna rispetto a un'altra tabella di riferimento.

Argomenti

- Opzioni di chiave esterna
- Esempi

Opzioni di chiave esterna

Le chiavi esterne sono supportate in Aurora PostgreSQL Limitless Database per alcune opzioni DDL. La tabella seguente elenca le opzioni supportate e non supportate tra le tabelle di Aurora PostgreSQL Limitless Database:

Opzione DDL	Da riferimento a riferimento	Da sottoposta a sharding a sottoposta a sharding (co-localizzate)	Da sottoposta a sharding a riferimento	Da standard a standard
`DEFERRABLE`	Sì	Sì	Sì	Sì
`INITIALLY DEFERRED`	Sì	Sì	Sì	Sì
`INITIALLY IMMEDIATE`	Sì	Sì	Sì	Sì
`MATCH FULL`	Sì	Sì	Sì	Sì
`MATCH PARTIAL`	No	No	No	No
`MATCH SIMPLE`	Sì	Sì	Sì	Sì
`NOT DEFERRABLE`	Sì	Sì	Sì	Sì
`NOT VALID`	Sì	No	No	Sì
`ON DELETE CASCADE`	Sì	Sì	Sì	Sì
`ON DELETE NO ACTION`	Sì	Sì	Sì	Sì
`ON DELETE RESTRICT`	Sì	Sì	Sì	Sì
`ON DELETE SET DEFAULT`	No	No	No	No

Opzione DDL	Da riferimento a riferimento	Da sottoposta a sharding a sottoposta a sharding (co-localizzate)	Da sottoposta a sharding a riferimento	Da standard a standard
`ON DELETE SET NULL`	Sì	No	No	Sì
`ON UPDATE CASCADE`	No	No	No	Sì
`ON UPDATE NO ACTION`	Sì	Sì	Sì	Sì
`ON UPDATE RESTRICT`	Sì	Sì	Sì	Sì
`ON UPDATE SET DEFAULT`	No	No	No	No
`ON UPDATE SET NULL`	Sì	No	No	Sì

Esempi

- Da standard a standard:

```
set rds_aurora.limitless_create_table_mode='standard';

CREATE TABLE products(
    product_no integer PRIMARY KEY,
    name text,
    price numeric
);

CREATE TABLE orders (
    order_id integer PRIMARY KEY,
    product_no integer REFERENCES products (product_no),
```

```
    quantity integer
);

SELECT constraint_name, table_name, constraint_type
FROM information_schema.table_constraints WHERE constraint_type='FOREIGN KEY';

 constraint_name         | table_name  | constraint_type
-------------------------+-------------+-----------------
 orders_product_no_fkey  | orders      | FOREIGN KEY
(1 row)
```

- Da sottoposta a sharding a sottoposta a sharding (co-localizzate):

```
set rds_aurora.limitless_create_table_mode='sharded';
set rds_aurora.limitless_create_table_shard_key='{"product_no"}';
CREATE TABLE products(
    product_no integer PRIMARY KEY,
    name text,
    price numeric
);

set rds_aurora.limitless_create_table_shard_key='{"order_id"}';
set rds_aurora.limitless_create_table_collocate_with='products';
CREATE TABLE orders (
    order_id integer PRIMARY KEY,
    product_no integer,
    quantity integer
);

ALTER TABLE orders ADD CONSTRAINT order_product_fk FOREIGN KEY (product_no)
 REFERENCES products (product_no);
```

- Da sottoposta a sharding a riferimento:

```
set rds_aurora.limitless_create_table_mode='reference';
CREATE TABLE products(
    product_no integer PRIMARY KEY,
    name text,
    price numeric
);

set rds_aurora.limitless_create_table_mode='sharded';
set rds_aurora.limitless_create_table_shard_key='{"order_id"}';
```

```
CREATE TABLE orders (
    order_id integer PRIMARY KEY,
    product_no integer,
    quantity integer
);

ALTER TABLE orders ADD CONSTRAINT order_product_fk FOREIGN KEY (product_no)
 REFERENCES products (product_no);
```

- Da riferimento a riferimento:

```
set rds_aurora.limitless_create_table_mode='reference';
CREATE TABLE products(
    product_no integer PRIMARY KEY,
    name text,
    price numeric
);
CREATE TABLE orders (
    order_id integer PRIMARY KEY,
    product_no integer,
    quantity integer
);

ALTER TABLE orders ADD CONSTRAINT order_product_fk FOREIGN KEY (product_no)
 REFERENCES products (product_no);
```

Funzioni

Le funzioni sono supportate in Aurora PostgreSQL Limitless Database.

I seguenti DDL sono supportati per le funzioni:

CREATE FUNCTION

Puoi creare funzioni, come in Aurora PostgreSQL, con l'eccezione di modificarne la volatilità durante la sostituzione.

Per ulteriori informazioni, consulta CREATE FUNCTION nella documentazione di PostgreSQL.

ALTER FUNCTION

Puoi creare funzioni, come in Aurora PostgreSQL, con l'eccezione di modificarne la volatilità.

Per ulteriori informazioni, consulta ALTER FUNCTION nella documentazione di PostgreSQL.

DROP FUNCTION

Puoi rimuovere le funzioni, come in Aurora PostgreSQL.

```
DROP FUNCTION [ IF EXISTS ] name [ ( [ [ argmode ] [ argname ] argtype [, ...] ] ) ]
 [, ...]
    [ CASCADE | RESTRICT ]
```

Per ulteriori informazioni, consulta DROP FUNCTION nella documentazione di PostgreSQL.

Argomenti

- Distribuzione delle funzioni
- Volatilità della funzione

Distribuzione delle funzioni

Quando tutte le istruzioni di una funzione sono indirizzate a un singolo shard, è conveniente eseguire l'intera funzione direttamente sullo shard di destinazione. In questo modo il risultato viene propagato al router, invece di eseguire la funzione direttamente sul router stesso. La capacità di pushdown di funzioni e stored procedure è utile per i clienti che desiderano eseguire la funzione o la stored procedure più vicino all'origine dati, ovvero allo shard.

Per distribuire una funzione, è necessario innanzitutto creare la funzione, quindi chiamare la procedura `rds_aurora.limitless_distribute_function` per distribuirla. Questa funzione usa la seguente sintassi:

```
SELECT rds_aurora.limitless_distribute_function('function_prototype',
 ARRAY['shard_key'], 'collocating_table');
```

La funzione accetta i seguenti parametri:

- *`function_prototype`*: la funzione da distribuire. Riporta solo gli argomenti in input, non quelli in output.

  Se uno qualsiasi degli argomenti è definito come parametro `OUT`, non includerne il tipo negli argomenti di `function_prototype`.
- `ARRAY['`*`shard_key`*`']`: l'elenco degli argomenti della funzione identificati come chiave shard per la funzione.
- *`collocating_table`*: la tabella condivisa che contiene l'intervallo di dati sullo shard di destinazione.

Per identificare lo shard su cui eseguire questa funzione, il sistema prende l'argomento `ARRAY['`*`shard_key`*`']`, ne esegue l'hash e individua lo shard da *`collocating_table`* che ospita l'intervallo contenente questo valore hash.

Restrizioni

Quando distribuisci una funzione o una procedura, questa opera solo sui dati delimitati dall'intervallo di chiavi shard in tale shard. Se la funzione o la procedura tenta di accedere ai dati da un altro shard, i risultati restituiti dalla funzione o dalla procedura distribuita saranno diversi rispetto a una funzione non distribuita.

Ad esempio, crea una funzione contenente query che interessano più shard, ma poi chiama la procedura `rds_aurora.limitless_distribute_function` per distribuirla. Quando invochi questa funzione fornendo argomenti per una chiave shard, è probabile che i risultati della sua esecuzione siano limitati dai valori presenti in tale shard. Questi risultati sono diversi da quelli prodotti senza distribuire la funzione.

Esempi

Considera la seguente funzione `func` con la tabella sottoposta a sharding `customers` con la chiave shard `customer_id`.

```
postgres_limitless=> CREATE OR REPLACE FUNCTION func(c_id integer, sc integer)
 RETURNS int
    language SQL
    volatile
    AS $$
    UPDATE customers SET score = sc WHERE customer_id = c_id RETURNING score;
    $$;
```

Ora distribuiamo questa funzione:

```
SELECT rds_aurora.limitless_distribute_function('func(integer, integer)',
 ARRAY['c_id'], 'customers');
```

Di seguito è riportato un esempio di piani di query.

```
EXPLAIN(costs false, verbose true) SELECT func(27+1,10);

                    QUERY PLAN
 ---------------------------------------------------
  Foreign Scan
    Output: (func((27 + 1), 10))
    Remote SQL:  SELECT func((27 + 1), 10) AS func
  Single Shard Optimized
 (4 rows)
```

```
EXPLAIN(costs false, verbose true)
 SELECT * FROM customers,func(customer_id, score) WHERE customer_id=10 AND score=27;

                          QUERY PLAN
 ---------------------------------------------------------------------
  Foreign Scan
    Output: customer_id, name, score, func
    Remote SQL:  SELECT customers.customer_id,
      customers.name,
      customers.score,
      func.func
```

```
     FROM public.customers,
      LATERAL func(customers.customer_id, customers.score) func(func)
    WHERE ((customers.customer_id = 10) AND (customers.score = 27))
  Single Shard Optimized
 (10 rows)
```

L'esempio seguente mostra una procedura con parametri `IN` e `OUT` come argomenti.

```
CREATE OR REPLACE FUNCTION get_data(OUT id INTEGER, IN arg_id INT)
    AS $$
    BEGIN
        SELECT customer_id,
        INTO id
        FROM customer
        WHERE customer_id = arg_id;
    END;
    $$ LANGUAGE plpgsql;
```

L'esempio seguente distribuisce la procedura utilizzando solo i parametri `IN`.

```
EXPLAIN(costs false, verbose true) SELECT * FROM get_data(1);

             QUERY PLAN
 -----------------------------------
  Foreign Scan
    Output: id
    Remote SQL:  SELECT customer_id
     FROM get_data(1) get_data(id)
  Single Shard Optimized
 (6 rows)
```

Volatilità della funzione

Puoi determinare se una funzione è immutabile, stabile o volatile controllando il valore `provolatile` nella vista pg_proc. Il valore `provolatile` indica se il risultato della funzione dipende solo dai suoi argomenti di input o è influenzato da fattori esterni.

Il valore è uno dei seguenti:

- `i`: funzioni immutabili, che restituiscono sempre lo stesso risultato per gli stessi input.
- `s`: funzioni stabili, i cui risultati (per input fissi) non cambiano durante una scansione.

- v: funzioni volatili, i cui risultati potrebbero cambiare in qualsiasi momento. Usa v anche per funzioni con effetti collaterali, in modo che le chiamate non possano essere ottimizzate ed eliminate.

Gli esempi seguenti mostrano le funzioni volatili.

```
SELECT proname, provolatile FROM pg_proc WHERE proname='pg_sleep';

 proname  | provolatile
----------+-------------
 pg_sleep | v
(1 row)

SELECT proname, provolatile FROM pg_proc WHERE proname='uuid_generate_v4';

     proname      | provolatile
------------------+-------------
 uuid_generate_v4 | v
(1 row)

SELECT proname, provolatile FROM pg_proc WHERE proname='nextval';

 proname | provolatile
---------+-------------
 nextval | v
(1 row)
```

La modifica della volatilità di una funzione esistente non è supportata in Aurora PostgreSQL Limitless Database. Questo vale sia per i comandi `ALTER FUNCTION` che per i comandi `CREATE OR REPLACE FUNCTION`, come illustrato negli esempi seguenti.

```
-- Create an immutable function
CREATE FUNCTION immutable_func1(name text) RETURNS text language plpgsql
AS $$
BEGIN
    RETURN name;
END;
$$IMMUTABLE;

-- Altering the volatility throws an error
ALTER FUNCTION immutable_func1 STABLE;
```

```
-- Replacing the function with altered volatility throws an error
CREATE OR REPLACE FUNCTION immutable_func1(name text) RETURNS text language plpgsql
AS $$
BEGIN
    RETURN name;
END;
$$VOLATILE;
```

Consigliamo vivamente di assegnare le volatilità corrette alle funzioni. Ad esempio, se la funzione utilizza `SELECT` da più tabelle o fa riferimento a oggetti del database, non impostarla come `IMMUTABLE`. Se il contenuto della tabella cambia, l'immutabilità viene compromessa.

Aurora PostgreSQL consente `SELECT` all'interno di funzioni immutabili, ma i risultati potrebbero essere errati. Aurora PostgreSQL Limitless Database può restituire sia errori sia risultati errati. Per ulteriori informazioni, consulta Function volatility categories nella documentazione di PostgreSQL.

Sequenze

Le sequenze nominate sono oggetti di database che generano numeri univoci in ordine crescente o decrescente. `CREATE SEQUENCE` crea un nuovo generatore di numeri di sequenza. I valori di sequenza sono garantiti come univoci.

Quando crei una sequenza denominata in Aurora PostgreSQL Limitless Database, viene creato un oggetto di sequenza distribuita. Successivamente, Aurora PostgreSQL Limitless Database distribuisce blocchi di valori di sequenza non sovrapposti su tutti i Distributed Transaction Router (router). I blocchi sono rappresentati come oggetti di sequenza locale sui router; pertanto, le operazioni di sequenza come `nextval` e `currval` vengono eseguite localmente. I router operano in modo indipendente e richiedono nuovi blocchi dalla sequenza distribuita, quando necessario.

Per ulteriori informazioni sulle sequenze, consulta CREATE SEQUENCE nella documentazione di PostgreSQL.

Argomenti

- Richiesta di un nuovo blocco
- Limitazioni
- Opzioni non supportate
- Esempi
- Viste delle sequenze
- Risoluzione dei problemi delle sequenze

Richiesta di un nuovo blocco

Puoi configurare le dimensioni dei blocchi allocati sui router utilizzando il parametro `rds_aurora.limitless_sequence_chunk_size`. Il valore predefinito è `250000`. Ogni router possiede inizialmente due blocchi: attivi e riservati. I blocchi attivi vengono utilizzati per configurare gli oggetti di sequenza locale (impostando `minvalue` e `maxvalue`) e i blocchi riservati vengono archiviati in una tabella interna del catalogo. Quando un blocco attivo raggiunge il valore minimo o massimo, viene sostituito dal blocco riservato. A tal fine, `ALTER SEQUENCE` viene utilizzato internamente, dunque `AccessExclusiveLock` viene acquisito.

I worker in background vengono eseguiti ogni 10 secondi sui nodi del router per scansionare le sequenze alla ricerca di blocchi riservati utilizzati. Se viene trovato un blocco utilizzato, il worker richiede un nuovo blocco dalla sequenza distribuita. Assicurati di impostare una dimensione dei

blocchi sufficientemente grande affinché i worker in background abbiano tempo sufficiente per richiederne di nuovi. Le richieste remote non avvengono mai nel contesto delle sessioni utente, quindi non puoi richiedere direttamente una nuova sequenza.

Limitazioni

Le limitazioni seguenti si applicano alle sequenze in Aurora PostgreSQL Limitless Database:

- Il catalogo `pg_sequence`, la funzione `pg_sequences` e l'istruzione `SELECT * FROM sequence_name` mostrano solo lo stato della sequenza locale, non quello distribuito.
- I valori della sequenza sono garantiti come univoci e monotoni all'interno di una sessione. Tuttavia, possono risultare errati rispetto alle istruzioni `nextval` eseguite in altre sessioni, se tali sessioni sono connesse ad altri router.
- Assicurati che la dimensione della sequenza (numero di valori disponibili) sia sufficientemente grande da poter essere distribuita su tutti i router. Usa il parametro `rds_aurora.limitless_sequence_chunk_size` per configurare `chunk_size`. Ogni router ha due blocchi.
- L'opzione `CACHE`è supportata, ma la cache deve essere inferiore a `chunk_size`.

Opzioni non supportate

Le seguenti opzioni non sono supportate per le sequenze in Aurora PostgreSQL Limitless Database.

Funzioni di manipolazione di sequenze

La funzione `setval` non è supportata. Per ulteriori informazioni, consulta Sequence Manipulation Functions nella documentazione di PostgreSQL.

CREATE SEQUENCE

Le seguenti opzioni non sono supportate.

```
CREATE [{ TEMPORARY | TEMP} | UNLOGGED] SEQUENCE
    [[ NO ] CYCLE]
```

Per ulteriori informazioni, consulta CREATE SEQUENCE nella documentazione di PostgreSQL.

ALTER SEQUENCE

Le seguenti opzioni non sono supportate.

```
ALTER SEQUENCE
    [[ NO ] CYCLE]
```

Per ulteriori informazioni, consulta ALTER SEQUENCE nella documentazione di PostgreSQL.

ALTER TABLE

Il comando `ALTER TABLE` non è supportato per le sequenze.

Esempi

CREATE/DROP SEQUENCE

```
postgres_limitless=> CREATE SEQUENCE s;
CREATE SEQUENCE

postgres_limitless=> SELECT nextval('s');

 nextval
---------
       1
(1 row)

postgres_limitless=> SELECT * FROM pg_sequence WHERE seqrelid='s'::regclass;

 seqrelid | seqtypid | seqstart | seqincrement | seqmax | seqmin | seqcache |
 seqcycle
----------+----------+----------+--------------+--------+--------+----------
+----------
    16960 |       20 |        1 |            1 |  10000 |      1 |        1 | f
(1 row)

% connect to another router
postgres_limitless=> SELECT nextval('s');

 nextval
---------
   10001
(1 row)

postgres_limitless=> SELECT * FROM pg_sequence WHERE seqrelid='s'::regclass;
```

```
 seqrelid | seqtypid | seqstart | seqincrement | seqmax | seqmin | seqcache |
 seqcycle
----------+----------+----------+--------------+--------+--------+----------
+----------
    16959 |       20 |    10001 |            1 |  20000 |  10001 |        1 | f
(1 row)

postgres_limitless=> DROP SEQUENCE s;
DROP SEQUENCE
```

ALTER SEQUENCE

```
postgres_limitless=> CREATE SEQUENCE s;
CREATE SEQUENCE

postgres_limitless=> ALTER SEQUENCE s RESTART 500;
ALTER SEQUENCE

postgres_limitless=> SELECT nextval('s');

 nextval
---------
     500
(1 row)

postgres_limitless=> SELECT currval('s');

 currval
---------
     500
(1 row)
```

Funzioni di manipolazione di sequenze

```
postgres=# CREATE TABLE t(a bigint primary key, b bigint);
CREATE TABLE

postgres=# CREATE SEQUENCE s minvalue 0 START 0;
CREATE SEQUENCE

postgres=# INSERT INTO t VALUES (nextval('s'), currval('s'));

INSERT 0 1
```

```
postgres=# INSERT INTO t VALUES (nextval('s'), currval('s'));
INSERT 0 1

postgres=# SELECT * FROM t;

 a | b
---+---
 0 | 0
 1 | 1
(2 rows)

postgres=# ALTER SEQUENCE s RESTART 10000;
ALTER SEQUENCE

postgres=# INSERT INTO t VALUES (nextval('s'), currval('s'));

INSERT 0 1

postgres=# SELECT * FROM t;

   a   |   b
-------+-------
     0 |     0
     1 |     1
 10000 | 10000
(3 rows)
```

Viste delle sequenze

Aurora PostgreSQL Limitless Database offre le seguenti viste per le sequenze.

rds_aurora.limitless_distributed_sequence

Questa vista mostra lo stato e la configurazione della sequenza distribuita. Le colonne `minvalue`, `maxvalue`, `start`, `inc` e `cache` hanno lo stesso significato della vista pg_sequences e mostrano le opzioni con cui è stata creata la sequenza. La colonna `lastval` mostra l'ultimo valore allocato o riservato in un oggetto di sequenza distribuita. Ciò non significa che il valore sia già stato utilizzato, poiché i router mantengono localmente i blocchi di sequenza.

```
postgres_limitless=> SELECT * FROM rds_aurora.limitless_distributed_sequence WHERE
 sequence_name='test_serial_b_seq';

 schema_name |   sequence_name   | lastval | minvalue |  maxvalue  | start | inc |
 cache
-------------+-------------------+---------+----------+------------+-------+-----
+-------
 public      | test_serial_b_seq | 1250000 |        1 | 2147483647 |     1 |   1 |
   1
(1 row)
```

rds_aurora.limitless_sequence_metadata

Questa vista mostra i metadati della sequenza distribuita e aggrega i metadati della sequenza dai nodi del cluster. Usa le seguenti colonne:

- `subcluster_id`: l'ID del nodo del cluster che possiede un blocco.
- Blocco attivo: un blocco di una sequenza attualmente in uso (`active_minvalue`, `active_maxvalue`).
- Blocco riservato: il blocco locale che verrà utilizzato successivamente (`reserved_minvalue`, `reserved_maxvalue`).
- `local_last_value`: l'ultimo valore osservato da una sequenza locale.
- `chunk_size`: la dimensione di un blocco, come configurato alla creazione.

```
postgres_limitless=> SELECT * FROM rds_aurora.limitless_sequence_metadata WHERE
 sequence_name='test_serial_b_seq' order by subcluster_id;
```

```
 subcluster_id |   sequence_name   | schema_name | active_minvalue | active_maxvalue
 | reserved_minvalue | reserved_maxvalue | chunk_size | chunk_state |
 local_last_value
---------------+-------------------+-------------+-----------------
+-----------------+-------------------+-------------------+------------
+-------------+------------------
 1             | test_serial_b_seq | public      |          500001 |          750000
 |           1000001 |           1250000 |     250000 |           1 |
 550010
 2             | test_serial_b_seq | public      |          250001 |          500000
 |            750001 |           1000000 |     250000 |           1 |

(2 rows)
```

Risoluzione dei problemi delle sequenze

Possono verificarsi i seguenti problemi con le sequenze:

Dimensione del blocco non sufficientemente grande

Se la dimensione del blocco non è sufficientemente grande e il tasso di transazione è elevato, i worker in background potrebbero non avere tempo sufficiente per richiedere nuovi blocchi prima che i blocchi attivi siano esauriti. Questo può causare eventi di contesa e attesa come `LIMITLESS:AuroraLimitlessSequenceReplace`, `LWLock:LockManager`, `Lockrelation` e `LWlock:bufferscontent`.

Aumenta il valore del parametro `rds_aurora.limitless_sequence_chunk_size`.

Cache della sequenza impostata troppo alta

In PostgreSQL, la memorizzazione nella cache delle sequenze avviene a livello di sessione. Ogni sessione alloca valori di sequenza successivi durante un accesso all'oggetto di sequenza e aumenta di conseguenza quelli dell'oggetto di sequenza di `last_value`. Quindi, gli usi successivi di `nextval` nella stessa sessione restituiscono semplicemente i valori preallocati senza interessare l'oggetto di sequenza.

Tutti i numeri allocati ma non utilizzati all'interno di una sessione vengono persi al termine di tale sessione, con conseguenti "buchi" nella sequenza. Ciò può consumare rapidamente sequence_chunk e causare eventi di contesa e attesa come `LIMITLESS:AuroraLimitlessSequenceReplace`, `LWLock:LockManager`, `Lockrelation` e `LWlock:bufferscontent`.

Riduci l'impostazione della cache della sequenza.

La figura seguente mostra gli eventi di attesa causati da problemi di sequenza.

Comandi SQL supportati e non supportati per Data Manipulation Language (DML) e per l'elaborazione delle query

La tabella seguente elenca i comandi DML supportati e non supportati da Aurora PostgreSQL Limitless Database, con riferimenti a limitazioni o ulteriori informazioni.

Comando	Supportato?	Limitazioni o ulteriori informazioni
ABORT	Sì	Nessuno
ANALYZE	Sì	ANALYZE
BEGIN	Sì	Nessuno
CALL	Sì	Nessuno
CHECKPOINT	Sì	Nessuno
CLOSE	Sì	Nessuno
CLUSTER	Sì	CLUSTER
COMMIT	Sì	Nessuno
COMMIT PREPARED	No	Non applicabile
COPY	Sì	Nessuno
DEALLOCATE	Sì	Nessuno
DECLARE	Sì	Nessuno
DELETE	Sì	Nessuno
DISCARD	Sì	Nessuno
DO	Sì	Nessuno
END	Sì	Nessuno

Comando	Supportato?	Limitazioni o ulteriori informazioni
EXECUTE	Sì	Nessuno
EXPLAIN	Sì	EXPLAIN
FETCH	Sì	Nessuno
IMPORT FOREIGN SCHEMA	No	Non applicabile
INSERT	Sì	INSERT
LISTEN	No	Non applicabile
LOCK	Sì	Nessuno
MERGE	No	Non applicabile
MOVE	Sì	Nessuno
NOTIFY	No	Non applicabile
OPEN	Sì	Nessuno
PREPARE	Sì	Nessuno
PREPARE TRANSACTION	No	Non applicabile
REFRESH MATERIALIZED VIEW	No	Non applicabile
REINDEX	No	Non applicabile
RELEASE SAVEPOINT	Sì	Nessuno
ROLLBACK	Sì	Nessuno
ROLLBACK PREPARED	No	Non applicabile
ROLLBACK TO SAVEPOINT	Sì	Nessuno

Comando	Supportato?	Limitazioni o ulteriori informazioni
SAVEPOINT	Sì	Nessuno
SELECT	Sì	Nessuno
SELECT INTO	Sì	Nessuno
MOSTRA	Sì	Nessuno
START TRANSACTION	Sì	Nessuno
UNLISTEN	No	Nessuno
UPDATE	Sì	UPDATE
UPDATE … WHERE CURRENT OF	No	Non applicabile
VACUUM	Sì	VACUUM
VALUES	Sì	Nessuno

Limitazioni DML e altre informazioni per Aurora PostgreSQL Limitless Database

I seguenti argomenti descrivono le limitazioni o forniscono ulteriori informazioni per i comandi DML e di elaborazione delle query SQL in Aurora PostgreSQL Limitless Database.

Argomenti

- ANALYZE
- CLUSTER
- EXPLAIN
- INSERT
- UPDATE
- VACUUM

ANALYZE

Il comando `ANALYZE` raccoglie statistiche sul contenuto delle tabelle nel database. Successivamente, il pianificatore di query utilizza queste statistiche per determinare i piani di esecuzione più efficienti per le query. Per ulteriori informazioni, consulta ANALYZE nella documentazione di PostgreSQL.

In Aurora PostgreSQL Limitless Database, il comando `ANALYZE` raccoglie le statistiche delle tabelle su tutti i router e gli shard durante l'esecuzione.

Per impedire il calcolo delle statistiche su ogni router durante le esecuzioni di `ANALYZE`, le statistiche delle tabelle vengono calcolate su uno dei router e quindi copiate sui router peer.

CLUSTER

Il comando `CLUSTER` riordina fisicamente una tabella in base a un indice. L'indice deve essere già stato definito nella tabella. In Aurora PostgreSQL Limitless Database, il clustering è locale alla parte dell'indice presente su ogni shard.

Per ulteriori informazioni, consulta CLUSTER nella documentazione di PostgreSQL.

EXPLAIN

Si utilizza il seguente parametro per configurare l'output del comando EXPLAIN:

- `rds_aurora.limitless_explain_options`: cosa includere nell'output EXPLAIN. Il valore predefinito è `single_shard_optimization`: viene mostrato se i piani sono ottimizzati per shard singolo, ma i piani shard non sono inclusi.

In questo esempio, l'output EXPLAIN non mostra i piani ricavati dagli shard.

```
postgres_limitless=> EXPLAIN SELECT * FROM employees where id =25;

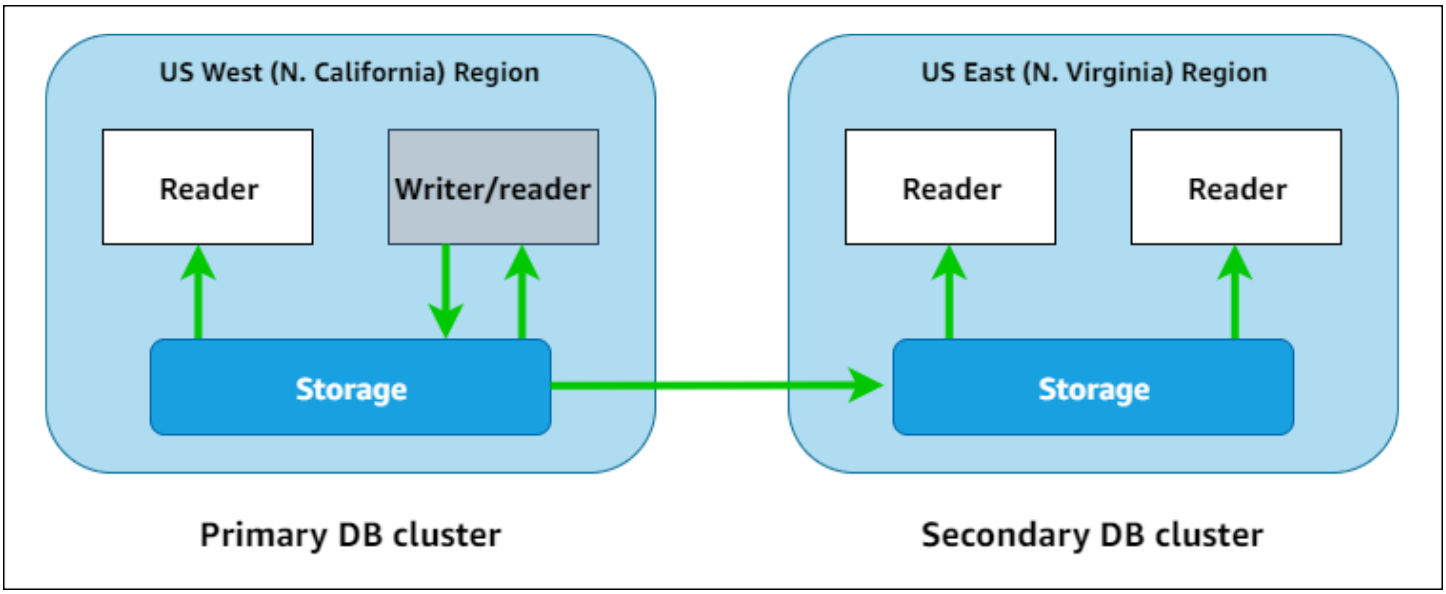
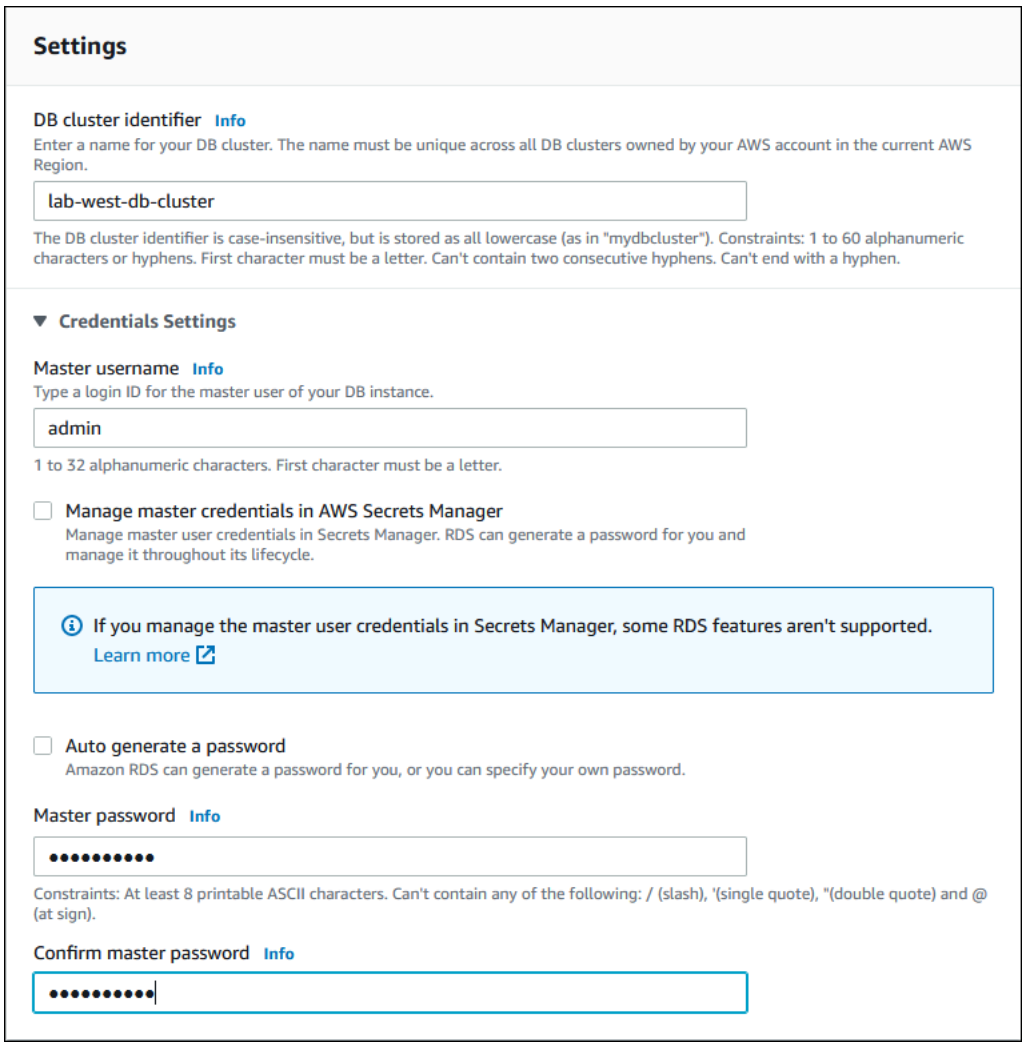
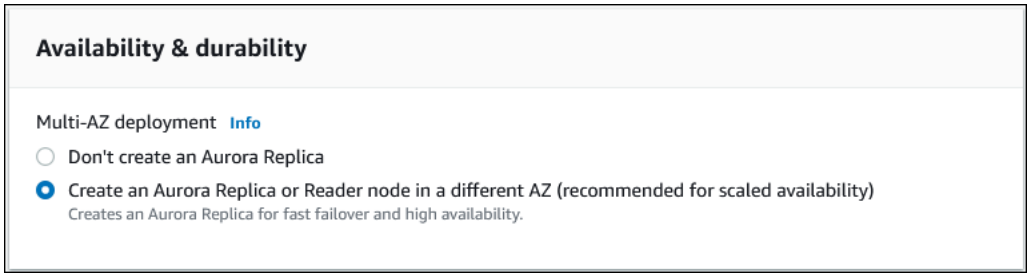
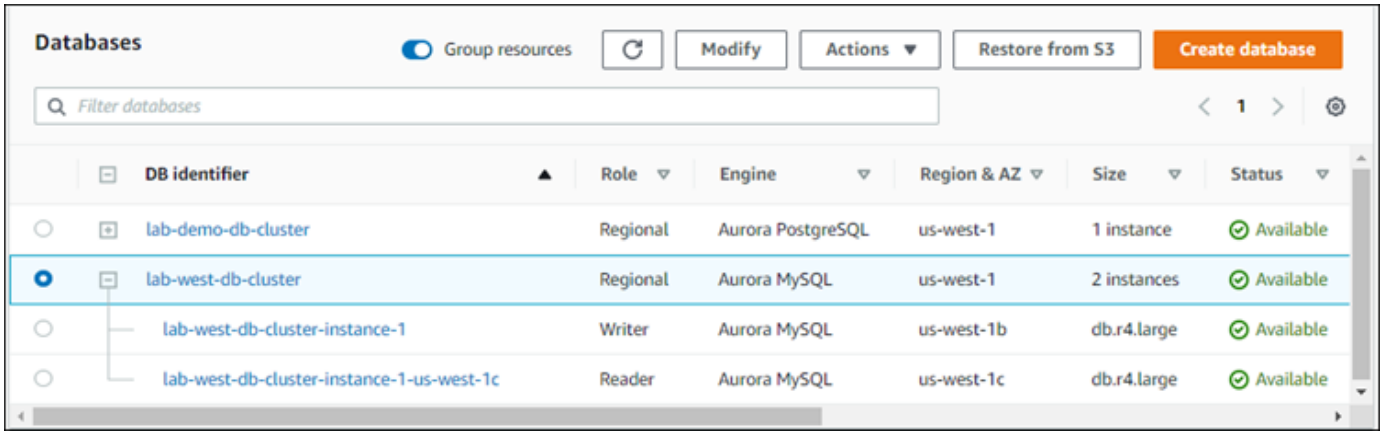
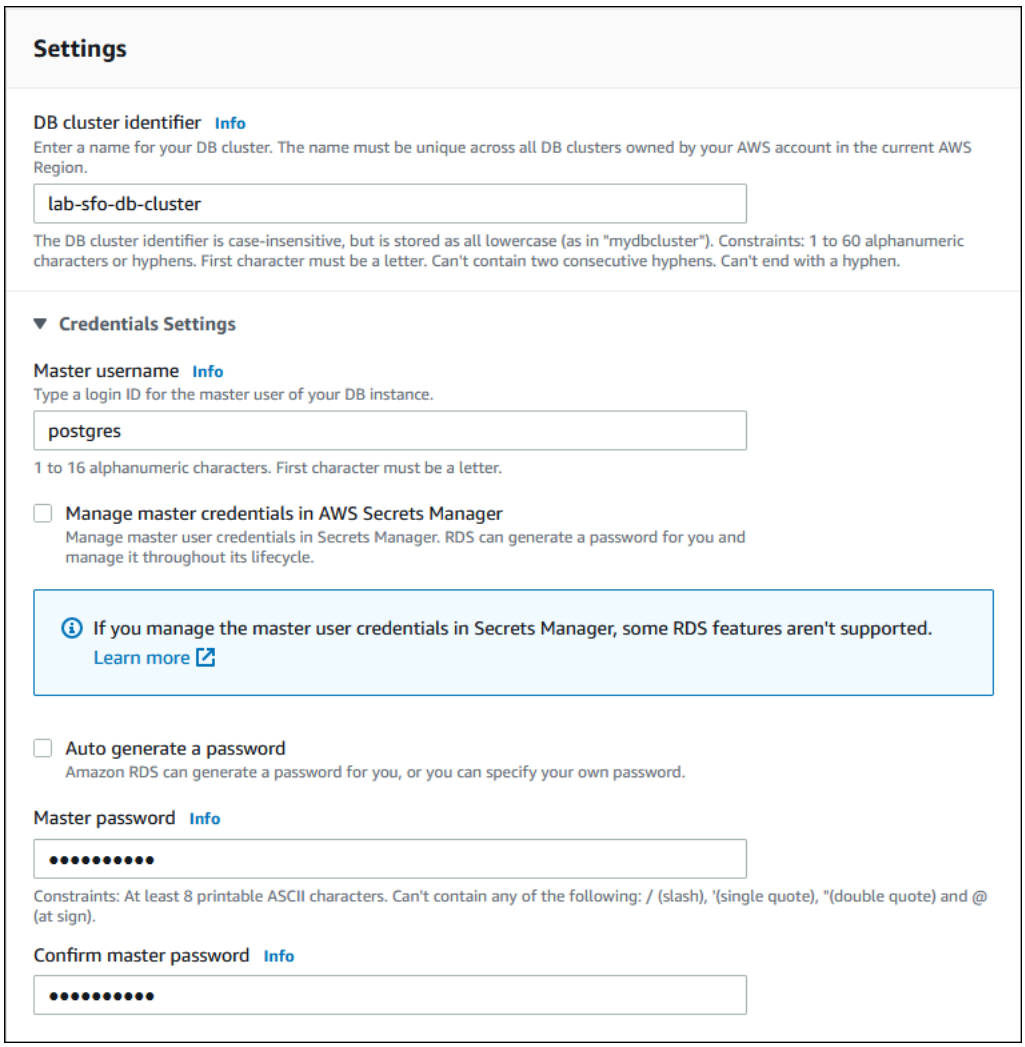
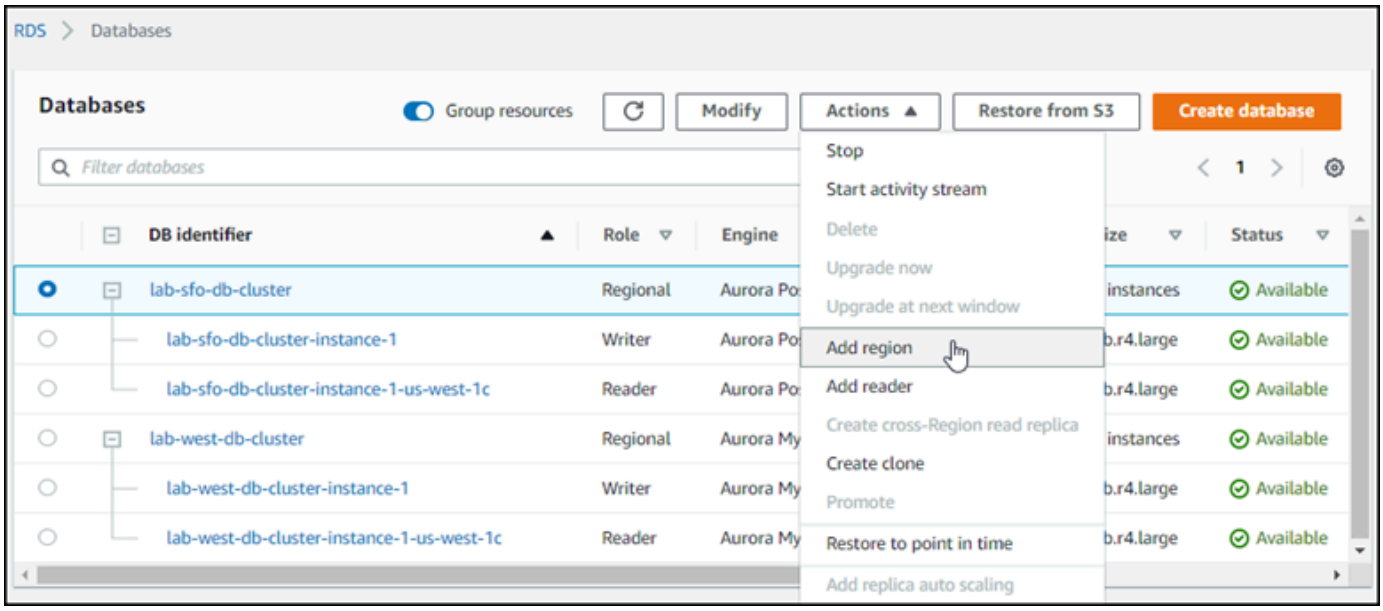
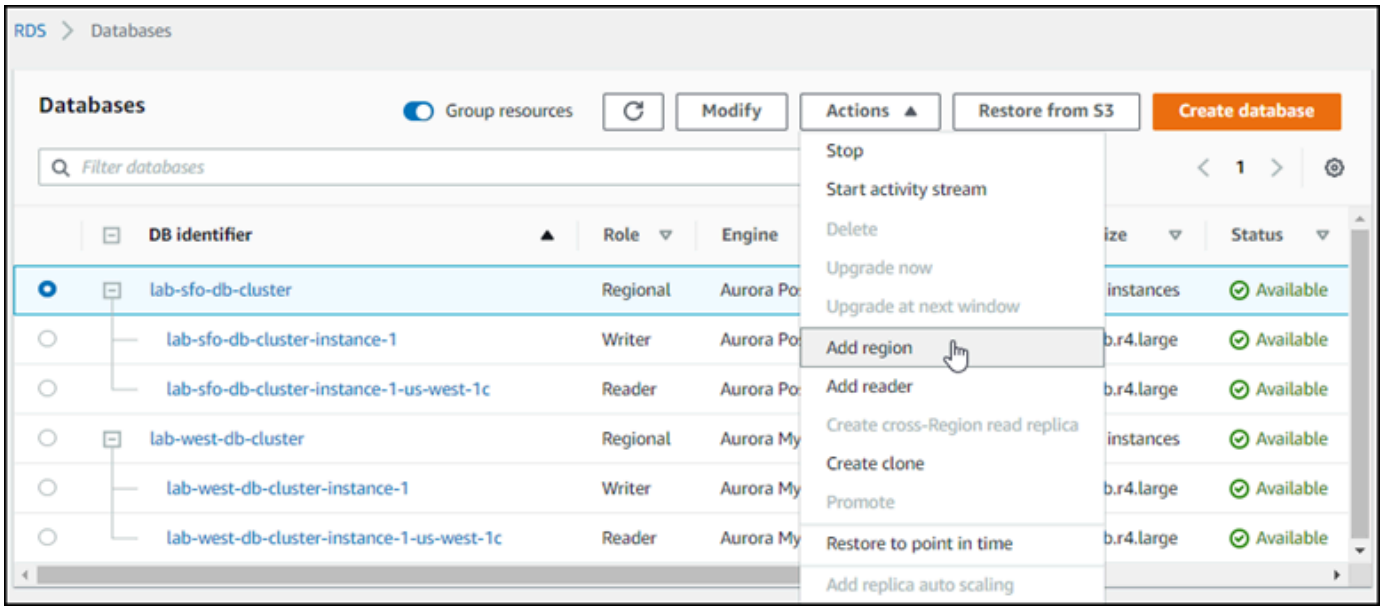
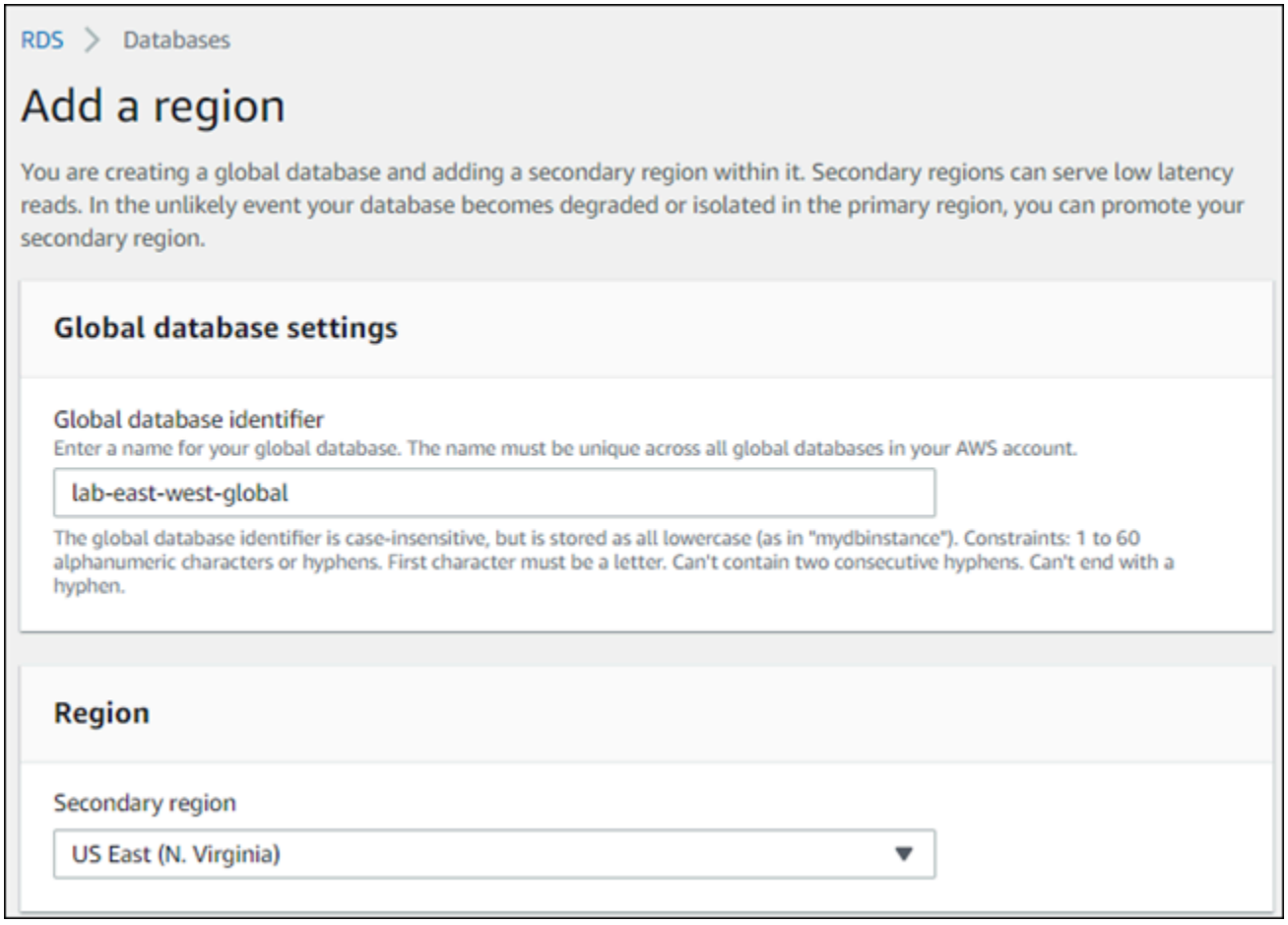
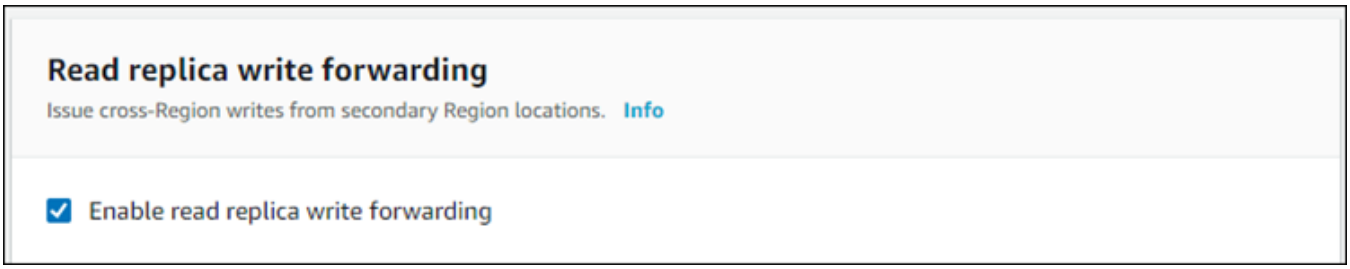
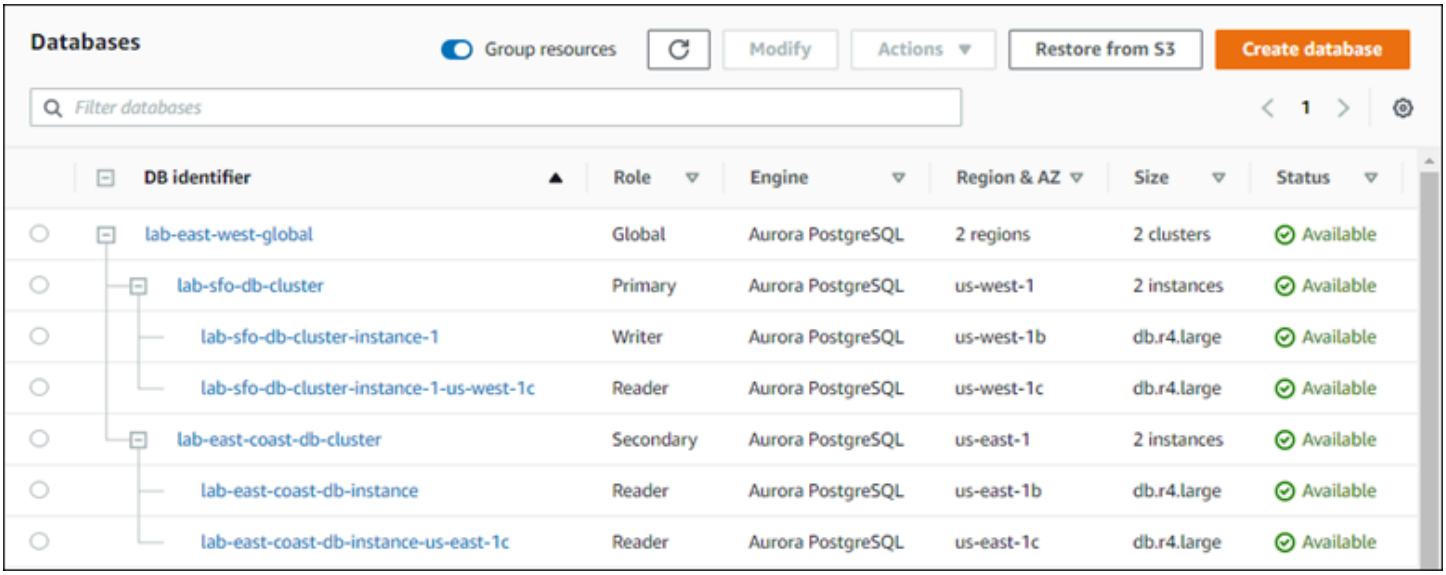
                      QUERY PLAN
------------------------------------------------------
 Foreign Scan  (cost=100.00..101.00 rows=100 width=0)
 Single Shard Optimized
(2 rows)
```

Ora impostiamo l'opzione `rds_aurora.limitless_explain_options` per includere `shard_plans` e`single_shard_optimization`. Possiamo visualizzare i piani di esecuzione delle istruzioni sia sui router che sugli shard. Inoltre, disabilitiamo il parametro `enable_seqscan` per far sì che la scansione dell'indice venga utilizzata sul livello shard.

```
postgres_limitless=> SET rds_aurora.limitless_explain_options = shard_plans,
 single_shard_optimization;
SET

postgres_limitless=> SET enable_seqscan = OFF;
SET

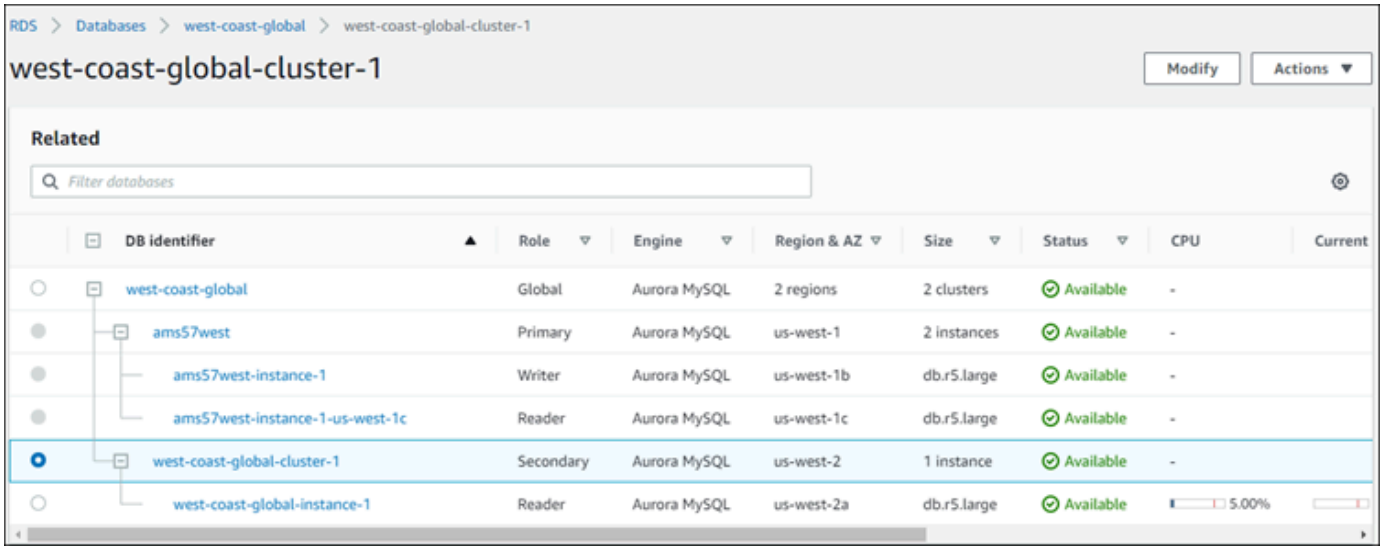
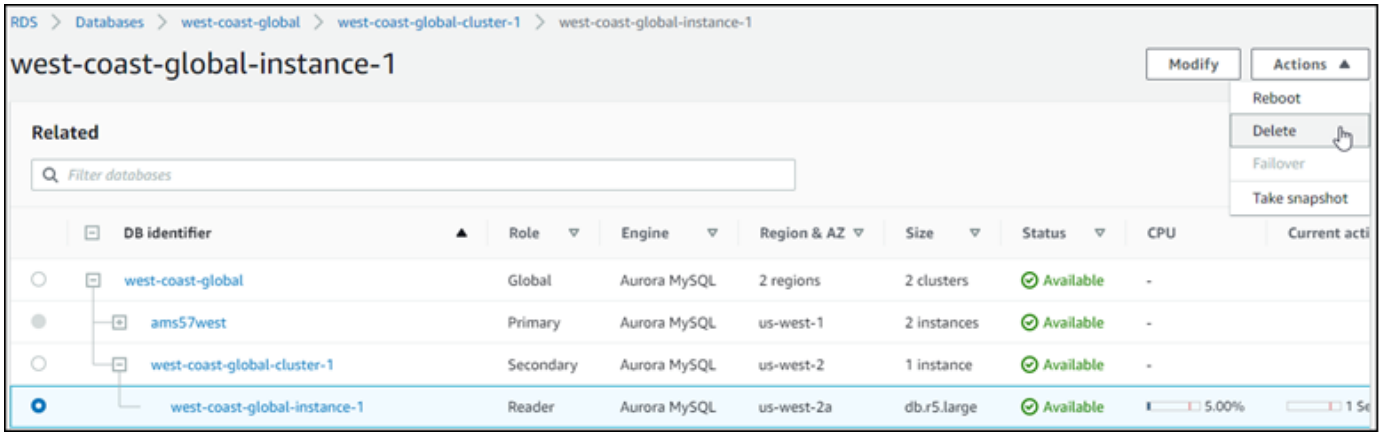
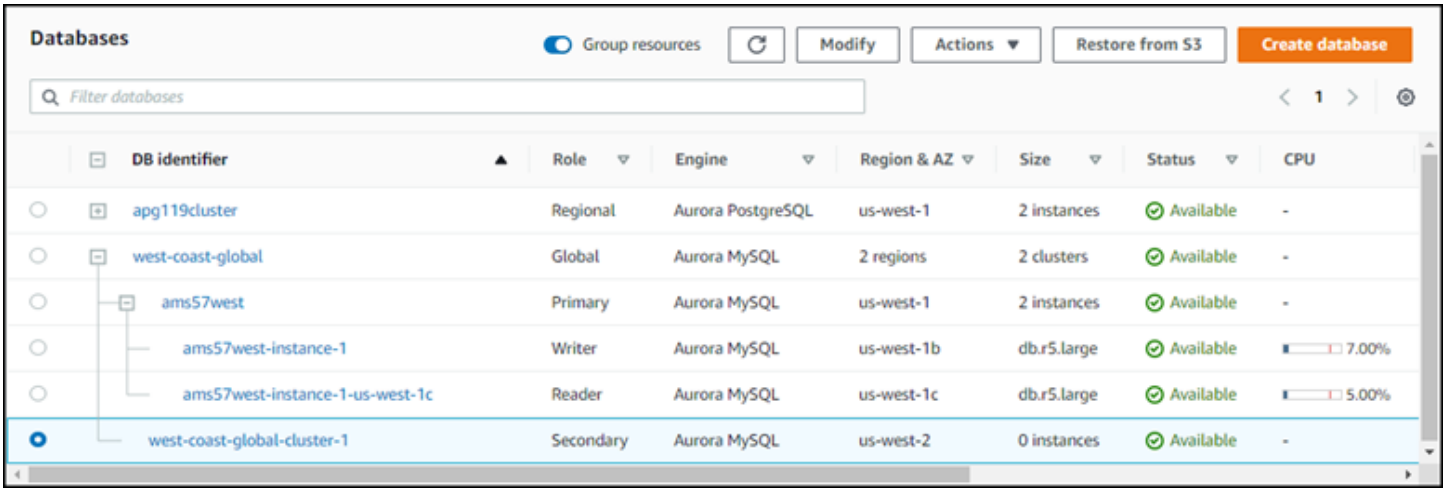
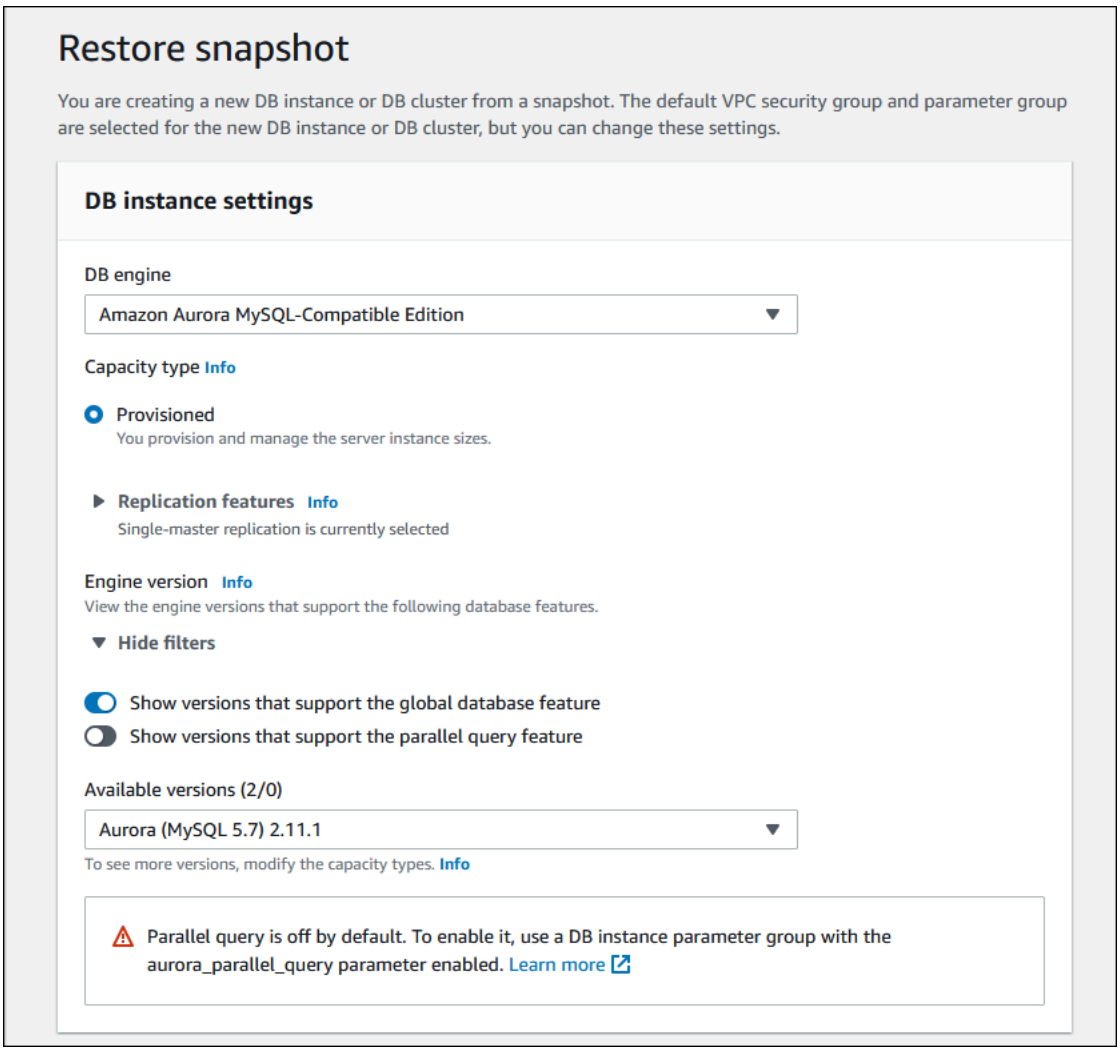
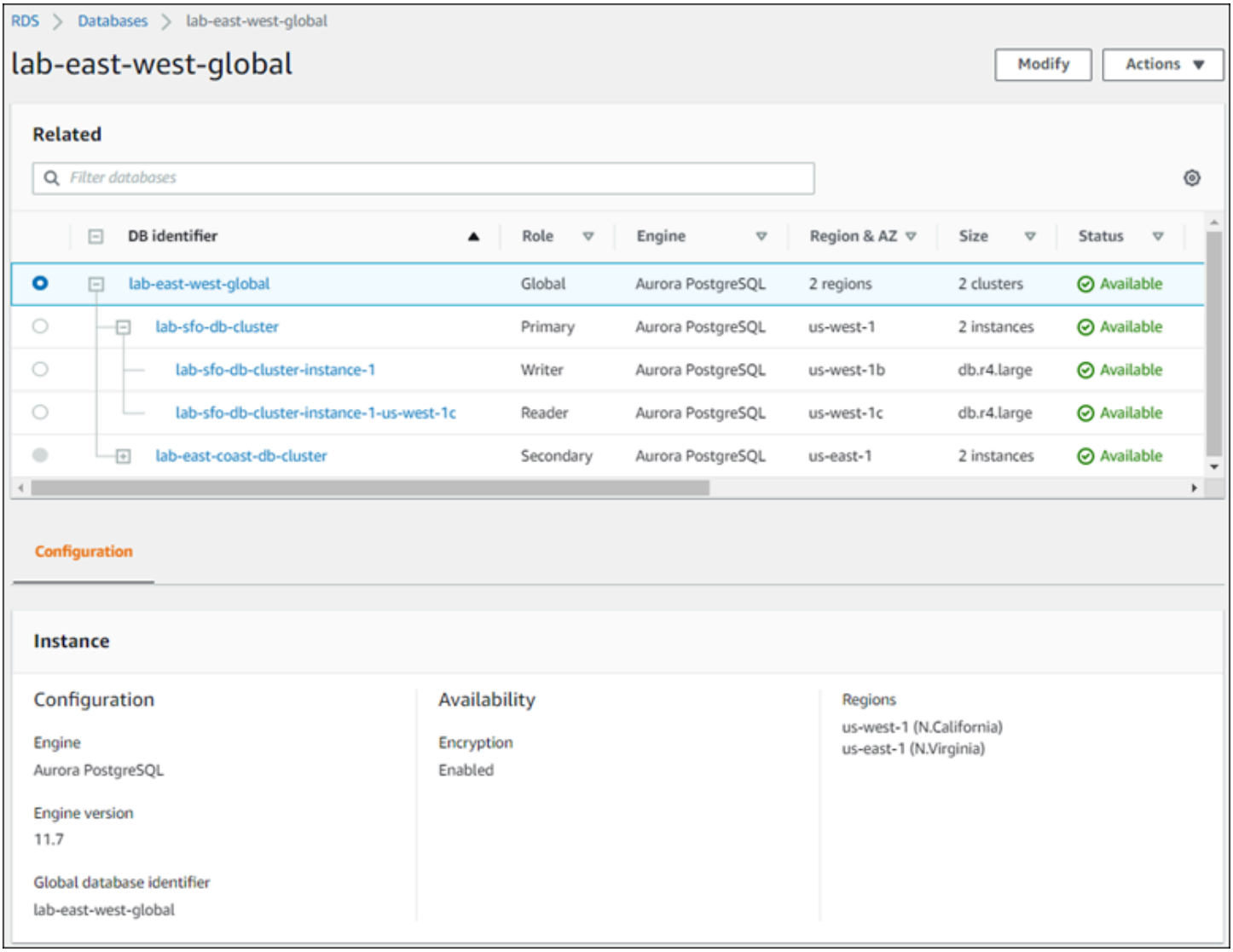
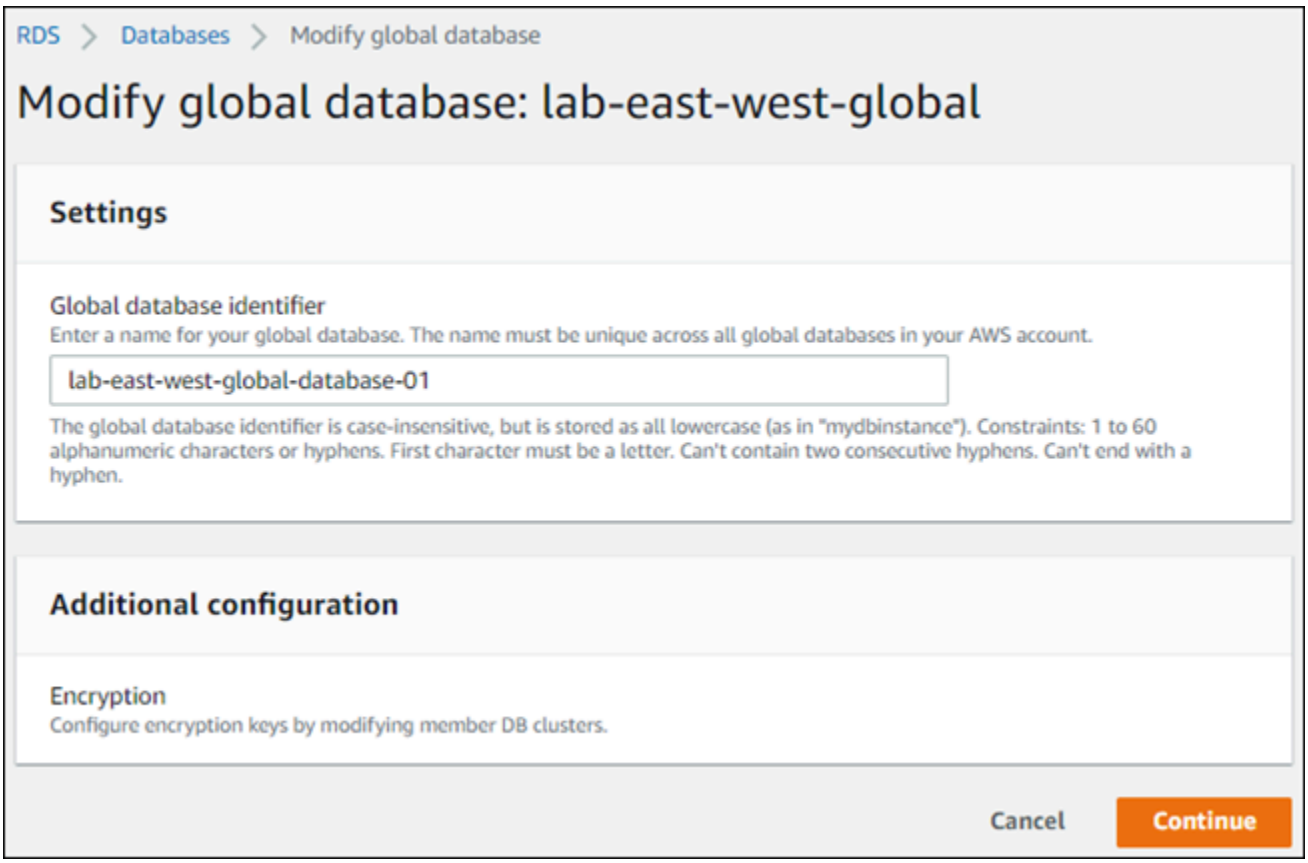
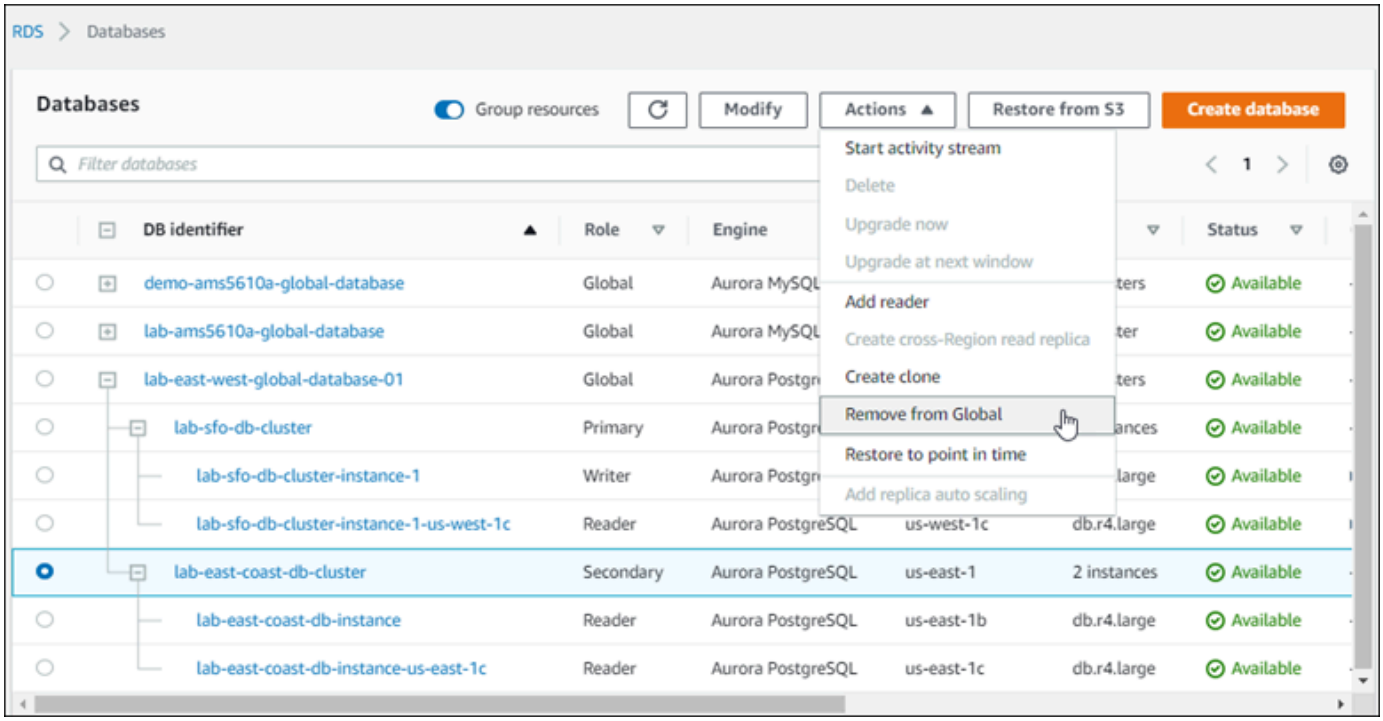
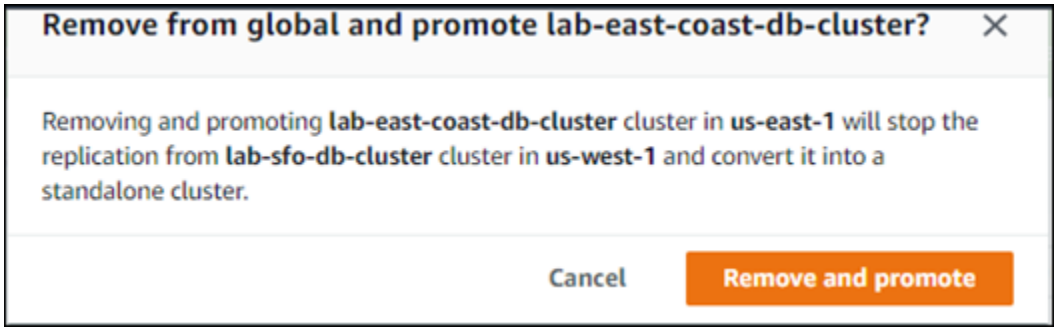
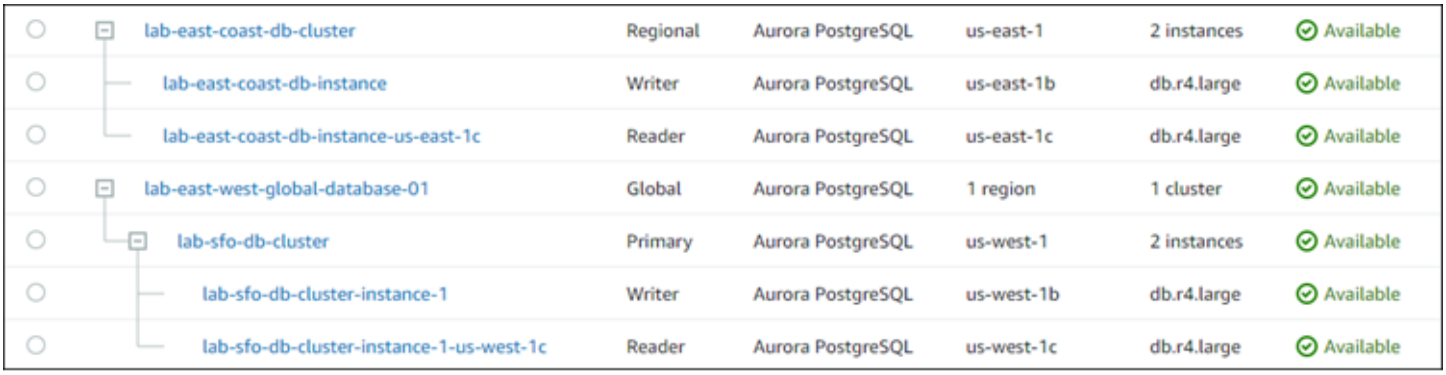
postgres_limitless=> EXPLAIN SELECT * FROM employees WHERE id = 25;

                                                    QUERY PLAN
-------------------------------------------------------------------------------------------------------
 Foreign Scan  (cost=100.00..101.00 rows=100 width=0)
   Remote Plans from Shard postgres_s4:
         Index Scan using employees_ts00287_id_idx on employees_ts00287
 employees_fs00003  (cost=0.14..8.16 rows=1 width=15)
           Index Cond: (id = 25)
 Single Shard Optimized
(5 rows)
```

Per ulteriori informazioni sul comando EXPLAIN, consulta EXPLAIN nella documentazione di PostgreSQL.

INSERT

La maggior parte dei comandi INSERT è supportata in Aurora PostgreSQL Limitless Database.

PostgreSQL non ha un comando UPSERT esplicito, ma supporta le istruzioni INSERT ... ON CONFLICT.

INSERT ... ON CONFLICT non è supportato se l'azione di conflitto ha una sottoquery o una funzione mutabile:

```
-- RANDOM is a mutable function.
INSERT INTO sharded_table VALUES (1, 100) ON CONFLICT (id) DO UPDATE SET other_id =
 RANDOM();

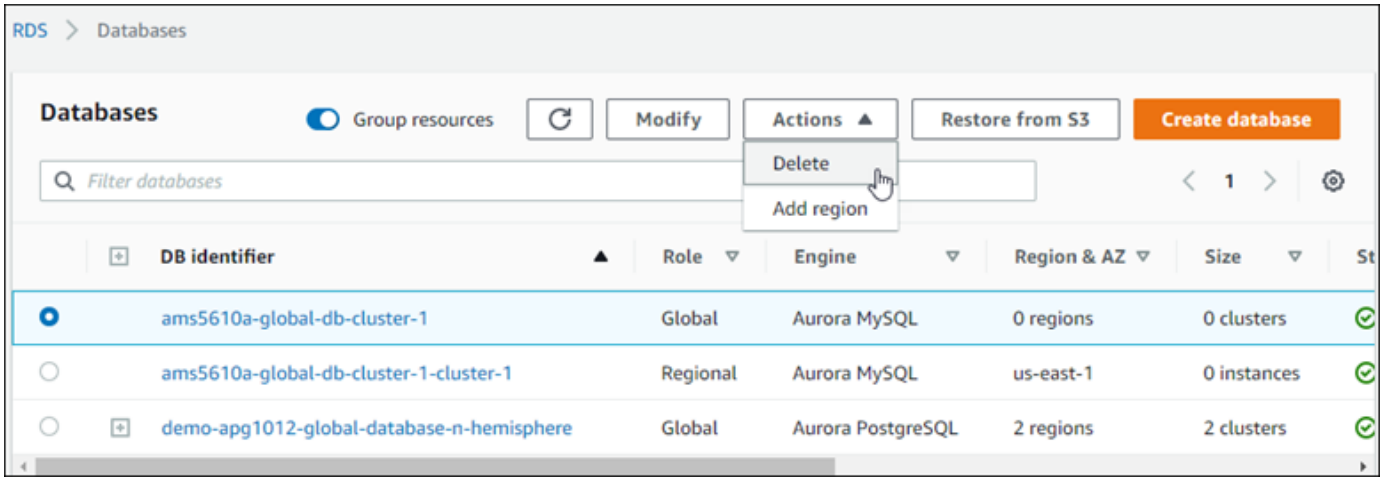
ERROR: Aurora Limitless Tables doesn't support pushdown-unsafe functions with DO UPDATE
 clauses.
```

Per ulteriori informazioni sul comando INSERT, consulta INSERT nella documentazione di PostgreSQL.

UPDATE

L'aggiornamento della chiave shard non è supportato. Ad esempio, si dispone di una tabella sottoposta a sharding denominata customers, con una chiave shard customer_id. Le seguenti istruzioni DML causano errori:

```
postgres_limitless=> UPDATE customers SET customer_id = 11 WHERE customer_id =1;
ERROR:  Shard key column update is not supported

postgres_limitless=> UPDATE customers SET customer_id = 11 WHERE customer_name='abc';
ERROR:  Shard key column update is not supported
```

Per aggiornare una chiave shard, è necessario prima DELETE la riga con la chiave shard key, quindi INSERT una nuova riga con il valore della chiave shard aggiornato.

Per ulteriori informazioni sul comando UPDATE, consulta Updating data nella documentazione di PostgreSQL.

VACUUM

È possibile eseguire l'operazione di vacuum sia sulle tabelle sottoposte a sharding che su quelle di riferimento. Le seguenti funzioni `VACUUM` sono completamente supportate in Aurora PostgreSQL Limitless Database.

- VACUUM
- ANALYZE
- DISABLE_PAGE_SKIPPING
- FREEZE
- FULL
- INDEX_CLEANUP
- PARALLEL
- PROCESS_TOAST
- TRUNCATE
- VERBOSE

`VACUUM` su Aurora PostgreSQL Limitless Database presenta le seguenti limitazioni:

- L'estensione pg_visibility_map non è supportata.
- Il controllo degli indici non utilizzati con la vista pg_stat_all_indexes non è supportato.
- Le viste consolidate per pg_stat_user_indexes, pg_class e pg_stats non sono implementate.

Per ulteriori informazioni sul comando `VACUUM`, consulta VACUUM nella documentazione di PostgreSQL. Per ulteriori informazioni su come funziona l'operazione di vacuum in Aurora PostgreSQL Limitless Database, consulta Recupero dello spazio di archiviazione tramite il processo di vacuum.

Variabili in Aurora PostgreSQL Limitless Database

È possibile utilizzare le seguenti variabili per configurare Aurora PostgreSQL Limitless Database.

rds_aurora.limitless_active_shard_key

Imposta una singola chiave shard durante l'esecuzione di query sul database, facendo sì che tutte le query `SELECT` e DML vengano aggiunte con la chiave shard come predicato costante. Per ulteriori informazioni, consulta Impostazione di una chiave di shard attiva.

rds_aurora.limitless_create_table_collocate_with

Imposta questa variabile su un nome di tabella specifico per collocare le tabelle appena create con quella tabella. Per ulteriori informazioni, consulta Creazione di tabelle Limitless utilizzando variabili.

rds_aurora.limitless_create_table_mode

Imposta la modalità di creazione della tabella. Per ulteriori informazioni, consulta Creazione di tabelle Limitless utilizzando variabili.

rds_aurora.limitless_create_table_shard_key

Imposta questa variabile su una matrice di nomi di colonna da utilizzare come chiavi shard. Per ulteriori informazioni, consulta Creazione di tabelle Limitless utilizzando variabili.

rds_aurora.limitless_explain_options

Cosa includere nell'output `EXPLAIN`. Per ulteriori informazioni, consulta EXPLAIN.

Parametri del cluster di database in Aurora PostgreSQL Limitless Database

È possibile utilizzare i seguenti parametri del cluster di database per configurare Aurora PostgreSQL Limitless Database.

rds_aurora.limitless_adaptive_fetch_size

Migliora il prefetching in batch. Se impostato su `true`, questo parametro consente una dimensione di recupero autoregolante (adattativa) per il prefetching. Se impostato su `false`, la dimensione di recupero è costante.

rds_aurora.limitless_auto_scale_options

Imposta le opzioni disponibili per aggiungere router o dividere gli shard in un gruppo di shard del database. Il valore può essere `add_router`, `split_shard` o entrambi.

Per ulteriori informazioni, consultare Aggiunta di un router a un gruppo di shard del database e Suddivisione di uno shard in un gruppo di shard di database.

rds_aurora.limitless_distributed_deadlock_timeout

Il periodo di tempo di attesa su un lock prima di verificare se esiste una condizione di deadlock distribuito, in millisecondi. Il valore predefinito è `1000` (1 secondo).

Per ulteriori informazioni, consulta Deadlock distribuiti in Aurora PostgreSQL Limitless Database.

rds_aurora.limitless_enable_auto_scale

Consente l'aggiunta di router e la suddivisione degli shard in un gruppo di shard del database.

Per ulteriori informazioni, consultare Aggiunta di un router a un gruppo di shard del database e Suddivisione di uno shard in un gruppo di shard di database.

rds_aurora.limitless_finalize_split_shard_mode

Determina come vengono finalizzate le suddivisioni degli shard avviate dal sistema. Per ulteriori informazioni, consulta Suddivisione di uno shard in un gruppo di shard di database.

rds_aurora.limitless_maximum_adaptive_fetch_size

Imposta il limite superiore per la dimensione di recupero adattivo. L'intervallo è `1–INT_MAX`. Il valore predefinito è `1000`.

Utilizzo di Database globale Amazon Aurora

Con la funzionalità Database globale Amazon Aurora, puoi configurare più cluster di database Aurora che si estendono su più cluster. Regioni AWS Aurora sincronizza automaticamente tutte le modifiche apportate nel cluster di database primario con uno o più cluster secondari. Un database globale Aurora ha un cluster di database primario in una Regione e fino a dieci cluster di database secondari in Regioni differenti. Questa configurazione multi-Regione consente un ripristino di emergenza rapido in caso di eventuali interruzioni che potrebbero interessare un'intera Regione AWS. La disponibilità di una copia completa di tutti i dati in più aree geografiche consente inoltre operazioni di lettura a bassa latenza per le applicazioni che si connettono da località molto diverse in tutto il mondo.

Argomenti

- Panoramica di Database globale Amazon Aurora
- Vantaggi di Database globale Amazon Aurora
- Disponibilità di regioni e versioni
- Limitazioni di Database globale Amazon Aurora
- Nozioni di base sul Database globale Amazon Aurora
- Gestione di un database globale Amazon Aurora
- Connessione a Database globale Amazon Aurora
- Utilizzo dell'inoltro di scrittura in un database globale Amazon Aurora
- Utilizzo dello switchover o failover in un Database globale Amazon Aurora
- Monitoraggio di un database globale Amazon Aurora
- Utilizzo Blue/Green delle distribuzioni per Amazon Aurora Global Database
- Utilizzo di database globali Amazon Aurora con altri servizi AWS
- Aggiornamento di un database globale Amazon Aurora

Panoramica di Database globale Amazon Aurora

Utilizzando la funzionalità Database globale Amazon Aurora, puoi eseguire le applicazioni distribuite globalmente utilizzando un singolo database Aurora che si sviluppa su più Regioni AWS.

Un database globale Aurora è costituito da un database primario Regione AWS in cui vengono scritti i dati e fino a 10 secondari di sola lettura. Regioni AWS Emetti operazioni di scrittura al cluster

di database primario nella Regione AWS primaria. Il modo più comodo per farlo è connettersi all'endpoint di scrittura del database globale Aurora, che punta sempre al cluster di database primario, anche dopo uno switchover o un failover verso un altro Regione AWS. Dopo ogni operazione di scrittura, Aurora replica i dati sul secondario Regioni AWS utilizzando un'infrastruttura dedicata, con una latenza in genere inferiore a un secondo.

Nel diagramma seguente, è possibile trovare un esempio di database globale Aurora che si estende su due. Regioni AWS



Puoi aumentare verticalmente le dimensioni del cluster secondario in maniera indipendente aggiungendo una o più istanze di lettura Aurora per servire carichi di lavoro di sola lettura. È possibile utilizzare Aurora Serverless v2 per le istanze di lettura per un dimensionamento ancora più granulare e flessibile.

Solo il cluster primario eseguire operazioni di scrittura. I client che eseguono operazioni di scrittura si connettono all'endpoint di scrittura del database globale Aurora, che punta sempre all'istanza di scrittura di database, che punta sempre all'istanza database del cluster di database. Come illustrato nel diagramma, Aurora utilizza il volume di archiviazione del cluster e non il motore di database per una replica rapida e con sovraccarico ridotto. Per ulteriori informazioni, consulta Panoramica dell'archiviazione di Amazon Aurora.

Il database globale Aurora è progettato per le applicazioni con una presenza globale. I cluster DB secondari multipli di sola lettura Regioni AWS aiutano a ottimizzare le operazioni di lettura più vicine agli utenti delle applicazioni. Attraverso la funzionalità di inoltro di scrittura, è possibile anche configurare un database globale in modo che i cluster secondari inviino i dati le richieste di scrittura

al primario. Per ulteriori informazioni, consulta Utilizzo dell'inoltro di scrittura in un database globale Amazon Aurora.

Un database globale Aurora supporta due diverse operazioni per modificare la Regione del cluster di database primario, a seconda dello scenario: switchover del Database globale Aurora e failover globale del database Aurora.

- Per le procedure operative pianificate come la rotazione delle Regioni, utilizza il meccanismo di switchover (precedentemente chiamato “failover pianificato gestito”). Con questa funzionalità, puoi rilocare il cluster primario di un Database globale Aurora integro in una delle Regioni secondarie senza alcuna perdita di dati. Per ulteriori informazioni, consulta Esecuzione di switchover per database globali Amazon Aurora.
- Per ripristinare il database globale Aurora dopo un’interruzione nella Regione primaria, utilizza il meccanismo di failover. Con questa funzionalità, esegui il failover del cluster di database primario in un’altra Regione (failover tra Regioni). Per ulteriori informazioni, consulta Esecuzione di failover gestiti per database globali Aurora.

Vantaggi di Database globale Amazon Aurora

Utilizzando Database globale Aurora, è possibile ottenere i seguenti vantaggi:

- Letture globali con latenza locale: le aziende che hanno uffici in tutto il mondo possono utilizzare Database globale Aurora per mantenere aggiornate le principali fonti di informazioni nella Regione AWS primaria. Gli uffici nelle altre regioni possono accedere alle informazioni nella propria regione, con una latenza locale.
- Cluster di database Aurora secondari scalabili: è possibile dimensionare i cluster secondari aggiungendo più istanze di sola lettura a una Regione AWS secondaria. Il cluster secondario è di sola lettura, quindi può supportare fino a 16 istanze database di sola lettura, anziché il limite abituale di 15 per un singolo cluster Aurora.
- Replica rapida dai cluster di database Aurora primari a quelli secondari – La replica eseguita da un database globale Aurora ha un impatto ridotto sulle prestazioni del cluster di database primario. Le risorse delle istanze database sono totalmente dedicate a servire carichi di lavoro di lettura e scrittura delle applicazioni.
- Ripristino da interruzioni a livello regionale: i cluster secondari consentono di creare un database globale Aurora disponibile in una nuova Regione AWS primaria più rapidamente (RTO inferiore) e con minore perdita di dati (RPO inferiore) rispetto alle soluzioni di replica tradizionali.

Disponibilità di regioni e versioni

Il supporto e la disponibilità di questa funzionalità variano a seconda delle versioni specifiche di ciascun motore di database Aurora e tra Regioni AWS. Per ulteriori informazioni sulla disponibilità di versioni e Regioni con database globali Aurora, consulta Regioni e motori di database supportati per i database globali Aurora.

Limitazioni di Database globale Amazon Aurora

Le seguenti limitazioni si applicano attualmente ai database globali Aurora:

- Aurora Global Database è disponibile in alcune Regioni AWS e per versioni specifiche di Aurora MySQL e Aurora PostgreSQL. Per ulteriori informazioni, consulta Regioni e motori di database supportati per i database globali Aurora.
- I database globali Aurora hanno determinati requisiti di configurazione per le classi di istanza database Aurora supportate, il numero massimo di Regioni AWS e così via. Per ulteriori informazioni, consulta Requisiti di configurazione di un database globale Amazon Aurora.
- Per Aurora MySQL con compatibilità MySQL 5.7, gli switchover di Database globale Aurora richiedono Aurora versione 2.09.1 o una versione secondaria superiore.
- È possibile eseguire uno switchover o un failover gestito tra Regioni su un database globale Aurora solo se i cluster di database primario e secondario hanno la stessa versione principale e secondaria e le stesse versioni di motore. A seconda del motore e delle versioni del motore, può essere necessario che i livelli di patch siano identici oppure possono essere diversi. Per un elenco dei motori e delle versioni dei motori che consentono queste operazioni tra cluster primari e secondari con diversi livelli di patch, consulta Compatibilità del livello di patch per switchover e failover gestiti tra regioni. Se le versioni del motore richiedono identici livelli di patch, puoi eseguire il failover manualmente seguendo i passaggi indicati in Esecuzione di failover manuali per i Database globali Aurora.
- Database globale Aurora attualmente non supporta le seguenti funzionalità di Aurora:
  - Aurora Serverless v1
  - Backtrack in Aurora
- Per le limitazioni all'utilizzo della funzionalità Server proxy per RDS con i database globali Aurora, consulta Limitazioni di Server proxy per RDS con i database globali.
- L'aggiornamento automatico della versione secondaria non si applica ai cluster Aurora MySQL e Aurora PostgreSQL che fanno parte di un database globale. Si noti che questa impostazione

può essere specificata per un'istanza database che fa parte di un cluster di database globale, ma l'impostazione non ha effetto.

- Database globale Aurora attualmente non supporta il dimensionamento automatico per i cluster di database secondari.
- Per utilizzare Database Activity Streams (DAS) su Database globale Aurora che esegue Aurora MySQL 5.7, la versione del motore deve essere la versione 2.08 o successiva. Per informazioni su DAS, consulta Monitoraggio di Amazon Aurora tramite i flussi di attività del database.
- Le seguenti limitazioni si applicano attualmente a Database globale Aurora:
  - Non è possibile applicare un gruppo di parametri personalizzato al cluster di database globale mentre si esegue un aggiornamento della versione principale del database globale Aurora. È possibile creare i gruppi di parametri personalizzati in ciascuna Regione del cluster globale e applicarli manualmente ai cluster regionali dopo l'aggiornamento.
  - Con un database globale Aurora basato su Aurora MySQL, non puoi eseguire un aggiornamento locale da Aurora MySQL versione 2 alla versione 3 se il parametro `lower_case_table_names` è attivato. Per ulteriori informazioni sui metodi disponibili all'uso, consulta Aggiornamenti di una versione principale.
  - Con Database globale Aurora basato su Aurora PostgreSQL, non è possibile eseguire un aggiornamento della versione principale del motore Aurora PostgreSQL DB se la funzionalità Obiettivo del punto di ripristino (RPO) è attivata. Per ulteriori informazioni sulla caratteristica RPO, consulta Gestione degli RPO per database globali basati su Aurora PostgreSQL–.
  - Con Database globale Aurora basato su Aurora MySQL, non è possibile eseguire un aggiornamento della versione secondaria dalla versione 3.01 o 3.02 alla versione 3.03 o successiva di Aurora MySQL utilizzando il processo standard. Per informazioni dettagliate sul processo da usare, consulta Aggiornamento di Aurora MySQL modificando la versione del motore.

  Per informazioni sull'aggiornamento di Database globale Aurora, consulta Aggiornamento di un database globale Amazon Aurora.
- Non è possibile interrompere o avviare i cluster di database Aurora nel database globale in modo individuale. Per ulteriori informazioni, consulta Avvio e arresto di un cluster di database Amazon Aurora.
- Le istanze database di lettura Aurora collegate al cluster di database Aurora secondario possono essere riavviate in determinate circostanze. Se l'istanza Writer DB principale Regione AWS viene riavviata o failover, vengono riavviate anche le istanze DB Reader nelle regioni secondarie. Il cluster secondario non sarà quindi disponibile fino a quando tutte le istanze database di lettura al

suo interno non sono nuovamente sincronizzate con l'istanza di scrittura del cluster di database primario. Il comportamento del cluster primario durante il riavvio o il failover è uguale a quello di un singolo cluster di database non globale. Per ulteriori informazioni, consulta Replica con Amazon Aurora.

Prima di apportare modifiche al cluster di database primario, assicurarsi di comprendere l'impatto sul database globale. Per ulteriori informazioni, consulta Ripristino di un database globale Amazon Aurora da un'interruzione non pianificata.

- Aurora Global Database attualmente non supporta lo `inaccessible-encryption-credentials-recoverable` stato in cui Amazon Aurora perde l'accesso alla chiave per AWS KMS il cluster DB. In questi casi, il cluster database crittografato entra nello stato terminale `inaccessible-encryption-credentials`. Per ulteriori informazioni su questi stati, consulta Visualizzazione dello stato del cluster del DB.
- Secrets Manager non supporta Database globale Aurora. Quando aggiungi una Regione a un database globale, è necessario prima disattivare l'integrazione di Secrets Manager per l'istanza database.
- I cluster di database basati su Aurora PostgreSQL che utilizzano un database globale Aurora hanno le seguenti limitazioni:
  - La gestione della cache del cluster non è supportata per i cluster di database secondari Aurora PostgreSQL che fanno parte dei database globali Aurora.
  - Se il cluster di database primario del database globale è basato su una replica di un'istanza PostgreSQL Amazon RDS, non è possibile creare un cluster secondario. Non tentare di creare un file secondario da quel cluster utilizzando l' Console di gestione AWS operazione AWS CLI, the o l'`CreateDBCluster`API. I tentativi di eseguire questa operazione scadono e il cluster secondario non viene creato.

Per i database globali si consiglia di creare cluster di database secondari utilizzando la stessa versione del motore di database Aurora del cluster primario. Per ulteriori informazioni, consulta Creazione di un database globale Amazon Aurora.

Nozioni di base sul Database globale Amazon Aurora

Per iniziare a utilizzare Database globale Aurora, è necessario innanzitutto decidere quale motore di database Aurora utilizzare e in quali Regioni AWS. Solo determinate versioni dei motori di database Aurora PostgreSQL e Aurora MySQL in alcune Regioni AWS supportano Database globale Aurora. Per l'elenco completo, consulta Regioni e motori di database supportati per i database globali Aurora.

È possibile creare un Database globale Aurora in uno dei modi seguenti:

- Creare un nuovo Database globale Aurora con nuovi cluster di database Aurora e istanze database Aurora: puoi eseguire questa operazione seguendo la procedura descritta in Creazione di un database globale Amazon Aurora. Dopo aver creato il cluster di database Aurora primario, aggiungere la Regione AWS secondaria seguendo la procedura descritta in Aggiunta di una Regione AWS a un database globale Amazon Aurora.
- Utilizza un cluster di database Aurora esistente che supporta la funzionalità Database globale Aurora e aggiungi una Regione AWS: puoi eseguire questa operazione solo se il cluster di database Aurora esistente utilizza una versione del motore database compatibile con il globale.

  Verifica se è possibile scegliere Aggiungi regione per Operazione nella Console di gestione AWS quando è selezionato il cluster database Aurora. Se è possibile, è possibile utilizzare tale cluster database Aurora per il cluster Aurora globale. Per ulteriori informazioni, consulta Aggiunta di una Regione AWS a un database globale Amazon Aurora.

Prima di creare un Database globale Aurora, è consigliabile conoscere tutti i requisiti di configurazione.

Argomenti

- Requisiti di configurazione di un database globale Amazon Aurora
- Creazione di un database globale Amazon Aurora
- Aggiunta di una Regione AWS a un database globale Amazon Aurora
- Creazione di un cluster database Aurora headless in una regione secondaria
- Creazione di un database globale Amazon Aurora da uno snapshot Amazon Aurora o Amazon RDS

Requisiti di configurazione di un database globale Amazon Aurora

Prima di iniziare a lavorare con il database globale, pianifica i nomi del cluster e delle istanze, le Regioni AWS, il numero di istanze e le classi di istanze che intendi utilizzare. Assicurati che le tue scelte siano conformi ai seguenti requisiti di configurazione.

Un database globale Aurora si sviluppa su almeno due Regioni AWS. La Regione AWS principale supporta un cluster di database Aurora con un'istanza di database writer Aurora. Una Regione AWS secondaria esegue un cluster di database Aurora di sola lettura costituito interamente da repliche

Aurora. È necessaria almeno una Regione AWS secondaria, tuttavia un database globale Aurora può avere fino a 10 Regioni AWS secondarie. Nella tabella sono elencati il numero massimo di cluster di database Aurora, di istanze database Aurora e repliche Aurora consentiti in un database globale Aurora.

Descrizione	Regione AWS principale	Regioni AWS secondarie
Cluster di database Aurora	1	10 (massimo)
Istanze di scrittura	1	0
Istanze di sola lettura (repliche Aurora), per cluster di database Aurora	15 (massimo)	16 (totali)
Istanze di sola lettura (massimo consentito, dato il numero effettivo di Regioni secondarie)	15 - s	s = numero totale di Regioni AWS secondarie

I cluster di database Aurora che compongono un database globale Aurora hanno i seguenti requisiti specifici:

- Requisiti delle classi di istanza database – Un database globale Aurora richiede classi di istanza database ottimizzate per le applicazioni che fanno un uso intensivo della memoria. Per informazioni sulle classi di istanza database ottimizzate per la memoria, consulta Tipi di classi di istanza database. Si consiglia di utilizzare una classe di istanza db.r5 o superiore.
- Requisiti di Regione AWS: un database globale Aurora richiede un cluster di database Aurora primario in una Regione AWS e almeno un cluster di database Aurora secondario in una regione diversa. È possibile creare fino a 10 cluster di database Aurora secondari (di sola lettura) e ciascuno deve trovarsi in una Regione diversa. In altre parole, nessun cluster di database Aurora in un database globale Aurora può trovarsi nella stessa Regione AWS.

  Per informazioni su quali combinazioni di motore di database Aurora, versione del motore e Regione AWS è possibile utilizzare con Database globale Aurora, consulta Regioni e motori di database supportati per i database globali Aurora

- Requisiti di denominazione: i nomi scelti per ciascuno dei cluster di database Aurora devono essere univoci in tutte le Regioni AWS. Non è possibile utilizzare lo stesso nome per cluster di database Aurora diversi anche se si trovano in regioni diverse.
- Requisiti di capacità per Aurora Serverless v2: per un database globale con Aurora Serverless v2, la capacità minima richiesta per il cluster di database nella Regione AWS primaria è di 8 ACU.

Prima di poter seguire le procedure descritte in questa sezione, è necessario disporre di un Account AWS. Completare le attività di configurazione per lavorare con Amazon Aurora. Per ulteriori informazioni, consulta Configurazione dell'ambiente per Amazon Aurora. È inoltre necessario completare altre fasi preliminari per la creazione di un cluster di database Aurora. Per ulteriori informazioni, consulta Creazione di un cluster database Amazon Aurora.

Quando sei pronto per configurare il tuo database globale, consulta Creazione di un database globale Amazon Aurora per la procedura per creare tutte le risorse necessarie. È inoltre possibile seguire la procedura illustrata in Aggiunta di una Regione AWS a un database globale Amazon Aurora per creare un database globale usando un cluster Aurora esistente come cluster primario.

Creazione di un database globale Amazon Aurora

Per creare un Database globale Aurora e le risorse associate utilizzando la Console di gestione AWS, la AWS CLI o l'API RDS, segui la procedura riportata di seguito.

> **Note**
>
> Se è presente un cluster di database Aurora che esegue un motore di database Aurora compatibile a livello globale, è possibile utilizzare una forma più breve di questa procedura. In tal caso, è possibile aggiungere un'altra Regione AWS al cluster di database esistente per creare un Database globale Aurora. A questo proposito, consulta Aggiunta di una Regione AWS a un database globale Amazon Aurora.

Console

La procedura per la creazione di un database globale Aurora inizia con l'accesso a una Regione AWS che supporta la funzionalità del database globale Aurora. Per un elenco completo, consulta Regioni e motori di database supportati per i database globali Aurora.

Uno dei passaggi seguenti consiste nel selezionare un Virtual Private Cloud (VPC) basato su Amazon VPC per il cluster di database Aurora. Per utilizzare il tuo VPC, ti consigliamo di crearlo in

anticipo in modo che sia possibile sceglierlo. Allo stesso tempo, crea tutte le sottoreti correlate e, in base alle esigenze, un gruppo di sottoreti e un gruppo di sicurezza. Per scoprire come, consulta Tutorial: Creazione di un Amazon VPC da utilizzare con un cluster database (solo IPv4).

Per informazioni generali sulla creazione di un cluster di database Aurora, consulta Creazione di un cluster database Amazon Aurora.

Per creare un database globale Aurora

1. Accedi alla Console di gestione AWS e apri la console Amazon RDS all'indirizzo https://console.aws.amazon.com/rds/.
2. Scegliere Create database (Crea database). Nella pagina Crea database completa le seguenti operazioni:
   - Scegli Standard create (Creazione standard) per il metodo di creazione del database. (Non selezionare Easy Create (Creazione rapida)).
   - Per `Engine type` nella sezione Opzioni motore scegli il tipo di motore applicabile, Aurora (compatibile con MySQL) o Aurora (compatibile con PostgreSQL).
3. Continua a creare il database globale Aurora utilizzando i passaggi descritti nelle procedure riportate di seguito.

Creazione di un database globale tramite Aurora MySQL

La procedura seguente si applica a tutte le versioni di Aurora MySQL.

Per creare un database globale Aurora tramite Aurora MySQL

Completare la pagina Create database (Crea database).

1. Per Opzioni motore, scegli quanto segue:
   - Per Engine version (Versione motore), scegli la versione di Aurora MySQL che desideri utilizzare per il database globale Aurora.
2. Per Templates (Modelli), scegliere Production (Produzione). In alternativa, puoi scegliere Dev/Test se appropriato per il tuo caso d'uso. Non utilizzare Dev/Test in ambienti di produzione.
3. In Settings (Impostazioni), eseguire la seguente operazione:

a. Immettere un nome specifico per l'identificatore del cluster di database. Al termine della creazione del database globale Aurora, questo nome identificherà il cluster di database primario.

b. Specifica la password per l'account utente `admin` per l'istanza database o lascia che Aurora ne generi una. Se scegli di generare automaticamente una password, apparirà l'opzione per copiare la password.



4. Per DB instance class (Classe dell'istanza database), scegli `db.r5.large` o un'altra classe dell'istanza database ottimizzata per la memoria. Si consiglia di utilizzare una classe di istanza db.r5 o superiore.

5. Per Disponibilità e durata, ti consigliamo di lasciare che Aurora crei autonomamente una replica Aurora in una zona di disponibilità differente. Se non crei subito una replica Aurora, sarà necessario farlo in un secondo momento.



6. Per Connectivity (Connettività), scegliere il Virtual Private Cloud (VPC) basato su Amazon VPC che definisce l'ambiente di rete virtuale per questa istanza di database. È possibile scegliere i valori predefiniti per semplificare questa attività.
7. Completa le impostazioni di autenticazione del database . Per semplificare il processo, puoi scegliere subito Autenticazione password e configurare AWS Identity and Access Management (IAM) in un secondo momento.
8. In Additional configuration (Configurazione aggiuntiva), eseguire le operazioni seguenti:

   a. Immettere un nome per Initial database name (Nome del database iniziale) per creare l'istanza database Aurora primaria per il cluster. Questo è il nodo di scrittura per il cluster di database Aurora primario.

      Lasciare i valori predefiniti selezionati per il gruppo di parametri del cluster di database e il gruppo di parametri di database, a meno che non si disponga di gruppi di parametri personalizzati che si desidera utilizzare.

   b. Se selezionata, deseleziona l’opzione Abilita backtrack. I database globali Aurora non supportano il backtracking. Puoi accettare tutte le altre impostazioni predefinite per Configurazione aggiuntiva.
9. Scegliere Create database (Crea database).

Il completamento del processo di creazione dell'istanza database Aurora, della replica Aurora e del cluster di database Aurora può richiedere alcuni minuti ad Aurora. È possibile sapere quando il cluster database Aurora è pronto per l'uso come cluster database primario in un database globale Aurora in base al relativo stato. Quando è così, il suo stato e quello del writer e del nodo di replica sarà Disponibile, come illustrato di seguito.



Quando il cluster di database primario è disponibile, crea il database globale Aurora aggiungendo un cluster secondario. A tale scopo, segui le fasi in Aggiunta di una Regione AWS a un database globale Amazon Aurora.

Creazione di un database globale tramite Aurora PostgreSQL

Per creare un database globale Aurora tramite Aurora PostgreSQL

Completare la pagina Create database (Crea database).

1. Per Opzioni motore, scegli quanto segue:
   - Per Engine version (Versione motore), scegli la versione di Aurora PostgreSQL che desideri utilizzare per il database globale Aurora.
2. Per Templates (Modelli), scegliere Production (Produzione). Oppure puoi scegliere Dev/Test se appropriato. Non utilizzare Dev/Test in ambienti di produzione.
3. In Settings (Impostazioni), eseguire la seguente operazione:
   a. Immettere un nome specifico per l'identificatore del cluster di database. Al termine della creazione del database globale Aurora, questo nome identificherà il cluster di database primario.
   b. Specifica la password per l'account amministratore predefinito per il cluster database o lascia che Aurora ne generi una. Se si sceglie Genera automaticamente una password, apparirà l'opzione per copiare la password.



4. Per DB instance class (Classe dell'istanza database), scegli `db.r5.large` o un'altra classe dell'istanza database ottimizzata per la memoria. Si consiglia di utilizzare una classe di istanza db.r5 o superiore.
5. Per Availability & durability (Disponibilità e durata), consigliamo di lasciare che Aurora crei autonomamente una replica Aurora in un'altra zona di disponibilità. Se non crei subito una replica Aurora, sarà necessario farlo in un secondo momento.
6. Per Connectivity (Connettività), scegliere il Virtual Private Cloud (VPC) basato su Amazon VPC che definisce l'ambiente di rete virtuale per questa istanza di database. È possibile scegliere i valori predefiniti per semplificare questa attività.
7. (Facoltativo) Completa le impostazioni di Database authentication (Autenticazione database). L'autenticazione password è sempre abilitata. Per semplificare il processo, puoi ignorare questa sezione e impostare IAM o Autenticazione di password e Kerberos in un secondo momento.
8. In Additional configuration (Configurazione aggiuntiva), eseguire le operazioni seguenti:

a. Immettere un nome per Initial database name (Nome del database iniziale) per creare l'istanza database Aurora primaria per il cluster. Questo è il nodo di scrittura per il cluster di database Aurora primario.

   Lasciare i valori predefiniti selezionati per il gruppo di parametri del cluster di database e il gruppo di parametri di database, a meno che non si disponga di gruppi di parametri personalizzati che si desidera utilizzare.

b. Accetta tutte le altre impostazioni predefinite per Configurazione aggiuntiva, ad esempio Crittografia, Esportazioni log e così via.

9. Scegliere Create database (Crea database).

   Il completamento del processo di creazione dell'istanza database Aurora, della replica Aurora e del cluster di database Aurora può richiedere alcuni minuti ad Aurora. Quando il cluster è pronto per l'uso, il cluster di database Aurora e i relativi nodi di scrittura e replica presentano tutti lo stato Disponibile. Questo diventa il cluster di database primario del database globale Aurora, dopo averne aggiunto uno secondario.



Dopo che il cluster di database primario è disponibile, creare uno o più cluster secondari seguendo le fasi in Aggiunta di una Regione AWS a un database globale Amazon Aurora.

AWS CLI

I comandi AWS CLI delle procedure riportate di seguito completano le seguenti attività:

1. Crea un database globale Aurora, assegnandogli un nome e specificando il tipo di motore del database Aurora che si desidera utilizzare.
2. Creare un cluster di database Aurora per il database globale Aurora.
3. Creare l'istanza database Aurora per il cluster. Questo è il cluster database Aurora primario per il database globale.
4. Creare una seconda istanza database per il cluster di database Aurora. Questo è un reader per completare il cluster di database Aurora.
5. Creare un secondo cluster di database Aurora in un'altra regione e aggiungerlo al database globale Aurora, seguendo la procedura descritta in Aggiunta di una Regione AWS a un database globale Amazon Aurora.

Segui la procedura per il motore di database Aurora.

Creazione di un database globale tramite Aurora MySQL

Per creare un database globale Aurora tramite Aurora MySQL

1. Utilizza il comando `create-global-cluster` della CLI, inviando il nome della Regione AWS, il motore di database Aurora e la versione.

   Per Linux, macOS o Unix:

   ```
   aws rds create-global-cluster --region primary_region \
       --global-cluster-identifier global_database_id \
       --engine aurora-mysql \
       --engine-version version # optional
   ```

   Per Windows:

   ```
   aws rds create-global-cluster ^
       --global-cluster-identifier global_database_id ^
       --engine aurora-mysql ^
       --engine-version version # optional
   ```

   Questo crea un database globale Aurora "vuoto", con solo un nome (identificatore) e un motore di database Aurora. Prima che il database globale Aurora sia disponibile, potrebbero essere necessari alcuni minuti. Prima di passare alla fase successiva, utilizzare il comando CLI `describe-global-clusters` per verificare se è disponibile.

```
aws rds describe-global-clusters --region primary_region --global-cluster-
identifier global_database_id
```

Quando il database globale Aurora è disponibile, è possibile creare il relativo cluster di database Aurora primario.

2. Per creare un cluster di database Aurora primario, utilizzare il comando CLI `create-db-cluster`. Includi il nome del database globale Aurora utilizzando il parametro `--global-cluster-identifier`.

   Per Linux, macOS o Unix:

   ```
   aws rds create-db-cluster \
     --region primary_region \
     --db-cluster-identifier primary_db_cluster_id \
     --master-username userid \
     --master-user-password password \
     --engine aurora-mysql \
     --engine-version version \
     --global-cluster-identifier global_database_id
   ```

   Per Windows:

   ```
   aws rds create-db-cluster ^
     --region primary_region ^
     --db-cluster-identifier primary_db_cluster_id ^
     --master-username userid ^
     --master-user-password password ^
     --engine aurora-mysql ^
     --engine-version version ^
     --global-cluster-identifier global_database_id
   ```

   Utilizzare il comando `describe-db-clusters` AWS CLI per confermare che il cluster database Aurora è pronto. Per individuare un cluster di database Aurora specifico, utilizzare il parametro `--db-cluster-identifier`. Oppure è possibile lasciare vuoto il nome del cluster di database Aurora nel comando per ottenere dettagli su tutti i cluster di database Aurora nella regione specificata.

```
aws rds describe-db-clusters --region primary_region --db-cluster-
identifier primary_db_cluster_id
```

Quando per il cluster viene visualizzata la risposta `"Status": "available"`, significa che è pronto per l'uso.

3. Creare l'istanza database per il cluster di database Aurora primario. A tale scopo, utilizza il comando della CLI `create-db-instance`. Assegnare al comando il nome del cluster di database Aurora e specificare i dettagli di configurazione per l'istanza. Nel comando non è necessario passare i parametri `--master-username` e `--master-user-password`, perché li ottiene dal cluster di database Aurora.

   Per `--db-instance-class`, è possibile utilizzare solo quelli dalle classi ottimizzate per la memoria, ad esempio `db.r5.large`. Si consiglia di utilizzare una classe di istanza db.r5 o superiore. Per informazioni su queste classi, consulta Tipi di classi di istanza database.

   Per Linux, macOS o Unix:

```
aws rds create-db-instance \
  --db-cluster-identifier primary_db_cluster_id \
  --db-instance-class instance_class \
  --db-instance-identifier db_instance_id \
  --engine aurora-mysql \
  --engine-version version \
  --region primary_region
```

   Per Windows:

```
aws rds create-db-instance ^
  --db-cluster-identifier primary_db_cluster_id ^
  --db-instance-class instance_class ^
  --db-instance-identifier db_instance_id ^
  --engine aurora-mysql ^
  --engine-version version ^
  --region primary_region
```

   L'operazione `create-db-instance` potrebbe richiedere alcuni minuti per il completamento. Prima di continuare, controllare lo stato per verificare se l'istanza di database Aurora è disponibile.

```
aws rds describe-db-clusters --db-cluster-identifier primary_db_cluster_id
```

Quando il comando restituisce lo stato `available`, è possibile creare un'altra istanza database Aurora per il cluster di database primario. Si tratta dell'istanza del reader (la replica Aurora) per il cluster di database Aurora.

4. Per creare un'altra istanza database Aurora per il cluster, utilizza il comando della CLI `create-db-instance`:

   Per Linux, macOS o Unix:

   ```
   aws rds create-db-instance \
     --db-cluster-identifier primary_db_cluster_id \
     --db-instance-class instance_class \
     --db-instance-identifier replica_db_instance_id \
     --engine aurora-mysql
   ```

   Per Windows:

   ```
   aws rds create-db-instance ^
     --db-cluster-identifier primary_db_cluster_id ^
     --db-instance-class instance_class ^
     --db-instance-identifier replica_db_instance_id ^
     --engine aurora-mysql
   ```

Quando l'istanza database è disponibile, la replica inizia dal nodo del writer al nodo di replica. Prima di continuare, verificare che l'istanza database sia disponibile utilizzando il comando CLI `describe-db-instances`.

A questo punto, si dispone di un database globale Aurora con il relativo cluster di database Aurora primario contenente un'istanza database writer e una replica Aurora. A questo punto è possibile aggiungere un cluster di database Aurora di sola lettura in una regione diversa per completare il database globale Aurora. A tale scopo, segui la procedura in Aggiunta di una Regione AWS a un database globale Amazon Aurora.

Creazione di un database globale tramite Aurora PostgreSQL

Quando si creano oggetti Aurora per un database globale Aurora utilizzando i comandi seguenti, possono essere necessari alcuni minuti prima che ciascuno diventi disponibile. Dopo aver completato

un determinato comando, si consiglia di controllare lo stato dell'oggetto Aurora specifico per assicurarsi che lo stato sia disponibile.

A tale scopo, utilizza il comando della CLI `describe-global-clusters`.

```
aws rds describe-global-clusters --region primary_region
    --global-cluster-identifier global_database_id
```

Per creare un database globale Aurora tramite Aurora PostgreSQL

1. Utilizza il comando della CLI `create-global-cluster`.

   Per Linux, macOS o Unix:

   ```
   aws rds create-global-cluster --region primary_region \
       --global-cluster-identifier global_database_id \
       --engine aurora-postgresql \
       --engine-version version # optional
   ```

   Per Windows:

   ```
   aws rds create-global-cluster ^
       --global-cluster-identifier global_database_id ^
       --engine aurora-postgresql ^
       --engine-version version # optional
   ```

   Quando il database globale Aurora è disponibile, è possibile creare il relativo cluster di database Aurora primario.

2. Per creare un cluster di database Aurora primario, utilizzare il comando CLI `create-db-cluster`. Includi il nome del database globale Aurora utilizzando il parametro `--global-cluster-identifier`.

   Per Linux, macOS o Unix:

   ```
   aws rds create-db-cluster \
     --region primary_region \
     --db-cluster-identifier primary_db_cluster_id \
     --master-username userid \
     --master-user-password password \
     --engine aurora-postgresql \
   ```

```
  --engine-version version \
  --global-cluster-identifier global_database_id
```

Per Windows:

```
aws rds create-db-cluster ^
  --region primary_region ^
  --db-cluster-identifier primary_db_cluster_id ^
  --master-username userid ^
  --master-user-password password ^
  --engine aurora-postgresql ^
  --engine-version version ^
  --global-cluster-identifier global_database_id
```

Verificare che il cluster di database Aurora sia pronto. Quando dal seguente comando viene visualizzata la risposta `"Status": "available"` per il cluster di database Aurora, è possibile continuare.

```
aws rds describe-db-clusters --region primary_region --db-cluster-
identifier primary_db_cluster_id
```

3. Creare l'istanza database per il cluster di database Aurora primario. A tale scopo, utilizza il comando della CLI `create-db-instance`.

   Invia il nome del cluster di database Aurora con il parametro `--db-cluster-identifier`.

   Nel comando non è necessario passare i parametri `--master-username` e `--master-user-password`, perché li ottiene dal cluster di database Aurora.

   Per `--db-instance-class`, è possibile utilizzare solo quelli dalle classi ottimizzate per la memoria, ad esempio `db.r5.large`. Si consiglia di utilizzare una classe di istanza db.r5 o superiore. Per informazioni su queste classi, consulta Tipi di classi di istanza database.

   Per Linux, macOS o Unix:

```
aws rds create-db-instance \
  --db-cluster-identifier primary_db_cluster_id \
  --db-instance-class instance_class \
  --db-instance-identifier db_instance_id \
  --engine aurora-postgresql \
  --engine-version version \
```

```
  --region primary_region
```

Per Windows:

```
aws rds create-db-instance ^
  --db-cluster-identifier primary_db_cluster_id ^
  --db-instance-class instance_class ^
  --db-instance-identifier db_instance_id ^
  --engine aurora-postgresql ^
  --engine-version version ^
  --region primary_region
```

4. Prima di continuare, controllare lo stato dell'istanza database Aurora.

```
aws rds describe-db-clusters --db-cluster-identifier primary_db_cluster_id
```

Se la risposta mostra che lo stato dell'istanza database Aurora è `available`, è possibile creare un'altra istanza database Aurora per il cluster di database primario.

5. Per creare una replica Aurora per cluster di database Aurora, utilizzare il comando CLI [create-db-instance](#).

Per Linux, macOS o Unix:

```
aws rds create-db-instance \
  --db-cluster-identifier primary_db_cluster_id \
  --db-instance-class instance_class \
  --db-instance-identifier replica_db_instance_id \
  --engine aurora-postgresql
```

Per Windows:

```
aws rds create-db-instance ^
  --db-cluster-identifier primary_db_cluster_id ^
  --db-instance-class instance_class ^
  --db-instance-identifier replica_db_instance_id ^
  --engine aurora-postgresql
```

Quando l'istanza database è disponibile, la replica inizia dal nodo del writer al nodo di replica. Prima di continuare, verificare che l'istanza database sia disponibile utilizzando il comando CLI `describe-db-instances`.

Il database globale Aurora esiste, ma ha solo la relativa regione principale con un cluster di database Aurora costituito da un'istanza database di scrittura e una replica Aurora. A questo punto è possibile aggiungere un cluster di database Aurora di sola lettura in una regione diversa per completare il database globale Aurora. A tale scopo, segui la procedura in Aggiunta di una Regione AWS a un database globale Amazon Aurora.

API RDS

Per creare un Aurora Global Database con l'API RDS, eseguire l'operazione CreateGlobalCluster.

Aggiunta di una Regione AWS a un database globale Amazon Aurora

È possibile utilizzare la procedura seguente per aggiungere un ulteriore cluster secondario a un database globale esistente. È anche possibile creare un database globale da un cluster di database Aurora autonomo utilizzando questa procedura per aggiungere la prima Regione AWS secondaria.

Un database globale Aurora richiede almeno un cluster di database Aurora secondario in una Regione AWS diversa da quella del cluster di database Aurora primario. È possibile collegare fino a 10 cluster di database secondari a un Database globale Aurora. Ripeti la procedura seguente per ogni nuovo cluster di database secondario. Per ogni cluster di database secondario aggiunto al database globale Aurora, ridurre di un'unità il numero di repliche Aurora consentite nel cluster di database primario.

Ad esempio, se il Database globale Aurora dispone di 10 Regioni secondarie, il cluster di database primario può avere solo 5 repliche Aurora (anziché 15). Per ulteriori informazioni, consulta Requisiti di configurazione di un database globale Amazon Aurora.

Il numero di repliche Aurora (istanze di lettura) nel cluster di database primario determina il numero di cluster database secondari che è possibile aggiungere. Il numero di istanze di lettura nel cluster database primario più il numero di cluster secondari potrebbe essere pari a 15. Ad esempio, se esistono 14 istanze di lettura nel cluster di database primario e un cluster secondario, non è possibile aggiungere un altro cluster secondario al database globale.

> **Note**
>
> Per Aurora MySQL versione 3, quando crei un cluster secondario, assicurati che il valore di `lower_case_table_names` corrisponda al valore nel cluster primario. Questa impostazione è un parametro del database che influisce sul modo in cui il server gestisce la distinzione tra maiuscole e minuscole. Per ulteriori informazioni sui parametri di database, vedi Gruppi di parametri per Amazon Aurora.
>
> Quando crei un cluster secondario, ti consigliamo di utilizzare la stessa versione del motore database per il primario e il secondario. Se necessario, aggiorna il primario in modo che abbia la stessa versione del secondario. Per ulteriori informazioni, consulta Compatibilità del livello di patch per switchover e failover gestiti tra regioni.

Console

Come aggiungere una Regione AWS a un database globale Aurora

1. Accedi alla Console di gestione AWS e apri la console Amazon RDS all'indirizzo https://console.aws.amazon.com/rds/.
2. Nel riquadro di navigazione della Console di gestione AWS, scegliere Databases (Database).
3. Scegliere il database globale Aurora che richiede un cluster di database Aurora secondario. Assicurarsi che il cluster di database Aurora primario sia `Available`.
4. In Azioni, scegli Aggiungi Regione AWS.



5. Nella pagina Add a region (Aggiungi una regione), scegli la Regione AWS secondaria.

   Non è possibile scegliere una Regione AWS che dispone già di un cluster di database Aurora secondario per lo stesso database globale Aurora. Inoltre, non può essere la stessa regione del cluster database Aurora primario.

   > **Note**
   >
   > I database globali Babelfish per Aurora PostgreSQL funzionano nelle Regioni secondarie solo se i parametri che controllano le preferenze di Babelfish sono attivati in tali Regioni. Per ulteriori informazioni, consulta Impostazioni del gruppo di parametri del cluster database per Babelfish



6. Completa i campi rimanenti per il cluster Aurora secondario nella nuova regione AWS. Queste sono le stesse opzioni di configurazione di qualsiasi istanza del cluster database Aurora, ad eccezione della seguente opzione solo per i database globali Aurora basati su Aurora MySQL–:

- Abilita l'inoltro in scrittura di replica in lettura – Questa impostazione facoltativa consente ai cluster database secondari del database Aurora globale di inoltrare le operazioni di scrittura al cluster primario. Per ulteriori informazioni, consulta Utilizzo dell'inoltro di scrittura in un database globale Amazon Aurora.



7. Scegli Aggiungi Regione AWS.

Dopo aver aggiunto la regione al database globale Aurora, potrai visualizzarla nell'elenco dei Database nella Console di gestione AWS come mostrato nella schermata.



AWS CLI

Come aggiungere una Regione AWS secondaria a un database globale Aurora

Per aggiungere un cluster secondario al database globale utilizzando la CLI, è necessario disporre già dell'oggetto container del cluster globale. Se non hai già eseguito il comando `create-global-cluster`, consulta la procedura CLI in Creazione di un database globale Amazon Aurora.

1. Utilizzare il comando CLI `create-db-cluster` con il nome (`--global-cluster-identifier`) del database globale Aurora. Per gli altri parametri, effettuare le seguenti operazioni:
2. Per `--region`, scegli una Regione AWS diversa dalla tua regione Aurora principale.
3. Scegli i valori specifici per i parametri `--engine` e `--engine-version`. Questi valori sono gli stessi di quelli del cluster database Aurora primario nel database globale Aurora.
4. Per un cluster crittografato, specificare la Regione AWS principale come `--source-region` per la crittografia.

Nell'esempio seguente viene creato un nuovo cluster di database Aurora e viene collegato a un database globale Aurora come cluster di database Aurora secondario di sola lettura. Nell'ultimo passaggio, viene aggiunta un'istanza database Aurora al nuovo cluster di database Aurora.

Per Linux, macOS o Unix:

```
aws rds --region secondary_region \
  create-db-cluster \
    --db-cluster-identifier secondary_cluster_id \
    --global-cluster-identifier global_database_id \
    --engine aurora-mysql | aurora-postgresql \
    --engine-version version

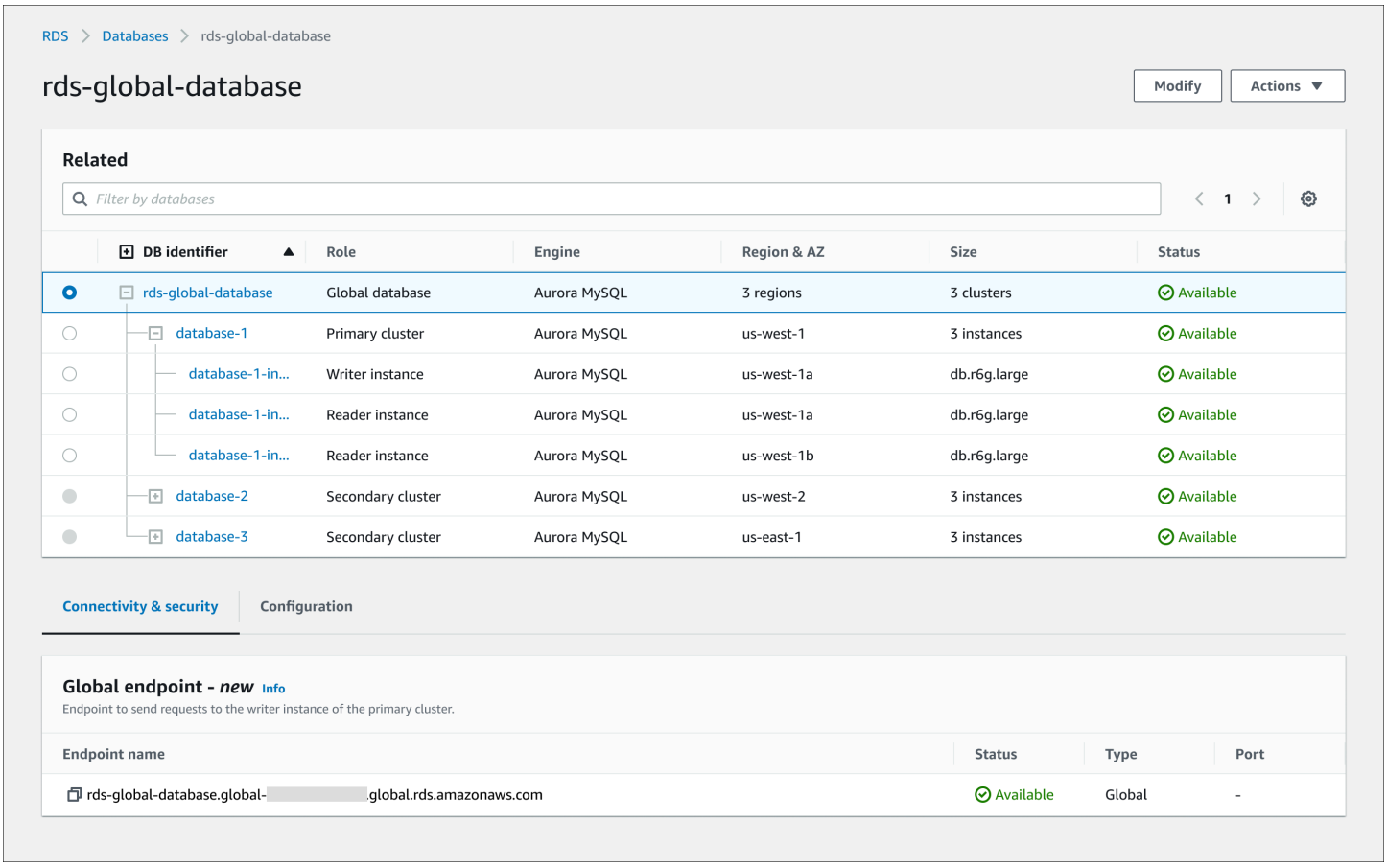
aws rds --region secondary_region \
  create-db-instance \
    --db-instance-class instance_class \
    --db-cluster-identifier secondary_cluster_id \
    --db-instance-identifier db_instance_id \
    --engine aurora-mysql | aurora-postgresql
```

Per Windows:

```
aws rds --region secondary_region ^
  create-db-cluster ^
    --db-cluster-identifier secondary_cluster_id ^
    --global-cluster-identifier global_database_id_id ^
    --engine aurora-mysql | aurora-postgresql ^
    --engine-version version

aws rds --region secondary_region ^
  create-db-instance ^
```

```
    --db-instance-class instance_class ^
    --db-cluster-identifier secondary_cluster_id ^
    --db-instance-identifier db_instance_id ^
    --engine aurora-mysql | aurora-postgresql
```

API RDS

Per aggiungere una nuova Regione AWS a un database globale Aurora con l'API RDS, esegui l'operazione CreateDBCluster. Specificare l'identificatore del database globale esistente utilizzando il parametro `GlobalClusterIdentifier`.

Creazione di un cluster database Aurora headless in una regione secondaria

Sebbene un database globale Aurora richieda almeno un cluster di database Aurora secondario in una Regione AWS diversa da quella primaria, è possibile utilizzare una configurazione headless per il cluster secondario. Un cluster database Aurora secondario headless è un cluster senza un'istanza database. Questo tipo di configurazione può ridurre le spese per un database globale Aurora. In un cluster database Aurora, il calcolo e l'archiviazione vengono disaccoppiati. Senza l'istanza database, non viene addebitato alcun costo per il calcolo, solo per lo storage. Se è configurato correttamente, il volume di archiviazione di un secondario headless viene mantenuto sincronizzato con il cluster database Aurora primario.

Puoi aggiungere il cluster secondario come si fa normalmente durante la creazione di un database globale Aurora. Se stai creando tutti i cluster nel database globale, segui la procedura riportata in Creazione di un database globale Amazon Aurora. Se disponi già di un cluster di database da utilizzare come cluster primario, segui la procedura riportata in Aggiunta di una Regione AWS a un database globale Amazon Aurora.

Dopo che il cluster di database Aurora primario inizia la replica nel secondario, è possibile eliminare l'istanza database Aurora di sola lettura dal cluster di database Aurora secondario. Questo cluster secondario è ora considerato "headless" perché non dispone più dell'istanza DB. Anche in assenza di istanze database Aurora, Aurora mantiene la sincronizzazione tra il volume di archiviazione e il cluster di database Aurora primario.

> **Warning**
>
> Con Aurora PostgreSQL, per creare un cluster headless in una Regione AWS secondaria, utilizza la AWS CLI o l'API RDS per aggiungere la Regione AWS secondaria. Salta il

passaggio per creare l'istanza DB di lettura per il cluster secondario. Attualmente, la creazione di un cluster headless non è supportata nella console RDS. Per l'utilizzo delle procedure CLI e API, consulta Aggiunta di una Regione AWS a un database globale Amazon Aurora.

Se il database globale utilizza una versione del motore Aurora PostgreSQL inferiore a 13.4, 12.8 o 11.13, la creazione di un'istanza database di lettura in una Regione secondaria e la successiva eliminazione potrebbero causare un problema di vacuum di Aurora PostgreSQL nell'istanza database di scrittura della Regione primaria. Se si verifica questo problema, riavviare l'istanza DB di scrittura della regione principale dopo aver eliminato l'istanza DB di lettura della regione secondaria.

Per aggiungere un cluster database Aurora secondario headless al database globale Aurora

1. Accedi alla Console di gestione AWS e apri la console Amazon RDS all'indirizzo https://console.aws.amazon.com/rds/.
2. Nel riquadro di navigazione della Console di gestione AWS, scegliere Databases (Database).
3. Scegliere il database globale Aurora che richiede un cluster di database Aurora secondario. Assicurarsi che il cluster di database Aurora primario sia `Available`.
4. In Azioni, scegli Aggiungi Regione AWS.
5. Nella pagina Add a region (Aggiungi una regione), scegli la Regione AWS secondaria.

   Non è possibile scegliere una Regione AWS che dispone già di un cluster di database Aurora secondario per lo stesso database globale Aurora. Inoltre, non può essere la stessa regione del cluster database Aurora primario.
6. Completa i campi rimanenti per il cluster Aurora secondario nella nuova Regione AWS. Queste sono le stesse opzioni di configurazione di qualsiasi istanza del cluster database Aurora.

   Per un database globale Aurora basato su Aurora MySQL–, ignora l'opzione Abilita inoltro in scrittura della replica in lettura. Questa opzione non ha alcuna funzione dopo aver eliminato l'istanza del lettore.
7. Scegli Aggiungi Regione AWS. Dopo aver aggiunto la regione al database globale Aurora, potrai visualizzarla nell'elenco dei Database nella Console di gestione AWS come mostrato nella schermata.



8. Prima di continuare, controlla lo stato del cluster database Aurora secondario e la relativa istanza di lettura tramite la Console di gestione AWS o la AWS CLI. Ad esempio:

```
$ aws rds describe-db-clusters --db-cluster-identifier secondary-cluster-id --query
 '*[].[Status]' --output text
```

   Il passaggio dello stato di un cluster database Aurora secondario appena aggiunto da `creating` a `available` può richiedere alcuni minuti. Quando il cluster database Aurora è disponibile, è possibile eliminare l'istanza di lettura.

9. Seleziona l'istanza di lettura nel cluster database Aurora secondario, quindi scegli Elimina.



Dopo aver eliminato l'istanza di lettura, il cluster secondario rimane parte del database globale Aurora. Non ha alcuna istanza associata, come illustrato di seguito.



Puoi utilizzare questo cluster di database Aurora secondario headless per ripristinare manualmente il database globale Amazon Aurora da un'interruzione non pianificata nella Regione AWS principale se si verifica un'interruzione del servizio.

Creazione di un database globale Amazon Aurora da uno snapshot Amazon Aurora o Amazon RDS

È possibile ripristinare uno snapshot di un cluster di database Aurora o di un'istanza database Amazon RDS da utilizzare come punto di partenza per il database globale Aurora. È possibile ripristinare uno snapshot e al contempo creare un nuovo cluster di database Aurora con provisioning. È quindi possibile aggiungere al cluster di database ripristinato un'altra Regione AWS trasformandolo in un database globale Aurora. Qualsiasi cluster di database Aurora creato utilizzando uno snapshot in questo modo diventa il cluster primario del database globale Aurora.

Lo snapshot utilizzato può essere preso da un cluster di database Aurora `provisioned` o `serverless`.

Durante il processo di ripristino, scegliere lo stesso tipo di motore di database dello snapshot. Ad esempio, si assuma di voler ripristinare uno snapshot creato da un cluster database Aurora Serverless che esegue Aurora PostgreSQL. In questo caso, si crea un cluster database Aurora PostgreSQL utilizzando lo stesso motore database Aurora e la stessa versione.

Quando si aggiunge una Regione AWS, il cluster database ripristinato assume il ruolo di cluster primario per il database globale Aurora. Tutti i dati contenuti in questo cluster primario vengono replicati in tutti i cluster secondari aggiunti al database globale Aurora.



Gestione di un database globale Amazon Aurora

È possibile eseguire la maggior parte delle operazioni di gestione sui singoli cluster che costituiscono un database globale Aurora. Quando selezioni Raggruppa risorse correlate nella pagina Database della console, i cluster primario e secondario vengono visualizzati come raggruppati nel database globale associato. Per trovare le Regioni AWS in cui sono in esecuzione i cluster di database di un database globale, il motore database Aurora e la versione e il relativo identificativo, utilizza la scheda Configurazione.

Il processo di failover del database tra regioni è disponibile solo per i database globali Aurora e non per un singolo cluster database Aurora. Per ulteriori informazioni, consulta Utilizzo dello switchover o failover in un Database globale Amazon Aurora.

Per ripristinare un database globale Aurora da un'interruzione non pianificata nella relativa regione principale, consulta Ripristino di un database globale Amazon Aurora da un'interruzione non pianificata.

Argomenti

- Modifica di un database globale Amazon Aurora
- Modifica dei parametri per un database globale Aurora
- Rimozione di un cluster da un database globale Amazon Aurora
- Eliminazione di un database globale Amazon Aurora
- Applicazione di tag alle risorse Database globale Amazon Aurora

Modifica di un database globale Amazon Aurora

La pagina Database in Console di gestione AWS contiene un elenco di tutti i database globali Aurora con il cluster primario e del cluster secondario per ciascuno di essi. Il database globale Aurora dispone di impostazioni di configurazione proprie. In particolare, ha delle Regioni AWS associate ai suoi cluster primari e secondari, come mostrato nello screenshot seguente.



Quando si apportano modifiche al database globale Aurora, si ha la possibilità di annullare le modifiche, come mostrato nella schermata seguente.



Quando si sceglie Continue (Continua), si confermano le modifiche.

Modifica dei parametri per un database globale Aurora

È possibile configurare i gruppi di parametri del cluster di database Aurora in modo indipendente per ogni cluster Aurora all'interno del database globale Aurora. La maggior parte dei parametri funzionano come per altri tipi di cluster Aurora: Si consiglia di mantenere le impostazioni coerenti tra tutti i cluster di un database globale. In questo modo è possibile evitare modifiche impreviste del comportamento se si promuove un cluster secondario come primario.

Ad esempio, utilizzare le stesse impostazioni per fusi orari e set di caratteri per evitare comportamenti incoerenti se un cluster diverso diventa un cluster primario.

Le impostazioni di configurazione `aurora_enable_repl_bin_log_filtering` e `aurora_enable_replica_log_compression` non hanno effetto.

Rimozione di un cluster da un database globale Amazon Aurora

È possibile rimuovere i cluster di database Aurora dal database globale Aurora per diversi motivi. Ad esempio, è possibile rimuovere un cluster di database Aurora da un database globale Aurora

se il cluster primario viene danneggiato o isolato. Diventa quindi un cluster database Aurora con provisioning autonomo che potrebbe essere utilizzato per creare un nuovo database globale Aurora. Per ulteriori informazioni, consulta Ripristino di un database globale Amazon Aurora da un'interruzione non pianificata.

È anche possibile rimuovere cluster di database Aurora se si desidera eliminare un database globale Aurora che non è più necessario. Non è possibile eliminare il database globale Aurora fino a quando non si rimuove (scollega) tutti i cluster database Aurora associati, lasciando il primario per ultimo. Per ulteriori informazioni, consulta Eliminazione di un database globale Amazon Aurora.

Quando un cluster database Aurora viene scollegato dal database globale Aurora, non è più sincronizzato con il primario. Diventa un cluster database Aurora con provisioning autonomo con funzionalità di lettura/scrittura complete.

È possibile rimuovere i cluster database Aurora dal database globale Aurora utilizzando la Console di gestione AWS, la AWS CLI o l'API RDS.

Console

Per rimuovere un cluster Aurora da un database globale Aurora

1. Accedi alla Console di gestione AWS e apri la console Amazon RDS all'indirizzo https://console.aws.amazon.com/rds/.
2. Scegli il cluster nella pagina Database .
3. Per Operazioni, scegli Rimuovi da globale.



Viene visualizzato un prompt per confermare che si desidera scollegare il cluster secondario dal database globale Aurora.



4. Scegli Rimuovi e promuovi per rimuovere il cluster dal database globale.

Il cluster di database Aurora non è più utilizzato come secondario nel database globale Aurora e non viene più sincronizzato con il cluster di database primario. Si tratta di un cluster di database Aurora autonomo con funzionalità di lettura/scrittura complete.



Dopo aver rimosso o eliminato tutti i cluster secondari, è possibile rimuovere il cluster primario nello stesso modo. Non è possibile scollegare (rimuovere) il cluster di database Aurora primario da un database globale Aurora fino a quando non si rimuovono tutti i cluster secondari.

Il database globale Aurora potrebbe rimanere nell'elenco Database, con 0 regioni e zone disponibilità. Laddove si desidera smettere di utilizzare questo database globale Aurora, è possibile eliminarlo. Per ulteriori informazioni, consulta Eliminazione di un database globale Amazon Aurora.

AWS CLI

Per rimuovere un cluster Aurora da un database globale Aurora, esegui il comando CLI remove-from-global-cluster con i seguenti parametri:

- `--global-cluster-identifier` – Il nome (identificatore) del database globale Aurora.
- `--db-cluster-identifier` – Il nome di ogni cluster di database Aurora da rimuovere dal database globale Aurora. Prima di rimuovere il cluster primario, rimuovere tutti i cluster di database Aurora secondari.

Negli esempi di seguito, da un database globale Aurora viene prima rimosso un cluster secondario e poi il cluster primario.

Per Linux, macOS o Unix:

```
aws rds --region secondary_region \
  remove-from-global-cluster \
    --db-cluster-identifier secondary_cluster_ARN \
    --global-cluster-identifier global_database_id

aws rds --region primary_region \
  remove-from-global-cluster \
```

```
    --db-cluster-identifier primary_cluster_ARN \
    --global-cluster-identifier global_database_id
```

Ripeti il comando `remove-from-global-cluster --db-cluster-identifier` *`secondary_cluster_ARN`* per ogni Regione AWS secondaria del database globale Aurora.

Per Windows:

```
aws rds --region secondary_region ^
  remove-from-global-cluster ^
    --db-cluster-identifier secondary_cluster_ARN ^
    --global-cluster-identifier global_database_id

aws rds --region primary_region ^
  remove-from-global-cluster ^
    --db-cluster-identifier primary_cluster_ARN ^
    --global-cluster-identifier global_database_id
```

Ripeti il comando `remove-from-global-cluster --db-cluster-identifier` *`secondary_cluster_ARN`* per ogni Regione AWS secondaria del database globale Aurora.

API RDS

Per rimuovere un cluster Aurora da un database globale Aurora con l'API RDS, esegui l'operazione RemoveFromGlobalCluster.

Eliminazione di un database globale Amazon Aurora

Poiché un database globale Aurora contiene in genere dati critici per l'azienda, non è possibile eliminare il database globale e i cluster associati in un singolo passaggio. Per eliminare un database globale Aurora, completa le seguenti operazioni:

- Rimuovere tutti i cluster di database secondari dal database globale Aurora. Ogni cluster diventa un cluster di database Aurora autonomo. Per scoprire come, consulta Rimozione di un cluster da un database globale Amazon Aurora.
- Da ogni cluster di database Aurora autonomo, eliminare tutte le repliche Aurora.
- Rimuovere il cluster di database primario dal database globale Aurora. Questo diventa un cluster di database Aurora autonomo.
- Dal cluster di database primario Aurora, per prima cosa eliminare tutte le repliche Aurora, quindi eliminare l'istanza del database di scrittura.

L'eliminazione dell'istanza di scrittura dal nuovo cluster di database Aurora autonomo generalmente rimuove anche il cluster di database Aurora e il database globale Aurora.

Per ulteriori informazioni generali, consulta Eliminazione di un'istanza database da un cluster database Aurora.

Per eliminare un database globale Aurora, è possibile utilizzare la Console di gestione AWS, la AWS CLI o l'API RDS.

Console

Per eliminare un database globale Aurora

1. Accedi alla Console di gestione AWS e apri la console Amazon RDS all'indirizzo https://console.aws.amazon.com/rds/.
2. Scegli Database e trova il database globale Aurora che desideri eliminare nell'elenco.
3. Confermare che tutti gli altri cluster vengono rimossi dal database globale Aurora. Il database globale Aurora dovrebbe avere 0 regioni e zone di disponibilità e una dimensione pari a 0 cluster.

   Se il database globale Aurora contiene cluster di database Aurora, non è possibile eliminarlo. Se necessario, scollegare i cluster di database Aurora primario e secondario dal database globale Aurora. Per ulteriori informazioni, consulta Rimozione di un cluster da un database globale Amazon Aurora.
4. Scegli il database globale Aurora nell'elenco, quindi scegli Elimina nel menu Operazioni.



AWS CLI

Per eliminare un database globale Aurora, esegui il comando della CLI delete-global-cluster con il nome della Regione AWS e l'identificatore del database globale Aurora, come illustrato nell'esempio seguente.

Per Linux, macOS o Unix:

```
aws rds --region primary_region delete-global-cluster \
   --global-cluster-identifier global_database_id
```

Per Windows:

```
aws rds --region primary_region delete-global-cluster ^
   --global-cluster-identifier global_database_id
```

API RDS

Per eliminare un cluster che fa parte di un database globale Aurora, esegui l'operazione API DeleteGlobalCluster.

Applicazione di tag alle risorse Database globale Amazon Aurora

Con la funzionalità Database globale Aurora, è possibile applicare tag RDS a risorse a diversi livelli all'interno di un database globale. Se non conosci il modo in cui i tag vengono utilizzati con AWS o le risorse Aurora, consulta Applicazione di tag alle risorse Amazon Aurora e Amazon RDS prima di applicare i tag all'interno del database globale.

> **Note**
>
> Poiché AWS elabora i dati dei tag come parte dei suoi meccanismi di rendicontazione dei costi, non includete dati sensibili o informazioni di identificazione personale (PII) nei nomi o nei valori dei tag.

È possibile applicare tag ai seguenti tipi di risorse in un database globale:

- L'oggetto container per l'intero database globale. Questa risorsa è nota come cluster globale.

  Dopo aver creato il cluster globale eseguendo un'operazione Aggiungi AWS Regione nella console, puoi aggiungere tag utilizzando la pagina dei dettagli per il cluster globale. Nella scheda

Tag all'interno della pagina dei dettagli del cluster globale, puoi aggiungere, rimuovere o modificare i tag e i relativi valori associati scegliendo Gestisci tag.

Con AWS CLI e l'API RDS, puoi aggiungere tag al cluster globale contemporaneamente alla sua creazione. È inoltre possibile aggiungere, rimuovere o modificare tag per un cluster globale esistente.

- Il cluster primario. Qui si utilizzano le stesse procedure per lavorare con i tag utilizzate per i cluster Aurora autonomi. È possibile configurare i tag prima di trasformare il cluster Aurora originale in un database globale. È possibile anche aggiungere, rimuovere o modificare i tag e i relativi valori associati scegliendo Gestisci tag nella scheda Tag all'interno della pagina dei dettagli del cluster di database
- Qualsiasi cluster secondario. Qui si utilizzano le stesse procedure per lavorare con i tag utilizzate per i cluster Aurora autonomi. È possibile configurare i tag contemporaneamente alla creazione di un cluster Aurora secondario utilizzando l'azione Aggiungi AWS Regione nella console. È possibile anche aggiungere, rimuovere o modificare i tag e i relativi valori associati scegliendo Gestisci tag nella scheda Tag all'interno della pagina dei dettagli del cluster di database
- Singole istanze database all'interno dei cluster primari o secondari. Qui si utilizzano le stesse procedure per lavorare con i tag utilizzate per le istanze database Aurora o RDS. È possibile configurare i tag contemporaneamente all'aggiunta di una nuova istanza database al cluster Aurora utilizzando l'azione Aggiungi lettura nella console. È possibile anche aggiungere, rimuovere o modificare i tag e i relativi valori associati scegliendo Gestisci tag nella scheda Tag all'interno della pagina dei dettagli dell'istanza database.

Ecco alcuni esempi dei tipi di tag che potresti assegnare all'interno di un database globale:

- È possibile aggiungere tag al cluster globale per registrare informazioni generali sull'applicazione, ad esempio identificatori anonimi che rappresentano proprietari e contatti all'interno dell'organizzazione. È possibile utilizzare i tag per rappresentare le proprietà dell'applicazione che utilizza il database globale.
- È possibile aggiungere tag al cluster primario e ai cluster secondari per tenere traccia dei costi dell'applicazione a livello di Regione AWS. Per i dettagli sulla procedura, consulta Come funziona la AWS fatturazione con i tag in Amazon RDS.
- È possibile aggiungere tag a istanze database specifiche con i cluster Aurora per indicarne lo scopo speciale. Ad esempio, all'interno del cluster primario, potresti avere un'istanza di lettura con una priorità di failover bassa che viene utilizzata esclusivamente per la generazione di report. Un

tag può distinguere questa istanza database per scopi speciali da altre istanze dedicate all'elevata disponibilità all'interno del cluster primario.

- È possibile utilizzare i tag a tutti i livelli delle risorse globali del database per controllare l'accesso tramite le policy IAM. Per ulteriori informazioni, consulta Controllo degli accessi alle risorse AWS nella Guida per l'utente di AWS Identity and Access Management.

> **Tip**
>
> In Console di gestione AWS, aggiungi tag al container del cluster globale come passaggio separato dopo averlo creato. Se desideri evitare intervalli di tempo in cui il cluster globale esiste senza tag di controllo degli accessi, puoi applicare i tag durante l'operazione `CreateGlobalCluster` creando quella risorsa tramite AWS CLI, l'API RDS o un modello CloudFormation.

- È possibile utilizzare i tag a livello di cluster o per il cluster globale per registrare informazioni sul controllo della qualità e sui test dell'applicazione. Ad esempio, è possibile specificare un tag su un cluster di database per registrare l'ultima volta che è stato eseguito uno switchover a quel cluster. È possibile specificare un tag sul cluster globale per registrare l'ora dell'ultima esercitazione di disaster recovery per l'intera applicazione.

Esempi AWS CLI di applicazione di tag per database globali

Gli esempi AWS CLI seguenti mostrano come specificare ed esaminare i tag per tutti i tipi di risorse Aurora nel database globale.

È possibile specificare i tag per il container del cluster globale con il comando `create-global-cluster`. Nell'esempio seguente viene creato un cluster globale a cui vengono assegnati due tag. I tag hanno chiavi `tag1` e `tag2`.

```
$  aws rds create-global-cluster --global-cluster-identifier my_global_cluster_id \
  --engine aurora-mysql --tags Key=tag1,Value=val1 Key=tag2,Value=val2
```

È possibile elencare i tag nel container del cluster globale con il comando `describe-global-clusters`. Quando si utilizzano i tag, in genere si esegue prima questo comando per recuperare il nome della risorsa Amazon (ARN) del cluster globale. L'ARN viene utilizzato come parametro nei comandi successivi per l'utilizzo dei tag. Il comando seguente mostra le informazioni sui tag nell'attributo `TagList`. Mostra anche l'ARN, utilizzato come parametro negli esempi successivi.

```
$  aws rds describe-global-clusters --global-cluster-identifier my_global_cluster_id
{
    "GlobalClusters": [
        {
            "Status": "available",
            "Engine": "aurora-mysql",
            "GlobalClusterArn": "my_global_cluster_arn",
            ...
            "TagList": [
                {
                    "Value": "val1",
                    "Key": "tag1"
                },
                {
                    "Value": "val2",
                    "Key": "tag2"
                }
            ]
        }
    ]
}
```

È possibile aggiungere nuovi tag con il comando `add-tags-to-resource`. Con questo comando, si specifica l'ARN del cluster globale anziché il suo identificatore. Aggiungendo un tag con lo stesso nome di un tag esistente viene sovrascritto il valore di quel tag. Se includi spazi o caratteri speciali nei valori dei tag, racchiudi i valori tra virgolette in base al sistema operativo o alla shell dei comandi in uso. L'esempio seguente modifica i tag del cluster globale rispetto all'esempio precedente. Originariamente, il cluster aveva tag con chiavi `tag1` e `tag2`. Al termine del comando, il cluster globale ha un nuovo tag con chiave `tag3` e il tag con chiave `tag1` ha un valore diverso.

```
$  aws rds add-tags-to-resource --resource-name my_global_cluster_arn \
  --tags Key=tag1,Value="new value for tag1" Key=tag3,Value="entirely new tag"

$  aws rds describe-global-clusters --global-cluster-identifier my_global_cluster_id
{
    "GlobalClusters": [
        {
            "Status": "available",
            "Engine": "aurora-mysql",
            ...
            "TagList": [
                {
```

```
                    "Value": "new value for tag1",
                    "Key": "tag1"
                },
                {
                    "Value": "val2",
                    "Key": "tag2"
                },
                {
                    "Value": "entirely new tag",
                    "Key": "tag3"
                }
            ]
        }
    ]
}
```

È possibile eliminare un tag dal cluster globale con il comando `remove-tags-from-resource`. Con questo comando, si specifica solo un set di chiavi di tag, senza alcun valore di tag. L'esempio seguente modifica i tag del cluster globale rispetto all'esempio precedente. Originariamente, il cluster aveva tag con chiavi `tag1`, `tag2` e `tag3`. Al termine del comando, rimane solo il tag con la chiave `tag1`.

```
$  aws rds remove-tags-from-resource --resource-name my_global_cluster_arn --tag-keys
 tag2 tag3

$  aws rds describe-global-clusters --global-cluster-identifier my_global_cluster_id
{
    "GlobalClusters": [
        {
            "Status": "available",
            "Engine": "aurora-mysql",
            ...
            "TagList": [
                {
                    "Value": "new value for tag1",
                    "Key": "tag1"
                }
            ]
        }
    ]
}
```

Connessione a Database globale Amazon Aurora

Ogni Database globale Aurora è dotato di un endpoint di scrittura che viene aggiornato automaticamente da Aurora per instradare le richieste all'istanza di scrittura corrente del cluster di database primario. Con l'endpoint di scrittura, non è necessario modificare la stringa di connessione dopo aver cambiato la posizione della Regione primaria utilizzando le funzionalità gestite di switchover e failover del Database globale Aurora. Per ulteriori informazioni sull'utilizzo dell'endpoint di scrittura insieme allo switchover e al failover del Database globale Aurora, consulta Utilizzo dello switchover o failover in un Database globale Amazon Aurora. Per informazioni sulla connessione a un Database globale Aurora con Server proxy per RDS, consulta Utilizzo del Server proxy per RDS con i database globali Aurora.

Argomenti

- Scelta dell'endpoint che soddisfa le esigenze dell'applicazione
- Visualizzazione degli endpoint di un database globale Amazon Aurora
- Considerazioni sull'utilizzo degli endpoint di scrittura globali

Scelta dell'endpoint che soddisfa le esigenze dell'applicazione

La connessione a un Database globale Aurora dipende dalla necessità di leggere o scrivere dal database e dalla Regione AWS a cui indirizzare le richieste. Ecco alcuni casi d'uso tipici:

- Instradamento delle richieste all'istanza di scrittura: connettiti all'endpoint di scrittura Database globale Aurora se devi eseguire istruzioni DML (Data Manipulation Language) e DDL (Data Definition Language) o se ti serve una forte coerenza tra letture e scritture. Tale endpoint instrada le richieste all'istanza di scrittura nel cluster primario del database globale. Questo endpoint viene aggiornato automaticamente per instradare le richieste all'istanza di scrittura, eliminando la necessità di aggiornare l'applicazione ogni volta che si modifica la posizione di scrittura nel cluster globale. Puoi anche utilizzare l'endpoint globale per inviare richieste di lettura/scrittura tra Regioni diverse al tuo endpoint di scrittura.

  > **Note**
  >
  > Se hai configurato il database globale prima che l'endpoint di scrittura del Database globale Aurora fosse disponibile, l'applicazione potrebbe connettersi all'endpoint cluster del cluster primario. In questo caso, è consigliabile modificare le impostazioni di connessione in modo da utilizzare l'endpoint di scrittura globale. In questo modo si evita la necessità di

modificare le impostazioni di connessione dopo ogni switchover o failover di Database globale Aurora.
La prima parte del nome dell'endpoint di scrittura è il nome del Database globale Aurora. Pertanto, se il Database globale Aurora viene rinominato, il nome dell'endpoint di scrittura cambia ed è necessario aggiornare tutto il codice che lo utilizza con il nuovo nome.

- Scalare le letture più vicino alla Regione dell'applicazione: per scalare le richieste di sola lettura nella stessa Regione AWS dell'applicazione o in una Regione vicina, connettiti all'endpoint di lettura dei cluster Aurora primari o secondari.
- Scalare le letture con scritture occasionali tra Regioni: per istruzioni DML occasionali, ad esempio per la manutenzione e la pulizia dei dati, connettiti all'endpoint di lettura di un cluster secondario con l'inoltro di scrittura abilitato. Con l'inoltro di scrittura, Aurora inoltra automaticamente le istruzioni di scrittura all'endpoint di scrittura nella Regione primaria del Database globale Aurora. L'inoltro di scrittura offre i seguenti vantaggi:
  - Non sono necessarie operazioni complesse per stabilire la connettività tra i cluster secondari e primari per inviare scritture tra Regioni.
  - Non è necessario dividere le richieste di lettura e scrittura nell'applicazione.
  - Non è necessario sviluppare una logica complessa per gestire la coerenza delle richieste di lettura dopo la scrittura.

  Tuttavia, con l'inoltro di scrittura, è necessario aggiornare il codice o la configurazione dell'applicazione per connettersi all'endpoint di lettura della Regione primaria appena promossa dopo avere eseguito un failover o uno switchover tra Regioni. È consigliabile monitorare la latenza delle operazioni eseguite tramite l'inoltro di scrittura per controllare il sovraccarico di elaborazione delle richieste di scrittura. Infine, l'inoltro di scrittura non supporta determinate operazioni di MySQL o PostgreSQL, come l'esecuzione di modifiche DDL (Data Definition Language) o istruzioni `SELECT FOR UPDATE`.

  Per ulteriori informazioni sull'utilizzo dell'inoltro di scrittura tra Regioni AWS, consulta Utilizzo dell'inoltro di scrittura in un database globale Amazon Aurora.

Per dettagli sui diversi tipi di endpoint Aurora, consulta Connessione a un cluster database Amazon Aurora.

Visualizzazione degli endpoint di un database globale Amazon Aurora

Quando nella console è presente un Database globale Aurora, è possibile visualizzare tutti gli endpoint generici associati a tutti i relativi cluster. La figura seguente mostra un esempio dei tipi di endpoint che vengono mostrati quando si visualizzano i dettagli del cluster di database primario:

- Endpoint di scrittura globale: il singolo endpoint di lettura/scrittura che punta sempre all'istanza database di scrittura corrente per il cluster di database globale.
- Endpoint di scrittura: l'endpoint di connessione per le richieste di lettura/scrittura al cluster di database primario nel cluster di database globale.
- Endpoint di lettura: l'endpoint di connessione per le richieste di sola lettura a un cluster di database primario o secondario nel cluster di database globale. Per ridurre al minimo la latenza, scegli qualsiasi endpoint di lettura che si trova nella tua Regione AWS o nella Regione AWS più vicina.



Console

Per visualizzare gli endpoint di un database globale

1. Accedi alla Console di gestione AWS e apri la console Amazon RDS all'indirizzo https://console.aws.amazon.com/rds/.
2. Nel riquadro di navigazione, scegliere Databases (Database).
3. Nell'elenco scegli il database globale o il cluster di database primario o secondario di cui vuoi visualizzare gli endpoint.
4. Scegli la scheda Connettività e sicurezza per visualizzare i dettagli dell'endpoint. Gli endpoint visualizzati dipendono dal tipo di cluster selezionato, come indicato di seguito:

   - Database globale: l'endpoint di scrittura globale.
   - Cluster di database primario: l'endpoint di scrittura globale e l'endpoint del cluster e l'endpoint di lettura per il cluster primario.
   - Cluster di database secondario: l'endpoint del cluster e l'endpoint di lettura per il cluster secondario. In un cluster secondario, lo stato dell'endpoint del cluster è inattivo perché non gestisce le richieste di scrittura. È comunque possibile connettersi all'endpoint del cluster, ma solo per le query di lettura.

AWS CLI

Per visualizzare l'endpoint di scrittura del cluster globale, usa il comando AWS CLI describe-global-clusters, come nell'esempio seguente.

```
aws rds describe-global-clusters --region aws_region
{
    "GlobalClusters": [
        {
            "GlobalClusterIdentifier": "global_cluster_id",
            "GlobalClusterResourceId": "cluster-unique_string",
            "GlobalClusterArn": "arn:aws:rds::123456789012:global-
cluster:global_cluster_id",
            "Status": "available",
            "Engine": "aurora-mysql",
            "EngineVersion": "5.7.mysql_aurora.2.11.2",
            "GlobalClusterMembers": [

              ...
```

```
            ],
            "Endpoint":
 "global_cluster_id.global-unique_string.global.rds.amazonaws.com"
        }
    ]
}
```

Per visualizzare gli endpoint del cluster e di lettura per i cluster di database che fanno parte del cluster globale, usa il comando AWS CLI describe-db-clusters, come nell'esempio seguente. I valori restituiti per `Endpoint` e `ReaderEndpoint` sono rispettivamente gli endpoint del cluster e di lettura.

```
aws rds describe-db-clusters --region primary_region --db-cluster-
identifier db_cluster_id
{
    "DBClusters": [
        {
            "AllocatedStorage": 1,
            "AvailabilityZones": [
                "az_1",
                "az_2",
                "az_3"
            ],
            "BackupRetentionPeriod": 1,
            "DBClusterIdentifier": "db_cluster_id",
            "DBClusterParameterGroup": "default.aurora-mysql5.7",
            "DBSubnetGroup": "default",
            "Status": "available",
            "EarliestRestorableTime": "2023-08-01T18:21:11.301Z",
            "Endpoint":
 "db_cluster_id.cluster-unique_string.primary_region.rds.amazonaws.com",
            "ReaderEndpoint": "db_cluster_id.cluster-
ro-unique_string.primary_region.rds.amazonaws.com",
            "MultiAZ": false,
            "Engine": "aurora-mysql",
            "EngineVersion": "5.7.mysql_aurora.2.11.2",

            "ReadReplicaIdentifiers": [
                "arn:aws:rds:secondary_region:123456789012:cluster:db_cluster_id"
            ],
            "DBClusterMembers": [
                {
                    "DBInstanceIdentifier": "db_instance_id",
                    "IsClusterWriter": true,
```

```
                    "DBClusterParameterGroupStatus": "in-sync",
                    "PromotionTier": 1
                }
            ],

            ...
            "TagList": [],
            "GlobalWriteForwardingRequested": false
        }
    ]
}
```

API RDS

Per visualizzare l'endpoint di scrittura del cluster globale, utilizza l'operazione DescribeGlobalClusters dell'API RDS. Per visualizzare gli endpoint del cluster e di lettura per i cluster di database che fanno parte del cluster globale, utilizza l'operazione DescribeDBClusters dell'API RDS.

Considerazioni sull'utilizzo degli endpoint di scrittura globali

Per utilizzare gli endpoint di scrittura del Database globale Aurora in modo efficiente, segui queste linee guida e best practice:

- Per ridurre al minimo le interruzioni dopo un failover o uno switchover tra Regioni, puoi configurare la connettività VPC tra l'elaborazione dell'applicazione e le Regioni AWS primarie e secondarie. Ad esempio, supponiamo di avere applicazioni o sistemi client in esecuzione nello stesso VPC del cluster primario. Se il cluster secondario viene promosso, l'endpoint di scrittura globale cambia automaticamente in modo da puntare a quel cluster. Sebbene l'endpoint di scrittura globale consenta di evitare di modificare le impostazioni di connessione per l'applicazione, le applicazioni non sono in grado di accedere agli indirizzi IP nel VPC della Regione AWS primaria appena promossa finché non si configura la connessione di rete tra i due VPC. Per valutare le diverse opzioni per configurare questa connettività, consulta Opzioni di connettività da Amazon VPC ad Amazon VPC.
- L'aggiornamento dell'endpoint di scrittura globale dopo uno switchover o un failover globale del database può richiedere molto tempo a seconda della durata della memorizzazione nella cache DNS (Domain Name Service). Per ulteriori informazioni, consulta Amazon Aurora MySQL Database Administrator's Handbook. Il Database globale Aurora emette un evento RDS quando rileva la modifica del DNS sull'endpoint di scrittura globale. È possibile utilizzare l'evento per elaborare strategie per garantire che la cache DNS non si estenda oltre il periodo successivo alla generazione dell'evento. Per ulteriori informazioni, consulta Eventi di cluster di database.

- Il Database globale Aurora replica i dati in modo asincrono. I metodi di failover tra Regioni possono comportare la presenza di dati delle transazioni di scrittura non replicati nella secondaria scelta prima dell'inizio del failover. Sebbene Aurora tenti con il massimo impegno di bloccare le scritture nella Regione AWS primaria originale, il failover può essere soggetto a problemi di split-brain. Le considerazioni per ridurre al minimo la perdita di dati e il rischio di split-brain si applicano anche agli endpoint di scrittura del Database globale Aurora. Per ulteriori informazioni, consulta Esecuzione di failover gestiti per database globali Aurora.

Utilizzo dell'inoltro di scrittura in un database globale Amazon Aurora

È possibile ridurre il numero di endpoint che è necessario gestire per le applicazioni in esecuzione nel database globale Aurora utilizzando l' inoltro in scrittura. Quando la funzionalità di inoltro di scrittura è abilitata, i cluster secondari di un database globale Aurora inoltrano le istruzioni SQL che eseguono operazioni di scrittura al cluster primario. Il cluster primario aggiorna l'origine e quindi propaga le modifiche risultanti a tutte le aree secondarie AWS.

La funzione di inoltro di scrittura consente di evitare di implementare il proprio meccanismo per inviare operazioni di scrittura da una regione AWS secondaria alla regione principale. Aurora gestisce la configurazione di rete tra regioni. Aurora trasmette anche tutte le sessioni necessarie e il contesto transazionale per ogni dichiarazione. I dati vengono sempre modificati prima nel cluster primario e quindi replicati nuovamente nei cluster secondari nel database globale Aurora. In questo modo, il cluster primario è sempre la fonte di attendibilità con la copia più aggiornata di tutti i dati.

Argomenti

- Utilizzo dell'inoltro di scrittura in un database globale Aurora MySQL
- Utilizzo dell'inoltro di scrittura in un database globale Aurora PostgreSQL

Utilizzo dell'inoltro di scrittura in un database globale Aurora MySQL

Argomenti

- Disponibilità di regioni e versioni dell'inoltro di scrittura in Aurora MySQL
- Abilitazione dell'inoltro di scrittura in Aurora MySQL
- Verifica se un cluster secondario ha abilitato l'inoltro di scrittura
- Compatibilità delle applicazioni e di SQL con l'inoltro di scrittura in Aurora MySQL

- Isolamento e coerenza per l'inoltro di scrittura in Aurora MySQL
- Esecuzione di istruzioni a più parti con inoltro scrittura in Aurora MySQL
- Transazioni con inoltro di scrittura in Aurora MySQL
- Parametri di configurazione per l'inoltro di scrittura in Aurora MySQL
- Parametri Amazon CloudWatch per l'inoltro di scrittura in Aurora MySQL
- Variabili di stato di Aurora MySQL per l'inoltro di scrittura

Disponibilità di regioni e versioni dell'inoltro di scrittura in Aurora MySQL

L'inoltro della scrittura è supportato con Aurora MySQL 2.08.1 e versioni successive in ogni regione in cui sono disponibili database globali basati su Aurora MySQL.

Per ulteriori informazioni sulla disponibilità di versioni e regioni dei database globali Aurora MySQL consulta Database globali Aurora con Aurora MySQL.

Abilitazione dell'inoltro di scrittura in Aurora MySQL

Per impostazione predefinita, l'inoltro di scrittura non è abilitato quando si aggiunge un cluster secondario a un database globale Aurora.

Per abilitare l'inoltro di scrittura mediante Console di gestione AWS, seleziona la casella di controllo Attiva l'inoltro di scrittura globale in Inoltro di scrittura di repliche di lettura quando aggiungi una regione per un database globale. Per un cluster secondario esistente, modifica il cluster in Attiva l'inoltro di scrittura globale. Per disattivare l'inoltro di scrittura, deseleziona la casella di controllo Attiva l'inoltro di scrittura globale durante l'aggiunta della regione o la modifica del cluster secondario.

Per abilitare l'inoltro di scrittura utilizzando l'AWS CLI, utilizzare l'opzione `--enable-global-write-forwarding`. Questa opzione funziona quando si crea un nuovo cluster secondario utilizzando il comando `create-db-cluster`. Funziona anche quando si modifica un cluster secondario esistente utilizzando il comando `modify-db-cluster`. Richiede che il database globale utilizzi una versione Aurora che supporti l'inoltro di scrittura. È possibile disattivare l'inoltro di scrittura utilizzando l'opzione `--no-enable-global-write-forwarding` con questi stessi comandi CLI.

Per abilitare l'inoltro di scrittura utilizzando l'API Amazon RDS, impostare il parametro `EnableGlobalWriteForwarding` su `true`. Questo parametro funziona quando si crea un nuovo cluster secondario utilizzando l'operazione `CreateDBCluster`. Funziona anche quando si modifica un cluster secondario esistente utilizzando l'operazione `ModifyDBCluster`. Richiede che il

database globale utilizzi una versione Aurora che supporti l'inoltro di scrittura. È possibile disattivare l'inoltro di scrittura impostando il parametro `EnableGlobalWriteForwarding` su `false`.

> **Note**
>
> Per utilizzare l'inoltro in scrittura in una sessione di database, specifica un'impostazione per il parametro di configurazione `aurora_replica_read_consistency`. Eseguire questa operazione in ogni sessione che utilizza la funzionalità di inoltro di scrittura. Per informazioni su questo parametro, consulta Isolamento e coerenza per l'inoltro di scrittura in Aurora MySQL.
>
> La funzionalità RDS Proxy non supporta il valore `SESSION` della variabile `aurora_replica_read_consistency`, la cui impostazione può causare un comportamento imprevisto.

Negli esempi seguenti di CLI viene illustrato come impostare un database globale Aurora con l'inoltro di scrittura abilitato o disabilitato. Gli elementi evidenziati rappresentano i comandi e le opzioni che è importanti specificare e mantenere coerenti quando si imposta l'infrastruttura per un Aurora Global Database.

Nell'esempio seguente viene creato un Aurora Global Database, un cluster primario e un cluster secondario con l'inoltro di scrittura abilitato. Sostituire il nome utente, la password e le regioni AWS primarie e secondarie.

```
# Create overall global database.
aws rds create-global-cluster --global-cluster-identifier write-forwarding-test \
  --engine aurora-mysql --engine-version 5.7.mysql_aurora.2.11.1 \
  --region us-east-1

# Create primary cluster, in the same AWS Region as the global database.
aws rds create-db-cluster --global-cluster-identifier write-forwarding-test \
  --db-cluster-identifier write-forwarding-test-cluster-1 \
  --engine aurora-mysql --engine-version 5.7.mysql_aurora.2.11.1 \
  --master-username user_name --master-user-password password \
  --region us-east-1

aws rds create-db-instance --db-cluster-identifier write-forwarding-test-cluster-1 \
  --db-instance-identifier write-forwarding-test-cluster-1-instance-1 \
  --db-instance-class db.r5.large \
  --engine aurora-mysql --engine-version 5.7.mysql_aurora.2.11.1 \
```

```
  --region us-east-1

aws rds create-db-instance --db-cluster-identifier write-forwarding-test-cluster-1 \
  --db-instance-identifier write-forwarding-test-cluster-1-instance-2 \
  --db-instance-class db.r5.large \
  --engine aurora-mysql --engine-version 5.7.mysql_aurora.2.11.1 \
  --region us-east-1

# Create secondary cluster, in a different AWS Region than the global database,
# with write forwarding enabled.
aws rds create-db-cluster --global-cluster-identifier write-forwarding-test \
  --db-cluster-identifier write-forwarding-test-cluster-2 \
  --engine aurora-mysql --engine-version 5.7.mysql_aurora.2.11.1 \
  --region us-east-2 \
  --enable-global-write-forwarding

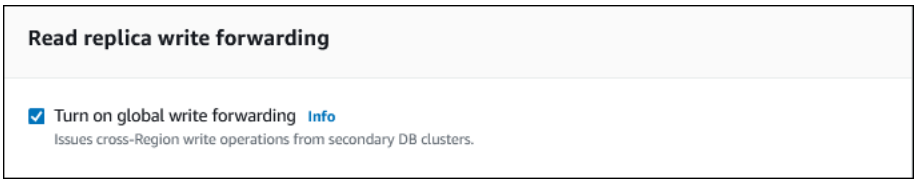
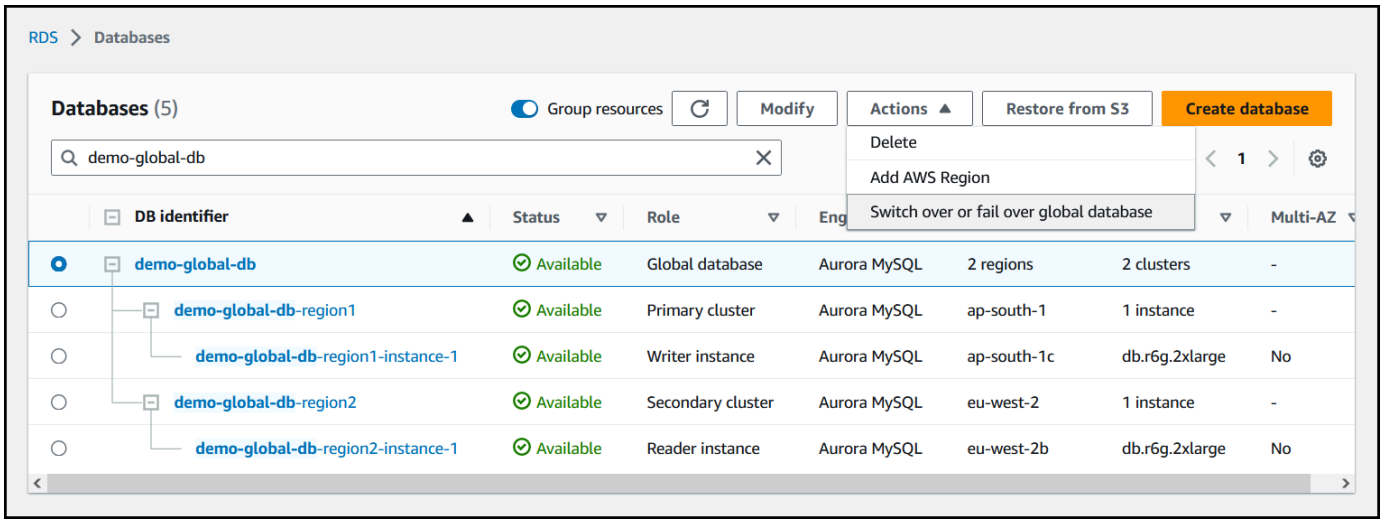
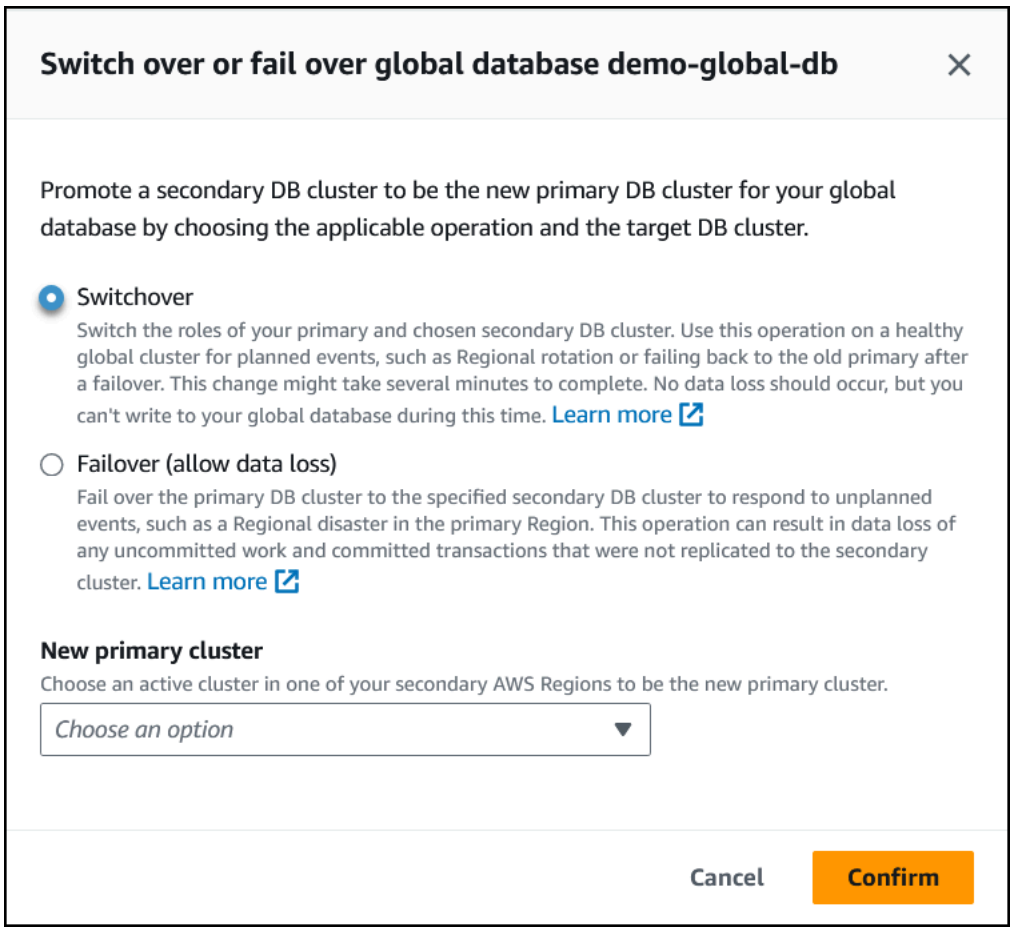
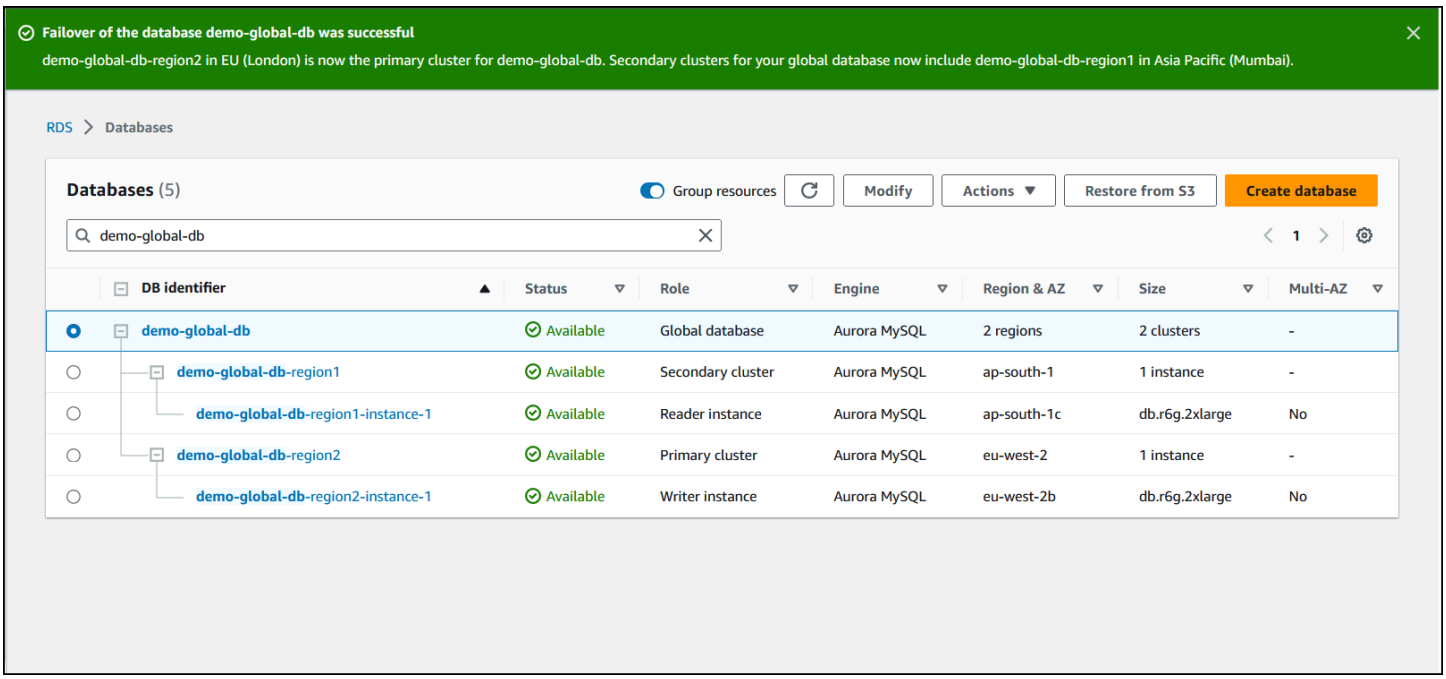
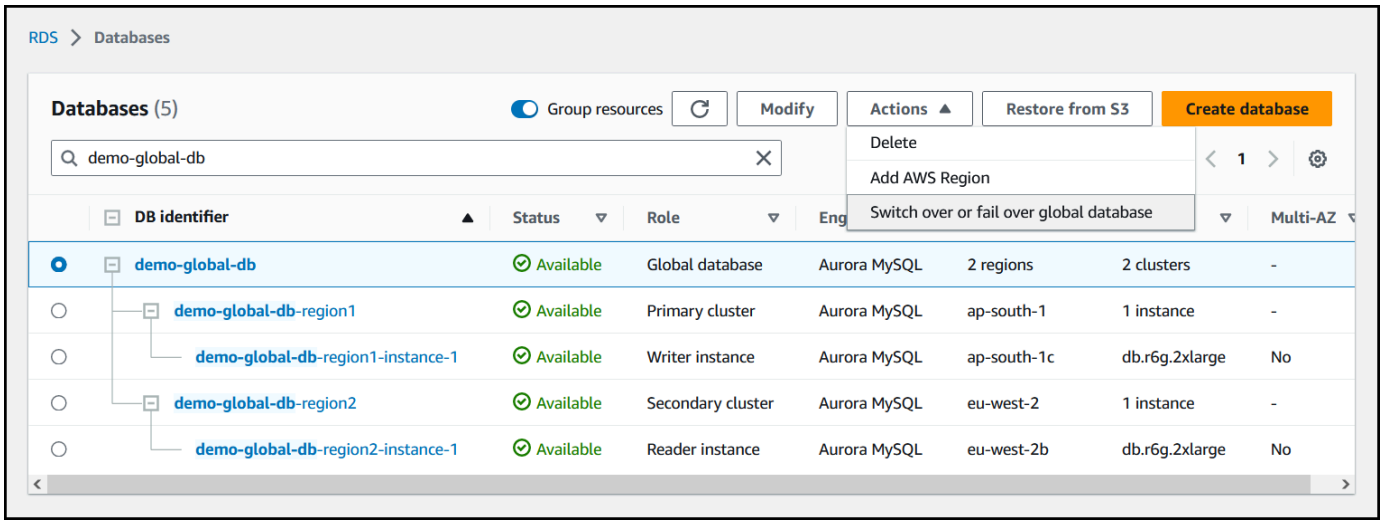
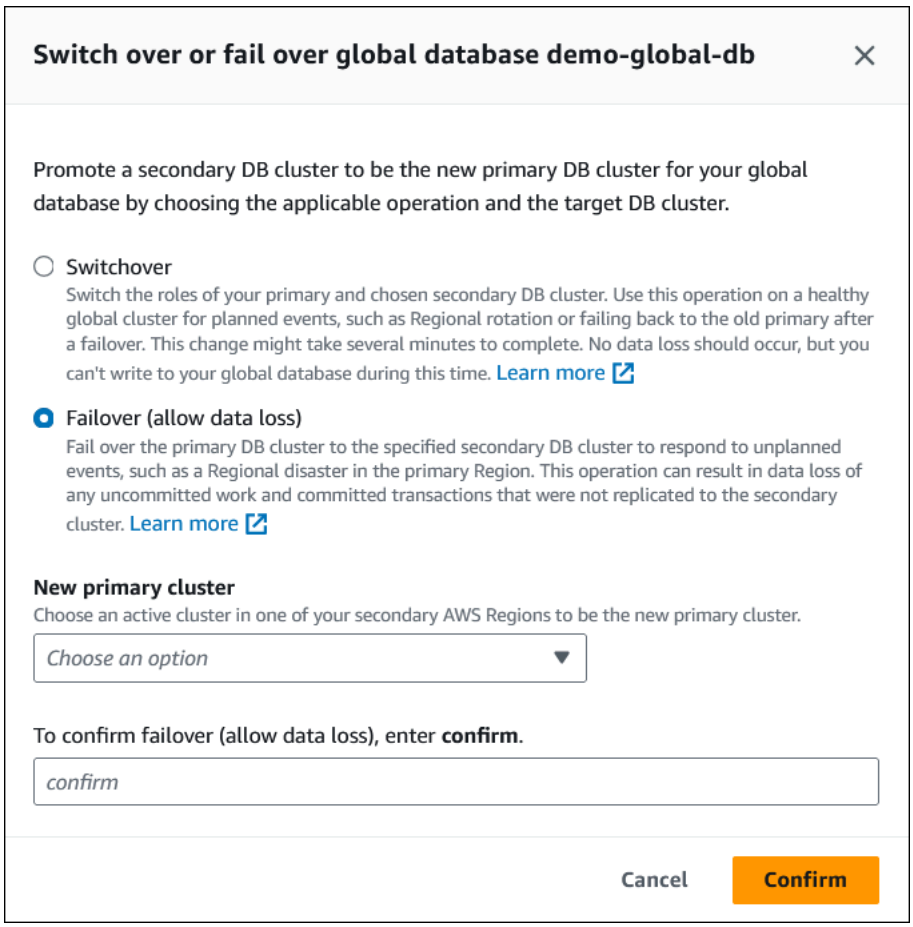
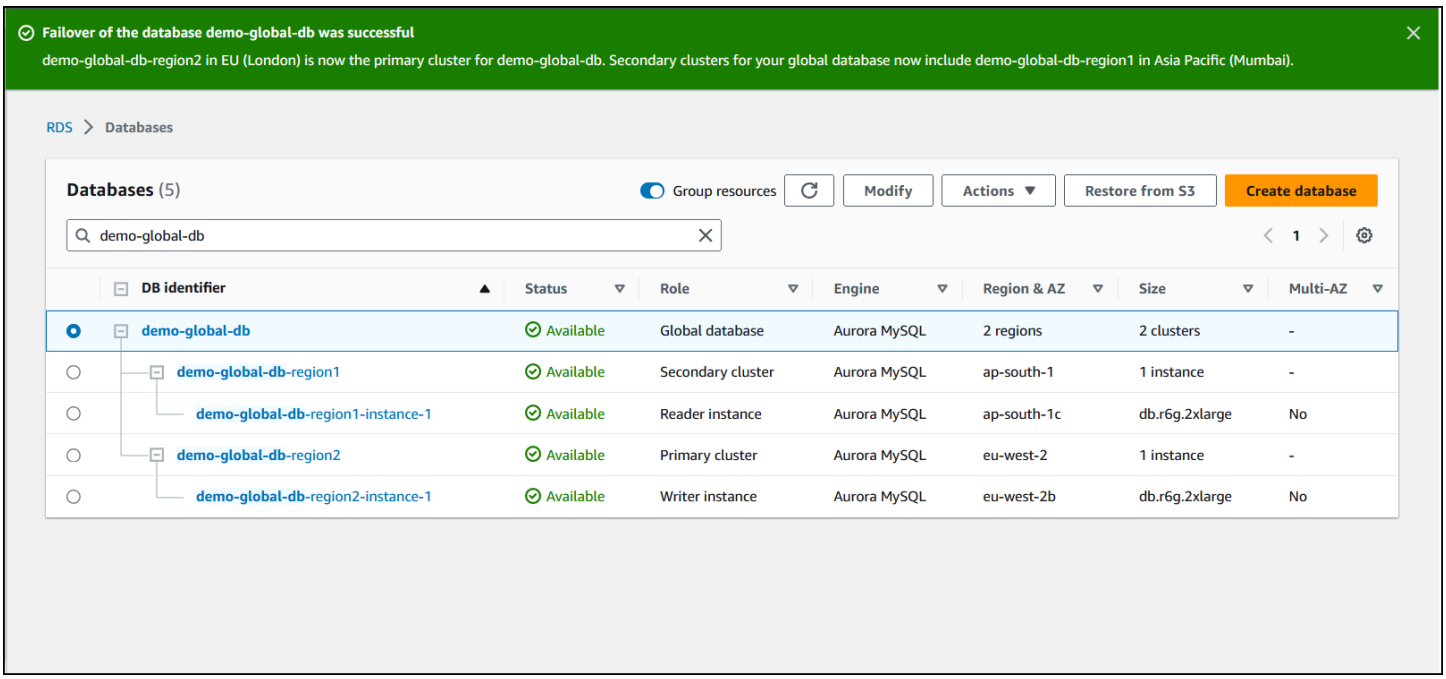
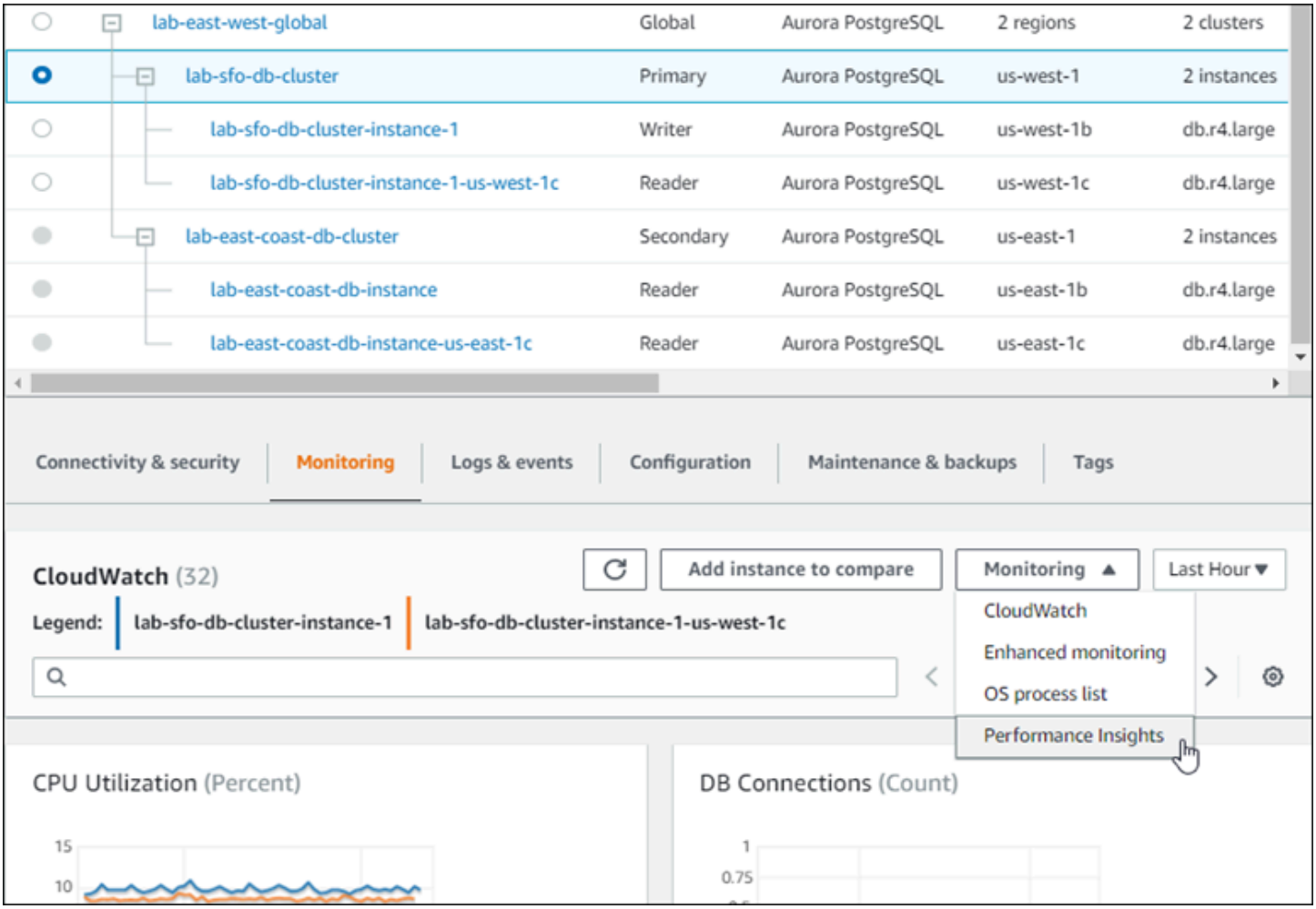
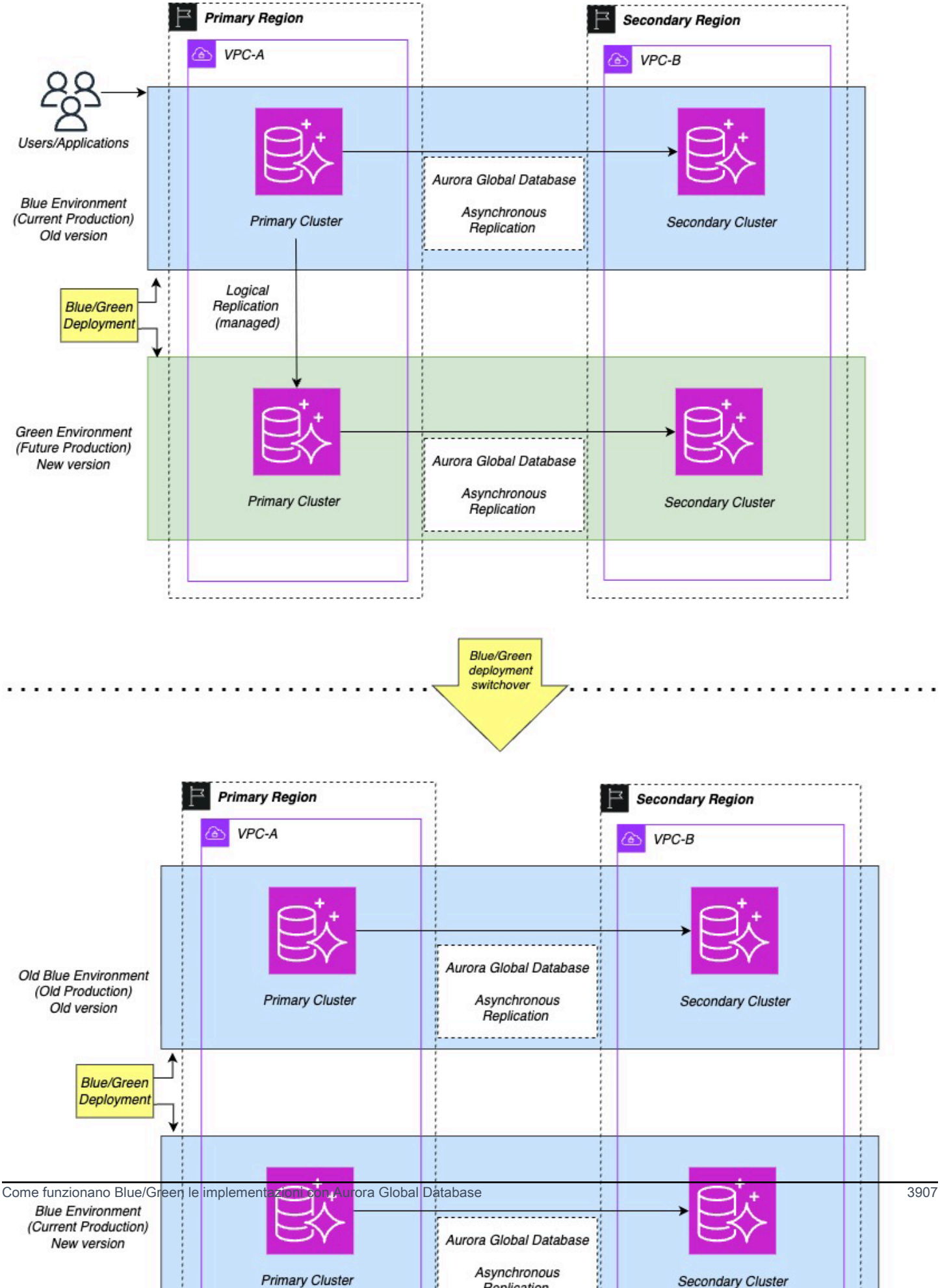
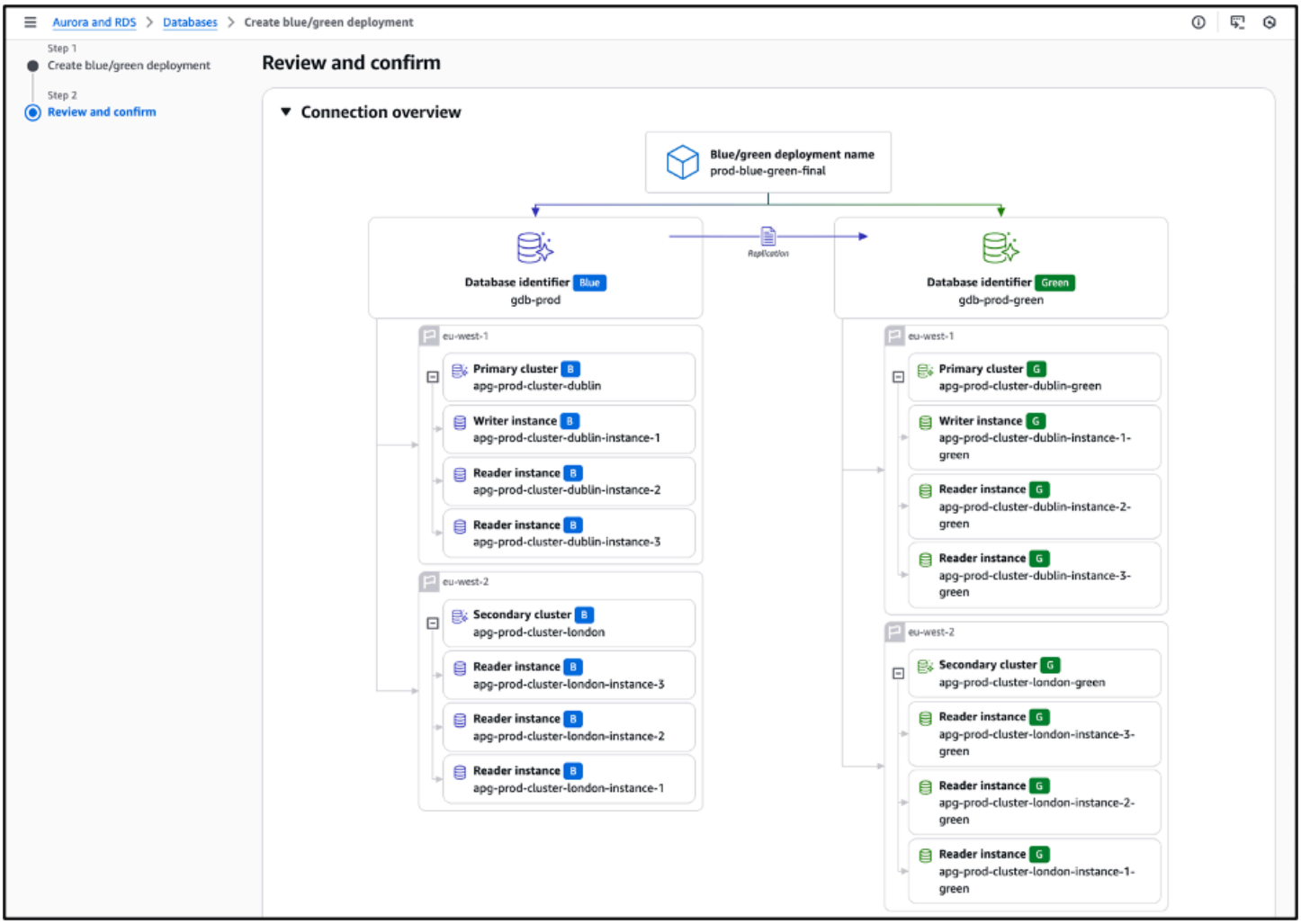
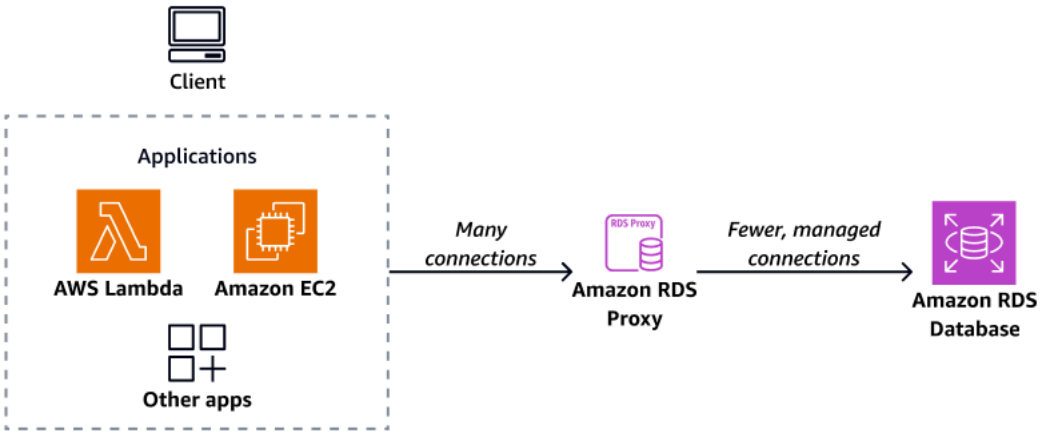
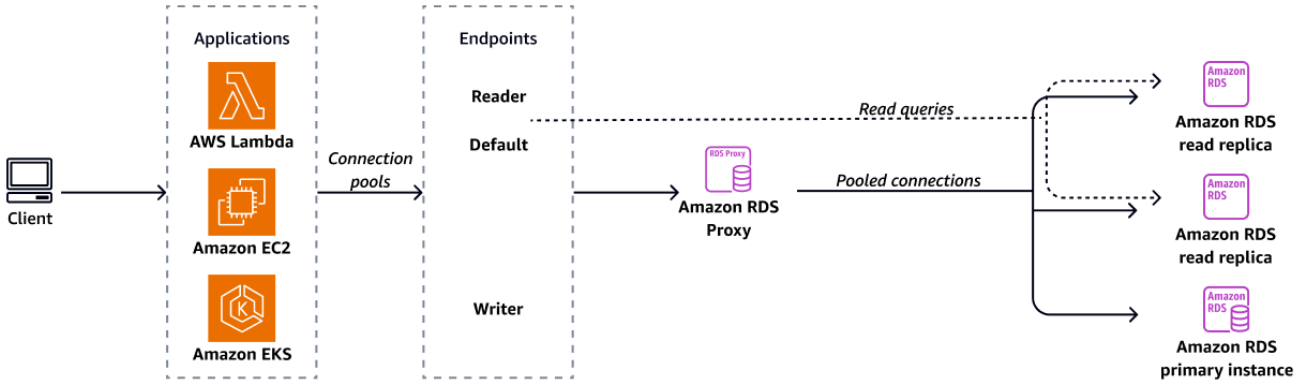
aws rds create-db-instance --db-cluster-identifier write-forwarding-test-cluster-2 \
  --db-instance-identifier write-forwarding-test-cluster-2-instance-1 \
  --db-instance-class db.r5.large \
  --engine aurora-mysql --engine-version 5.7.mysql_aurora.2.11.1 \
  --region us-east-2

aws rds create-db-instance --db-cluster-identifier write-forwarding-test-cluster-2 \
  --db-instance-identifier write-forwarding-test-cluster-2-instance-2 \
  --db-instance-class db.r5.large \
  --engine aurora-mysql --engine-version 5.7.mysql_aurora.2.11.1 \
  --region us-east-2
```

L'esempio seguente continua dal precedente. Crea un cluster secondario senza l'inoltro di scrittura abilitato, quindi abilita l'inoltro di scrittura. Al termine di questo esempio, tutti i cluster secondari del database globale hanno attivato l'inoltro di scrittura.

```
# Create secondary cluster, in a different AWS Region than the global database,
# without write forwarding enabled.
aws rds create-db-cluster --global-cluster-identifier write-forwarding-test \
  --db-cluster-identifier write-forwarding-test-cluster-2 \
  --engine aurora-mysql --engine-version 5.7.mysql_aurora.2.11.1 \
  --region us-west-1

aws rds create-db-instance --db-cluster-identifier write-forwarding-test-cluster-2 \
  --db-instance-identifier write-forwarding-test-cluster-2-instance-1 \
  --db-instance-class db.r5.large \
  --engine aurora-mysql --engine-version 5.7.mysql_aurora.2.11.1 \
```

```
  --region us-west-1

aws rds create-db-instance --db-cluster-identifier write-forwarding-test-cluster-2 \
  --db-instance-identifier write-forwarding-test-cluster-2-instance-2 \
  --db-instance-class db.r5.large \
  --engine aurora-mysql --engine-version 5.7.mysql_aurora.2.11.1 \
  --region us-west-1

aws rds modify-db-cluster --db-cluster-identifier write-forwarding-test-cluster-2 \
  --region us-east-2 \
  --enable-global-write-forwarding
```

Verifica se un cluster secondario ha abilitato l'inoltro di scrittura

Per determinare se è possibile utilizzare l'inoltro di scrittura da un cluster secondario, è possibile verificare se il cluster dispone dell'attributo `"GlobalWriteForwardingStatus": "enabled"`.

Nella Console di gestione AWS, nella scheda Configurazione della pagina dei dettagli del cluster, viene visualizzato lo stato Abilitato per Inoltro di scrittura di repliche di lettura globali.

Per visualizzare lo stato dell'impostazione di inoltro di scrittura globale per tutti i cluster, eseguire il comando AWS CLI seguente.

Un cluster secondario mostra il valore `"enabled"` o `"disabled"` per indicare se l'inoltro di scrittura è attivato o disattivato. Il valore `null` indica che l'inoltro di scrittura non è disponibile per il cluster. Il cluster non fa parte di un database globale oppure è il cluster primario anziché un cluster secondario. Il valore può anche essere `"enabling"` o `"disabling"` se l'inoltro di scrittura è in procinto di essere attivato o disattivato.

Example

```
aws rds describe-db-clusters \
--query '*[].
{DBClusterIdentifier:DBClusterIdentifier,GlobalWriteForwardingStatus:GlobalWriteForwardingStatu

[
    {
        "GlobalWriteForwardingStatus": "enabled",
        "DBClusterIdentifier": "aurora-write-forwarding-test-replica-1"
    },
    {
        "GlobalWriteForwardingStatus": "disabled",
```

```
        "DBClusterIdentifier": "aurora-write-forwarding-test-replica-2"
    },
    {
        "GlobalWriteForwardingStatus": null,
        "DBClusterIdentifier": "non-global-cluster"
    }
]
```

Per trovare tutti i cluster secondari per i quali è abilitato l'inoltro di scrittura globale, eseguire il comando seguente. Questo comando restituisce anche l'endpoint di lettura del cluster. È possibile utilizzare l'endpoint di lettura del cluster secondario quando si utilizza l'inoltro di scrittura dal cluster secondario al primario nel database globale Aurora.

Example

```
aws rds describe-db-clusters --query 'DBClusters[].
{DBClusterIdentifier:DBClusterIdentifier,GlobalWriteForwardingStatus:GlobalWriteForwardingStatu
 | [?GlobalWriteForwardingStatus == `enabled`]'
[
    {
        "GlobalWriteForwardingStatus": "enabled",
        "ReaderEndpoint": "aurora-write-forwarding-test-replica-1.cluster-ro-
cnpexample.us-west-2.rds.amazonaws.com",
        "DBClusterIdentifier": "aurora-write-forwarding-test-replica-1"
    }
]
```

Compatibilità delle applicazioni e di SQL con l'inoltro di scrittura in Aurora MySQL

È possibile utilizzare i seguenti tipi di istruzioni SQL con l'inoltro di scrittura:

- Istruzioni DML (Data Manipulation Language), ad esempio `INSERT`, `DELETE` e `UPDATE`. Esistono alcune restrizioni sulle proprietà di queste istruzioni che è possibile utilizzare con l'inoltro di scrittura, come descritto di seguito.
- Istruzioni `SELECT ... LOCK IN SHARE MODE` e `SELECT FOR UPDATE`.
- Istruzioni `PREPARE` e `EXECUTE`.

Alcune istruzioni non sono consentite o possono produrre risultati non aggiornati quando vengono utilizzate in un database globale con inoltro di scrittura. Pertanto, l'impostazione

`EnableGlobalWriteForwarding` è disattivata in modo predefinito per i cluster secondari. Prima di attivarlo, verificare che il codice dell'applicazione non sia interessato da nessuna di queste restrizioni.

Le seguenti restrizioni si applicano alle istruzioni SQL utilizzate con l'inoltro di scrittura. In alcuni casi, è possibile utilizzare le istruzioni su cluster secondari con l'inoltro di scrittura abilitato a livello di cluster. Questo approccio funziona se l'inoltro di scrittura non è attivato all'interno della sessione dal parametro di configurazione `aurora_replica_read_consistency`. Se si tenta di utilizzare un'istruzione quando non è consentito a causa dell'inoltro di scrittura, viene generato un messaggio di errore con il formato seguente.

```
ERROR 1235 (42000): This version of MySQL doesn't yet support 'operation with write
 forwarding'.
```

DDL (Data Definition Language)

Connettersi al cluster primario per eseguire le istruzioni DDL. Non è possibile eseguirle da istanze database di lettura.

Aggiornamento di una tabella permanente utilizzando i dati di una tabella temporanea

È possibile utilizzare tabelle temporanee in cluster secondari con l'inoltro di scrittura abilitato. Tuttavia, non è possibile utilizzare un'istruzione DML per modificare una tabella permanente se l'istruzione fa riferimento a una tabella temporanea. Ad esempio, non è possibile utilizzare un'istruzione `INSERT ... SELECT` che accetta i dati da una tabella temporanea. La tabella temporanea esiste nel cluster secondario e non è disponibile quando l'istruzione viene eseguita nel cluster primario.

Transazioni XA

Non è possibile utilizzare le istruzioni seguenti in un cluster secondario quando l'inoltro di scrittura è attivato all'interno della sessione. È possibile utilizzare queste istruzioni su cluster secondari per i quali non è abilitato l'inoltro di scrittura o all'interno di sessioni in cui l'impostazione `aurora_replica_read_consistency` è vuota. Prima di attivare l'inoltro di scrittura all'interno di una sessione, verificare se il codice utilizza queste istruzioni.

```
XA {START|BEGIN} xid [JOIN|RESUME]
XA END xid [SUSPEND [FOR MIGRATE]]
XA PREPARE xid
XA COMMIT xid [ONE PHASE]
XA ROLLBACK xid
XA RECOVER [CONVERT XID]
```

Istruzioni LOAD per tabelle permanenti

Non è possibile utilizzare le istruzioni seguenti in un cluster secondario con l'inoltro di scrittura abilitato.

```
LOAD DATA INFILE 'data.txt' INTO TABLE t1;
        LOAD XML LOCAL INFILE 'test.xml' INTO TABLE t1;
```

È possibile caricare i dati in una tabella temporanea in un cluster secondario. Tuttavia, assicurarsi di eseguire tutte le istruzioni LOAD che fanno riferimento a tabelle permanenti solo nel cluster primario.

Istruzioni plugin

Non è possibile utilizzare le istruzioni seguenti in un cluster secondario con l'inoltro di scrittura abilitato.

```
INSTALL PLUGIN example SONAME 'ha_example.so';
UNINSTALL PLUGIN example;
```

Istruzioni SAVEPOINT

Non è possibile utilizzare le istruzioni seguenti in un cluster secondario quando l'inoltro di scrittura è attivato all'interno della sessione. È possibile utilizzare queste istruzioni su cluster secondari per i quali non è abilitato l'inoltro di scrittura o all'interno di sessioni in cui l'impostazione `aurora_replica_read_consistency` è vuota. Controlla se il tuo codice utilizza queste istruzioni prima di attivare l'inoltro di scrittura all'interno di una sessione.

```
SAVEPOINT t1_save;
ROLLBACK TO SAVEPOINT t1_save;
RELEASE SAVEPOINT t1_save;
```

Isolamento e coerenza per l'inoltro di scrittura in Aurora MySQL

Nelle sessioni che utilizzano l'inoltro di scrittura, è possibile utilizzare solo il livello di isolamento `REPEATABLE READ`. Sebbene sia anche possibile utilizzare il livello di isolamento `READ COMMITTED` con cluster di sola lettura nelle regioni AWS secondarie, tale livello di isolamento non funziona con l'inoltro di scrittura. Per informazioni sui livelli di isolamento `REPEATABLE READ` e `READ COMMITTED`, consulta Livelli di isolamento di Aurora MySQL.

È possibile controllare il grado di coerenza di lettura in un cluster secondario. Il livello di coerenza di lettura determina la quantità di attesa di un cluster secondario prima di ogni operazione di lettura per garantire che alcune o tutte le modifiche vengano replicate dal cluster primario. È possibile regolare il livello di coerenza di lettura per garantire che tutte le operazioni di scrittura inoltrate dalla sessione siano visibili nel cluster secondario prima di qualsiasi query successiva. È inoltre possibile utilizzare questa impostazione per garantire che le query sul cluster secondario visualizzino sempre gli aggiornamenti più recenti dal cluster primario. Ciò si verifica anche per quelli inviati da altre sessioni o altri cluster. Per specificare questo tipo di comportamento per l'applicazione, scegliere un valore per il parametro a livello di sessione `aurora_replica_read_consistency`.

> **Important**
>
> Impostare sempre il parametro `aurora_replica_read_consistency` per qualsiasi sessione per la quale si desidera inoltrare le scritture. In caso contrario, Aurora non abilita l'inoltro di scrittura per quella sessione. Questo parametro ha un valore vuoto per impostazione predefinita, quindi scegli un valore specifico quando utilizzi questo parametro. Il parametro `aurora_replica_read_consistency` ha effetto solo sui cluster secondari in cui è abilitato l'inoltro di scrittura.
>
> Per Aurora MySQL versione 2 e versione 3 inferiori alla 3.04, usa `aurora_replica_read_consistency` come variabile di sessione. Per Aurora MySQL versione 3.04 e successive, è possibile usare `aurora_replica_read_consistency` come variabile di sessione o come un parametro del cluster database.

Per il parametro `aurora_replica_read_consistency`, è possibile specificare i valori `EVENTUAL`, `SESSION` e `GLOBAL`.

Aumentando il livello di coerenza, l'applicazione impiega più tempo in attesa della propagazione delle modifiche tra regioni AWS. È possibile scegliere il bilanciamento tra tempi di risposta rapidi e garantire che le modifiche apportate in altre posizioni siano completamente disponibili prima dell'esecuzione delle query.

Con la coerenza di lettura impostata su `EVENTUAL`, le query in una regione AWS secondaria che utilizza l'inoltro di scrittura potrebbero vedere dati leggermente obsoleti a causa del ritardo di replica. I risultati delle operazioni di scrittura nella stessa sessione non sono visibili fino a quando l'operazione di scrittura non viene eseguita nella regione primaria e replicata nella regione corrente. La query non attende la disponibilità dei risultati aggiornati. Pertanto, potrebbe recuperare i dati meno recenti o i dati aggiornati, a seconda della tempistica delle istruzioni e della quantità di ritardo di replica.

Con la coerenza di lettura impostata su `SESSION`, tutte le query in una regione AWS secondaria, che utilizza l'inoltro di scrittura, visualizzano i risultati di tutte le modifiche apportate in tale sessione. Le modifiche sono visibili indipendentemente dal fatto che la transazione sia stata impegnata. Se necessario, la query attende che i risultati delle operazioni di scrittura inoltrate vengano replicati nell'area corrente. Non attende i risultati aggiornati delle operazioni di scrittura eseguite in altre regioni o in altre sessioni all'interno dell'area corrente.

Con la coerenza di lettura impostata su `GLOBAL`, una sessione in una regione AWS secondaria vede le modifiche apportate da quella sessione. Vede inoltre tutte le modifiche impegnate sia dalla regione AWS primaria che da altre regioni AWS secondarie. Ogni query potrebbe attendere un periodo che varia a seconda della quantità di ritardo della sessione. La query procede quando il cluster secondario è aggiornato con tutti i dati di commit dal cluster primario, a partire dal momento in cui la query è iniziata.

Per ulteriori informazioni su tutti i parametri coinvolti nell'inoltro di scrittura, consulta Parametri di configurazione per l'inoltro di scrittura in Aurora MySQL.

Esempi di utilizzo dell'inoltro di scrittura

Questi esempi utilizzano `aurora_replica_read_consistency` come una variabile di sessione. Per Aurora MySQL versione 3.04 e successive, è possibile usare `aurora_replica_read_consistency` come variabile di sessione o come un parametro del cluster database.

Nell'esempio seguente, il cluster primario si trova nella regione US East (N. Virginia). Il cluster secondario si trova nella regione Stati Uniti orientali (Ohio). L'esempio mostra gli effetti delle istruzioni `INSERT` in esecuzione seguite da istruzioni `SELECT`. A seconda del valore dell'impostazione `aurora_replica_read_consistency`, i risultati potrebbero differire a seconda della tempistica delle istruzioni. Per ottenere una maggiore coerenza, è possibile attendere brevemente prima di rilasciare l'istruzione `SELECT`. Oppure Aurora può attendere automaticamente il completamento della replica dei risultati prima di procedere con `SELECT`.

In questo esempio, è disponibile un'impostazione di coerenza di lettura di `eventual`. L'esecuzione di una istruzione `INSERT` immediatamente seguita da una istruzione `SELECT` restituisce ancora il valore di `COUNT(*)`. Questo valore riflette il numero di righe prima dell'inserimento della nuova riga. Se poco dopo si esegue nuovamente `SELECT`, viene restituito il conteggio delle righe aggiornato. Le istruzioni `SELECT` non attendono.

```
mysql> set aurora_replica_read_consistency = 'eventual';
mysql> select count(*) from t1;
```

```
+----------+
| count(*) |
+----------+
|        5 |
+----------+
1 row in set (0.00 sec)
mysql> insert into t1 values (6); select count(*) from t1;
+----------+
| count(*) |
+----------+
|        5 |
+----------+
1 row in set (0.00 sec)
mysql> select count(*) from t1;
+----------+
| count(*) |
+----------+
|        6 |
+----------+
1 row in set (0.00 sec)
```

Con un'impostazione di coerenza di lettura di `session`, un'istruzione `SELECT` immediatamente dopo un `INSERT` attende fino a quando le modifiche dall'istruzione `INSERT` diventano visibili. Le istruzioni `SELECT` successive non attendono.

```
mysql> set aurora_replica_read_consistency = 'session';
mysql> select count(*) from t1;
+----------+
| count(*) |
+----------+
|        6 |
+----------+
1 row in set (0.01 sec)
mysql> insert into t1 values (6); select count(*) from t1; select count(*) from t1;
Query OK, 1 row affected (0.08 sec)
+----------+
| count(*) |
+----------+
|        7 |
+----------+
1 row in set (0.37 sec)
+----------+
| count(*) |
```

```
+----------+
|        7 |
+----------+
1 row in set (0.00 sec)
```

Con l'impostazione di coerenza di lettura ancora impostata su `session`, l'introduzione di una breve attesa dopo l'esecuzione di un'istruzione `INSERT` rende disponibile il conteggio delle righe aggiornate dal momento dell'esecuzione dell'istruzione `SELECT` successiva.

```
mysql> insert into t1 values (6); select sleep(2); select count(*) from t1;
Query OK, 1 row affected (0.07 sec)
+----------+
| sleep(2) |
+----------+
|        0 |
+----------+
1 row in set (2.01 sec)
+----------+
| count(*) |
+----------+
|        8 |
+----------+
1 row in set (0.00 sec)
```

Con un'impostazione di coerenza di lettura di `global`, ogni istruzione `SELECT` attende di assicurarsi che tutte le modifiche ai dati a partire dall'ora di inizio dell'istruzione siano visibili prima di eseguire la query. La quantità di attesa per ogni istruzione `SELECT` varia a seconda della quantità di ritardo di replica tra i cluster primario e secondario.

```
mysql> set aurora_replica_read_consistency = 'global';
mysql> select count(*) from t1;
+----------+
| count(*) |
+----------+
|        8 |
+----------+
1 row in set (0.75 sec)
mysql> select count(*) from t1;
+----------+
| count(*) |
+----------+
|        8 |
```

```
+----------+
1 row in set (0.37 sec)
mysql> select count(*) from t1;
+----------+
| count(*) |
+----------+
|        8 |
+----------+
1 row in set (0.66 sec)
```

Esecuzione di istruzioni a più parti con inoltro scrittura in Aurora MySQL

Un'istruzione DML può essere costituita da più parti, ad esempio un'istruzione `INSERT ... SELECT` o `DELETE ... WHERE`. In questo caso, l'intera istruzione viene inoltrata al cluster primario ed eseguita lì.

Transazioni con inoltro di scrittura in Aurora MySQL

Se la transazione viene inoltrata al cluster primario dipende dalla modalità di accesso della transazione. È possibile specificare la modalità di accesso per la transazione utilizzando l'istruzione `SET TRANSACTION` o `START TRANSACTION`. È anche possibile specificare la modalità di accesso alle transazioni modificando il valore della variabile di sessione transaction_read_only. È possibile modificare questo valore di sessione solo quando si è connessi a un cluster database in cui l'inoltro di scrittura è abilitato.

Se una transazione di lunga durata non emette alcuna dichiarazione per un periodo di tempo considerevole, potrebbe superare il periodo di timeout inattivo. Questo periodo ha un valore predefinito di un minuto. Puoi aumentarlo fino a un giorno. Una transazione che supera il timeout di inattività viene annullata dal cluster primario. L'istruzione successiva inviata riceve un errore di timeout. Quindi Aurora esegue il rollback della transazione.

Questo tipo di errore può verificarsi in altri casi in cui l'inoltro di scrittura non diventa disponibile. Ad esempio, Aurora annulla tutte le transazioni che utilizzano l'inoltro di scrittura se si riavvia il cluster primario o se si disattiva l'impostazione di configurazione dell'inoltro di scrittura.

Parametri di configurazione per l'inoltro di scrittura in Aurora MySQL

I gruppi di parametri del cluster Aurora contengono delle impostazioni per la funzionalità di inoltro di scrittura. Poiché si tratta di parametri cluster, tutte le istanze database in ogni cluster hanno gli stessi valori per queste variabili. I dettagli su questi parametri sono riepilogati nella tabella seguente, con note di utilizzo dopo la tabella.

Nome	Ambito	Tipo	Valore predefinito	Valori validi
`aurora_fwd_master_idle_timeout` (Aurora MySQL versione 2)	Globale	unsigned integer	60	1–86.400
`aurora_fwd_master_max_connections_pct` (Aurora MySQL versione 2)	Globale	intero lungo senza segno	10	0–90
`aurora_fwd_writer_idle_timeout` (Aurora MySQL version 3)	Globale	unsigned integer	60	1–86.400
`aurora_fwd_writer_max_connections_pct` (Aurora MySQL version 3)	Globale	intero lungo senza segno	10	0–90
`aurora_replica_read_consistency`	Sessione per versione 2 e versione 3 inferiori alla 3.04, Global per versione 3.04 e successive	Enum	" (null)	`EVENTUAL`, `SESSION`, `GLOBAL`

Per controllare le richieste di scrittura in ingresso da cluster secondari, utilizza le seguenti impostazioni nel cluster primario:

- `aurora_fwd_master_idle_timeout`, `aurora_fwd_writer_idle_timeout`: il numero di secondi in cui il cluster primario attende l'attività su una connessione inoltrata da un cluster secondario prima di chiuderla. Se la sessione rimane inattiva oltre questo periodo, la sessione viene annullata da Aurora.

- `aurora_fwd_master_max_connections_pct`, `aurora_fwd_writer_max_connections_pct`: il limite superiore delle connessioni al database che può essere utilizzato su un'istanza database writer per gestire le query inoltrate dai lettori. Viene espresso come percentuale dell'impostazione `max_connections` per l'istanza database writer nel cluster primario. Ad esempio, se `max_connections` è 800 e `aurora_fwd_master_max_connections_pct` o `aurora_fwd_writer_max_connections_pct` è 10, allora lo scrittore consente un massimo di 80 sessioni inoltrate simultanee. Queste connessioni provengono dallo stesso pool di connessioni gestito dall'impostazione `max_connections`.

  Questa impostazione si applica solo al cluster primario, quando uno o più cluster secondari hanno abilitato l'inoltro di scrittura. Se si riduce il valore, le connessioni esistenti non vengono influenzate. Aurora tiene in considerazione il nuovo valore dell'impostazione quando si prova a creare una nuova connessione da un cluster secondario. Il valore predefinito è 10, che rappresenta il 10% del valore `max_connections`. Se si abilita l'inoltro di query su uno qualsiasi dei cluster secondari, questa impostazione deve avere un valore diverso da zero affinché le operazioni di scrittura dai cluster secondari abbiano esito positivo. Se il valore è zero, le operazioni di scrittura ricevono il codice di errore `ER_CON_COUNT_ERROR` con il messaggio `Not enough connections on writer to handle your request`.

Il `aurora_replica_read_consistency` parametro consente l'inoltro di scrittura. Lo si utilizza in ogni sessione. È possibile specificare `EVENTUAL`, `SESSION` o `GLOBAL` per il livello di coerenza di lettura. Per ulteriori informazioni sui livelli di coerenza, consulta Isolamento e coerenza per l'inoltro di scrittura in Aurora MySQL. A questo parametro si applicano le seguenti regole:

- Il valore predefinito è " (empty).
- L'inoltro di scrittura è disponibile in una sessione solo se `aurora_replica_read_consistency` è impostato su `EVENTUAL` o `SESSION` o `GLOBAL`. Questo parametro è rilevante solo nelle istanze di lettura di cluster secondari che hanno l'inoltro di scrittura abilitato e che si trovano in un Aurora Global Database.
- Non è possibile impostarlo come variabile (quando vuoto) o unset (se già impostato) all'interno di una transazione multiistruzione. Tuttavia, è possibile modificarlo da un valore valido (`EVENTUALSESSION`, o `GLOBAL`) a un altro valore valido (`EVENTUALSESSION`, o `GLOBAL`) durante tale transazione.
- La variabile non può essere `SET` quando l'inoltro di scrittura non è abilitato nel cluster secondario.

Parametri Amazon CloudWatch per l'inoltro di scrittura in Aurora MySQL

I parametri Amazon CloudWatch seguenti si applicano al cluster primario quando si utilizza l'inoltro di scrittura su uno o più cluster secondari. Questi parametri sono tutti misurati sull'istanza database writer nel cluster primario.

Parametro CloudWatch	Unità	Descrizione
`AuroraDMLRejectedMasterFull`	Conteggio	Il numero di query inoltrate che vengono rifiutate perché la sessione è piena sull'istanza database di scrittura. Aurora MySQL versione 2
`AuroraDMLRejectedWriterFull`	Conteggio	Il numero di query inoltrate che vengono rifiutate perché la sessione è piena sull'istanza database di scrittura. Per Aurora MySQL versione 3.
`ForwardingMasterDMLLatency`	Millisecondi	Tempo medio per elaborare ogni istruzione DML inoltrata sull'istanza database writer. Non include il tempo impiegato dal cluster secondario per inoltrare la richiesta di scrittura o il tempo necessario per replicare le modifiche al cluster secondario. Aurora MySQL versione 2
`ForwardingMasterDMLThroughput`	Conteggio al secondo	Numero di istruzioni DML inoltrate elaborate ogni secondo da questa istanza database writer.

Parametro CloudWatch	Unità	Descrizione
		Aurora MySQL versione 2
`ForwardingMasterOpenSessions`	Conteggio	Numero di sessioni inoltrate sull'istanza database writer. Aurora MySQL versione 2
`ForwardingWriterDMLLatency`	Millisecondi	Tempo medio per elaborare ogni istruzione DML inoltrata sull'istanza database writer. Non include il tempo impiegato dal cluster secondario per inoltrare la richiesta di scrittura o il tempo necessario per replicare le modifiche al cluster secondario. Per Aurora MySQL versione 3.
`ForwardingWriterDMLThroughput`	Conteggio al secondo	Numero di istruzioni DML inoltrate elaborate ogni secondo da questa istanza database writer. Per Aurora MySQL versione 3.
`ForwardingWriterOpenSessions`	Conteggio	Numero di sessioni inoltrate sull'istanza database writer. Per Aurora MySQL versione 3.

I parametri CloudWatch seguenti si applicano a ciascun cluster secondario. Questi parametri vengono misurati su ogni istanza database del lettore in un cluster secondario con l'inoltro di scrittura abilitato.

Parametro CloudWatch	Unità	Descrizione
`ForwardingReplicaDMLLatency`	Millisecondi	Tempo medio di risposta di DML inoltrati sulla replica.
`ForwardingReplicaDMLThroughput`	Conteggio al secondo	Numero di istruzioni DML inoltrate elaborate al secondo.
`ForwardingReplicaOpenSessions`	Conteggio	Numero di sessioni che utilizzano l'inoltro di scrittura su un'istanza database di lettura.
`ForwardingReplicaReadWaitLatency`	Millisecondi	Tempo medio di attesa di un'istruzione `SELECT` su un'istanza del database di scrittura prima di raggiungere il cluster primario. Il grado di attesa dell'istanza database del lettore prima di elaborare una query dipende dall'impostazione `aurora_replica_read_consistency` .
`ForwardingReplicaReadWaitThroughput`	Conteggio al secondo	Numero totale di istruzioni `SELECT` elaborate ogni secondo in tutte le sessioni che inoltrano scritture.
`ForwardingReplicaSelectLatency`	Millisecondi	Latenza inoltrata `SELECT`, media su tutte le istruzioni `SELECT` inoltrate entro il periodo di monitoraggio.

Parametro CloudWatch	Unità	Descrizione
`ForwardingReplicaSelectThroughput`	Conteggio al secondo	Throughput inoltrato medio per secondo `SELECT` all'interno del periodo di monitoraggio.

Variabili di stato di Aurora MySQL per l’inoltro di scrittura

Le variabili di stato Aurora MySQL seguenti si applicano al cluster primario quando si utilizza l’inoltro di scrittura su uno o più cluster secondari. Questi parametri sono tutti misurati sull'istanza database writer nel cluster primario.

Variabile di stato Aurora MySQL	Unità	Descrizione
`Aurora_fwd_master_dml_stmt_count`	Conteggio	Numero totale di istruzioni DML inoltrate a questa istanza database writer. Aurora MySQL versione 2
`Aurora_fwd_master_dml_stmt_duration`	Microsecondi	Durata totale delle istruzioni DML inoltrate a questa istanza database writer. Aurora MySQL versione 2
`Aurora_fwd_master_open_sessions`	Conteggio	Numero di sessioni inoltrate sull'istanza database writer. Aurora MySQL versione 2
`Aurora_fwd_master_select_stmt_count`	Conteggio	Numero totale di istruzioni `SELECT` inoltrate a questa istanza database writer. Aurora MySQL versione 2

Variabile di stato Aurora MySQL	Unità	Descrizione
`Aurora_fwd_master_select_stmt_duration`	Microsecondi	Durata totale delle istruzioni `SELECT` inoltrate a questa istanza database writer. Aurora MySQL versione 2
`Aurora_fwd_writer_dml_stmt_count`	Conteggio	Numero totale di istruzioni DML inoltrate a questa istanza database writer. Per Aurora MySQL versione 3.
`Aurora_fwd_writer_dml_stmt_duration`	Microsecondi	Durata totale delle istruzioni DML inoltrate a questa istanza database writer.
`Aurora_fwd_writer_open_sessions`	Conteggio	Numero di sessioni inoltrate sull'istanza database writer. Per Aurora MySQL versione 3.
`Aurora_fwd_writer_select_stmt_count`	Conteggio	Numero totale di istruzioni `SELECT` inoltrate a questa istanza database writer. Per Aurora MySQL versione 3.
`Aurora_fwd_writer_select_stmt_duration`	Microsecondi	Durata totale delle istruzioni `SELECT` inoltrate a questa istanza database writer. Per Aurora MySQL versione 3.

Le variabili di stato Aurora MySQL seguenti si applicano a ciascun cluster secondario. Questi parametri vengono misurati su ogni istanza database del lettore in un cluster secondario con l'inoltro di scrittura abilitato.

Variabile di stato Aurora MySQL	Unità	Descrizione
`Aurora_fwd_replica_dml_stmt_count`	Conteggio	Numero totale di istruzioni DML inoltrate da questa istanza database del lettore.
`Aurora_fwd_replica_dml_stmt_duration`	Microsecondi	Durata totale delle istruzioni DML inoltrate da questa istanza database del lettore.
`Aurora_fwd_replica_errors_session_limit`	Conteggio	Numero di sessioni rifiutate dal cluster primario a causa di una delle seguenti condizioni di errore: • istanza di scrittura piena • Troppe istruzioni inoltrate in corso.
`Aurora_fwd_replica_open_sessions`	Conteggio	Numero di sessioni che utilizzano l'inoltro di scrittura su un'istanza database di lettura.
`Aurora_fwd_replica_read_wait_count`	Conteggio	Numero totale di attese di lettura-post-scrittura su questa istanza database del lettore.
`Aurora_fwd_replica_read_wait_duration`	Microsecondi	Durata totale delle attese a causa dell'impostazione di coerenza di lettura su questa istanza database del lettore.
`Aurora_fwd_replica_select_stmt_count`	Conteggio	Numero totale di istruzioni `SELECT` inoltrate dall'istanza database del lettore.

Variabile di stato Aurora MySQL	Unità	Descrizione
`Aurora_fwd_replica_select_stmt_duration`	Microsecondi	Durata totale delle istruzioni `SELECT` inoltrate da questa istanza database del lettore.

Utilizzo dell'inoltro di scrittura in un database globale Aurora PostgreSQL

Argomenti

- Disponibilità di regioni e versioni dell'inoltro di scrittura in Aurora PostgreSQL
- Abilitazione dell'inoltro di scrittura in Aurora PostgreSQL
- Verifica se un cluster secondario ha abilitato l'inoltro di scrittura in Aurora PostgreSQL
- Compatibilità delle applicazioni e di SQL con l'inoltro di scrittura in Aurora PostgreSQL
- Isolamento e coerenza per l'inoltro di scrittura in Aurora PostgreSQL
- Modalità di accesso alle transazioni con inoltro scrittura
- Esecuzione di istruzioni a più parti con inoltro scrittura in Aurora PostgreSQL
- Parametri di configurazione per l'inoltro di scrittura in Aurora PostgreSQL
- Parametri Amazon CloudWatch per l'inoltro di scrittura in Aurora PostgreSQL
- Eventi di attesa per l'inoltro di scrittura in Aurora PostgreSQL

Disponibilità di regioni e versioni dell'inoltro di scrittura in Aurora PostgreSQL

In Aurora PostgreSQL versione 16 e versioni principali successive, l'inoltro globale di scrittura è supportato in tutte le versioni secondarie. Per versioni precedenti di Aurora PostgreSQL, l'inoltro di scrittura è supportato con la versione 15.4 e versioni secondarie successive e versione 14.9 e versioni secondarie successive. L'inoltro di scrittura è disponibile in tutte le Regioni AWS in cui sono disponibili database globali Aurora PostgreSQL.

Per ulteriori informazioni sulla disponibilità di versioni e regioni dei database globali Aurora PostgreSQL consulta Database globali Aurora con Aurora PostgreSQL.

Abilitazione dell'inoltro di scrittura in Aurora PostgreSQL

Per impostazione predefinita, l'inoltro di scrittura non è abilitato quando si aggiunge un cluster secondario a un database globale Aurora. È possibile abilitare l'inoltro di scrittura per il cluster di database secondario durante la creazione o in qualsiasi momento dopo averlo creato. Se necessario, puoi disabilitarlo in un secondo momento. L'attivazione o la disabilitazione dell'inoltro di scrittura non causa tempi di inattività o il riavvio.

> **Note**
>
> Puoi utilizzare l'inoltro di scrittura locale per le applicazioni che hanno occasionalmente scrittura e richiedono la coerenza lettura dopo scrittura, che è la capacità di leggere l'ultima scrittura in una transazione.

Console

Nella console è possibile attivare o disattivare l'inoltro di scrittura quando si crea o si modifica un cluster di database secondario.

Abilitazione o disabilitazione dell'inoltro di scrittura durante la creazione di un cluster di database secondario

Quando crei un nuovo cluster di database secondario, abiliti l'inoltro di scrittura selezionando la casella di controllo Attiva l'inoltro di scrittura globale in Abilita inoltro in scrittura della replica in lettura. Oppure deseleziona la casella di controllo per disabilitarlo. Per creare un cluster di database secondario, segui le istruzioni in Creazione di un cluster database Amazon Aurora.

Lo screenshot seguente mostra la sezione Abilita inoltro in scrittura della replica in lettura con la casella di controllo Attiva l'inoltro di scrittura globale selezionata.



Abilitazione o disabilitazione dell'inoltro di scrittura durante la modifica di un cluster di database secondario

È possibile modificare un cluster di database secondario nella console per abilitare o disabilitare l'inoltro di scrittura.

Per abilitare o disabilitare l'inoltro di scrittura per un cluster di database secondario utilizzando la console

1. Accedi alla Console di gestione AWS e apri la console Amazon RDS all'indirizzo https://console.aws.amazon.com/rds/.
2. Scegli Database.
3. Scegli il cluster di database secondario e scegli Modifica.
4. Nella sezione Abilita inoltro in scrittura della replica in lettura, seleziona o deseleziona la casella di controllo Attiva l'inoltro di scrittura globale.
5. Scegli Continua.
6. In Pianificazione delle modifiche, scegli Applica immediatamente. Se scegli Applica durante la prossima finestra di manutenzione pianificata, Aurora ignora questa impostazione e attiva immediatamente l'inoltro di scrittura.
7. Scegliere Modify cluster (Modifica cluster).

AWS CLI

Per abilitare l'inoltro di scrittura utilizzando la AWS CLI, utilizza l'opzione `--enable-global-write-forwarding`. Questa opzione funziona quando crei un nuovo cluster secondario utilizzando il comando create-db-cluster. Funziona anche quando modifichi un cluster secondario esistente utilizzando il comando modify-db-cluster. Richiede che il database globale utilizzi una versione Aurora che supporti l'inoltro di scrittura. Puoi disattivare l'inoltro di scrittura mediante l'opzione `--no-enable-global-write-forwarding` con questi stessi comandi CLI.

Le seguenti procedure descrivono come abilitare o disabilitare l'inoltro di scrittura per un cluster di database secondario nel cluster globale utilizzando la AWS CLI.

Per abilitare o disabilitare l'inoltro di scrittura per un cluster di database secondario esistente

- Chiama il comando della AWS CLI modify-db-cluster e specifica i seguenti valori:
  - `--db-cluster-identifier`: il nome del cluster di database.
  - `--enable-global-write-forwarding` per attivare o `--no-enable-global-write-forwarding` per disattivare.

  L'esempio seguente mostra come abilitare l'inoltro di scrittura per un cluster di database `sample-secondary-db-cluster`.

Per Linux, macOS o Unix:

```
aws rds modify-db-cluster \
    --db-cluster-identifier sample-secondary-db-cluster \
    --enable-global-write-forwarding
```

Per Windows:

```
aws rds modify-db-cluster ^
    --db-cluster-identifier sample-secondary-db-cluster ^
    --enable-global-write-forwarding
```

API RDS

Per abilitare l'inoltro di scrittura utilizzando l'API Amazon RDS, impostare il parametro `EnableGlobalWriteForwarding` su `true`. Questo parametro funziona quando crei un nuovo cluster secondario utilizzando l'operazione CreateDBCluster. Funziona anche quando modifichi un cluster secondario esistente utilizzando l'operazione ModifyDBCluster. Richiede che il database globale utilizzi una versione Aurora che supporti l'inoltro di scrittura. È possibile disattivare l'inoltro di scrittura impostando il parametro `EnableGlobalWriteForwarding` su `false`.

Verifica se un cluster secondario ha abilitato l'inoltro di scrittura in Aurora PostgreSQL

Per determinare se è possibile utilizzare l'inoltro di scrittura da un cluster secondario, è possibile verificare se il cluster dispone dell'attributo `"GlobalWriteForwardingStatus": "enabled"`.

In Console di gestione AWS, viene visualizzato `Read replica write forwarding` nella scheda Configurazione della pagina dei dettagli per il cluster. Per visualizzare lo stato dell'impostazione di inoltro di scrittura globale per tutti i cluster, eseguire il comando AWS CLI seguente.

Un cluster secondario mostra il valore `"enabled"` o `"disabled"` per indicare se l'inoltro di scrittura è attivato o disattivato. Il valore `null` indica che l'inoltro di scrittura non è disponibile per il cluster. Il cluster non fa parte di un database globale oppure è il cluster primario anziché un cluster secondario. Il valore può anche essere `"enabling"` o `"disabling"` se l'inoltro di scrittura è in procinto di essere attivato o disattivato.

Example

```
aws rds describe-db-clusters --query '*[].
{DBClusterIdentifier:DBClusterIdentifier,GlobalWriteForwardingStatus:GlobalWriteForwardingStatu
[
    {
        "GlobalWriteForwardingStatus": "enabled",
        "DBClusterIdentifier": "aurora-write-forwarding-test-replica-1"
    },
    {
        "GlobalWriteForwardingStatus": "disabled",
        "DBClusterIdentifier": "aurora-write-forwarding-test-replica-2"
    },
    {
        "GlobalWriteForwardingStatus": null,
        "DBClusterIdentifier": "non-global-cluster"
    }
]
```

Per trovare tutti i cluster secondari per i quali è abilitato l'inoltro di scrittura globale, eseguire il comando seguente. Questo comando restituisce anche l'endpoint di lettura del cluster. È possibile utilizzare l'endpoint di lettura del cluster secondario quando si utilizza l'inoltro di scrittura dal cluster secondario al primario nel database globale Aurora.

Example

```
aws rds describe-db-clusters --query 'DBClusters[].
{DBClusterIdentifier:DBClusterIdentifier,GlobalWriteForwardingStatus:GlobalWriteForwardingStatu
 | [?GlobalWriteForwardingStatus == `enabled`]'
[
    {
        "GlobalWriteForwardingStatus": "enabled",
        "ReaderEndpoint": "aurora-write-forwarding-test-replica-1.cluster-ro-
cnpexample.us-west-2.rds.amazonaws.com",
        "DBClusterIdentifier": "aurora-write-forwarding-test-replica-1"
    }
]
```

Compatibilità delle applicazioni e di SQL con l'inoltro di scrittura in Aurora PostgreSQL

Alcune istruzioni non sono consentite o possono produrre risultati non aggiornati quando vengono utilizzate in un database globale con inoltro di scrittura. Inoltre, le funzioni e le procedure definite

dall'utente non sono supportate. Pertanto, l'impostazione `EnableGlobalWriteForwarding` è disattivata in modo predefinito per i cluster secondari. Prima di attivarlo, verificare che il codice dell'applicazione non sia interessato da nessuna di queste restrizioni.

È possibile utilizzare i seguenti tipi di istruzioni SQL con l'inoltro di scrittura:

- Istruzioni DML (Data Manipulation Language), ad esempio `INSERT`, `DELETE` e `UPDATE`
- Istruzioni `SELECT FOR { UPDATE | NO KEY UPDATE | SHARE | KEY SHARE }`
- Istruzioni `PREPARE` e `EXECUTE`.
- Istruzioni `EXPLAIN` con le istruzioni in questo elenco

I seguenti tipi di istruzioni SQL non sono supportati con l'inoltro di scrittura:

- Istruzioni DDL (Data Definition Language)
- `ANALYZE`
- `CLUSTER`
- `COPY`
- Cursori: i cursori non sono supportati, quindi assicurati di chiuderli prima di utilizzare l'inoltro di scrittura.
- `GRANT|REVOKE|REASSIGN OWNED|SECURITY LABEL`
- `LOCK`
- Istruzioni `SAVEPOINT`
- `SELECT INTO`
- `SET CONSTRAINTS`
- `TRUNCATE`
- `VACUUM`

Isolamento e coerenza per l'inoltro di scrittura in Aurora PostgreSQL

Nelle sessioni che utilizzano l'inoltro di scrittura, è possibile utilizzare solo i livelli di isolamento `REPEATABLE READ` e `READ COMMITTED`. Tuttavia, il livello di isolamento `SERIALIZABLE` non è supportato.

È possibile controllare il grado di coerenza di lettura in un cluster secondario. Il livello di coerenza di lettura determina la quantità di attesa di un cluster secondario prima di ogni operazione di lettura

per garantire che alcune o tutte le modifiche vengano replicate dal cluster primario. È possibile regolare il livello di coerenza di lettura per garantire che tutte le operazioni di scrittura inoltrate dalla sessione siano visibili nel cluster secondario prima di qualsiasi query successiva. È inoltre possibile utilizzare questa impostazione per garantire che le query sul cluster secondario visualizzino sempre gli aggiornamenti più recenti dal cluster primario. Ciò si verifica anche per quelli inviati da altre sessioni o altri cluster. Per specificare questo tipo di comportamento per l'applicazione, scegli il valore appropriato per il parametro `apg_write_forward.consistency_mode`. Il parametro `apg_write_forward.consistency_mode` ha effetto solo sui cluster secondari in cui è abilitato l'inoltro di scrittura.

> **Note**
>
> Per il parametro `apg_write_forward.consistency_mode`, è possibile specificare i valori `SESSION`, `EVENTUAL`, `GLOBAL` o `OFF`. Per impostazione predefinita, il valore è impostato su `SESSION`. L'impostazione del valore su `OFF` disabilita l'inoltro di scrittura nella sessione.

Aumentando il livello di coerenza, l'applicazione impiega più tempo in attesa della propagazione delle modifiche tra regioni AWS. È possibile scegliere il bilanciamento tra tempi di risposta rapidi e garantire che le modifiche apportate in altre posizioni siano completamente disponibili prima dell'esecuzione delle query.

Ogni impostazione della modalità di coerenza disponibile, produce un effetto come descritto di seguito:

- `SESSION`: tutte le query in una regione AWS secondaria che utilizza l'inoltro di scrittura vedono i risultati di tutte le modifiche apportate durante tale sessione. Le modifiche sono visibili indipendentemente dal fatto che la transazione sia stata impegnata. Se necessario, la query attende che i risultati delle operazioni di scrittura inoltrate vengano replicati nell'area corrente. Non attende i risultati aggiornati delle operazioni di scrittura eseguite in altre regioni o in altre sessioni all'interno dell'area corrente.
- `EVENTUAL`: le query in una regione AWS secondaria che utilizza l'inoltro di scrittura potrebbero vedere dati leggermente obsoleti a causa del ritardo di replica. I risultati delle operazioni di scrittura nella stessa sessione non sono visibili fino a quando l'operazione di scrittura non viene eseguita nella regione primaria e replicata nella regione corrente. La query non attende la disponibilità dei risultati aggiornati. Pertanto, potrebbe recuperare i dati meno recenti o i dati aggiornati, a seconda della tempistica delle istruzioni e della quantità di ritardo di replica.

- `GLOBAL`: una sessione in una regione AWS secondaria vede le modifiche apportate da quella sessione. Vede inoltre tutte le modifiche impegnate sia dalla regione AWS primaria che da altre regioni AWS secondarie. Ogni query potrebbe attendere un periodo che varia a seconda della quantità di ritardo della sessione. La query procede quando il cluster secondario è aggiornato con tutti i dati di commit dal cluster primario, a partire dal momento in cui la query è iniziata.
- `OFF`: l'inoltro di scrittura durante la sessione è disabilitato.

Per ulteriori informazioni su tutti i parametri coinvolti nell'inoltro di scrittura, consulta Parametri di configurazione per l'inoltro di scrittura in Aurora PostgreSQL.

Modalità di accesso alle transazioni con inoltro scrittura

Se la modalità di accesso alle transazioni è impostata su sola lettura, l'inoltro di scrittura non viene utilizzato. È possibile impostare la modalità di accesso in lettura e scrittura quando si è connessi a un cluster di database e a una sessione in cui l'inoltro di scrittura è abilitato.

Per ulteriori informazioni sulle modalità di accesso alle transazioni, consultare SET TRANSACTION.

Esecuzione di istruzioni a più parti con inoltro scrittura in Aurora PostgreSQL

Un'istruzione DML può essere costituita da più parti, ad esempio un'istruzione `INSERT ... SELECT` o `DELETE ... WHERE`. In questo caso, l'intera istruzione viene inoltrata al cluster primario ed eseguita lì.

Parametri di configurazione per l'inoltro di scrittura in Aurora PostgreSQL

I gruppi di parametri del cluster Aurora contengono delle impostazioni per la funzionalità di inoltro di scrittura. Poiché si tratta di parametri cluster, tutte le istanze database in ogni cluster hanno gli stessi valori per queste variabili. I dettagli su questi parametri sono riepilogati nella tabella seguente, con note di utilizzo dopo la tabella.

Nome	Ambito	Tipo	Valore predefinito	Valori validi
`apg_write_forward.connect_timeout`	Sessione	secondi	30	0–2147483647
`apg_write_forward.consistency_mode`	Sessione	enum	Sessione	`SESSION`, `EVENTUAL`,

Nome	Ambito	Tipo	Valore predefinito	Valori validi
				GLOBAL, OFF
apg_write_forward.idle_in_transaction_session_timeout	Sessione	millisecondi	86.400.000	0–2147483647
apg_write_forward.idle_session_timeout	Sessione	millisecondi	300.000	0–2147483647
apg_write_forward.max_forwarding_connections_percent	Globale	int	25	1-100

Il parametro apg_write_forward.max_forwarding_connections_percent rappresenta il limite superiore degli slot di connessione al database che possono essere utilizzati per gestire le query inoltrate dalle istanze di lettura. Viene espresso come percentuale dell'impostazione max_connections per l'istanza database di scrittura nel cluster primario. Ad esempio, se max_connections è 800 e apg_write_forward.max_forwarding_connections_percent è 10, allora l'istanza di scrittura consente un massimo di 80 sessioni inoltrate simultanee. Queste connessioni provengono dallo stesso pool di connessioni gestito dall'impostazione max_connections. Questa impostazione si applica solo al cluster primario, quando almeno un cluster ha abilitato l'inoltro di scrittura.

Utilizza le seguenti impostazioni sul cluster secondario:

- apg_write_forward.consistency_mode: un parametro a livello di sessione che controlla il grado di coerenza di lettura sul cluster secondario. I valori validi sono SESSION, EVENTUAL, GLOBAL o OFF. Per impostazione predefinita, il valore è impostato su SESSION. L'impostazione del valore su OFF disabilita l'inoltro di scrittura nella sessione. Per ulteriori informazioni sui livelli di coerenza, consulta Isolamento e coerenza per l'inoltro di scrittura in Aurora PostgreSQL. Questo parametro è rilevante solo nelle istanze di lettura di cluster secondari che hanno l'inoltro di scrittura abilitato e che si trovano in un Aurora Global Database.
- apg_write_forward.connect_timeout: il numero massimo di secondi che il cluster secondario attende per stabilire una connessione al cluster primario prima di rinunciare. Il valore 0 indica un tempo di attesa infinito.

- `apg_write_forward.idle_in_transaction_session_timeout`: il numero di millisecondi che il cluster primario attende l'attività su una connessione inoltrata da un cluster secondario con una transazione aperta prima di chiuderla. Se la sessione continua ad avere una transazione inattiva oltre questo periodo, Aurora la chiude. Il valore `0` disabilita il timeout.
- `apg_write_forward.idle_session_timeout`: il numero di millisecondi che il cluster primario attende l'attività su una connessione inoltrata da un cluster secondario prima di chiuderla. Se la sessione rimane inattiva oltre questo periodo, Aurora la chiude. Il valore `0` disabilita il timeout.

Parametri Amazon CloudWatch per l'inoltro di scrittura in Aurora PostgreSQL

I parametri Amazon CloudWatch seguenti si applicano al cluster primario quando si utilizza l'inoltro di scrittura su uno o più cluster secondari. Questi parametri sono tutti misurati sull'istanza database writer nel cluster primario.

Parametro CloudWatch	Unità e descrizione
`AuroraForwardingWriterDMLThroughput`	Conteggio (al secondo) Numero di istruzioni DML inoltrate elaborate ogni secondo da questa istanza database writer.
`AuroraForwardingWriterOpenSessions`	Conteggio Numero di sessioni aperte su questa istanza database di scrittura che elabora le query inoltrate.
`AuroraForwardingWriterTotalSessions`	Conteggio Numero totale di sessioni inoltrate sull'istanza database di scrittura.

I parametri CloudWatch seguenti si applicano a ciascun cluster secondario. Questi parametri vengono misurati su ogni istanza database del lettore in un cluster secondario con l'inoltro di scrittura abilitato.

Parametro CloudWatch	Unità e descrizione
`AuroraForwardingReplicaCommitThroughput`	Conteggio (al secondo) Numero di commit in sessioni inoltrate da questa replica ogni secondo.

Parametro CloudWatch	Unità e descrizione
`AuroraForwardingReplicaDMLLatency`	Millisecondi Tempo medio di risposta in millisecondi di DML inoltrati sulla replica.
`AuroraForwardingReplicaDMLThroughput`	Conteggio (al secondo) Numero di istruzioni DML inoltrate sulla replica elaborate ogni secondo.
`AuroraForwardingReplicaErrorSessionsLimit`	Conteggio Numero di sessioni rifiutate dal cluster primario quando viene raggiunto il limite massimo di connessioni o di connessioni create per l'inoltro di scrittura.
`AuroraForwardingReplicaOpenSessions`	Conteggio Il numero di sessioni che utilizzano l'inoltro di scrittura su un'istanza di replica.
`AuroraForwardingReplicaReadWaitLatency`	Millisecondi Tempo medio di attesa in millisecondi della replica per garantire la coerenza con l'LSN del cluster primario. Il grado di attesa dell'istanza database in lettura dipende dall'impostazione `apg_write_forward.consistency_mode` . Per ulteriori informazioni su questa impostazione, consulta the section called “Parametri di configurazione per l'inoltro di scrittura in Aurora PostgreSQL”.

Eventi di attesa per l'inoltro di scrittura in Aurora PostgreSQL

Amazon Aurora genera i seguenti eventi di attesa quando si utilizza l'inoltro di scrittura con Aurora PostgreSQL.

Argomenti

- IPC:Aurora WriteForwardConnect
- IPC:AuroraWriteForwardConsistencyPoint
- IPC:AuroraWriteForwardExecute
- IPC:AuroraWriteForwardGetGlobalConsistencyPoint

- IPC:AuroraWriteForwardXactAbort
- IPC:AuroraWriteForwardXactCommit
- IPC:AuroraWriteForwardXactStart

IPC:Aurora WriteForwardConnect

L'evento `IPC:AuroraWriteForwardConnect` si verifica quando un processo di backend sul cluster di database secondario è in attesa dell'apertura della connessione del cluster di database primario al nodo di scrittura.

Probabili cause di aumento delle attese

Questo evento diventa più frequente all'aumentare del numero dei tentativi di connessione dal nodo di lettura di una regione secondaria al nodo di scrittura del cluster di database primario.

Azioni

Riduci il numero di connessioni simultanee da un nodo secondario al nodo di scrittura della regione primaria.

IPC:AuroraWriteForwardConsistencyPoint

L'evento `IPC:AuroraWriteForwardConsistencyPoint` descrive il tempo di attesa di una query generata da un nodo sul cluster di database secondario affinché i risultati delle operazioni di scrittura inoltrate vengano replicati nella regione attuale. Questo evento viene generato solo se il parametro `apg_write_forward.consistency_mode` a livello di sessione è impostato su uno dei seguenti:

- `SESSION`: le query su un nodo secondario attendono i risultati di tutte le modifiche apportate in quella sessione.
- `GLOBAL`: le query su un nodo secondario attendono i risultati delle modifiche apportate da quella sessione, oltre a tutte le modifiche richieste dalla regione primaria e dalle altre regioni secondarie del cluster globale.

Per ulteriori informazioni sull'impostazione del parametro `apg_write_forward.consistency_mode`, consulta the section called “Parametri di configurazione per l'inoltro di scrittura in Aurora PostgreSQL”.

Probabili cause di aumento delle attese

Alcune cause comuni dei tempi di attesa più lunghi sono:

- Aumento del ritardo di replica, misurato dalla metrica `ReplicaLag` di Amazon CloudWatch. Per ulteriori informazioni su questa metrica, consulta Monitoraggio della replica Aurora PostgreSQL..
- Aumento del carico sul nodo di scrittura della regione primaria o sul nodo secondario.

Azioni

Modifica la modalità di coerenza in base ai requisiti dell'applicazione.

IPC:AuroraWriteForwardExecute

L'evento `IPC:AuroraWriteForwardExecute` si verifica quando un processo di backend sul cluster di database secondario è in attesa del completamento di una query inoltrata e di ottenere risultati dal nodo di scrittura del cluster di database primario.

Probabili cause di aumento delle attese

Alcune cause comuni dell'aumento dei tempi di attesa sono:

- Recupero di un numero elevato di righe dal nodo di scrittura primario della regione.
- L'aumento della latenza di rete tra il nodo secondario e il nodo di scrittura della regione primaria ritarda la ricezione dei dati del nodo di scrittura da parte del nodo secondario.
- L'aumento del carico sul nodo secondario può ritardare la trasmissione della richiesta di query dal nodo secondario al nodo di scrittura della regione primaria.
- Un carico maggiore sul nodo di scrittura della regione primaria può ritardare la trasmissione dei dati dal nodo di scrittura al nodo secondario.

Azioni

Consigliamo azioni diverse a seconda delle cause dell'evento di attesa.

- Ottimizza le query per recuperare solo i dati necessari.
- Ottimizza le operazioni DML (Data Manipulation Language) per modificare solo i dati necessari.
- Se il nodo secondario o il nodo di scrittura della regione primaria è limitato dalla CPU o dalla larghezza di banda di rete, valuta la possibilità di modificarlo in un tipo di istanza con maggiore capacità di CPU o maggiore larghezza di banda di rete.

IPC:AuroraWriteForwardGetGlobalConsistencyPoint

L'evento `IPC:AuroraWriteForwardGetGlobalConsistencyPoint` si verifica quando un processo di backend sul cluster di database secondario che utilizza la modalità di coerenza GLOBAL è in attesa di ottenere il punto di coerenza globale dal nodo di scrittura prima di eseguire una query.

Probabili cause di aumento delle attese

Alcune cause comuni dell'aumento dei tempi di attesa sono:

- L'aumento della latenza di rete tra il nodo secondario e il nodo di scrittura della regione primaria ritarda la ricezione dei dati del nodo di scrittura da parte del nodo secondario.
- L'aumento del carico sul nodo secondario può ritardare la trasmissione della richiesta di query dal nodo secondario al nodo di scrittura della regione primaria.
- Un carico maggiore sul nodo di scrittura della regione primaria può ritardare la trasmissione dei dati dal nodo di scrittura al nodo secondario.

Azioni

Consigliamo azioni diverse a seconda delle cause dell'evento di attesa.

- Modifica la modalità di coerenza in base ai requisiti dell'applicazione.
- Se il nodo secondario o il nodo di scrittura della regione primaria è limitato dalla CPU o dalla larghezza di banda di rete, valuta la possibilità di modificarlo in un tipo di istanza con maggiore capacità di CPU o maggiore larghezza di banda di rete.

IPC:AuroraWriteForwardXactAbort

L'evento `IPC:AuroraWriteForwardXactAbort` si verifica quando un processo di backend sul cluster di database secondario è in attesa del risultato di una query di pulizia remota. Le query di pulizia vengono emesse per riportare il processo allo stato ottimale dopo l'interruzione di una transazione di scrittura inoltrata. Amazon Aurora le esegue perché è stato rilevato un errore o perché un utente ha chiamato un comando `ABORT` esplicito o ha annullato una query in esecuzione.

Probabili cause di aumento delle attese

Alcune cause comuni dell'aumento dei tempi di attesa sono:

- L'aumento della latenza di rete tra il nodo secondario e il nodo di scrittura della regione primaria ritarda la ricezione dei dati del nodo di scrittura da parte del nodo secondario.
- L'aumento del carico sul nodo secondario può ritardare la trasmissione della richiesta di query di pulizia dal nodo secondario al nodo di scrittura della regione primaria.
- Un carico maggiore sul nodo di scrittura della regione primaria può ritardare la trasmissione dei dati dal nodo di scrittura al nodo secondario.

Azioni

Consigliamo azioni diverse a seconda delle cause dell'evento di attesa.

- Indaga la causa della transazione interrotta.
- Se il nodo secondario o il nodo di scrittura della regione primaria è limitato dalla CPU o dalla larghezza di banda di rete, valuta la possibilità di modificarlo in un tipo di istanza con maggiore capacità di CPU o maggiore larghezza di banda di rete.

IPC:AuroraWriteForwardXactCommit

L'evento `IPC:AuroraWriteForwardXactCommit` si verifica quando un processo di backend sul cluster di database secondario è in attesa del risultato di un comando di transazione di commit inoltrato.

Probabili cause di aumento delle attese

Alcune cause comuni dell'aumento dei tempi di attesa sono:

- L'aumento della latenza di rete tra il nodo secondario e il nodo di scrittura della regione primaria ritarda la ricezione dei dati del nodo di scrittura da parte del nodo secondario.
- L'aumento del carico sul nodo secondario può ritardare la trasmissione della richiesta di query dal nodo secondario al nodo di scrittura della regione primaria.
- Un carico maggiore sul nodo di scrittura della regione primaria può ritardare la trasmissione dei dati dal nodo di scrittura al nodo secondario.

Azioni

Se il nodo secondario o il nodo di scrittura della regione primaria è limitato dalla CPU o dalla larghezza di banda di rete, valuta la possibilità di modificarlo in un tipo di istanza con maggiore capacità di CPU o maggiore larghezza di banda di rete.

IPC:AuroraWriteForwardXactStart

L'evento `IPC:AuroraWriteForwardXactStart` si verifica quando un processo di backend sul cluster di database secondario è in attesa del risultato di un comando di avvio della transazione inoltrato.

Probabili cause di aumento delle attese

Alcune cause comuni dell'aumento dei tempi di attesa sono:

- L'aumento della latenza di rete tra il nodo secondario e il nodo di scrittura della regione primaria ritarda la ricezione dei dati del nodo di scrittura da parte del nodo secondario.
- L'aumento del carico sul nodo secondario può ritardare la trasmissione della richiesta di query dal nodo secondario al nodo di scrittura della regione primaria.
- Un carico maggiore sul nodo di scrittura della regione primaria può ritardare la trasmissione dei dati dal nodo di scrittura al nodo secondario.

Azioni

Se il nodo secondario o il nodo di scrittura della regione primaria è limitato dalla CPU o dalla larghezza di banda di rete, valuta la possibilità di modificarlo in un tipo di istanza con maggiore capacità di CPU o maggiore larghezza di banda di rete.

Utilizzo dello switchover o failover in un Database globale Amazon Aurora

La funzionalità Database globale Aurora offre una maggiore protezione a livello di continuità aziendale e disaster recovery (BCDR) rispetto alla disponibilità elevata standard fornita da un cluster di database Aurora in una singola Regione AWS. Database globale Aurora consente di pianificare un ripristino più rapido in caso di rari eventi non pianificati o di interruzioni complete dei livelli di servizio a livello regionale.

È possibile consultare le seguenti linee guida e procedure per pianificare, testare e implementare la strategia BCDR utilizzando la funzionalità Database globale Aurora.

Argomenti

- Pianificazione della continuità aziendale e del disaster recovery
- Esecuzione di switchover per database globali Amazon Aurora
- Ripristino di un database globale Amazon Aurora da un'interruzione non pianificata
- Gestione degli RPO per database globali basati su Aurora PostgreSQL–
- Resilienza tra Regioni per i cluster secondari del database globale

Pianificazione della continuità aziendale e del disaster recovery

Per pianificare la strategia di continuità aziendale e disaster recovery, è utile comprendere la seguente terminologia di settore e il modo in cui tali termini si applicano alle funzionalità del Database globale Aurora.

Il ripristino di emergenza è in genere determinato dai due obiettivi aziendali seguenti:

- Obiettivo del tempo di ripristino (RTO): il tempo necessario a un sistema per tornare a uno stato funzionante dopo un guasto o un'interruzione del servizio. In altre parole, l'RTO misura i tempi di inattività. Per Database globale Aurora, l'RTO può essere nell'ordine di minuti.
- Obiettivo del punto di ripristino (RPO): la quantità di dati che può venire persa (misurata nel tempo) dopo un guasto o un'interruzione del servizio. Questa perdita di dati è in genere dovuta al ritardo della replica asincrona. Per un database globale Aurora, l'RPO viene in genere misurato in secondi. Con un database globale basato su Aurora PostgreSQL–, puoi utilizzare il parametro `rds.global_db_rpo` per impostare e tenere traccia del limite superiore su RPO, ma ciò potrebbe influire sull'elaborazione delle transazioni sul nodo writer del cluster primario. Per ulteriori informazioni, consulta Gestione degli RPO per database globali basati su Aurora PostgreSQL–.

L'esecuzione di uno switchover o failover con Database globale Aurora implica la promozione di un cluster di database secondario a cluster di database primario. Il termine "interruzione a livello regionale" viene spesso utilizzato per descrivere una serie di scenari di errore. Lo scenario peggiore potrebbe essere un'interruzione generalizzata causata da un evento catastrofico che interessa un'area geografica particolarmente estesa. Tuttavia, la maggior parte delle interruzioni è molto più localizzata e riguarda solo un piccolo sottoinsieme di servizi cloud o sistemi dei clienti. Valuta l'ambito dell'interruzione nel suo insieme per assicurarti che il failover tra regioni rappresenti la soluzione più appropriata e scegli il metodo di failover più adatto alla situazione. L'utilizzo del failover o dello switchover dipende dallo scenario di interruzione specifico:

- Failover: utilizza questo approccio per il ripristino dopo un'interruzione non pianificata. Con questo approccio, puoi eseguire un failover tra regioni su uno dei cluster database secondari del database globale Aurora. Il valore RPO, espresso in secondi, per questo approccio è in genere diverso da zero. La quantità di dati persi dipende dal ritardo di replica del database globale Aurora tra Regioni AWS al momento del guasto. Per ulteriori informazioni, consulta Ripristino di un database globale Amazon Aurora da un'interruzione non pianificata.
- Switchover: questa operazione era precedentemente denominata “failover pianificato gestito”. Usa questo approccio per scenari controllati, ad esempio la manutenzione operativa e altre procedure operative pianificate, in cui tutti i cluster Aurora e gli altri servizi con cui interagiscono sono in stato integro. Poiché questa funzionalità sincronizza i cluster di database secondari con il primario prima di apportare altre modifiche, l’RPO è 0 (nessuna perdita di dati). Per ulteriori informazioni, consulta Esecuzione di switchover per database globali Amazon Aurora.

> **Note**
>
> Se desideri eseguire lo switchover o il failover su un cluster di database Aurora secondario headless, devi prima aggiungervi un’istanza database. Per ulteriori informazioni sui cluster database headless, consulta Creazione di un cluster database Aurora headless in una regione secondaria.

Esecuzione di switchover per database globali Amazon Aurora

> **Note**
>
> Gli switchover erano precedentemente denominati failover pianificati gestiti.

Utilizzando gli switchover, è possibile modificare regolarmente la Regione del cluster primario. Questo approccio è destinato agli scenari controllati, ad esempio durante la manutenzione operativa e altre procedure operative pianificate.

Esistono tre casi d'uso comuni per l'utilizzo degli switchover.

- Per i requisiti relativi alla "rotazione regionale" imposti a settori specifici. Ad esempio, le normative sui servizi finanziari potrebbero imporre che i sistemi di livello 0 passino a un'altra regione per diversi mesi per garantire l'esecuzione regolare delle procedure di ripristino di emergenza.

- Per applicazioni con approccio "follow-the-sun" in più regioni. Ad esempio, un'azienda potrebbe voler fornire scritture con latenza inferiore in diverse regioni in base all'orario di lavoro nei vari fusi orari.
- Come metodo senza perdita di dati per eseguire il failback alla regione principale originale dopo un failover.

> **Note**
>
> Gli switchover sono progettati per essere utilizzati su un Database globale Aurora in cui tutti i cluster Aurora e gli altri servizi con cui interagiscono sono in stato integro. Per eseguire il ripristino da un'interruzione non pianificata, puoi eseguire la procedura appropriata descritta in Ripristino di un database globale Amazon Aurora da un'interruzione non pianificata.
> È possibile eseguire switchover gestiti tra Regioni con il Database globale Aurora solo se i cluster di database primario e secondario hanno le stesse versioni principale e secondaria del motore. A seconda del motore e delle versioni del motore, può essere necessario che i livelli di patch siano identici oppure possono essere diversi. Per un elenco dei motori e delle versioni dei motori che consentono queste operazioni tra cluster primari e secondari con diversi livelli di patch, consulta Compatibilità del livello di patch per switchover e failover gestiti tra regioni. Prima di iniziare lo switchover, controlla le versioni del motore nel cluster globale per assicurarti che supportino lo switchover gestito tra regioni e, se necessario, aggiornale.

Durante uno switchover, Aurora imposta il cluster nella Regione secondaria scelta come cluster primario. Il meccanismo di switchover mantiene la topologia di replica esistente del database globale: ha ancora lo stesso numero di cluster Aurora nelle stesse Regioni. Prima di avviare il processo di switchover, Aurora attende che tutti i cluster della Regione secondaria di destinazione siano completamente sincronizzati con il cluster della Regione primaria. In seguito, il cluster di database nella Regione primaria diventa di sola lettura. Il cluster secondario scelto promuove uno dei suoi nodi di sola lettura allo stato di scrittura completo, consentendo così al cluster secondario di assumere il ruolo di cluster primario. Poiché tutti i cluster secondari di destinazione sono stati sincronizzati con quello primario all'inizio del processo, il nuovo primario continua le operazioni per il Database globale Aurora senza perdere alcun dato. Il database non è disponibile per un breve periodo, mentre i cluster primario e secondario selezionati assumono i loro nuovi ruoli.

> **Note**
>
> Per gestire gli slot di replica per Aurora PostgreSQL dopo aver eseguito uno switchover, consulta Gestione degli slot logici per Aurora PostgreSQL .

Per ottimizzare la disponibilità delle applicazioni, è consigliabile eseguire le seguenti operazioni prima di utilizzare questa funzionalità:

- Esegui questa operazione durante le ore non di punta o in un altro momento quando le scritture nel cluster DB primario sono minime.
- Controllare i tempi di ritardo per tutti i cluster di database Aurora secondari nel database globale Aurora. Per tutti i database globali basati su Aurora PostgreSQL e per i database globali basati su Aurora MySQL a partire dalle versioni del motore 3.04.0 e successive o 2.12.0 e successive, usa Amazon CloudWatch per visualizzare la metrica `AuroraGlobalDBRPOLag` per tutti i cluster database secondari. Per le versioni minori precedenti dei database globali basati su Aurora MySQL, puoi invece visualizzare la metrica `AuroraGlobalDBReplicationLag`. Questa metrica indica il ritardo (in millisecondi) della replica tra un cluster secondario e il cluster database primario. Il suo valore è direttamente proporzionale al tempo necessario ad Aurora per completare lo switchover. Di conseguenza, maggiore è il valore del ritardo, maggiore sarà il tempo necessario per lo switchover. Quando esamini queste metriche, procedi dal cluster primario corrente.

  Per ulteriori informazioni sulle metriche di CloudWatch per Aurora, consulta Parametri a livello di cluster per Amazon Aurora.
- Il cluster di database secondario promosso durante uno switchover potrebbe avere impostazioni di configurazione diverse rispetto al cluster di database primario precedente. È consigliabile mantenere i seguenti tipi di impostazioni di configurazione coerenti tra tutti i cluster nei cluster di Database globale Aurora. Questo consente di ridurre al minimo i problemi con le prestazioni, le incompatibilità dei carichi di lavoro e gli altri comportamenti anomali che seguono uno switchover.
  - Configura il gruppo di parametri del cluster di database Aurora per il nuovo cluster primario, se necessario: quando si promuove un cluster di database secondario perché assuma il ruolo di primario, il gruppo di parametri del secondario potrebbe essere configurato in modo diverso rispetto al cluster primario. In tal caso, modifica il gruppo di parametri del cluster di database secondario promosso in modo che sia conforme alle impostazioni del cluster primario. Per scoprire come, consulta Modifica dei parametri per un database globale Aurora.

- Configura gli strumenti e le opzioni di monitoraggio, ad esempio Amazon CloudWatch Events e allarmi – Configura il cluster di database promosso con la stessa capacità di registrazione, gli allarmi e così via in base alle esigenze del database globale. Come per i gruppi di parametri, la configurazione di queste funzionalità non viene ereditata dal ruolo primario durante il processo di switchover. Alcune metriche CloudWatch, come il ritardo di replica, sono disponibili solo per le regioni secondarie. Pertanto, uno switchover modifica il modo in cui visualizzare tali metriche e impostare i relativi allarmi e potrebbe richiedere modifiche da apportare a qualsiasi dashboard predefinito. Per ulteriori informazioni sui cluster di database Aurora e sul monitoraggio, consulta Monitoraggio dei parametri di Amazon Aurora con Amazon CloudWatch.
- Configura le integrazioni con altri servizi AWS: se il database globale Aurora si integra con i servizi AWS, ad esempio Gestione dei segreti AWS, AWS Identity and Access Management, Amazon S3 e AWS Lambda, assicurati di configurare le integrazioni con questi servizi in base alle esigenze. Per ulteriori informazioni sull'integrazione dei database globali Aurora con IAM, Amazon S3 e Lambda, consulta Utilizzo di database globali Amazon Aurora con altri servizi AWS. Per ulteriori informazioni su Secrets Manager, consulta Come automatizzare la replica dei segreti in Gestione dei segreti AWS tra Regioni AWS.

Se utilizzi l'endpoint di scrittura del Database globale Aurora, non occorre cambiare le impostazioni di connessione nell'applicazione. Verifica che le modifiche al DNS si siano propagate e che sia possibile connettersi ed eseguire operazioni di scrittura sul nuovo cluster primario. Potrai quindi riprendere la piena operatività dell'applicazione.

Supponiamo che le connessioni alle applicazioni utilizzino l'endpoint cluster del cluster primario precedente, anziché l'endpoint di scrittura globale. In tal caso, assicurati di modificare le impostazioni di connessione dell'applicazione in modo da utilizzare l'endpoint cluster del nuovo cluster primario. Se hai accettato i nomi forniti al momento della creazione del database globale Aurora, puoi modificare l'endpoint rimuovendo `-ro` dalla stringa endpoint del cluster promosso nell'applicazione. Ad esempio, l'endpoint del cluster secondario `my-global.cluster-ro-aaaaaabbbbbb.us-west-1.rds.amazonaws.com` diventa `my-global.cluster-aaaaaabbbbbb.us-west-1.rds.amazonaws.com` quando tale cluster viene promosso a primario.

Se utilizzi Server proxy per RDS, assicurati di reindirizzare le operazioni di scrittura dell'applicazione all'endpoint di lettura/scrittura appropriato del proxy associato al nuovo cluster primario. Questo endpoint del proxy può essere l'endpoint predefinito o un endpoint di lettura/scrittura personalizzato. Per ulteriori informazioni, consulta Come funzionano gli endpoint Server proxy per RDS con i database globali.

Puoi eseguire lo switchover del Database globale Aurora utilizzando la Console di gestione AWS, la AWS CLI o l'API RDS.

Console

Esecuzione dello switchover nel database globale Aurora

1. Accedi alla Console di gestione AWS e apri la console Amazon RDS all'indirizzo https://console.aws.amazon.com/rds/.
2. Scegli Database e individua il Database globale Aurora in cui intendi eseguire lo switchover.
3. Scegli Switchover o failover database globale nel menu Operazioni.



4. Scegli Switchover.



5. In Nuovo cluster primario, scegli un cluster attivo in una delle Regioni AWS secondarie da promuovere a nuovo cluster primario.
6. Scegli Conferma.

Al termine dello switchover, potrai visualizzare i cluster database Aurora e il relativo stato corrente nell'elenco Database, come illustrato nella figura seguente.



AWS CLI

Esecuzione dello switchover nel database globale Aurora

Utilizza il comando CLI `switchover-global-cluster` per eseguire lo switchover per il Database globale Aurora. Questo comando consente di passare i valori per i seguenti parametri.

- `--region`: specifica la Regione AWS in cui è in esecuzione il cluster di database primario del database globale Aurora.
- `--global-cluster-identifier` – Specifica il nome del database globale Aurora.
- `--target-db-cluster-identifier` – Specifica l'Amazon Resource Name (ARN) del cluster di database Aurora che si desidera promuovere come principale per il database globale Aurora.

Per Linux, macOS o Unix:

```
aws rds --region region_of_primary \
   switchover-global-cluster --global-cluster-identifier global_database_id \
  --target-db-cluster-identifier arn_of_secondary_to_promote
```

Per Windows:

```
aws rds --region region_of_primary ^
   switchover-global-cluster --global-cluster-identifier global_database_id ^
```

```
  --target-db-cluster-identifier arn_of_secondary_to_promote
```

API RDS

Per eseguire lo switchover per il Database globale Aurora, esegui l'operazione API SwitchoverGlobalCluster.

Ripristino di un database globale Amazon Aurora da un'interruzione non pianificata

In rare occasioni, per il Database globale Aurora potrebbe verificarsi un'interruzione inaspettata nella Regione AWS primaria. In questo caso, il cluster database Aurora primario e il relativo nodo di scrittura non sono disponibili e la replica tra il cluster database primario e secondari viene interrotta. Per ridurre al minimo i tempi di inattività (RTO) e la perdita di dati (RPO), puoi eseguire un failover tra regioni.

Database globale Aurora offre due metodi di failover da utilizzare in una situazione di disaster recovery:

- Failover gestito: questo metodo è consigliato in situazioni che prevedono il ripristino di emergenza. Quando si utilizza questo metodo, Aurora aggiunge automaticamente la vecchia regione primaria al database globale come regione secondaria quando diventa nuovamente disponibile. Pertanto, viene mantenuta la topologia originale del cluster globale. Per informazioni su come utilizzare questo metodo, consulta Esecuzione di failover gestiti per database globali Aurora.
- Failover manuale: questo metodo alternativo può essere utilizzato quando il failover gestito non è un'opzione, ad esempio quando le regioni primarie e secondarie utilizzano versioni del motore non compatibili. Per informazioni su come utilizzare questo metodo, consulta Esecuzione di failover manuali per i Database globali Aurora.

> **Important**
>
> Entrambi i metodi di failover possono comportare la perdita dei dati delle transazioni di scrittura che non sono stati replicati sul dispositivo secondario scelto prima che si verificasse l'evento di failover. Tuttavia, il processo di ripristino che promuove un'istanza database sul cluster database secondario scelto come istanza database di scrittura principale garantisce che i dati si trovino in uno stato transazionale coerente. I failover sono inoltre soggetti a problemi di tipo split-brain.

Esecuzione di failover gestiti per database globali Aurora

Questo approccio è destinato alla continuità aziendale in caso di una reale emergenza a livello regionale o di un'interruzione completa del livello di servizio.

Durante un failover gestito, il cluster secondario nella Regione secondaria scelta diventa il nuovo cluster primario. Il cluster secondario scelto promuove uno dei suoi nodi di sola lettura allo stato di istanza di scrittura completa. Questo passaggio consente al cluster di assumere il ruolo di cluster primario. Il database non sarà disponibile per un breve periodo di tempo mentre il cluster sta assumendo il suo nuovo ruolo. Non appena la precedente Regione primaria è di nuovo integra e disponibile, Aurora la aggiunge automaticamente al cluster globale come Regione secondaria. Pertanto, viene mantenuta la topologia di replica esistente del Database globale Aurora.

> **Note**
>
> Per gestire gli slot di replica per Aurora PostgreSQL dopo aver eseguito un failover, consulta Gestione degli slot logici per Aurora PostgreSQL .

> **Note**
>
> È possibile eseguire failover gestiti tra Regioni con il Database globale Aurora solo se i cluster di database primario e secondario hanno le stesse versioni principale e secondaria del motore. A seconda del motore e delle versioni del motore, può essere necessario che i livelli di patch siano identici oppure possono essere diversi. Per un elenco dei motori e delle versioni dei motori che consentono queste operazioni tra cluster primari e secondari con diversi livelli di patch, consulta Compatibilità del livello di patch per switchover e failover gestiti tra regioni. Prima di iniziare il failover, controlla le versioni del motore nel cluster globale per assicurarti che supportino lo switchover gestito tra Regioni e, se necessario, aggiornale. Se le versioni dei motori richiedono livelli di patch identici ma utilizzano livelli di patch diversi, puoi eseguire il failover manualmente tramite la procedura indicata in Esecuzione di failover manuali per i Database globali Aurora.

Il failover gestito non attende la sincronizzazione dei dati tra la Regione secondaria scelta e la Regione primaria corrente. Poiché il Database globale Aurora replica i dati in modo asincrono, è possibile che non tutte le transazioni vengano replicate nella Regione AWS secondaria scelta prima della promozione per accettare funzionalità complete di lettura/scrittura.

Per garantire che i dati siano in uno stato coerente, Aurora crea un nuovo volume di archiviazione per la precedente Regione primaria dopo il ripristino. Prima di creare il nuovo volume di archiviazione nella Regione AWS, Aurora tenta di acquisire uno snapshot del volume di archiviazione precedente nel punto in cui si è verificato l'errore. Questo consente di ripristinare lo snapshot e recuperare i dati mancanti. Se questa operazione ha esito positivo, Aurora inserisce questo snapshot denominato `rds:unplanned-global-failover-`*`name-of-old-primary-DB-cluster`*`-`*`timestamp`* nella sezione Snapshot della Console di gestione AWS. È inoltre possibile utilizzare il comando AWS CLI `describe-db-cluster-snapshots` o l'operazione API `DescribeDBClusterSnapshots` per visualizzare i dettagli dello snapshot.

Quando si avvia un failover gestito, Aurora tenta anche di arrestare il traffico di scrittura nel livello di archiviazione Aurora ad alta disponibilità. Questo meccanismo è chiamato "write fencing". Se il processo ha esito positivo, Aurora emette un evento RDS per indicare che le scritture sono state interrotte. Nell'improbabile eventualità che si verifichino più errori AZ in una Regione, è possibile che il processo di write fencing non abbia esito positivo tempestivamente. In tal caso, Aurora emette un evento RDS che informa l'utente che si è verificato il timeout del processo di interruzione delle scritture. Se il cluster primario precedente è raggiungibile sulla rete, Aurora vi registra questi eventi. In caso contrario, Aurora registra gli eventi sul nuovo cluster primario. Per ulteriori informazioni su questi eventi, consulta Eventi di cluster di database. Poiché il fencing di scritture è un tentativo migliore possibile, è possibile che le scritture vengano momentaneamente accettate nella Regione primaria precedente, causando problemi di tipo split-brain.

È consigliabile completare le seguenti attività prima di eseguire un failover con Database globale Aurora. In questo modo si riduce al minimo l'incidenza di problemi di tipo split-brain o il ripristino di dati non replicati dallo snapshot del cluster primario precedente.

- Per impedire l'invio di scrittura al cluster primario del Database globale Aurora, metti offline le applicazioni.
- Assicurati che tutte le applicazioni che si connettono al cluster di database primario utilizzino l'endpoint di scrittura globale. Questo endpoint ha un valore che rimane invariato anche quando una nuova Regione diventa il cluster primario a causa dello switchover o del failover. Aurora implementa misure di protezione aggiuntive per ridurre al minimo la possibilità di perdita di dati per le operazioni di scrittura inviate tramite l'endpoint globale. Per ulteriori informazioni sugli endpoint di scrittura globali, consulta Connessione a Database globale Amazon Aurora.
- Se utilizzi l'endpoint di scrittura globale e i livelli applicativi o di rete memorizzano nella cache i valori DNS, imposta il time-to-live (TTL) della cache DNS su un valore più basso, ad esempio 5 secondi. In questo modo, l'applicazione registra rapidamente le modifiche DNS con l'endpoint di

scrittura globale. Sebbene Aurora tenti di bloccare le scritture nella vecchia Regione primaria, non è garantito che l'azione riesca. La riduzione della durata della cache DNS riduce ulteriormente la probabilità di problemi di tipo split-brain. In alternativa, puoi verificare la presenza dell'evento RDS che ti indica quando Aurora ha osservato le modifiche DNS per l'endpoint di scrittura globale. In questo modo, puoi verificare che l'applicazione abbia registrato anche la modifica DNS prima di riavviare il traffico di scrittura dell'applicazione.

- Controllare i tempi di ritardo per tutti i cluster di database Aurora secondari nel Database globale Aurora. La scelta della regione secondaria con il minor ritardo di replica può ridurre al minimo la perdita di dati relativamente all'attuale regione primaria in stato di errore.

  Per tutte le versioni dei database globali basati su Aurora PostgreSQL e per i database globali basati su Aurora MySQL con versioni del motore 3.04.0 e successive o 2.12.0 e successive, usa Amazon CloudWatch per visualizzare la metrica `AuroraGlobalDBRPOLag` per tutti i cluster di database secondari. Per le versioni minori precedenti dei database globali basati su Aurora MySQL, puoi invece visualizzare la metrica `AuroraGlobalDBReplicationLag`. Questa metrica indica il ritardo (in millisecondi) della replica tra un cluster secondario e il cluster database primario.

  Per ulteriori informazioni sulle metriche di CloudWatch per Aurora, consulta Parametri a livello di cluster per Amazon Aurora.

Durante un failover gestito, il cluster database secondario scelto viene promosso al nuovo ruolo primario. Tuttavia, non eredita le varie opzioni di configurazione del cluster di database primario. Una mancata corrispondenza nella configurazione può causare problemi di prestazioni, incompatibilità dei carichi di lavoro e altri comportamenti anomali. Per evitare tali problemi, è consigliabile risolvere le differenze tra i cluster di database globali Aurora per quanto segue:

- Configura il gruppo di parametri del cluster di database Aurora per il nuovo primario, se necessario – Puoi configurare i gruppi di parametri del cluster di database Aurora in modo indipendente per ogni cluster Aurora del Database globale Aurora. Pertanto, quando si promuove un cluster database secondario perché assuma il ruolo primario, il gruppo di parametri dal cluster secondario potrebbe essere configurato in modo diverso rispetto al cluster primario. In tal caso, modifica il gruppo di parametri del cluster di database secondario promosso in modo che sia conforme alle impostazioni del cluster primario. Per scoprire come, consulta Modifica dei parametri per un database globale Aurora.
- Configura gli strumenti e le opzioni di monitoraggio, ad esempio Amazon CloudWatch Events e allarmi – Configura il cluster di database promosso con la stessa capacità di registrazione, gli allarmi e così via in base alle esigenze del database globale. Come per i gruppi di parametri,

la configurazione di queste funzionalità non viene ereditata dal primario durante il processo di failover. Alcune metriche CloudWatch, come il ritardo di replica, sono disponibili solo per le regioni secondarie. Pertanto, un failover modifica il modo in cui visualizzare tali metriche e impostare i relativi allarmi e potrebbe richiedere modifiche da apportare a qualsiasi dashboard predefinito. Per ulteriori informazioni sul monitoraggio dei cluster di database Aurora, consulta Monitoraggio dei parametri di Amazon Aurora con Amazon CloudWatch.

- Configura le integrazioni con altri servizi AWS: se il Database globale Aurora si integra con altri servizi AWS, ad esempio Gestione dei segreti AWS, AWS Identity and Access Management, Amazon S3 e AWS Lambda, è necessario che questi siano configurati per l'accesso da qualsiasi Regione secondaria. Per ulteriori informazioni sull'integrazione dei database globali Aurora con IAM, Amazon S3 e Lambda, consulta Utilizzo di database globali Amazon Aurora con altri servizi AWS. Per ulteriori informazioni su Secrets Manager, consulta Come automatizzare la replica dei segreti in Gestione dei segreti AWS tra Regioni AWS.

In genere, il cluster secondario scelto assume il ruolo primario entro pochi minuti. Non appena l'istanza database di scrittura della nuova Regione primaria è disponibile, puoi connettervi le tue applicazioni e riprendere i tuoi carichi di lavoro. Dopo aver promosso il nuovo cluster primario, Aurora ricostruisce automaticamente tutti i cluster secondari regionali aggiuntivi.

Poiché i database globali Aurora utilizzano la replica asincrona, il ritardo di replica in ciascuna regione secondaria può variare. Aurora ricostruisce queste regioni secondarie in modo che abbiano esattamente gli stessi dati point-in-time del nuovo cluster regionale primario. La durata dell'attività di ricostruzione completa può richiedere da alcuni minuti a diverse ore, a seconda delle dimensioni del volume di archiviazione e della distanza tra regioni. Quando i cluster regionali secondari terminano la ricostruzione in base alla nuova regione primaria, diventano disponibili per l'accesso in lettura.

Non appena la nuova istanza di scrittura primaria viene promossa e risulta disponibile, il cluster della nuova regione primaria può gestire le operazioni di lettura e scrittura per il database globale Aurora.

Se utilizzi l'endpoint globale, non occorre cambiare le impostazioni di connessione nell'applicazione. Verifica che le modifiche al DNS si siano propagate e che sia possibile connettersi ed eseguire operazioni di scrittura sul nuovo cluster primario. Potrai quindi riprendere la piena operatività dell'applicazione.

Se non utilizzi l'endpoint globale, assicurati di modificare l'endpoint dell'applicazione in modo che utilizzi l'endpoint del cluster per il cluster di database primario appena promosso. Se hai accettato i nomi forniti al momento della creazione del database globale Aurora, puoi modificare l'endpoint rimuovendo `-ro` dalla stringa endpoint del cluster promosso nell'applicazione.

Ad esempio, l'endpoint del cluster secondario `my-global.cluster-ro-aaaaaabbbbbb.us-west-1.rds.amazonaws.com` diventa `my-global.cluster-aaaaaabbbbbb.us-west-1.rds.amazonaws.com` quando tale cluster viene promosso a primario.

Se utilizzi il proxy RDS, assicurati di reindirizzare le operazioni di scrittura dell'applicazione all'endpoint di lettura/scrittura appropriato del proxy associato al nuovo cluster primario. Questo endpoint del proxy può essere l'endpoint predefinito o un endpoint di lettura/scrittura personalizzato. Per ulteriori informazioni, consulta Come funzionano gli endpoint Server proxy per RDS con i database globali.

Per ripristinare la topologia originale del cluster database globale, Aurora monitora la disponibilità della vecchia regione primaria. Non appena tale regione è di nuovo integra e disponibile, Aurora la aggiunge automaticamente al cluster globale come regione secondaria. Prima di creare il nuovo volume di archiviazione nella vecchia regione primaria, Aurora tenta di acquisire uno snapshot del vecchio volume di archiviazione nel punto in cui si è verificato l'errore. Ciò consente di usare lo snapshot per recuperare i dati mancanti. Se questa operazione ha esito positivo, Aurora crea uno snapshot denominato `rds:unplanned-global-failover-`*`name-of-old-primary-DB-cluster`*`-`*`timestamp`*. Questo snapshot è disponibile nella sezione Snapshot di Console di gestione AWS. Questo snapshot è disponibile anche nelle informazioni restituite dall'operazione API DescribeDBClusterSnapshots.

> **Note**
>
> Lo snapshot del vecchio volume di archiviazione è uno snapshot del sistema soggetto al periodo di conservazione del backup configurato sul vecchio cluster primario. Per conservare questo snapshot oltre il periodo di conservazione, puoi copiarlo e salvarlo come snapshot manuale. Per ulteriori informazioni sulla copia degli snapshot, inclusi i prezzi, consulta Copia di uno snapshot del cluster di database.

Dopo il ripristino della topologia originale, puoi eseguire il failback del database globale nella regione primaria originale eseguendo un'operazione di switchover nel momento più opportuno per l'azienda e il carico di lavoro. A tale scopo, segui la procedura in Esecuzione di switchover per database globali Amazon Aurora.

Puoi eseguire un failover con il Database globale Aurora utilizzando la Console di gestione AWS, la AWS CLI o l'API RDS.

Console

Esecuzione del failover gestito nel database globale Aurora

1. Accedi alla Console di gestione AWS e apri la console Amazon RDS all'indirizzo https://console.aws.amazon.com/rds/.
2. Scegli Database e individua il Database globale Aurora di cui desideri eseguire il failover.
3. Scegli Switchover o failover database globale nel menu Operazioni.



4. Scegli Failover (consenti perdita di dati).



5. In Nuovo cluster primario, scegli un cluster attivo in una delle Regioni AWS secondarie da promuovere a nuovo cluster primario.
6. Immetti **`confirm`**, quindi scegli Conferma.

Al termine del failover, potrai visualizzare i cluster di database Aurora e il relativo stato corrente nell'elenco Database, come illustrato nella figura seguente.



AWS CLI

Esecuzione del failover gestito nel database globale Aurora

Utilizza il comando della CLI `failover-global-cluster` per eseguire il failover con il Database globale Aurora. Questo comando consente di passare i valori per i seguenti parametri.

- `--region`: specifica la Regione AWS in cui è in esecuzione il cluster database secondario che vuoi promuovere a nuovo cluster primario per il database globale Aurora.
- `--global-cluster-identifier` – Specifica il nome del database globale Aurora.
- `--target-db-cluster-identifier`: specifica il nome della risorsa Amazon (ARN) del cluster database Aurora che desideri promuovere a nuovo cluster primario per il database globale Aurora.
- `--allow-data-loss`: imposta in modo esplicito un'operazione di failover anziché un'operazione di switchover. Un'operazione di failover può causare una perdita di dati se i componenti della replica asincrona non hanno completato l'invio di tutti i dati replicati alla regione secondaria.

Per Linux, macOS o Unix:

```
aws rds --region region_of_selected_secondary \
   failover-global-cluster --global-cluster-identifier global_database_id \
   --target-db-cluster-identifier arn_of_secondary_to_promote \
   --allow-data-loss
```

Per Windows:

```
aws rds --region region_of_selected_secondary ^
   failover-global-cluster --global-cluster-identifier global_database_id ^
   --target-db-cluster-identifier arn_of_secondary_to_promote ^
   --allow-data-loss
```

API RDS

Per eseguire un failover con il Database globale Aurora, esegui l'operazione API FailoverGlobalCluster.

Esecuzione di failover manuali per i Database globali Aurora

In alcuni scenari, puoi non essere in grado di utilizzare il processo di failover gestito. Un esempio è quando i cluster database primario e secondari non utilizzano versioni del motore compatibili. In questo caso, puoi usare questa procedura manuale per eseguire un failover nella Regione secondaria di destinazione.

> **Tip**
>
> Si consiglia di comprendere questo processo prima di utilizzarlo. Prepara un piano per procedere rapidamente al primo segno di un problema a livello regionale. Puoi essere pronto a identificare la regione secondaria con il minor ritardo di replica utilizzando Amazon CloudWatch con regolarità per monitorare i tempi di ritardo dei cluster secondario. Assicurati di testare il piano per verificare che le procedure siano complete e accurate e che il personale sia addestrato per eseguire un failover in caso di ripristino di emergenza prima che ciò avvenga realmente.

Per eseguire un failover manuale in un cluster secondario dopo un'interruzione non pianificata nella Regione primaria

1. Interrompi l'emissione di istruzioni DML e altre operazioni di scrittura sul cluster di database Aurora primario nella Regione AWS con l'interruzione.
2. Identifica un cluster di database Aurora da una Regione AWS secondaria da utilizzare come nuovo cluster di database primario. Se nel database globale Aurora sono presenti due o più Regioni AWS secondarie, scegli il cluster secondario con il ritardo di replica minore.

3. Scollega il cluster di database secondario scelto dal database globale Aurora.

   La rimozione di un cluster di database secondario da un database globale Aurora interrompe immediatamente la replica dal primario al secondario e la promuove a cluster di database Aurora con provisioning autonomo con funzionalità di lettura/scrittura complete. Tutti gli altri cluster di database Aurora secondari associati al cluster primario nella regione con interruzione sono ancora disponibili e possono accettare chiamate dall'applicazione. Inoltre consumano risorse. Poiché si sta ricreando il database globale Aurora, rimuovi gli altri cluster di database secondari prima di creare il nuovo database globale Aurora nei passaggi seguenti. In questo modo, si evitano incongruenze di dati tra i cluster di database nel database globale Aurora (problemi di split-brain).

   Per i passaggi dettagliati per lo scollegamento, consulta Rimozione di un cluster da un database globale Amazon Aurora.

4. Riconfigura l'applicazione per inviare tutte le operazioni di scrittura a questo cluster di database Aurora ora autonomo utilizzando il nuovo endpoint. Se sono stati accettati i nomi forniti al momento della creazione del database globale Aurora, puoi modificare l'endpoint rimuovendo `-ro` dalla stringa endpoint del cluster nell'applicazione.

   Ad esempio, l'endpoint del cluster secondario `my-global.cluster-ro-aaaaaabbbbbb.us-west-1.rds.amazonaws.com` diventa `my-global.cluster-aaaaaabbbbbb.us-west-1.rds.amazonaws.com` quando tale cluster viene scollegato dal database globale Aurora.

   Questo cluster di database Aurora diventa il cluster primario di un nuovo database globale Aurora quando si inizia ad aggiungere regioni nel passaggio successivo.

   Se utilizzi il proxy RDS, assicurati di reindirizzare le operazioni di scrittura dell'applicazione all'endpoint di lettura/scrittura appropriato del proxy associato al nuovo cluster primario. Questo endpoint del proxy può essere l'endpoint predefinito o un endpoint di lettura/scrittura personalizzato. Per ulteriori informazioni, consulta Come funzionano gli endpoint Server proxy per RDS con i database globali.

5. Aggiungi una Regione AWS al cluster di database. Quando esegui questa operazione, inizia il processo di replica da primario a secondario. Per i passaggi dettagliati per aggiungere una regione, consulta Aggiunta di una Regione AWS a un database globale Amazon Aurora.

6. Aggiungi altre Regioni AWS in base alle esigenze per ricreare la topologia necessaria per supportare l'applicazione.

Assicurati che le scritture delle applicazioni vengano inviate al cluster di database Aurora corretto prima, durante e dopo aver apportato queste modifiche. In questo modo, si evitano incongruenze di dati tra i cluster di database nel database globale Aurora (problemi di split-brain).

Se hai eseguito la riconfigurazione in risposta a un'interruzione di una Regione AWS, puoi nuovamente rendere la Regione AWS come principale dopo la risoluzione dell'interruzione. A tale scopo, aggiungi la precedente Regione AWS al nuovo database globale e quindi usa il processo di switchover per cambiare il ruolo. Il database globale Aurora deve utilizzare una versione di Aurora PostgreSQL o Aurora MySQL che supporti gli switchover. Per ulteriori informazioni, consulta Esecuzione di switchover per database globali Amazon Aurora.

Gestione degli RPO per database globali basati su Aurora PostgreSQL–

Con un database globale basato su Aurora PostgreSQL, puoi gestire l'obiettivo del punto di ripristino (RPO) per il database globale Aurora utilizzando il parametro `rds.global_db_rpo`. RPO rappresenta la quantità massima di dati che possono essere persi in caso di interruzione.

Quando si imposta un RPO per il database globale basato su Aurora PostgreSQL–, Aurora controlla il tempo di ritardo RPO di tutti i cluster secondari per assicurarsi che almeno un cluster secondario rimanga all'interno della finestra RPO di destinazione. Il tempo di ritardo RPO è un altro parametro basato sul tempo.

L'RPO viene utilizzato quando il database riprende le operazioni in una nuova Regione AWS dopo un failover. Aurora valuta i tempi di ritardo RPO e RPO per eseguire il commit (o il blocco) delle transazioni sulla regione principale come segue:

- Conferma la transazione se almeno un cluster di database secondario ha un tempo di ritardo RPO inferiore rispetto all'RPO.
- Blocca la transazione se tutti i cluster di database secondari hanno tempi di ritardo RPO superiori all'RPO. Registra inoltre l'evento nel file di log di PostgreSQL ed emette eventi di “attesa” che mostrano le sessioni bloccate.

In altre parole, se tutti i cluster secondari sono dietro l'RPO di destinazione, Aurora sospende le transazioni sul cluster primario fino al raggiungimento di almeno uno dei cluster secondari. Le transazioni in pausa vengono nuovamente eseguite non appena il tempo di ritardo di almeno un cluster di database secondario diventa inferiore all'RPO. Il risultato è che nessuna transazione può eseguire il commit fino al raggiungimento dell'RPO.

Il parametro `rds.global_db_rpo` è dinamico. Se decidi che non vuoi che tutte le transazioni di scrittura si blocchino fino a quando il ritardo non diminuisce a un livello sufficiente, puoi ripristinarlo rapidamente. In questo caso, Aurora riconosce e implementa la modifica dopo un breve ritardo.

> **Important**
>
> In un database globale con solo due Regioni AWS, è consigliabile mantenere il valore predefinito del parametro `rds.global_db_rpo` nel gruppo di parametri della Regione secondaria. Diversamente, l'esecuzione di un failover a causa della perdita della Regione AWS primaria potrebbe causare la sospensione delle transazioni da parte di Aurora. Attendi invece che Aurora completi la ricostruzione del cluster nella vecchia Regione AWS in cui si è verificato l'errore prima di modificare questo parametro per imporre un RPO massimo.

Se imposti questo parametro come descritto di seguito, puoi monitorare anche le metriche generate. Puoi eseguire questa operazione utilizzando `psql` o un altro strumento per interrogare il cluster di database primario del database globale Aurora e ottenere informazioni dettagliate sulle operazioni del database globale basato su Aurora PostgreSQL–. Per scoprire come, consulta Monitoraggio dei database globali Aurora basati su PostgreSQL.

Argomenti

- Impostazione dell'obiettivo del punto di ripristino
- Visualizzazione dell'obiettivo del punto di ripristino
- Disattivazione dell'obiettivo del punto di ripristino

Impostazione dell'obiettivo del punto di ripristino

Il parametro `rds.global_db_rpo` controlla l'impostazione RPO per un database PostgreSQL. Questo parametro è supportato da Aurora PostgreSQL. I valori validi per `rds.global_db_rpo` vanno da 20 secondi a 2.147.483.647 secondi (68 anni). Scegli un valore realistico per soddisfare le tue esigenze aziendali e il caso d'uso. Ad esempio, puoi consentire fino a 10 minuti per l'RPO, nel qual caso si imposta il valore su 600.

Puoi impostare questo valore per il database globale basato su Aurora PostgreSQL utilizzando la Console di gestione AWS, la AWS CLI o l'API RDS.

Console

Per impostare l'RPO

1. Accedi alla Console di gestione AWS e apri la console Amazon RDS all'indirizzo https://console.aws.amazon.com/rds/.
2. Sceglie il cluster primario del database Aurora globale e apri la scheda Configurazione per trovare il relativo gruppo di parametri del cluster di database. Ad esempio, il gruppo di parametri predefinito per un cluster di database primario che esegue Aurora PostgreSQL 11.7 è `default.aurora-postgresql11`.

   I gruppi di parametri non possono essere modificati direttamente. Puoi invece procedere come descritto di seguito:

   - Crea un gruppo di parametri cluster di database personalizzato utilizzando il gruppo di parametri predefinito appropriato come punto di partenza. Ad esempio, crea un gruppo di parametri cluster di database personalizzato basato su `default.aurora-postgresql11`.
   - Nel gruppo di parametri database personalizzato, imposta il valore del parametro rds.global_db_rpo in modo da soddisfare il caso d'uso. I valori validi vanno da 20 secondi fino al valore intero massimo di 2.147.483.647 (68 anni).
   - Applica il gruppo di parametri del cluster di database modificato al cluster di database Aurora.

Per ulteriori informazioni, consulta Modifica dei parametri in un gruppo di parametri del cluster di database in Amazon Aurora.

AWS CLI

Per impostare il parametro `rds.global_db_rpo`, utilizza il comando della CLI modify-db-cluster-parameter-group. Nel comando specifica il nome del gruppo di parametri del cluster primario e i valori per il parametro RPO.

Nell'esempio seguente l'RPO viene impostato su 600 secondi per il gruppo di parametri cluster di database primario denominato `my_custom_global_parameter_group`.

Per Linux, macOS o Unix:

```
aws rds modify-db-cluster-parameter-group \
    --db-cluster-parameter-group-name my_custom_global_parameter_group \
```

```
    --parameters
 "ParameterName=rds.global_db_rpo,ParameterValue=600,ApplyMethod=immediate"
```

Per Windows:

```
aws rds modify-db-cluster-parameter-group ^
    --db-cluster-parameter-group-name my_custom_global_parameter_group ^
    --parameters
 "ParameterName=rds.global_db_rpo,ParameterValue=600,ApplyMethod=immediate"
```

API RDS

Per modificare il parametro `rds.global_db_rpo`, utilizzare l'operazione API Amazon RDS ModifyDBClusterParameterGroup.

Visualizzazione dell'obiettivo del punto di ripristino

L'obiettivo del punto di ripristino (RPO) di un database globale viene memorizzato nel parametro `rds.global_db_rpo`. Puoi connettersi all'endpoint per il cluster secondario che si desidera visualizzare e utilizzare per `psql` eseguire una query sull'istanza per questo valore.

```
db-name=>show rds.global_db_rpo;
```

Se questo parametro non è impostato, la query restituirà quanto segue:

```
rds.global_db_rpo
-------------------
 -1
(1 row)
```

Questa risposta successiva proviene da un cluster di database secondario con impostazione RPO di 1 minuto.

```
rds.global_db_rpo
-------------------
 60
(1 row)
```

Puoi inoltre utilizzare la CLI per ottenere valori per scoprire se `rds.global_db_rpo` è attivo su uno qualsiasi dei cluster di database Aurora utilizzando l'interfaccia a riga di comando per ottenere i valori di tutti i parametri `user` per il cluster.

Per Linux, macOS o Unix:

```
aws rds describe-db-cluster-parameters \
 --db-cluster-parameter-group-name lab-test-apg-global \
 --source user
```

Per Windows:

```
aws rds describe-db-cluster-parameters ^
 --db-cluster-parameter-group-name lab-test-apg-global *
 --source user
```

Il comando restituisce un output simile al seguente per tutti i parametri `user` che non sono parametri del cluster di database `default-engine` o `system`.

```
{
    "Parameters": [
        {
            "ParameterName": "rds.global_db_rpo",
            "ParameterValue": "60",
            "Description": "(s) Recovery point objective threshold, in seconds, that
 blocks user commits when it is violated.",
            "Source": "user",
            "ApplyType": "dynamic",
            "DataType": "integer",
            "AllowedValues": "20-2147483647",
            "IsModifiable": true,
            "ApplyMethod": "immediate",
            "SupportedEngineModes": [
                "provisioned"
            ]
        }
    ]
}
```

Per ulteriori informazioni sulla visualizzazione dei parametri del gruppo di parametri del cluster, consulta Visualizzazione dei valori dei parametri per un gruppo di parametri del cluster di database in Amazon Aurora.

Disattivazione dell'obiettivo del punto di ripristino

Per disabilitare l'RPO, reimpostare il parametro `rds.global_db_rpo`. È possibile reimpostare i parametri utilizzando Console di gestione AWS, AWS CLI o l'API RDS.

Console

Per disabilitare l'RPO

1. Accedi alla Console di gestione AWS e apri la console Amazon RDS all'indirizzo https://console.aws.amazon.com/rds/.
2. Nel riquadro di navigazione scegliere Parameter groups (Gruppi di parametri).
3. Nell'elenco scegliere il gruppo di parametri cluster DB primario.
4. Scegliere Edit parameters (Modifica parametri).
5. Scegliere la casella accanto al parametro rds.global_db_rpo.
6. Scegliere Reimposta.
7. Quando la schermata mostra Reimposta parametri nel gruppo di parametri DB, scegliere Reimposta parametri.

Per ulteriori informazioni su come reimpostare un parametro con la console, vedere Modifica dei parametri in un gruppo di parametri del cluster di database in Amazon Aurora.

AWS CLI

Per reimpostare il parametro `rds.global_db_rpo`, utilizzare il comando reset-db-cluster-parameter-group.

Per Linux, macOS o Unix:

```
aws rds reset-db-cluster-parameter-group \
    --db-cluster-parameter-group-name global_db_cluster_parameter_group \
    --parameters "ParameterName=rds.global_db_rpo,ApplyMethod=immediate"
```

Per Windows:

```
aws rds reset-db-cluster-parameter-group ^
    --db-cluster-parameter-group-name global_db_cluster_parameter_group ^
    --parameters "ParameterName=rds.global_db_rpo,ApplyMethod=immediate"
```

API RDS

Per reimpostare il parametro `rds.global_db_rpo`, utilizzare l'operazione Amazon RDS API ResetDBClusterParameterGroup.

Resilienza tra Regioni per i cluster secondari del database globale

Le versioni 16.6, 15.10, 14.15, 13.18, 12.22 o successive di Aurora PostgreSQL e le versioni 3.09 o successive di Aurora MySQL contengono miglioramenti della disponibilità che consentono alle repliche di lettura regionali secondarie di mantenere la continuità del servizio durante eventi non pianificati come guasti hardware, interruzioni della rete tra Regioni AWS, grandi volumi di trasferimenti di dati tra i cluster e altro.

Sebbene le repliche di lettura rimangano disponibili per le richieste delle applicazioni, il ritardo di replica potrebbe continuare ad aumentare fino alla risoluzione dell'evento non pianificato. È possibile monitorare il ritardo tra i cluster primari e secondari utilizzando la metrica CloudWatch di `AuroraGlobalDBProgressLag`. Per misurare il ritardo end-to-end, incluso qualsiasi ritardo tra il volume del cluster e le istanze database del cluster secondario, aggiungi i valori delle metriche CloudWatch di `AuroraGlobalDBProgressLag` e `AuroraReplicaLag`. Per ulteriori informazioni sulle metriche, consulta Riferimento per i parametri per Amazon Aurora.

La disponibilità di lettura del database globale per Aurora MySQL e le versioni precedenti di Aurora PostgreSQL potrebbe risentire di tali eventi non pianificati.

Per ulteriori informazioni sulle nuove funzionalità di Aurora PostgreSQL 16.6, 15.10, 14.15, 13.18 e 12.22, consulta PostgreSQL 16.6 nelle Note di rilascio di Aurora PostgreSQL.

Per ulteriori informazioni sulle nuove funzionalità di Aurora MySQL versione 3.09 e successive, consulta Aggiornamenti del motore del database per Amazon Aurora MySQL versione 3 nelle Note di rilascio di Aurora MySQL.

Monitoraggio di un database globale Amazon Aurora

Quando si creano i cluster di database Aurora che costituiscono il database globale Aurora, è possibile scegliere molte opzioni che consentono di monitorare le prestazioni del cluster di database. Queste opzioni includono:

- Amazon RDS Performance Insights – Supporta lo schema delle prestazioni nel motore di database Aurora sottostante. Per ulteriori informazioni su Performance Insights e database globali Aurora,

consulta Monitoraggio di un database globale Amazon Aurora con Amazon RDS Performance Insights.

- Monitoraggio avanzato – Genera parametri per l'utilizzo di processi o thread sulla CPU. Per ulteriori informazioni sul monitoraggio avanzato, consulta Monitoraggio dei parametri del sistema operativo con il monitoraggio avanzato.
- Amazon CloudWatch Logs – Pubblica i tipi di log specificati in CloudWatch Logs. I log degli errori vengono pubblicati per impostazione predefinita, ma è possibile scegliere altri log specifici per il motore di database Aurora.
  - Per i cluster di database Aurora basati su Aurora MySQL è possibile esportare il log di controllo, il log generale e il log delle query lente.
  - Per i cluster database Aurora basati su Aurora PostgreSQL, puoi esportare il log PostgreSQL.
- Per i database globali basati su Aurora MySQL, è possibile eseguire query su tabelle `information_schema` specifiche per verificare lo stato del database globale Aurora e delle relative istanze. Per scoprire come, consulta Monitoraggio dei database globali basati su Aurora MySQL.
- Per i database globali basati su Aurora PostgreSQL, è possibile utilizzare funzioni specifiche per verificare lo stato del database globale Aurora e delle relative istanze. Per scoprire come, consulta Monitoraggio dei database globali Aurora basati su PostgreSQL.

La seguente schermata mostra alcune delle opzioni disponibili nella scheda Monitoraggio di un cluster di database Aurora primario in un database globale Aurora.



Per ulteriori informazioni, consulta Monitoraggio dei parametri in un cluster di database Amazon Aurora.

Monitoraggio di un database globale Amazon Aurora con Amazon RDS Performance Insights

È possibile utilizzare Amazon RDS Performance Insights per i database globali Aurora. È possibile abilitare questa funzionalità singolarmente, per ogni cluster di database Aurora nel database globale Aurora. A tale scopo, scegliere Enable Performance Insights (Abilita Performance Insights) nella sezione Additional configuration (Configurazioni aggiuntive) della pagina Crea database. In alternativa, una volta operativi, è possibile modificare i cluster di database Aurora per utilizzare questa funzionalità. È possibile abilitare o disattivare Performance Insights per ciascun cluster che fa parte del database globale Aurora.

I report creati da Performance Insights si applicano a ciascun cluster del database globale. Quando si aggiunge una nuova Regione AWS secondaria a un database globale Aurora che sta già utilizzando

Performance Insights, sarà necessario abilitare Performance Insights nel cluster appena aggiunto. Non eredita l'impostazione Performance Insights dal database globale esistente.

È possibile cambiare le Regioni AWS mentre si visualizza la pagina Performance Insights per un'istanza database allegata a un database globale. Tuttavia, potreste non vedere le informazioni sulle prestazioni subito dopo aver cambiato Regioni AWS. Anche se le istanze database potrebbero avere gli stessi nomi in ciascuna Regione AWS, l'URL Performance Insights associato è diverso per ogni istanza database. Dopo aver cambiato Regioni AWS, scegliere di nuovo il nome dell'istanza database nel riquadro di navigazione Performance Insights.

Per le istanze database associate a un database globale, i fattori che influiscono sulle prestazioni potrebbero essere diversi in ciascuna Regione AWS. Ad esempio, le istanze database in ciascuna Regione AWS potrebbero avere una diversa capacità.

Per ulteriori informazioni sull'utilizzo di Performance Insights, consulta Monitoraggio del carico DB con Performance Insights su Amazon Aurora.

Monitoraggio dei database globali Aurora con i flussi di attività di database

Con i flussi di attività del database è possibile monitorare e impostare gli allarmi per l'attività di audit nei cluster database del database globale. Avvia un flusso di attività del database su ciascun cluster database separatamente. Ciascun cluster fornisce i dati di audit al proprio flusso Kinesis all'interno della propria Regione AWS. Per ulteriori informazioni, consulta Monitoraggio di Amazon Aurora tramite i flussi di attività del database.

Monitoraggio dei database globali basati su Aurora MySQL

Per visualizzare lo stato di un database globale basato su Aurora MySQL, eseguire query delle tabelle information_schema.aurora_global_db_status e information_schema.aurora_global_db_instance_status.

> **Note**
>
> Le tabelle `information_schema.aurora_global_db_status` e `information_schema.aurora_global_db_instance_status` sono disponibili solo con i database globali Aurora MySQL 3.04.0 e versioni successive.

Per monitorare un database globale basato su Aurora MySQL

1. Esegui la connessione all'endpoint del cluster primario del database globale utilizzando un client MySQL. Per ulteriori informazioni su come connettersi, consulta Connessione a Database globale Amazon Aurora.
2. Esegui la query sulla tabella `information_schema.aurora_global_db_status` in un comando mysql per elencare i volumi primari e secondari. Questa query restituisce i tempi di ritardo dei cluster database secondari del database globale, come nell'esempio seguente.

```
mysql> select * from information_schema.aurora_global_db_status;
```

```
AWS_REGION | HIGHEST_LSN_WRITTEN | DURABILITY_LAG_IN_MILLISECONDS |
 RPO_LAG_IN_MILLISECONDS | LAST_LAG_CALCULATION_TIMESTAMP | OLDEST_READ_VIEW_TRX_ID
-----------+---------------------+--------------------------------
+-------------------------+---------------------------------
+------------------------
us-east-1  |           183537946 |                              0   |
      0 |  1970-01-01 00:00:00.000000     |               0
us-west-2  |           183537944 |                              428 |
      0 |  2023-02-18 01:26:41.925000     |               20806982
(2 rows)
```

L'output include una riga per ogni cluster DB del database globale contenente le seguenti colonne:

- AWS_REGION: la Regione AWS in cui si trova questo cluster database. Per l'elenco delle tabelle Regioni AWS per motore, consulta Disponibilità nelle regioni.
- HIGHEST_LSN_WRITTEN: il numero di sequenza di log più alto (LSN) attualmente scritto in questo cluster database.

  Numero di sequenza di log (LSN) è un numero sequenziale univoco che identifica un record nel log delle transazioni del database. Gli LSN sono ordinati in modo tale che un LSN più grande rappresenti una transazione successiva.
- DURABILITY_LAG_IN_MILLISECONDS: la differenza nei valori di timestamp tra il parametro `HIGHEST_LSN_WRITTEN` su un cluster database secondario e il parametro `HIGHEST_LSN_WRITTEN` sul cluster database primario. Questo valore è sempre 0 sul cluster database primario del database globale Aurora.

- RPO_LAG_IN_MILLISECONDS: il ritardo dell'obiettivo del punto di ripristino (RPO). Il ritardo dell'obiettivo del punto di ripristino (RPO) è il tempo necessario per memorizzare il COMMIT delle transazioni utente più recenti dopo la sua memorizzazione nel cluster di database primario del database globale Aurora. Questo valore è sempre 0 sul cluster database primario del database globale Aurora.

  In sintesi, questo parametro calcola l'obiettivo del punto di ripristino per ciascun cluster database Aurora MySQL nel database globale Aurora, ovvero quanti dati potrebbero andare perduti in caso di interruzione. Come per il ritardo, l'obiettivo del punto di ripristino (RPO) viene misurato nel tempo.
- LAST_LAG_CALCULATION_TIMESTAMP: il timestamp che specifica l'ultima volta in cui i valori stati calcolati per `DURABILITY_LAG_IN_MILLISECONDS` e `RPO_LAG_IN_MILLISECONDS`. Il valore temporale `1970-01-01 00:00:00+00` indica che questo è il cluster di database primario.
- OLDEST_READ_VIEW_TRX_ID: l'ID della transazione più vecchia che può essere rimossa dall'istanza database di scrittura.

3. Esegui una query sulla tabella `information_schema.aurora_global_db_instance_status` per elencare tutte le istanze database secondarie per il cluster database primario e i cluster database secondari.

```
mysql> select * from information_schema.aurora_global_db_instance_status;
```

```
SERVER_ID            |              SESSION_ID              | AWS_REGION
 | DURABLE_LSN | HIGHEST_LSN_RECEIVED | OLDEST_READ_VIEW_TRX_ID |
 OLDEST_READ_VIEW_LSN | VISIBILITY_LAG_IN_MSEC
---------------------+--------------------------------------+------------
+-------------+----------------------+------------------------
+----------------------+-----------------------
ams-gdb-primary-i2   | MASTER_SESSION_ID                    | us-east-1  |
 183537698   |                    0 |                      0 |
 0 |                      0
ams-gdb-secondary-i1 | cc43165b-bdc6-4651-abbf-4f74f08bf931 | us-west-2  |
 183537689   |           183537692 |                20806928 |
 183537682 |                      0
ams-gdb-secondary-i2 | 53303ff0-70b5-411f-bc86-28d7a53f8c19 | us-west-2  |
 183537689   |           183537692 |                20806928 |
 183537682 |                    677
```

```
ams-gdb-primary-i1   | 5af1e20f-43db-421f-9f0d-2b92774c7d02 | us-east-1  |
 183537697   |            183537698 |                20806930 |
 183537691 |                    21
(4 rows)
```

L'output include una riga per ogni istanza DB del database globale contenente le colonne seguenti:

- SERVER_ID: identificatore del server per l'istanza database.
- SESSION_ID: un identificatore univoco per la sessione corrente. Il valore di `MASTER_SESSION_ID` identifica l'istanza database di lettura (primaria).
- AWS_REGION: la Regione AWS in cui si trova questa istanza database. Per l'elenco delle tabelle Regioni AWS per motore, consulta Disponibilità nelle regioni.
- DURABLE_LSN: l'LSN è stato reso durevole nello storage.
- HIGHEST_LSN_RECEIVED: l'LSN più alto ricevuto dall'istanza database dall'istanza database di scrittura.
- OLDEST_READ_VIEW_TRX_ID: l'ID della transazione più vecchia che può essere rimossa dall'istanza database di scrittura.
- OLDEST_READ_VIEW_LSN: l'LSN più vecchio utilizzato dall'istanza database per leggere dallo storage.
- VISIBILITY_LAG_IN_MSEC: per le istanze di lettura nel cluster database primario, il ritardo in millisecondi di questa istanza database rispetto all'istanza database di scrittura. Per istanze di lettura in un cluster database secondario, il ritardo in millisecondi di questa istanza database rispetto al volume secondario.

Per vedere come questi valori cambiano nel tempo, considerare il seguente blocco di transazioni in cui un inserimento di tabella richiede un'ora:

```
mysql> BEGIN;
mysql> INSERT INTO table1 SELECT Large_Data_That_Takes_1_Hr_To_Insert;
mysql> COMMIT;
```

In alcuni casi, potrebbe esserci una disconnessione di rete tra il cluster DB primario e il cluster DB secondario dopo l'istruzione `BEGIN`. In tal caso, il valore DURABILITY_LAG_IN_MILLISECONDS del cluster database secondario inizia ad aumentare. Alla fine dell'istruzione `INSERT`, il valore DURABILITY_LAG_IN_MILLISECONDS è di 1 ora. Tuttavia, il

valore RPO_LAG_IN_MILLISECONDS è 0 perché tutti i dati utente confermati tra il cluster database primario e il cluster database secondario sono ancora gli stessi. Al termine dell'istruzione `COMMIT`, il valore RPO_LAG_IN_MILLISECONDS aumenta.

Monitoraggio dei database globali Aurora basati su PostgreSQL

Per visualizzare lo stato di un database globale Aurora basato su PostgreSQL, occorre utilizzare le funzioni `aurora_global_db_status` e `aurora_global_db_instance_status`.

> **Note**
>
> Solo Aurora PostgreSQL supporta le funzioni `aurora_global_db_status` e `aurora_global_db_instance_status`.

Per monitorare un database globale basato su Aurora PostgreSQL

1. Connettersi all'endpoint cluster primario del database globale utilizzando un'utilità PostgreSQL come psql. Per ulteriori informazioni su come connettersi, consulta Connessione a Database globale Amazon Aurora.
2. Utilizzare la funzione `aurora_global_db_status` in un comando psql per elencare i volumi primari e secondari. Mostra i tempi di ritardo dei cluster DB secondari del database globale.

```
postgres=> select * from aurora_global_db_status();
```

```
aws_region | highest_lsn_written | durability_lag_in_msec | rpo_lag_in_msec |
 last_lag_calculation_time  | feedback_epoch | feedback_xmin
------------+---------------------+------------------------+-----------------
+----------------------------+----------------+---------------
us-east-1  |         93763984222 |                     -1 |              -1 |
 1970-01-01 00:00:00+00     |              0 |              0
us-west-2  |         93763984222 |                    900 |            1090 |
 2020-05-12 22:49:14.328+00 |              2 |    3315479243
(2 rows)
```

L'output include una riga per ogni cluster DB del database globale contenente le seguenti colonne:

- aws_region: la Regione AWS in cui si trova questo cluster di database. Per l'elenco delle tabelle Regioni AWS per motore, consulta Disponibilità nelle regioni.
- highest_lsn_written – Il numero di sequenza di log più alto (LSN) attualmente scritto in questo cluster DB.

  Numero di sequenza di log (LSN) è un numero sequenziale univoco che identifica un record nel log delle transazioni del database. Gli LSN sono ordinati in modo tale che un LSN più grande rappresenti una transazione successiva.
- durability_lag_in_msec – La differenza di timestamp tra il numero di sequenza di log più alto scritto su un cluster DB secondario (`highest_lsn_written`) e il `highest_lsn_written` sul cluster DB primario.
- rpo_lag_in_msec – Il ritardo dell'obiettivo del punto di ripristino (RPO). Questo ritardo è la differenza di tempo tra il commit delle transazioni utente più recenti memorizzate in un cluster DB secondario e il commit delle transazioni utente più recenti memorizzate nel cluster DB primario.
- last_lag_calculation_time: il timestamp in cui sono stati calcolati i valori per `durability_lag_in_msec` e `rpo_lag_in_msec`.
- feedback_epoch – L'epoca utilizzata da un cluster di database secondario quando genera informazioni di standby a caldo.

  Hot Standby – Si verifica quando un cluster DB può connettersi e interrogare mentre il server è in modalità di ripristino o standby. Il feedback hot standby è costituito da informazioni sul cluster DB quando è in standby. Per ulteriori informazioni, consulta Hot Standby nella documentazione di PostgreSQL.
- feedback_xmin – L'ID minimo della transazione attiva (meno recente) utilizzato da un cluster di database secondario.

3. Utilizzare la funzione `aurora_global_db_instance_status` per elencare tutte le istanze database secondarie sia per il cluster DB primario che per i cluster DB secondari.

```
postgres=> select * from aurora_global_db_instance_status();
```

```
server_id                                    |              session_id
  | aws_region | durable_lsn | highest_lsn_rcvd | feedback_epoch | feedback_xmin |
 oldest_read_view_lsn | visibility_lag_in_msec
```

```
-----------------------------------------+-------------------------------------
+-----------+------------+-----------------+---------------+--------------
+---------------------+-----------------------
apg-global-db-rpo-mammothrw-elephantro-1-n1 | MASTER_SESSION_ID
 | us-east-1  | 93763985102 |                  |               |              |
                    |
apg-global-db-rpo-mammothrw-elephantro-1-n2 | f38430cf-6576-479a-b296-dc06b1b1964a
 | us-east-1  | 93763985099 |      93763985102 |             2 |    3315479243 |
        93763985095 |                     10
apg-global-db-rpo-elephantro-mammothrw-n1   | 0d9f1d98-04ad-4aa4-8fdd-e08674cbbbfe
 | us-west-2  | 93763985095 |      93763985099 |             2 |    3315479243 |
        93763985089 |                   1017
(3 rows)
```

L'output include una riga per ogni istanza DB del database globale contenente le colonne seguenti:

- server_id – Identificatore del server per l'istanza DB.
- session_id – Un identificatore univoco per la sessione corrente.
- aws_region: la Regione AWS in cui si trova questa istanza database. Per l'elenco delle tabelle Regioni AWS per motore, consulta Disponibilità nelle regioni.
- durable_lsn – LSN è stato reso durevole nello storage.
- highest_lsn_rcvd – L'LSN più alto che l'istanza database ha ricevuto dall'istanza database di scrittura.
- feedback_epoch – L'epoca utilizzata dall'istanza DB quando genera informazioni di hot standby.

  Standby a caldo è quando un'istanza database può connettersi ed eseguire query mentre il server è in modalità di ripristino o standby. Il feedback hot standby consiste in informazioni sull'istanza DB quando è in hot standby. Per ulteriori informazioni, consulta la documentazione di PostgreSQL su Hot Standby.
- feedback_xmin – L'ID della transazione attiva minimo (meno recente) utilizzato dall'istanza DB.
- oldest_read_view_lsn – L'LSN più vecchio utilizzato dall'istanza DB per leggere dallo storage.
- visibility_lag_in_msec – Quanto questa istanza DB è in ritardo rispetto all'istanza DB di scrittura.

Per vedere come questi valori cambiano nel tempo, considerare il seguente blocco di transazioni in cui un inserimento di tabella richiede un'ora:

```
psql> BEGIN;
psql> INSERT INTO table1 SELECT Large_Data_That_Takes_1_Hr_To_Insert;
psql> COMMIT;
```

In alcuni casi, potrebbe esserci una disconnessione di rete tra il cluster DB primario e il cluster DB secondario dopo l'istruzione `BEGIN`. In tal caso, il valore `durability_lag_in_msec` del cluster DB secondario inizia ad aumentare. Alla fine dell'istruzione `INSERT`, il valore `durability_lag_in_msec` è 1 ora. Tuttavia, il valore `rpo_lag_in_msec` è 0 perché tutti i dati utente impegnati tra il cluster DB primario e il cluster DB secondario sono ancora gli stessi. Non appena l'istruzione `COMMIT` viene completata, il valore `rpo_lag_in_msec` aumenta.

Utilizzo Blue/Green delle distribuzioni per Amazon Aurora Global Database

Amazon RDS Blue/Green Deployments offre la possibilità di testare le modifiche al database in modo sicuro. Per il tuo database globale, le blue/green implementazioni ti consentono di eseguire aggiornamenti e operazioni di manutenzione con tempi di inattività minimi. È possibile creare un ambiente di staging completamente gestito (verde) che rispecchi l'intero ambiente di produzione (blu), incluso il cluster principale e tutte le regioni secondarie associate in più regioni. AWS L'ambiente di staging riflette la configurazione di produzione e consente di testare in modo affidabile le modifiche prima di passare al nuovo ambiente. Durante tutto il processo, le blue/green implementazioni mantengono sincronizzati gli ambienti di staging e di produzione. Quando sei pronto a trasformare l'ambiente di staging nel nuovo ambiente di produzione, esegui un passaggio. blue/green Questa operazione trasferisce le aree primarie e tutte le aree secondarie all'ambiente verde, con tempi di inattività in genere inferiori a un minuto. Il servizio rinomina automaticamente i cluster, le istanze e gli endpoint in base all'ambiente di produzione originale, consentendo alle applicazioni di accedere al nuovo ambiente di produzione senza richiedere modifiche alla configurazione e riducendo al minimo il sovraccarico operativo.

Argomenti

- Vantaggi dell'utilizzo delle Blue/Green implementazioni con Aurora Global Database
- Come funzionano Blue/Green le implementazioni con Aurora Global Database

Vantaggi dell'utilizzo delle Blue/Green implementazioni con Aurora Global Database

- Esegui aggiornamenti di versioni principali, versioni secondarie e di sistema, tra cui patch di database e aggiornamenti del sistema operativo su Aurora Global Databases, mantenendo al contempo tempi di inattività minimi.
- Crea un ambiente di staging (verde) pronto per la produzione nelle aree primarie e secondarie del database globale per testare e implementare nuove funzionalità del database.
- Eseguite il passaggio in modo sicuro utilizzando i guardrail di commutazione integrati con tempi di inattività in genere inferiori a un minuto, a seconda del carico di lavoro.
- Mantiene le funzionalità di disaster recovery durante il processo di switchover, permettendo il failover di Global Database durante lo Blue/Green switchover. Blue/Green
- Il traffico verrà indirizzato al nuovo ambiente di produzione senza richiedere alcuna modifica all'applicazione.

Come funzionano Blue/Green le implementazioni con Aurora Global Database

Per i dettagli su come creare, visualizzare, cambiare ed eliminare una Blue/Green distribuzione, consulta. Utilizzo di Amazon Blue/Green Aurora Deployments per gli aggiornamenti del database È possibile utilizzarlo per aggiornamenti di versioni principali o secondarie, miglioramenti delle prestazioni del database tramite aggiornamenti dei parametri e adozione di nuove funzionalità del database, con tempi di inattività minimi.

Di seguito è riportata una rappresentazione dell'aspetto di una blue/green distribuzione per Aurora Global Database con un'area secondaria prima e dopo lo blue/green switchover.


Primary Region
VPC-A
Secondary Region
VPC-B
Users/Applications
Blue Environment
(Current Production)
Old version
Primary Cluster
Aurora Global Database
Asynchronous
Replication
Secondary Cluster
Blue/Green
Deployment
Logical
Replication
(managed)
Green Environment
(Future Production)
New version
Primary Cluster
Aurora Global Database
Asynchronous
Replication
Secondary Cluster
Blue/Green
deployment
switchover
Primary Region
VPC-A
Secondary Region
VPC-B
Old Blue Environment
(Old Production)
Old version
Primary Cluster
Aurora Global Database
Asynchronous
Replication
Secondary Cluster
Blue/Green
Deployment
Blue Environment
(Current Production)
New version
Primary Cluster
Aurora Global Database
Asynchronous
Replication
Secondary Cluster

È possibile creare una blue/green distribuzione dalla regione principale del database globale. Seleziona le configurazioni del motore, ad esempio la versione principale o secondaria del motore, il gruppo di parametri DB e il gruppo di parametri del cluster DB per l'ambiente verde. Amazon RDS copia la topologia dell'ambiente blu per l'ambiente verde. Una rappresentazione visiva Console di gestione AWS è quella mostrata di seguito.



> **Note**
>
> Il failover globale è supportato durante uno blue/green switchover, ma lo switchover globale non è supportato durante uno switchover. blue/green

Quando si avvia un failover globale durante uno blue/green switchover RDS, la regione di destinazione torna automaticamente all'ambiente blu o torna all'ambiente verde prima che si verifichi il failover globale.

Per informazioni sulla creazione, la visualizzazione, il cambio e l'eliminazione delle distribuzioni, vedere. blue/green Utilizzo di Amazon Blue/Green Aurora Deployments per gli aggiornamenti del database Segui lo stesso flusso di lavoro per Global Databases, con istruzioni specifiche riportate nei passaggi pertinenti.

Utilizzo di database globali Amazon Aurora con altri servizi AWS

È possibile utilizzare i database globali Aurora con altri servizi AWS, ad esempio Amazon S3 e AWS Lambda. Ciò richiede che tutti i cluster di database Aurora nel database globale abbiano gli stessi privilegi, funzioni esterne e così via nelle rispettive Regioni AWS. Poiché un cluster di database Aurora secondario di sola lettura in un database globale Aurora può essere promosso al ruolo primario, è consigliabile impostare i privilegi di scrittura in anticipo su tutti i cluster di database Aurora per tutti i servizi che si intende utilizzare con il database globale Aurora.

Le procedure seguenti riepilogano le azioni da intraprendere per ciascun Servizio AWS.

Come richiamare le funzioni AWS Lambda da un database globale Aurora

1. Per tutti i cluster Aurora che costituiscono il database globale l'Aurora, esegui le procedure in Chiamare una funzione Lambda da un cluster DB Amazon Aurora MySQL.
2. Per ogni cluster nel database globale Aurora, imposta l'Amazon Resource Name (nome della risorsa Amazon (ARN)) del nuovo ruolo IAM (IAM).
3. Per consentire agli utenti di database in un database globale Aurora di invocare funzioni Lambda, associa il ruolo creato in Creazione di un ruolo IAM per consentire ad Amazon Aurora di accedere ai servizi AWS a ogni cluster nel database globale Aurora.
4. Configura ogni cluster nel database globale Aurora per consentire le connessioni in uscita a Lambda. Per istruzioni, consultare Abilitazione delle comunicazioni di rete da Amazon Aurora ad altri servizi AWS.

Per caricare i dati da Amazon S3

1. Per tutti i cluster Aurora che costituiscono il database globale l'Aurora, esegui le procedure in Caricamento dei dati in un cluster DB Amazon Aurora MySQL da file di testo in un bucket Amazon S3.
2. Per ogni cluster Aurora nel database globale, imposta il parametro del cluster database `aurora_load_from_s3_role` o `aws_default_s3_role` sull'ARN del nuovo ruolo IAM. Se

un ruolo IAM non è specificato per `aurora_load_from_s3_role`, Aurora utilizza il ruolo IAM specificato in `aws_default_s3_role`.

3. Per consentire agli utenti di database in un Aurora Global Database di accedere ad S3, associa il ruolo creato in Creazione di un ruolo IAM per consentire ad Amazon Aurora di accedere ai servizi AWS a ogni cluster Aurora nel database globale.
4. Configura ogni cluster nel database globale Aurora per consentire le connessioni in uscita a S3. Per istruzioni, consultare Abilitazione delle comunicazioni di rete da Amazon Aurora ad altri servizi AWS.

Per salvare i dati interrogati in Amazon S3

1. Per tutti i cluster Aurora che costituiscono il database globale Aurora, esegui le procedure in Salvataggio dei dati da un cluster DB Amazon Aurora MySQL nei file di testo in un bucket Amazon S3 o Esportazione di dati da del cluster di database Aurora PostgreSQLRDS per PostgreSQL a Amazon S3.
2. Per ogni cluster Aurora nel database globale, imposta il parametro del cluster database `aurora_select_into_s3_role` o `aws_default_s3_role` sull'ARN del nuovo ruolo IAM. Se un ruolo IAM non è specificato per `aurora_select_into_s3_role`, Aurora utilizza il ruolo IAM specificato in `aws_default_s3_role`.
3. Per consentire agli utenti di database in un Aurora Global Database di accedere ad S3, associa il ruolo creato in Creazione di un ruolo IAM per consentire ad Amazon Aurora di accedere ai servizi AWS a ogni cluster Aurora nel database globale.
4. Configura ogni cluster nel database globale Aurora per consentire le connessioni in uscita a S3. Per istruzioni, consultare Abilitazione delle comunicazioni di rete da Amazon Aurora ad altri servizi AWS.

Aggiornamento di un database globale Amazon Aurora

L'aggiornamento di un database globale Aurora segue le stesse procedure dell'aggiornamento dei cluster di database Aurora. Tuttavia, di seguito sono riportate alcune importanti differenze di cui prendere nota prima di iniziare il processo.

Ti consigliamo di eseguire l'aggiornamento dei cluster di database primario e secondario alla stessa versione. È possibile eseguire un failover gestito del database tra regioni su un database globale Aurora solo se i cluster di database primario e secondario hanno la stessa versione principale e secondaria e lo stesso livello di patch del motore. Tuttavia, i livelli di patch possono essere diversi, a

seconda della versione secondaria del motore. Per ulteriori informazioni, consulta Compatibilità del livello di patch per switchover e failover gestiti tra regioni.

Aggiornamenti di una versione principale

Quando esegui un aggiornamento della versione principale di un database globale Amazon Aurora, aggiorni il cluster di database globale invece dei singoli cluster in esso contenuti.

Per informazioni su come aggiornare un database globale Aurora PostgreSQL a una versione principale superiore, consulta Principali aggiornamenti per database globali.

> **Note**
>
> Con un database globale Aurora basato su Aurora PostgreSQL, non è possibile eseguire un aggiornamento della versione principale del motore Aurora DB se la caratteristica Recovery point objective (RPO) (Obiettivo del punto di ripristino (RPO)) è attivata. Per ulteriori informazioni sulla caratteristica RPO, consulta Gestione degli RPO per database globali basati su Aurora PostgreSQL–.

Per informazioni su come aggiornare un database globale Aurora MySQL a una versione principale superiore, consulta Principali aggiornamenti in loco per database globali.

> **Note**
>
> Con un database globale Aurora basato su Aurora MySQL, puoi eseguire un aggiornamento in loco da Aurora MySQL versione 2 alla versione 3 solo se il parametro `lower_case_table_names` è attivato come predefinito e il database globale viene riavviato.
> Per eseguire un aggiornamento della versione principale ad Aurora MySQL versione 3 usando `lower_case_table_names`, utilizza la seguente procedura:
>
> 1. Rimuovi tutte le regioni secondarie dal cluster globale. Seguire la procedura riportata in Rimozione di un cluster da un database globale Amazon Aurora.
> 2. Esegui l'aggiornamento della versione del motore della regione principale ad Aurora MySQL versione 3. Seguire la procedura riportata in Come eseguire un aggiornamento in loco.
> 3. Aggiungi le regioni secondarie al cluster globale. Seguire la procedura riportata in Aggiunta di una Regione AWS a un database globale Amazon Aurora.

Puoi anche utilizzare il metodo di ripristino snapshot. Per ulteriori informazioni, consulta Ripristino da uno snapshot cluster database.

Aggiornamenti della versione secondaria

Per un aggiornamento secondario su un database globale Aurora, è necessario aggiornare tutti i cluster secondari prima di aggiornare il cluster primario.

Per informazioni su come aggiornare un database globale Aurora PostgreSQL a una versione secondaria superiore, consulta Come eseguire aggiornamenti della versione secondaria e applicare patch. Per informazioni su come aggiornare un database globale Aurora MySQL a una versione secondaria superiore, consulta Aggiornamento di Aurora MySQL modificando la versione del motore.

Prima di eseguire l'aggiornamento, esamina le seguenti considerazioni:

- L'aggiornamento della versione secondaria di un cluster secondario non influisce in alcun modo sulla disponibilità o sull'utilizzo del cluster primario.
- Un cluster secondario deve disporre almeno di un'istanza database per eseguire un aggiornamento a una versione secondaria.
- Se aggiorni un database globale Aurora MySQL alla versione 2.11.*, è necessario aggiornare i tuoi cluster di database primari e secondari alla stessa identica versione, incluso il livello di patch.
- Se aggiorni un database globale Aurora PostgreSQL, è necessario aggiornare i tuoi cluster di database primari e secondari alla stessa identica versione, incluso il livello di patch. Per aggiornare il livello di patch, applica tutte le azioni di manutenzione in sospeso sul cluster secondario.
- Per supportare switchover o failover gestiti tra Regioni, potrebbe essere necessario eseguire l'aggiornamento dei cluster di database primario e secondari alla stessa versione, incluso il livello di patch. Questo requisito si applica ad Aurora MySQL e ad alcune versioni di Aurora PostgreSQL. Per un elenco di versioni che consentono lo switchover e il failover tra cluster che eseguono livelli di patch diversi, consulta Compatibilità del livello di patch per switchover e failover gestiti tra regioni.

Compatibilità del livello di patch per switchover e failover gestiti tra regioni

Se il database globale Aurora sta operando una delle seguenti versioni secondarie del motore, puoi eseguire switchover o failover gestiti tra Regioni anche se i livelli di patch dei cluster di database

primario e secondari non corrispondono. Per le versioni secondarie del motore precedenti a quelle presenti in questo elenco, i cluster di database primario e secondari devono eseguire la stessa versione principale e secondaria e allo stesso livello di patch per eseguire switchover o failover gestiti tra Regioni. Assicurati di esaminare le informazioni sulla versione e le note nella tabella seguente quando pianifichi gli aggiornamenti per il cluster primario, i cluster secondari o entrambi.

> **Note**
>
> Per i failover manuali tra regioni, è possibile eseguire il processo di failover purché il cluster database secondario di destinazione esegua la stessa versione principale e secondaria del motore del cluster database primario. In questo caso, i livelli di patch non devono necessariamente corrispondere.
> Se le versioni del motore richiedono identici livelli di patch, puoi eseguire il failover manualmente seguendo i passaggi indicati in Esecuzione di failover manuali per i Database globali Aurora.

Motore del database	Versioni secondarie del motore	Note
Aurora MySQL	Nessuna versione secondaria	Nessuna delle versioni secondari e di Aurora MySQL consente di eseguire switchover o failover gestiti tra Regioni con livelli differenti di patch tra cluster di database primario e secondari.
Aurora PostgreSQL	• Versione 15 o versioni principali successive • Versione 14.5 o versioni secondari e successive • Versione 13.8 o versioni secondari e successive • Versione 12.12 o versioni secondarie successive	Con le versioni del motore elencate nella colonna precedente, è possibile eseguire switchover o failover gestiti tra Regioni da un cluster di database primario con un livello di patch a un cluster di database secondario con un livello di patch diverso. Con le versioni secondarie precedent i a queste, è possibile eseguire

Motore del database	Versioni secondarie del motore	Note
	• Versione 11.17 o versioni secondarie successive	switchover o failover gestiti tra Regioni solo se i livelli di patch dei cluster di database primari e secondari corrispondono. **Warning** Quando si aggiorna un cluster nel database globale a una delle seguenti versioni di patch, non sarà possibile eseguire switchover o failover tra Regioni finché tutti i cluster di database globale non eseguiranno una di queste versioni di patch o una più recente. • Versioni di patch 16.1.6, 16.2.4, 16.3.2 e 16.4.2 • Versioni di patch 15.3.8, 15.4.9, 15.5.6, 15.6.4, 15.7.2 e 15.8.2 • Versioni di patch 14.8.8, 14.9.9, 14.10.6, 14.11.4, 14.12.2 e 14.13.2

Server proxy per Amazon RDS per Aurora

Con Server proxy per Amazon RDS, puoi consentire alle tue applicazioni di eseguire il pool e condividere connessioni di database per migliorare la loro capacità di dimensionamento. RDS Proxy rende le applicazioni più resilienti agli errori del database connettendosi automaticamente a un'istanza database di standby, mantenendo al contempo le connessioni delle applicazioni. Utilizzando RDS Proxy, è possibile applicare l'autenticazione AWS Identity and Access Management (IAM) per i client che si connettono al proxy e il proxy può connettersi ai database utilizzando l'autenticazione del database IAM o le credenziali archiviate in. Gestione dei segreti AWS



Con Server proxy per RDS, puoi gestire picchi imprevedibili nel traffico del database che altrimenti potrebbero causare problemi dovuti all'eccessiva sottoscrizione delle connessioni o alla creazione di nuove connessioni a velocità elevata. Server proxy per RDS stabilisce un pool di connessioni al database e riutilizza le connessioni di questo pool. Questo approccio evita il sovraccarico della memoria e della CPU per aprire ogni volta una nuova connessione al database. Per proteggere un database da sottoscrizione eccessiva, è possibile controllare il numero di connessioni al database create.

Server proxy per RDS crea sequenze o limita le connessioni delle applicazioni che non possono essere servite immediatamente dal pool di connessioni. Sebbene le latenze possano aumentare, l'applicazione può continuare a dimensionare senza errori improvvisi o senza sovraccaricare il database. Se le richieste di connessione superano i limiti specificati, RDS Proxy rifiuta le connessioni dell'applicazione (genera il carico). Allo stesso tempo, mantiene prestazioni prevedibili per il carico che può essere servito da RDS con la capacità disponibile.



Puoi ridurre il sovraccarico per elaborare le credenziali e stabilire una connessione sicura per ogni nuova connessione. RDS Proxy può gestire alcune di queste operazioni per conto del database.

RDS Proxy è compatibile con le versioni di motore supportate. È possibile abilitare RDS Proxy per la maggior parte delle applicazioni senza modifiche al codice. Per un elenco delle versioni di motore supportate, consulta Regioni e motori di database Aurora supportati per Server proxy per Amazon RDS.

Argomenti

- Disponibilità di regioni e versioni
- Quote e limiti per RDS Proxy
- Pianificazione sull'utilizzo di RDS Proxy
- Concetti e terminologia RDS Proxy
- Nozioni di base su RDS Proxy
- Gestire un RDS Proxy
- Utilizzo degli endpoint Amazon RDS Proxy
- Monitoraggio dei parametri RDS Proxy con Amazon CloudWatch
- Utilizzo degli eventi RDS Proxy
- Esempi della riga di comando per RDS Proxy
- Risoluzione dei problemi per RDS Proxy
- Utilizzo di RDS Proxy con AWS CloudFormation
- Utilizzo del Server proxy per RDS con i database globali Aurora

Disponibilità di regioni e versioni

Per informazioni sul supporto della versione del motore di database e sulla disponibilità di RDS Proxy in un determinato periodoRegione AWS, vedere. Regioni e motori di database Aurora supportati per Server proxy per Amazon RDS

Quote e limiti per RDS Proxy

A RDS Proxy si applicano i seguenti limiti:

- Ogni Account AWS ID è limitato a 20 proxy. Se la tua applicazione richiede più proxy, richiedi un aumento tramite la pagina Service Quotas all'interno di Console di gestione AWS. Nella pagina Service Quotas, seleziona Amazon Relational Database Service (Amazon RDS) e individua i proxy per richiedere un aumento della quota. AWS può aumentare automaticamente la tua quota o è in attesa di revisione della tua richiesta entro Supporto.
- Ogni proxy può avere fino a 200 segreti di Secrets Manager associati, limitando così le connessioni a un massimo di 200 account utente diversi quando si utilizzano segreti.
- Ciascun proxy ha un endpoint predefinito. Puoi anche aggiungere fino a 20 endpoint proxy per ogni proxy. Puoi creare, visualizzare, modificare ed eliminare questi endpoint.
- In un cluster Aurora, tutte le connessioni che utilizzano l'endpoint proxy predefinito vengono gestite dall'istanza scrittore di Aurora. Per eseguire il bilanciamento del carico per carichi di lavoro a uso intensivo di lettura, puoi creare un endpoint di sola lettura per un proxy. Tale endpoint passa le connessioni all'endpoint di lettura del cluster. In questo modo, le connessioni proxy possono sfruttare i vantaggi della scalabilità di lettura di Aurora. Per ulteriori informazioni, consulta Panoramica degli endpoint proxy.
- È possibile utilizzare Server proxy per RDS con cluster Aurora Serverless v2 ma non con cluster Aurora Serverless v1.
- RDS Proxy deve essere nello stesso virtual private cloud (VPC) del database. Sebbene il database non sia accessibile pubblicamente, il proxy può esserlo. Ad esempio, se stai eseguendo la prototipazione del database su un host locale, non puoi connetterti al proxy se non hai configurato i requisiti di rete necessari per consentire la connessione al proxy. Ciò avviene perché il tuo host locale è esterno al VPC del proxy.

> **Note**
>
> Per i cluster database Aurora, puoi attivare l'accesso tra VPC. Per fare ciò, creare un endpoint aggiuntivo per un proxy e specificare un VPC, sottoreti e gruppi di sicurezza diversi con tale endpoint. Per ulteriori informazioni, consulta Accesso ai database Aurora RDS VPCs.

- Non è possibile utilizzare RDS Proxy con un VPC con tenancy impostato a `dedicated`.
- Non puoi utilizzare RDS Proxy in un VPC con controlli `Enforce Mode` di crittografia abilitati.
- Per i tipi di rete IPv6 endpoint, configura il VPC e le sottoreti in modo che supportino solo. IPv6 Per entrambi i IPv4 tipi di rete con connessione di IPv6 destinazione, configura il VPC e le sottoreti per supportare la modalità dual-stack.
- Se si utilizza RDS Proxy con un cluster Aurora con autenticazione IAM abilitata, il proxy può connettersi al database utilizzando l'autenticazione IAM o le credenziali archiviate in Secrets Manager. I client che si connettono al proxy devono autenticarsi utilizzando le credenziali IAM. Per istruzioni dettagliate sulla configurazione, consulta e Impostazione delle credenziali del database per RDS Proxy Configurazione dell'autenticazione IAM per Server proxy per RDS
- Non puoi utilizzare RDS Proxy con DNS personalizzati quando utilizzi la convalida del nome host SSL.
- Ogni proxy può essere associato a un singolo cluster di database di destinazione. Tuttavia, è possibile associare più proxy allo stesso cluster di database.
- Qualsiasi istruzione con una dimensione del testo maggiore di 16 KB fa sì che il proxy effettui il pinning della sessione nella connessione corrente.
- Per alcune regioni sono presenti restrizioni relative alla zona di disponibilità (AZ) da considerare durante la creazione del proxy. La Regione Stati Uniti orientali (Virginia settentrionale) non supporta Server proxy per RDS nella zona di disponibilità `use1-az3`. La Regione Stati Uniti occidentali (California settentrionale) non supporta Server proxy per RDS nella zona di disponibilità `usw1-az2`. Quando selezioni le sottoreti durante la creazione del proxy, assicurati di non scegliere sottoreti nelle zone di disponibilità sopra menzionate.
- Attualmente, Server proxy per RDS non supporta nessuna delle chiavi di contesto delle condizioni globali.

  Per ulteriori informazioni sulle chiavi di contesto delle condizioni globali, consulta Chiavi di contesto delle condizioni globali AWS nella Guida per l'utente di IAM.

- Non è possibile utilizzare Server proxy per RDS con RDS Custom per SQL Server.
- Per riflettere qualsiasi modifica del gruppo di parametri del database apportata al proxy, è necessario riavviare l'istanza anche se si sceglie di applicare immediatamente le modifiche. Per i parametri a livello di cluster, è necessario un riavvio a livello di cluster.
- Il proxy crea automaticamente l'utente database `rdsproxyadmin` quando si registra una destinazione proxy. Si tratta di un utente protetto essenziale per la funzionalità del proxy. È necessario evitare di manomettere l'utente `rdsproxyadmin` a qualsiasi titolo. L'eliminazione o la modifica dell'utente `rdsproxyadmin` o delle relative autorizzazioni può comportare la completa indisponibilità del proxy per l'applicazione.

Per le altre limitazioni di ciascun motore di database, consulta le sezioni riportate di seguito:

- Limitazioni aggiuntive per Aurora MySQL
- Limitazioni aggiuntive per Aurora PostgreSQL

Limitazioni aggiuntive per Aurora MySQL

Le seguenti limitazioni aggiuntive si applicano a Server proxy per RDS con database Aurora MySQL:

- Il supporto del proxy RDS per l'autenticazione `caching_sha2_password` richiede una connessione sicura (TLS).
- È noto che il supporto Server proxy per RDS per `caching_sha2_password` presenta problemi di compatibilità con alcune versioni del driver go-sql.
- Quando si utilizza il driver MySQL 8.4 C, l'API `mysql_stmt_bind_named_param` potrebbe formare pacchetti non validi se il conteggio dei parametri supera il conteggio dei segnaposto in un'istruzione preparata. Ciò si traduce in risposte errate. Per ulteriori informazioni, consulta il report sui bug di MySQL.
- Attualmente, tutti i proxy sono in ascolto di MySQL sulla porta 3306. I proxy si connettono ancora al database utilizzando la porta specificata nelle impostazioni del database.
- Non è possibile utilizzare RDS Proxy con database MySQL autogestiti in istanze. EC2
- Non è possibile utilizzare RDS Proxy con un'istanza database di RDS per MySQL con il parametro `read_only` nel suo gruppo di parametri database impostato su `1`.
- RDS Proxy non supporta la modalità compressa MySQL. Ad esempio, non supporta la compressione utilizzata dalle opzioni `--compress` o `-C` del comando `mysql`.

- Le connessioni al database che elaborano un comando `GET DIAGNOSTIC` potrebbero restituire informazioni imprecise quando Server proxy per RDS riutilizza la stessa connessione al database per eseguire un'altra query. Questo può accadere quando Server proxy per RDS crea multiplex delle connessioni al database.
- Alcune istruzioni e funzioni SQL, ad esempio `SET LOCAL`, possono modificare lo stato della connessione senza causare l'aggiunta. Per il comportamento del pinning più aggiornato, consulta Evitare di effettuare il pinning di un Server proxy per RDS.
- L'utilizzo della funzione `ROW_COUNT()` in una query con più istruzioni non è supportato.
- Server proxy per RDS non supporta le applicazioni client che non sono in grado di gestire più messaggi di risposta in un unico record TLS.
- Server proxy per RDS non supporta le doppie password MySQL.
- Il proxy RDS potrebbe non funzionare come previsto quando si configura il parametro `init_connect` nel gruppo di parametri RDS DB per impostare le variabili dello stato della sessione. Imposta invece la query di inizializzazione per il proxy in modo che esegua le istruzioni di inizializzazione della sessione quando utilizzi il proxy per connetterti al database.

> **Important**
>
> Per i proxy associati ai database MySQL, non impostare il parametro di configurazione `sql_auto_is_null` su `true` o un valore diverso da zero nella query di inizializzazione. Ciò potrebbe causare un comportamento non corretto dell'applicazione.

Limitazioni aggiuntive per Aurora PostgreSQL

Le seguenti limitazioni aggiuntive si applicano a Server proxy per RDS con database Aurora PostgreSQL:

- RDS Proxy non supporta i filtri di pinning della sessione per PostgreSQL.
- Attualmente, tutti i proxy sono in ascolto di PostgreSQL sulla porta 5432.
- Per PostgreSQL, RDS Proxy attualmente non supporta l'annullamento di una query da un client tramite l'emissione di una `CancelRequest`. Questo è il caso, ad esempio, di quando si annulla una query a esecuzione prolungata in una sessione psql interattiva utilizzando Ctrl+C.
- I risultati della funzione PostgreSQL lastval non sono sempre accurati. Per risolvere il problema, utilizzare l'istruzione INSERT con la clausola `RETURNING`.

- Server proxy per RDS attualmente non supporta la modalità di replica in streaming.
- Il `postgres` database predefinito deve esistere nell'istanza RDS for PostgreSQL affinché RDS Proxy funzioni. Non eliminare questo database anche se l'applicazione utilizza database diversi.
- Se si utilizza `ALTER ROLE` per modificare il ruolo dell'utente con `SET ROLE`, le connessioni successive dell'utente al proxy potrebbero non utilizzare questa impostazione del ruolo, se tali connessioni presentano un blocco. Per evitare che ciò si verifichi, quando si utilizza il proxy, è necessario utilizzarlo `SET ROLE` nella query di inizializzazione del proxy. Per ulteriori informazioni, consulta Query di inizializzazione in Creazione di un proxy per Amazon Aurora.

> **Important**
>
> Per i proxy esistenti con database PostgreSQL, se si modifica l'autenticazione del database in modo da utilizzare solo `SCRAM`, il proxy diventa non disponibile per un massimo di 60 secondi. Per evitare il problema, procedi in uno dei seguenti modi:
>
> - Assicurati che il database consenta entrambe le autenticazioni `SCRAM` e `MD5`.
> - Per utilizzare solo l'autenticazione `SCRAM`, crea un nuovo proxy, esegui la migrazione del traffico dell'applicazione sul nuovo proxy, quindi elimina il proxy precedentemente associato al database.

Pianificazione sull'utilizzo di RDS Proxy

Puoi determinare quali tra le istanze DB, i cluster e le applicazioni possono trarre maggior vantaggio dall'utilizzo di RDS Proxy. Per fare ciò, considera questi fattori:

- Qualsiasi cluster database che genera errori del tipo "Troppe connessioni" è un buon candidato per l'associazione con un proxy. Questo è spesso caratterizzato da un valore elevato della `ConnectionAttempts` CloudWatch metrica. Il proxy consente alle applicazioni di aprire molte connessioni client, mentre il proxy gestisce un numero minore di connessioni di lunga durata al cluster database.
- Per DB, i cluster che utilizzano classi di AWS istanze più piccole, come T2 o T3, l'uso di un proxy può aiutare a evitare condizioni. out-of-memory Può anche contribuire a ridurre il sovraccarico della CPU per stabilire connessioni. Queste condizioni possono verificarsi quando vi è un numero elevato di connessioni.

- Puoi monitorare determinati CloudWatch parametri di Amazon per determinare se un cluster di DB si avvicina a determinati tipi di limite. Questi limiti riguardano il numero di connessioni e la memoria associata alla gestione delle connessioni. Puoi anche monitorare determinati CloudWatch parametri per determinare se un cluster di DB gestisce molte connessioni di breve durata. L'apertura e la chiusura di tali connessioni possono determinare un sovraccarico delle prestazioni sul database. Per informazioni sui parametri da monitorare, consulta Monitoraggio dei parametri RDS Proxy con Amazon CloudWatch.
- AWS Lambda le funzioni possono anche essere buone candidate per l'utilizzo di un proxy. Queste funzioni effettuano frequenti connessioni del database brevi che beneficiano del pool di connessioni offerto da RDS Proxy. Puoi sfruttare qualsiasi autenticazione IAM che già possiedi per le funzioni Lambda, invece di gestire le credenziali del database nel codice dell'applicazione Lambda.
- Le applicazioni che in genere aprono e chiudono un numero elevato di connessioni al database e non dispongono di meccanismi di pooling delle connessioni incorporati sono ottimi candidati per l'utilizzo di un proxy.
- Le applicazioni che mantengono un numero elevato di connessioni aperte per lunghi periodi sono in genere buoni candidati per l'utilizzo con un proxy. Applicazioni in ambiti come software as a service (SaaS) o e-commerce spesso riducono al minimo la latenza per le richieste del database lasciando aperte le connessioni. Con Server proxy per RDS, un’applicazione può mantenere aperte un maggior numero di connessioni rispetto a quando si connette direttamente al cluster di database.
- Potresti non aver adottato l'autenticazione IAM e Secrets Manager a causa della complessità della configurazione di tale autenticazione per tutti i cluster di DB. Il proxy può applicare le policy di autenticazione per le connessioni client per applicazioni particolari. Puoi sfruttare qualsiasi autenticazione IAM che già possiedi per le funzioni Lambda, invece di gestire le credenziali del database nel codice dell'applicazione Lambda.
- Server proxy per RDS può contribuire a rendere le applicazioni più resilienti e trasparenti agli errori del database. Server proxy per RDS ignora le cache del sistema dei nomi di dominio (DNS) per ridurre i tempi di failover fino al 66% per i database Multi-AZ Aurora. Server proxy per RDS inoltre instrada automaticamente il traffico a una nuova istanza database, preservando al contempo le connessioni dell'applicazione. In tal modo i failover sono più trasparenti per le applicazioni.

Concetti e terminologia RDS Proxy

Puoi semplificare la gestione delle connessioni per i cluster di database Amazon Aurora utilizzando Server proxy per RDS.

RDS Proxy gestisce il traffico di rete tra l'applicazione client e il database. Lo fa in modo attivo prima comprendendo il protocollo del database. Quindi regola il suo comportamento in base alle operazioni SQL dell'applicazione e ai set di risultati dal database.

RDS Proxy riduce il sovraccarico di memoria e CPU per la gestione delle connessioni nel database. Il database richiede meno memoria e risorse della CPU quando le applicazioni aprono molte connessioni simultanee. Inoltre, non richiede alcuna logica nelle applicazioni al fine di chiudere e riaprire le connessioni che rimangono inattive per un lungo periodo di tempo. Allo stesso modo, richiede meno operazioni logiche dell'applicazione per ristabilire le connessioni in caso di problemi di database.

L'infrastruttura per RDS Proxy è altamente disponibile e distribuita su più zone di disponibilità ()AZs. Il calcolo, la memoria e l'archiviazione per server proxy per RDS sono indipendenti dal cluster di un cluster di database Aurora. Questa separazione consente di ridurre il sovraccarico sui server di database, in modo da poter dedicare le risorse al servizio dei carichi di lavoro del database. Le risorse di calcolo RDS Proxy sono serverless e vengono scalate automaticamente in base al carico di lavoro del database.

Argomenti

- Panoramica dei concetti RDS Proxy
- Pooling di connessioni
- Sicurezza di RDS Proxy
- Failover
- Transazioni

Panoramica dei concetti RDS Proxy

RDS Proxy gestisce l'infrastruttura per eseguire il pool di connessioni e le altre funzionalità descritte nelle sezioni seguenti. I proxy rappresentati nella console RDS vengono visualizzati nella pagina Proxy.

Ogni proxy gestisce le connessioni a un singolo cluster di database Aurora. Il proxy determina l'istanza di scrittura corrente per i cluster di provisioning Aurora.

Le connessioni che un proxy mantiene aperte e disponibili per le applicazioni di database da utilizzare formano il pool di connessioni.

Per impostazione predefinita, RDS Proxy può riutilizzare una connessione dopo ogni transazione nella sessione. Questo riutilizzo a livello di transazione viene definito multiplexing. Quando RDS Proxy rimuove temporaneamente una connessione dal pool di connessioni per riutilizzarla, tale operazione viene chiamata prestito della connessione. Quando è sicuro farlo, RDS Proxy restituisce tale connessione al pool di connessioni.

In alcuni casi, per RDS Proxy non è possibile avere la certezza di riutilizzare una connessione al database al di fuori della sessione corrente. In questi casi, mantiene la sessione sulla stessa connessione fino al termine della sessione. Questo comportamento di fallback viene definito pinning.

Un proxy ha un endpoint predefinito. Ti connetti a questo endpoint quando lavori con un cluster database Amazon Aurora Gli endpoint per finalità speciali per un cluster Aurora rimangono disponibili per l'utilizzo. Per i cluster Aurora DB e i cluster aggiuntivi e di sola lettura. read/write Per ulteriori informazioni, consulta Panoramica degli endpoint proxy.

Ad esempio, è comunque possibile connettersi all'endpoint del cluster per connessioni senza pool di connessioni. read/write Puoi comunque connetterti all'endpoint di lettura per le connessioni di sola lettura con bilanciamento del carico. Puoi comunque connetterti agli endpoint dell'istanza per la diagnosi e la risoluzione dei problemi di istanze database specifiche all'interno di un cluster. Se utilizzi altri AWS servizi, ad esempio AWS Lambda per connetterti ai database RDS, modifica le relative impostazioni di connessione per utilizzare l'endpoint proxy. Ad esempio, se specifichi l'endpoint proxy per consentire alle funzioni Lambda di accedere al database sfruttando al contempo le funzionalità RDS Proxy.

Ogni proxy contiene un gruppo di destinazione. Questo gruppo di destinazione rappresenta il cluster di database Aurora a cui il proxy può connettersi. Per un cluster Aurora, per impostazione predefinita, il gruppo di destinazione è associato a tutte le istanze DB in tale cluster. In questo modo, il proxy può connettersi a qualsiasi istanza database Aurora venga promossa a istanza di scrittura nel cluster. Il cluster di database Aurora associato a un proxy è la destinazione di tale proxy. Per comodità, quando crei un proxy attraverso la console, RDS Proxy crea anche il gruppo di destinazione corrispondente e registra automaticamente le destinazioni associate.

Una famiglia di motori è un insieme correlato di motori di database che utilizzano lo stesso protocollo di DB. Scegli la famiglia di motori per ogni proxy creato.

Pooling di connessioni

Ogni proxy esegue il pooling delle connessioni separatamente per l'istanza di scrittura e di lettura del relativo database Aurora associato. Il pooling delle connessioni è un'ottimizzazione che riduce il sovraccarico associato all'apertura e alla chiusura delle connessioni e mantiene aperte contemporaneamente molte connessioni. Questo sovraccarico include la memoria necessaria per gestire ogni nuova connessione. Ciò implica un sovraccarico della CPU anche per chiudere la connessione e aprirne una nuova. Gli esempi includono l'handshake di Transport Layer Security/Secure Sockets Layer (TLS/SSL), l'autenticazione, le funzionalità di negoziazione e così via. Il pooling delle connessioni semplifica la logica dell'applicazione. Non devi scrivere codice dell'applicazione per ridurre al minimo il numero di connessioni aperte simultanee.

Ogni proxy esegue anche il multiplexing delle connessioni, noto anche come riutilizzo della connessione. Con il multiplexing, Server proxy per RDS esegue tutte le operazioni per una transazione utilizzando una connessione al database sottostante, quindi può utilizzare una connessione diversa per la transazione successiva. Puoi aprire molte connessioni simultanee al proxy e il proxy mantiene un numero minore di connessioni aperte all'istanza database o al cluster. In questo modo si riduce ulteriormente il sovraccarico di memoria per le connessioni sul server di database. Questa tecnica riduce anche la possibilità di errori del tipo «troppe connessioni».

Sicurezza di RDS Proxy

RDS Proxy utilizza i meccanismi di sicurezza RDS esistenti come and (IAM). TLS/SSL AWS Identity and Access Management Per informazioni generali sulle funzionalità di sicurezza, consulta Sicurezza in Amazon Aurora. Inoltre, assicurati di familiarizzare con il modo in cui Aurora utilizza l'autenticazione, l'autorizzazione e altri metodi di protezione.

RDS Proxy può fungere da ulteriore livello di sicurezza tra le applicazioni client e il database sottostante. Ad esempio, puoi connetterti al proxy utilizzando TLS 1.3, anche se l'istanza database sottostante supporta una versione precedente di TLS. È possibile connettersi al proxy utilizzando un ruolo IAM anche se il proxy si connette al database utilizzando il metodo di autenticazione utente e password del database. Utilizzando questa tecnica, puoi applicare requisiti di autenticazione avanzata per le applicazioni di database senza un costoso sforzo di migrazione per le istanze DB medesime.

È possibile utilizzare i seguenti metodi di autenticazione con RDS Proxy:

- Credenziali del database

- Autenticazione IAM standard
- End-to-end Autenticazione IAM

Utilizzo di IAM con RDS Proxy

RDS Proxy offre due metodi di autenticazione IAM:

- Autenticazione IAM standard: applica l'autenticazione IAM per le connessioni al proxy mentre il proxy si connette al database utilizzando le credenziali archiviate in Secrets Manager. Ciò applica l'autenticazione IAM per l'accesso al database anche se i database utilizzano l'autenticazione con password nativa. Il proxy recupera le credenziali del database da Secrets Manager e gestisce l'autenticazione al database per conto dell'applicazione.
- End-to-end Autenticazione IAM: applica l'autenticazione IAM per le connessioni direttamente dalle applicazioni al database tramite il proxy. End-to-end L'autenticazione IAM semplifica la configurazione di sicurezza ed evita la gestione delle credenziali del database in Secrets Manager. Questo ulteriore livello di sicurezza impone il controllo degli accessi basato su IAM dall'applicazione client al database.

Per utilizzare l'autenticazione IAM standard, configura il proxy per utilizzare i segreti di Secrets Manager per l'autenticazione e abilitare l'autenticazione IAM per le connessioni client. Le applicazioni si autenticano sul proxy utilizzando IAM, mentre il proxy si autentica nel database utilizzando le credenziali recuperate da Secrets Manager.

Per utilizzare l'autenticazione end-to-end IAM, configura il proxy in modo che utilizzi l'autenticazione IAM quando imposti lo schema di autenticazione predefinito durante la creazione o la modifica del proxy.

Per l'autenticazione end-to-end IAM, devi aggiornare il ruolo IAM associato al proxy per concedere l'`rds-db:connect`autorizzazione. Con l'autenticazione end-to-end IAM, questo elimina la necessità di registrare i singoli utenti del database con il proxy tramite Secrets Manager secrets.

Utilizzo TLS/SSL con RDS Proxy

È possibile connettersi a RDS Proxy utilizzando il TLS/SSL protocollo.

> **Note**
>
> RDS Proxy utilizza i certificati di AWS Certificate Manager (ACM). Se si utilizza RDS Proxy, non è necessario scaricare certificati Amazon RDS o aggiornare applicazioni che utilizzano connessioni RDS Proxy.

Per applicare TLS a tutte le connessioni tra il proxy e il database, è possibile specificare un'impostazione Richiedi Transport Layer Security quando si crea o si modifica un proxy nella Console di gestione AWS.

RDS Proxy può anche garantire che la sessione venga utilizzata TLS/SSL tra il client e l'endpoint RDS Proxy. Per fare in modo che RDS Proxy proceda, specificare il requisito sul lato client. Le variabili di sessione SSL non sono configurate per le connessioni SSL a un database che utilizza RDS Proxy.

- Per Aurora MySQL, specifica il requisito sul lato client con il parametro `--ssl-mode` quando esegui il comando `mysql`.
- Per e Aurora PostgreSQL, specifica `sslmode=require` come parte della stringa `conninfo` quando esegui il comando `psql`.

Server proxy per RDS supporta il protocollo TLS versione 1.0, 1.1, 1.2 e 1.3. È possibile connettersi al proxy utilizzando una versione superiore di TLS rispetto a quella utilizzata nel database sottostante.

Per impostazione predefinita, i programmi client stabiliscono una connessione crittografata con RDS Proxy, con controllo aggiuntivo disponibile tramite l'opzione `--ssl-mode`. Dal lato client, RDS Proxy supporta tutte le modalità SSL.

Per il client, le modalità SSL sono le seguenti:

PREFERRED

SSL è la prima scelta, ma non obbligatoria.

DISABLED

Nessun SSL è abilitato.

REQUIRED

Applica SSL.

VERIFY_CA

Applicare SSL e verificare l'autorità di certificazione (CA).

VERIFY_IDENTITY

Applica SSL e verifica CA e nome host CA.

Quando si utilizza un client con `--ssl-mode` `VERIFY_CA` o `VERIFY_IDENTITY`, specificare l'opzione `--ssl-ca` puntando a una CA in formato `.pem`. Per utilizzare il `.pem` file, scarica tutte le CA root PEMs da Amazon Trust Services e inseriscile in un unico `.pem` file.

Server proxy per RDS utilizza certificati jolly, che si applicano sia a un dominio sia ai relativi sottodomini. Se utilizzi il client `mysql` per eseguire la connessione in modalità SSL `VERIFY_IDENTITY`, al momento devi utilizzare il comando compatibile `mysql` con MySQL 8.0.

Failover

Il failover è una funzionalità ad alta disponibilità che sostituisce un'istanza di database con un'altra quando l'istanza originale diventa non disponibile. Un failover potrebbe verificarsi a causa di un problema con un'istanza di database. Potrebbe anche essere parte delle normali procedure di manutenzione, ad esempio durante un aggiornamento del database. Il failover si applica ai cluster di database Aurora con una o più istanze di lettura oltre all'istanza di scrittura.

La connessione tramite un proxy rende le applicazioni più resilienti ai failover del database. Quando l'istanza database originale diventa non disponibile, RDS Proxy si connette al database di standby senza far cadere le connessioni dell'applicazione inattiva. Ciò consente di velocizzare e semplificare il processo di failover. Il risultato è un failover meno dannoso per l’applicazione rispetto a un tipico problema di riavvio o del database.

Senza RDS Proxy, un failover comporta una breve interruzione. Durante l’interruzione, non è possibile eseguire operazioni di scrittura sul database in stato di failover. Tutte le connessioni al database esistenti vengono interrotte e l'applicazione deve riaprirle. Il database diventa disponibile per le nuove connessioni e le operazioni di scrittura quando un'istanza database di sola lettura viene promossa al posto di quella non disponibile.

Durante i failover DB, RDS Proxy continua ad accettare connessioni allo stesso indirizzo IP e indirizza automaticamente le connessioni alla nuova istanza database primaria. I client che si connettono tramite RDS Proxy non sono soggetti a quanto segue:

- Ritardi di propagazione DNS (Domain Name System) durante il failover.
- Cache DNS locale.
- Timeout di connessione.
- Incertezza su quale istanza database è il writer corrente.
- In attesa di una risposta di query da un precedente writer che è diventato non disponibile senza chiudere le connessioni.

Per le applicazioni che mantengono il proprio pool di connessioni, passare attraverso RDS Proxy significa che la maggior parte delle connessioni rimane attiva durante i failover o altre interruzioni. Vengono annullate solo le connessioni che si trovano nel mezzo di una transazione o istruzione SQL. RDS Proxy accetta immediatamente nuove connessioni. Quando l'istanza di scrittura del database non è disponibile, RDS Proxy accoda le richieste in arrivo.

Per le applicazioni che non mantengono i propri pool di connessioni, RDS Proxy offre velocità di connessione più rapide e connessioni più aperte. Si evita il costoso sovraccarico dovuto a frequenti riconnessioni dal database. Ciò avviene riutilizzando le connessioni al database mantenute nel pool di connessioni del RDS Proxy. Questo approccio è particolarmente importante per le connessioni TLS, dove i costi di installazione sono significativi.

Transazioni

Tutte le istruzioni all'interno di una singola transazione utilizzano sempre la stessa connessione al database sottostante. La connessione diventa disponibile per l'utilizzo da parte di una sessione diversa al termine della transazione. L'utilizzo della transazione come unità di granularità ha le seguenti conseguenze:

- Il riutilizzo della connessione può avvenire dopo ogni singola istruzione quando l'impostazione Aurora MySQL `autocommit` è attivata.
- Al contrario, quando l'impostazione `autocommit` è disattivata, la prima istruzione che viene emessa in una sessione inizia una nuova transazione. Ad esempio, supponi di immettere una sequenza di `SELECT`, `INSERT`, `UPDATE` e altre istruzioni DML (Data Manipulation Language). In questo caso, il riutilizzo della connessione non avviene fino a quando non invii un'istruzione come `COMMIT` o `ROLLBACK` per terminare la transazione.

- L'immissione di un'istruzione DDL (Data Definition Language) fa terminare la transazione dopo il completamento dell'istruzione.

RDS Proxy rileva quando una transazione termina attraverso il protocollo di rete utilizzato dall'applicazione client del database. Il rilevamento delle transazioni non si basa su parole chiave come `COMMIT` o `ROLLBACK` che appaiono nel testo dell'istruzione SQL.

In alcuni casi, RDS Proxy potrebbe rilevare una richiesta di database che rende impossibile spostare la sessione a una connessione diversa. In questi casi, disattiva il multiplexing per quella connessione al resto della sessione. La stessa regola si applica se RDS Proxy non può avere la certezza che il multiplexing sia praticabile per la sessione. Questa operazione è chiamata pinning. Per informazioni su come rilevare e ridurre al minimo il pinning, consulta Evitare di effettuare il pinning di un Server proxy per RDS.

Nozioni di base su RDS Proxy

Utilizza le informazioni nelle pagine seguenti per configurare, gestire Server proxy per Amazon RDS per Aurora e impostare le opzioni di sicurezza correlate. La opzioni di sicurezza controllano chi può accedere a ciascun proxy e come ciascun proxy si connette alle istanze database.

Se non conosci Server proxy per RDS, ti consigliamo di seguire le pagine nell'ordine in cui le presentiamo.

Argomenti

- Configurazione dei prerequisiti di rete per Server proxy per RDS
- Impostazione delle credenziali del database per RDS Proxy
- Configurazione dell'autenticazione IAM per Server proxy per RDS
- Creazione di un proxy per Amazon Aurora
- Visualizzazione di un proxy
- Connessione a un database tramite RDS Proxy

Configurazione dei prerequisiti di rete per Server proxy per RDS

L'utilizzo di Server proxy per RDS richiede un cloud privato virtuale (VPC) comune tra il cluster di database Aurora e Server proxy per RDS. Questo VPC deve avere un minimo di due sottoreti che si

trovano in zone di disponibilità diverse. Il tuo account può possedere queste sottoreti o condividerle con altri account. Per informazioni sulla condivisione di VPC, consulta Work with shared. VPCs

Per il IPv6 supporto, è richiesta una configurazione di rete aggiuntiva:

- IPv6 tipi di rete endpoint: il VPC e le sottoreti devono essere configurati per il supporto. IPv6 Ciò include l'assegnazione di blocchi IPv6 CIDR al VPC e alle sottoreti.
- Tipi di rete endpoint dual-stack: il VPC e le sottoreti devono supportare sia l'indirizzamento che l'indirizzamento. IPv4 IPv6
- IPv6 tipi di rete di connessione di destinazione: il database deve essere configurato per la modalità dual-stack per supportare le connessioni dal proxy. IPv6

Le risorse delle applicazioni client come Amazon EC2, Lambda o Amazon ECS possono trovarsi nello stesso VPC del proxy. In alternativa, possono trovarsi in un VPC separato dal proxy. Se hai effettuato correttamente la connessione a un cluster di database Aurora, sono già disponibili le risorse di rete necessarie.

Argomenti

- Recupero delle informazioni sulle sottoreti
- Pianificazione della capacità degli indirizzi IP

Recupero delle informazioni sulle sottoreti

Se hai appena iniziato a utilizzare Aurora, consulta le informazioni di base per la connessione a un database seguendo le procedure descritte in Configurazione dell'ambiente per Amazon Aurora. Puoi inoltre seguire il tutorial in Nozioni di base su Amazon Aurora.

Per creare un proxy, è necessario specificare le sottoreti e il VPC in cui opera il proxy. Il seguente esempio di Linux mostra AWS CLI i comandi che esaminano le sottoreti VPCs e le sottoreti di proprietà dell'utente. Account AWS In particolare, si passa la sottorete IDs come parametri quando si crea un proxy utilizzando la CLI.

```
aws ec2 describe-vpcs
aws ec2 describe-internet-gateways
aws ec2 describe-subnets --query '*[].[VpcId,SubnetId]' --output text | sort
```

Il seguente esempio di Linux mostra AWS CLI i comandi per determinare la sottorete IDs corrispondente a una specifica istanza DB.

Per un cluster Aurora, è innanzitutto possibile trovare l'ID per una delle istanze database associate. È possibile estrarre la sottorete IDs utilizzata da quell'istanza DB. A tale scopo, esamina i campi nidificati all'interno degli attributi `DBSubnetGroup` e `Subnets` nell'output di descrizione dell'istanza database. È possibile specificare alcune o tutte queste sottoreti IDs quando si configura un proxy per quel server di database.

```
$ # Find the ID of any DB instance in the cluster.
$  aws rds describe-db-clusters --db-cluster-identifier my_cluster_id --query '*[].
[DBClusterMembers]||[0]||[0][*].DBInstanceIdentifier' --output text
```

```
my_instance_id
instance_id_2
instance_id_3
```

Dopo aver trovato l'identificatore dell'istanza database, esamina il VPC associato per individuare le sottoreti. Il seguente esempio Linux mostra come.

```
$ #From the DB instance, trace through the DBSubnetGroup and Subnets to find the subnet
 IDs.
$ aws rds describe-db-instances --db-instance-identifier my_instance_id --query '*[].
[DBSubnetGroup]||[0]||[0]||[Subnets]||[0]||[*].SubnetIdentifier' --output text
```

```
subnet_id_1
subnet_id_2
subnet_id_3
...
```

```
$ #From the DB instance, find the VPC.
$ aws rds describe-db-instances --db-instance-identifier my_instance_id --query '*[].
[DBSubnetGroup]||[0]||[0].VpcId' --output text
```

```
my_vpc_id
```

```
$ aws ec2 describe-subnets --filters Name=vpc-id,Values=my_vpc_id --query '*[].
[SubnetId]' --output text
```

```
subnet_id_1
subnet_id_2
```

```
subnet_id_3
subnet_id_4
subnet_id_5
subnet_id_6
```

Pianificazione della capacità degli indirizzi IP

RDS Proxy regola automaticamente la propria capacità in base alla configurazione delle istanze DB registrate con esso. Per le istanze assegnate, ciò è determinato dalla dimensione dell'istanza e, per le Aurora Serverless v2 istanze, è determinato dalla capacità massima dell'ACU. Alcune operazioni potrebbero richiedere anche capacità aggiuntiva, ad esempio l'aumento delle dimensioni di un database registrato o operazioni di manutenzione interne del proxy RDS. Durante queste operazioni, il proxy può aver bisogno di più indirizzi IP per fornire la capacità aggiuntiva. Questi indirizzi aggiuntivi consentono al proxy di dimensionare senza interessare il carico di lavoro. La mancanza di indirizzi IP liberi nelle sottoreti impedisce al proxy di aumentare le dimensioni, causando eventuali latenze elevate nell'esecuzione di query o errori di connessione del client. RDS ti avvisa tramite l'evento `RDS-EVENT-0243` quando non ci sono abbastanza indirizzi IP liberi nelle tue sottoreti. Per informazioni su questo evento, consulta Utilizzo degli eventi RDS Proxy.

Riserva il seguente numero minimo consigliato di indirizzi IP liberi nelle sottoreti per il proxy, in base alle dimensioni della classe di istanza database.

DB instance class (Classe istanza database)	Numero minimo di indirizzi IP liberi
db.*.xlarge o più piccola	10
db.*.2xlarge	15
db.*.4xlarge	25
db.*.8xlarge	45
db.*.12xlarge	60
db.*.16xlarge	75
db.*.24xlarge	110

PertantoAurora Serverless v2, riservate al proxy il seguente numero minimo di indirizzi IP gratuiti nelle sottoreti, in base alla capacità massima dell'ACU.

`Maximum ACU Capacity`	Numero minimo di indirizzi IP liberi
16 o inferiore	10
32	15
64	25
96	30
128	40
160	50
192	55
224	65
256	75

> **Note**
>
> Il proxy RDS non utilizza più di 215 indirizzi IP per ogni proxy in un VPC.

Il proxy RDS richiede un minimo di 10 indirizzi IP per il database Aurora. Questi numeri consigliati per indirizzi IP sono stime per un proxy con solo l'endpoint predefinito. Per ogni endpoint personalizzato aggiuntivo, consigliamo di prenotare altri tre indirizzi IP. Per ogni istanza del lettore Aurora, si consiglia di riservare indirizzi IP aggiuntivi come specificato nella tabella in base al numero massimo di quel lettore ACUs per la Aurora Serverless v2 destinazione o alla dimensione dell'istanza DB per una destinazione predisposta.

Per stimare gli indirizzi IP richiesti per un proxy associato a un cluster Aurora DB con:

- 1 istanza provisioned writer di dimensioni adeguate `db.r5.8xlarge` e 1 istanza provisioned reader di dimensioni adeguate. `db.r5.2xlarge`

- Il proxy collegato a questo cluster ha un endpoint predefinito e un endpoint personalizzato con ruolo di sola lettura.

In questo caso, il proxy richiede circa 63 indirizzi IP liberi (45 per l'istanza di scrittura, 15 per l'istanza di lettura e 3 per l'endpoint personalizzato aggiuntivo).

Per stimare gli indirizzi IP richiesti per un proxy associato a un cluster Aurora DB che dispone di:

- 1 istanza di Aurora Serverless v2 scrittura con capacità massima di 256 ACUs e 1 istanza di lettura Serverless v2 con capacità massima di 192. ACUs
- Il proxy collegato a questo cluster ha l'endpoint predefinito e 1 endpoint personalizzato con il ruolo di sola lettura.

In questo caso, il proxy necessita di circa 133 indirizzi IP gratuiti (75 per l'istanza writer, 55 per l'istanza reader e 3 per l'endpoint personalizzato aggiuntivo).

Per stimare gli indirizzi IP richiesti per un proxy associato a un cluster Aurora che dispone di:

- 1 istanza di scripting fornita con dimensione dell'istanza DB di db.r5.4xlarge e 1 istanza di lettura Serverless v2 con capacità massima di 64. ACUs
- Il proxy collegato a questo cluster ha l'endpoint predefinito e 1 endpoint personalizzato con il ruolo di sola lettura.

In questo caso, il proxy necessita di circa 53 indirizzi IP gratuiti (25 per l'istanza writer, 25 per l'istanza reader e 3 per l'endpoint personalizzato aggiuntivo).

Per stimare gli indirizzi IP richiesti per un proxy associato a un cluster Aurora DB che dispone di:

- 1 istanza provisioned writer di dimensione db.r5.24xlarge e 3 istanze provisioned reader di dimensione db.r5.8xlarge.
- Il proxy collegato a questo cluster database ha l'endpoint predefinito e 1 endpoint personalizzato con il ruolo di sola lettura.

In questo caso, il proxy necessita di 215 indirizzi IP gratuiti. Sebbene i calcoli suggeriscano 248 IPs (110 + (3*45) + 3), RDS Proxy non utilizza più di 215 indirizzi IP per ogni proxy in un VPC.

Impostazione delle credenziali del database per RDS Proxy

Il proxy RDS in Amazon RDS viene utilizzato Gestione dei segreti AWS per archiviare e gestire le credenziali del database in modo sicuro. Anziché essere incorporate nell'applicazione, le credenziali vengono impostate associando un proxy a un segreto Secrets Manager contenente i dettagli di autenticazione necessari. È necessario creare un segreto Secrets Manager separato per ogni account utente del database a cui si connette il proxy sull'sul cluster di database Aurora.

In alternativa, puoi configurare RDS Proxy per utilizzare l'autenticazione end-to-end IAM, che elimina la necessità di archiviare le credenziali del database in Secrets Manager. RDS Proxy utilizza l'autenticazione IAM per entrambe client-to-proxy le connessioni. proxy-to-database Ciò fornisce una soluzione di autenticazione basata su IAM completamente integrata che non richiede la gestione di segreti o password. Per informazioni sull'aggiunta di un nuovo utente IAM DB, consulta. Creazione di un account database tramite l'autenticazione IAM

Argomenti

- Creazione di segreti da utilizzare con Server proxy per RDS

Creazione di segreti da utilizzare con Server proxy per RDS

Prima di creare un proxy, devi creare almeno un segreto in cui archiviare le credenziali del database.

Console

Per creare un segreto

1. Apri la console Secrets Manager all'indirizzo https://console.aws.amazon.com/secretsmanager/.
2. Scegli Archivia un nuovo segreto.
3. Scegli Credenziali per database Amazon RDS.
4. Inserisci un nome utente e una password. Le credenziali inserite devono corrispondere a quelle di un utente esistente nel database RDS associato. Server proxy per RDS utilizza queste credenziali per autenticare e stabilire connessioni al database per conto delle applicazioni.

   In caso di mancata corrispondenza, puoi aggiornare il segreto in modo che corrisponda alla password del database. Finché non aggiorni il segreto, i tentativi di connessione effettuati tramite il proxy utilizzando tale segreto avranno esito negativo, ma le connessioni che utilizzano altri segreti validi continueranno a funzionare.

> **Note**
>
> Per RDS per SQL Server, Server proxy per RDS richiede un segreto con distinzione tra maiuscole e minuscole in Secrets Manager, indipendentemente dalle impostazioni di confronto delle istanze database. Se l'applicazione consente nomi utente con uso di maiuscole e minuscole diverse, come "Admin" e "admin", è necessario creare segreti separati per ciascuno. Server proxy per RDS non supporta l'autenticazione senza distinzione tra maiuscole e minuscole tra client e proxy.
> Per ulteriori informazioni sul confronto in SQL Server, consulta la documentazione di Microsoft SQL Server.

5. Per Database, seleziona il database Amazon RDS a cui accederà il segreto.
6. Inserisci le altre impostazioni per il segreto, quindi scegli Archivia. Per le istruzioni complete, consulta Creazione di un segreto Gestione dei segreti AWS nella Guida per l'utente di Gestione dei segreti AWS .

AWS CLI

Quando crei un proxy tramite AWS CLI, specifichi gli Amazon Resource Names (ARNs) dei segreti corrispondenti. per tutti gli account utente del database a cui il proxy può accedere. In Console di gestione AWS, scegli i segreti in base ai loro nomi descrittivi.

- Per creare un segreto Secrets Manager da utilizzare con Server proxy per RDS, utilizza il comando create-secret:

```
aws secretsmanager create-secret \
  --name "secret_name" \
  --description "secret_description" \
  --region region_name \
  --secret-string '{"username":"db_user","password":"db_user_password"}'
```

- Puoi anche creare una chiave personalizzata per crittografare il segreto Secrets Manager. Il comando seguente crea una chiave di esempio.

```
aws kms create-key --description "test-key" --policy '{
  "Id":"kms-policy",
  "Version": "2012-10-17",
  "Statement":
```

```
    [
      {
        "Sid":"Enable IAM User Permissions",
        "Effect":"Allow",
        "Principal":{"AWS":"arn:aws:iam::account_id:root"},
        "Action":"kms:*","Resource":"*"
      },
      {
        "Sid":"Allow access for Key Administrators",
        "Effect":"Allow",
        "Principal":
          {
            "AWS":
              ["$USER_ARN","arn:aws:iam:account_id::role/Admin"]
          },
        "Action":
          [
            "kms:Create*",
            "kms:Describe*",
            "kms:Enable*",
            "kms:List*",
            "kms:Put*",
            "kms:Update*",
            "kms:Revoke*",
            "kms:Disable*",
            "kms:Get*",
            "kms:Delete*",
            "kms:TagResource",
            "kms:UntagResource",
            "kms:ScheduleKeyDeletion",
            "kms:CancelKeyDeletion"
          ],
        "Resource":"*"
      },
      {
        "Sid":"Allow use of the key",
        "Effect":"Allow",
        "Principal":{"AWS":"$ROLE_ARN"},
        "Action":["kms:Decrypt","kms:DescribeKey"],
        "Resource":"*"
      }
    ]
}'
```

Ad esempio, i comandi seguenti creano segreti Secrets Manager per due utenti di database:

```
aws secretsmanager create-secret \
  --name secret_name_1 --description "db admin user" \
  --secret-string '{"username":"admin","password":"choose_your_own_password"}'

aws secretsmanager create-secret \
  --name secret_name_2 --description "application user" \
  --secret-string '{"username":"app-user","password":"choose_your_own_password"}'
```

Per creare questi segreti crittografati con la tua AWS KMS chiave personalizzata, usa i seguenti comandi:

```
aws secretsmanager create-secret \
  --name secret_name_1 --description "db admin user" \
  --secret-string '{"username":"admin","password":"choose_your_own_password"}' \
  --kms-key-id arn:aws:kms:us-east-2:account_id:key/key_id

aws secretsmanager create-secret \
  --name secret_name_2 --description "application user" \
  --secret-string '{"username":"app-user","password":"choose_your_own_password"}' \
  --kms-key-id arn:aws:kms:us-east-2:account_id:key/key_id
```

Per vedere i segreti di proprietà del tuo AWS account, usa il comando list-secrets:

```
aws secretsmanager list-secrets
```

Quando crei un proxy utilizzando la CLI, passi gli Amazon Resource Names (ARNs) di uno o più segreti al `--auth` parametro. L'esempio seguente mostra come preparare un report con solo il nome e l'ARN di ogni segreto di proprietà del tuo account AWS . Questo esempio utilizza il `--output table` parametro disponibile nella AWS CLI versione 2. Se stai usando la AWS CLI versione 1, usa `--output text` invece.

```
aws secretsmanager list-secrets --query '*[].[Name,ARN]' --output table
```

Per confermare che il segreto contenga le credenziali corrette nel formato corretto, utilizzate il get-secret-valuecomando. Sostituisci *`your_secret_name`* con il nome breve o l'ARN del segreto.

```
aws secretsmanager get-secret-value --secret-id your_secret_name
```

L'output contiene una riga con un valore con codifica JSON simile al seguente:

```
...
"SecretString": "{\"username\":\"your_username\",\"password\":\"your_password\"}",
...
```

Configurazione dell'autenticazione IAM per Server proxy per RDS

Per configurare l'autenticazione AWS Identity and Access Management (IAM) per RDS Proxy in Amazon RDS, crea e configura una policy IAM che conceda le autorizzazioni necessarie.

Questo argomento fornisce i passaggi per configurare l'autenticazione IAM per Server proxy per RDS, inclusa la creazione della policy IAM richiesta e il suo collegamento a un ruolo IAM.

> **Tip**
>
> Questa procedura è necessaria solo se desideri creare il tuo ruolo IAM. Altrimenti, RDS può creare automaticamente il ruolo richiesto quando configuri il proxy, quindi puoi saltare questi passaggi.

Prerequisiti

Prima di configurare l'autenticazione IAM per Server proxy per RDS, assicurati di disporre di quanto segue:

- Gestione dei segreti AWS: almeno un segreto archiviato che contiene le credenziali del database. Per istruzioni su come creare segreti, consulta the section called "Impostazione delle credenziali del database".

  Questo non è necessario se utilizzi l'autenticazione IAM. end-to-end
- Autorizzazioni IAM: un ruolo o un utente IAM con autorizzazioni per creare e gestire policy, ruoli e segreti IAM in Gestione dei segreti AWS.

Creazione di una policy IAM per l'autenticazione end-to-end IAM

Quando si utilizza l'autenticazione end-to-end IAM, RDS Proxy si connette al database utilizzando l'autenticazione IAM anziché recuperare le credenziali da Secrets Manager. Ciò richiede la

configurazione del ruolo IAM con `rds-db:connect` le autorizzazioni per gli account di database che desideri utilizzare con il proxy.

Per autenticare il proxy RDS nel database utilizzando IAM, crea un ruolo IAM con una policy che conceda le autorizzazioni di connessione al database necessarie.

Console

Per creare un ruolo per l'autenticazione end-to-end IAM con il tuo proxy

1. Accedi Console di gestione AWS e apri la console IAM all'indirizzo https://console.aws.amazon.com/iam/.
2. Creazione di una policy delle autorizzazioni per il ruolo. Per i passaggi generali, consulta Creare policy IAM (console).

   Incolla la seguente policy nell'editor JSON ed effettua le seguenti modifiche:

   - Sostituisci il tuo ID account.
   - Sostituiscilo `us-east-2` con il luogo in cui deve risiedere il proxy.
   - Sostituisci la risorsa del database IDs e i nomi utente con quelli che desideri utilizzare. Il formato dell'ID della risorsa è diverso tra le istanze RDS e. Aurora clusters

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": "rds-db:connect",
            "Resource": [
                "arn:aws:rds-db:us-
east-2:account_id:dbuser:db_instance_resource_id/db_user_name_1",
                "arn:aws:rds-db:us-
east-2:account_id:dbuser:db_instance_resource_id/db_user_name_2"
            ]
        }
    ]
}
```

3. Crea il ruolo e collega ad esso una policy delle autorizzazioni. Per i passaggi generali, consulta Creare un ruolo per delegare le autorizzazioni a un servizio. AWS

Per Tipo di entità attendibile, scegli AWS Servizio. In Caso d'uso, seleziona RDS e scegli RDS - Aggiungi ruolo al database per il caso d'uso.

4. In Policy di autorizzazione, scegli la policy creata.
5. In Seleziona entità attendibili, inserisci la seguente policy di attendibilità per il ruolo:

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "",
      "Effect": "Allow",
      "Principal": {
        "Service": "rds.amazonaws.com"
      },
      "Action": "sts:AssumeRole"
    }
  ]
}
```

AWS CLI

Per creare il ruolo utilizzando ilAWS CLI, invia la seguente richiesta:

```
aws iam create-role \
  --role-name my_e2e_iam_role_name \

  --assume-role-policy-document '{"Version":"2012-10-17",
     "Statement":[{"Effect":"Allow","Principal":{"Service":
["rds.amazonaws.com"]},"Action":"sts:AssumeRole"}]}'
```

Quindi, collega tale policy al ruolo:

```
aws iam put-role-policy \
  --role-name my_e2e_iam_role_name \
  --policy-name e2e_iam_db_connect_policy \
  --policy-document '{

    "Version": "2012-10-17",
    "Statement": [
        {
```

```
            "Effect": "Allow",
            "Action": "rds-db:connect",
            "Resource": [
                "arn:aws:rds-db:us-
east-2:account_id:dbuser:db_instance_resource_id/db_user_name_1",
                "arn:aws:rds-db:us-
east-2:account_id:dbuser:db_instance_resource_id/db_user_name_2"
            ]
        }
    ]
}'
```

Con il ruolo e le autorizzazioni end-to-end IAM configurati per l'autenticazione IAM, ora puoi creare un proxy con `DefaultAuthScheme` set to`IAM_AUTH`. Questo proxy si autentica direttamente nel database utilizzando IAM senza richiedere i segreti di Secrets Manager. Per istruzioni, consulta the section called “Creazione di un proxy”.

Quando utilizzi l'autenticazione end-to-end IAM, assicurati che gli utenti del database siano configurati per l'autenticazione IAM come descritto in. Creazione di un account database tramite l’autenticazione IAM

Creazione di una policy IAM per l’accesso a Secrets Manager

Per consentire a Server proxy per RDS di recuperare le credenziali del database da Secrets Manager, devi creare un ruolo IAM con una policy che concede le autorizzazioni necessarie.

Console

Per creare un ruolo che acceda ai segreti da utilizzare con il proxy

1. Accedi a Console di gestione AWS e apri la console IAM all'indirizzo https://console.aws.amazon.com/iam/.
2. Creazione di una policy delle autorizzazioni per il ruolo. Per i passaggi generali, consulta Creare policy IAM (console).

   Incolla la seguente policy nell’editor JSON ed effettua le seguenti modifiche:

   - Sostituisci il tuo ID account.
   - Sostituisci `us-east-2` con la Regione in cui risiederà il proxy.
   - Sostituisci i nomi segreti con quelli che hai creato. Per ulteriori informazioni, consulta Specificazione delle chiavi KMS nelle istruzioni della policy IAM.

- Sostituisci l'ID della chiave KMS con quello utilizzato per crittografare i segreti di Secrets Manager, la chiave di default o la propria chiave.

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": "secretsmanager:GetSecretValue",
            "Resource": [
                "arn:aws:secretsmanager:us-
east-2:111122223333:secret:secret_name_1",
                "arn:aws:secretsmanager:us-
east-2:111122223333:secret:secret_name_2"
            ]
        },
        {
            "Effect": "Allow",
            "Action": "kms:Decrypt",
            "Resource": "arn:aws:kms:us-east-2:111122223333:key/key_id",
            "Condition": {
                "StringEquals": {
                    "kms:ViaService": "secretsmanager.us-
east-2.amazonaws.com"
                }
            }
        }
    ]
}
```

3. Crea il ruolo e collega ad esso una policy delle autorizzazioni. Per i passaggi generali, consulta Creare un ruolo per delegare le autorizzazioni a un AWS servizio.

   Per Tipo di entità attendibile, scegli AWS Servizio. In Caso d'uso, seleziona RDS e scegli RDS - Aggiungi ruolo al database per il caso d'uso.

4. In Policy di autorizzazione, scegli la policy creata.
5. In Seleziona entità attendibili, inserisci la seguente policy di attendibilità per il ruolo:

JSON

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Sid": "",
      "Effect": "Allow",
      "Principal": {
        "Service": "rds.amazonaws.com"
      },
      "Action": "sts:AssumeRole"
    }
  ]
}
```

AWS CLI

Per creare il ruolo utilizzando ilAWS CLI, invia la seguente richiesta:

```
aws iam create-role \
  --role-name my_role_name \
  --assume-role-policy-document '{"Version": "2012-10-17",
      "Statement":[{"Effect":"Allow","Principal":{"Service":
["rds.amazonaws.com"]},"Action":"sts:AssumeRole"}]}'
```

Quindi, collega tale policy al ruolo:

```
aws iam put-role-policy \
  --role-name my_role_name \
  --policy-name secret_reader_policy \
  --policy-document '{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "VisualEditor0",
            "Effect": "Allow",
            "Action": "secretsmanager:GetSecretValue",
            "Resource": [
                "arn:aws:secretsmanager:us-east-2:account_id:secret:secret_name_1",
```

```
                "arn:aws:secretsmanager:us-east-2:account_id:secret:secret_name_2"
            ]
        },
        {
            "Sid": "VisualEditor1",
            "Effect": "Allow",
            "Action": "kms:Decrypt",
            "Resource": "arn:aws:kms:us-east-2:account_id:key/key_id",
            "Condition": {
                "StringEquals": {
                    "kms:ViaService": "secretsmanager.us-east-2.amazonaws.com"
                }
            }
        }
    ]
}'
```

Con il ruolo e le autorizzazioni IAM configurati, ora puoi creare un proxy e associarlo a questo ruolo. Ciò consente al proxy di recuperare le credenziali del database in modo sicuro Gestione dei segreti AWS e di abilitare l'autenticazione IAM per le applicazioni. Per istruzioni, consulta the section called “Creazione di un proxy”.

Creazione di un proxy per Amazon Aurora

Puoi utilizzare Server proxy per Amazon RDS per migliorare la scalabilità, la disponibilità e la sicurezza delle tue applicazioni di database raggruppando le connessioni e gestendo i failover del database in modo più efficiente. In questo argomento viene introdotta la creazione di un proxy. Prima di iniziare, assicurati che il database soddisfi i prerequisiti necessari, tra cui le autorizzazioni IAM e la configurazione del VPC.

Puoi associare un proxy a un cluster database Aurora MySQL o Aurora PostgreSQL.

Console

Per creare un proxy

1. Accedi a Console di gestione AWS e apri la console Amazon RDS all'indirizzo https://console.aws.amazon.com/rds/.
2. Nel riquadro di navigazione scegli Proxy.
3. Scegli Create proxy (Crea proxy).
4. Configura il proxy utilizzando le impostazioni indicate di seguito.

Impostazione	Description
Famiglia di motori	Il protocollo di rete del database riconosciuto dal proxy quando interpreta il traffico di rete verso e dal database. **Note** Per utilizzare Aurora PostgreSQL, assicurati di mantenere il database `postgres` nella tua istanza. Consulta Risoluzione dei problemi relativi al database `postgres` eliminato.
Identificatore proxy	Un nome univoco all'interno dell'ID AWS dell'account e AWS della regione corrente.
Timeout di connessione client inattivo	Il proxy chiude una connessione client se questa rimane inattiva per un determinato periodo. Il valore predefinito è 1.800 secondi (30 minuti). Una connessione client è inattiva quando l'applica zione non invia una nuova richiesta entro il tempo specifica to dopo il completamento della richiesta precedente. Il proxy mantiene aperta la connessione al database sottostante e la restituisce al pool di connessioni, rendendola disponibile per nuove connessioni client. Per eliminare in modo proattivo le connessioni obsolete, riduci il timeout della connessione client per inattività. Per ridurre al minimo i costi di connessione durante i picchi dei carichi di lavoro, aumenta il timeout.
Database	Scegli un cluster di database Aurora per l'accesso tramite questo proxy. L'elenco include solo istanze DB e cluster con motori di database compatibili, versioni del motore e altre impostazioni. Se l'elenco è vuoto, crea una nuova istanza database o un cluster compatibile con RDS Proxy. A tale scopo, segui la procedura in Creazione di un cluster database Amazon Aurora. Quindi prova a creare nuovamente il proxy.

Impostazione	Description
Numero massimo di connessioni del pool di connessione	Un valore compreso tra 1 e 100 per definire il limite per `max_connections` , espresso in percentuale, che Server proxy per RDS può utilizzare. Se intendi utilizzare un solo proxy con questa istanza o cluster di database, puoi impostare il limite su 100. Per ulteriori informazioni su come Server proxy per RDS utilizza questa impostazione, consulta the section called “MaxConnectionsPercent”.
Filtri per l’aggiunta di sessioni	Impedisce a Server proxy per RDS di bloccare determinati stati di sessione rilevati, ignorando così le misure di sicurezza predefinite per le connessioni multiplexing. Attualmente, PostgreSQL non supporta questa impostazione e l’unica opzione disponibile è `EXCLUDE_VARIABLE_SETS` . La sua attivazione potrebbe far sì che le variabili di sessione di una connessione influiscano sulle altre, causando errori o problemi di correttezza se le query si basano su variabili di sessione impostate al di fuori della transazione corrente. Utilizza questa opzione solo dopo aver verificato che le applicazioni possano condividere in modo sicuro le connessioni al database. I seguenti modelli sono considerati sicuri: • Istruzioni `SET` in cui non viene apportata alcuna modifica al valore della variabile di sessione effettiva. In altre parole, la variabile di sessione rimane invariata. • Modifichi il valore della variabile di sessione ed esegui un'istruzione nella stessa transazione. Per ulteriori informazioni, consulta Evitare di effettuare il pinning di un Server proxy per RDS.

Impostazione	Description
Timeout del prestito della connessione	Se prevedi che il proxy utilizzerà tutte le connessioni al database disponibili, imposta il tempo di attesa prima della restituzione di un errore di timeout. Puoi specificare fino a 5 minuti. Questa impostazione si applica solo quando il proxy ha il numero massimo di connessioni aperte e tutte le connessioni sono già in uso.
Query di inizializzazione	(Opzionale) Aggiungi una query di inizializzazione o modifica quella corrente. Puoi specificare una o più istruzioni SQL per l'esecuzione del proxy all'apertura di ogni nuova connessione al database. L'impostazione è in genere utilizzata con le istruzioni SET per assicurarsi che ogni connessione abbia impostazioni identiche. Assicurati che la query che aggiungi sia valida. Per includere più variabili in una singola istruzione SET, utilizza la virgola come separatore. Ad esempio: `SET variable1 =value1, variable2 =value2` Per più istruzioni, utilizzare il punto e virgola come separatore. **Important** Poiché è possibile accedervi nell'ambito della configurazione del gruppo di destinazione, la query di inizializzazione non è protetta da metodi di autenticazione o crittografia. Chiunque abbia accesso alla visualizzazione o alla gestione della configurazione del gruppo di destinazione proxy può visualizzare la query di inizializzazione. È opportuno evitare di aggiungere a questa opzione dati sensibili, ad esempio password o chiavi di crittografia di lunga durata.

Impostazione	Description
AWS Identity and Access Management Ruolo (IAM)	Un ruolo IAM con autorizzazione di accesso ai segreti Secrets Manager, che rappresentano le credenziali per gli account utente del database che il proxy può utilizzare. In alternativa, è possibile creare un nuovo ruolo IAM dalla Console di gestione AWS.
Segreti Secrets Manager	Crea o scegli i segreti di Secrets Manager che rappresentano le credenziali per gli account degli utenti del database che possono utilizzare il proxy. Quando lo schema di autenticazione predefinito è impostato su Nessuno, questo campo è obbligatorio. Quando lo schema di autenticazione predefinito è impostato sull'autenticazione IAM, questo campo diventa facoltativo e viene contrassegnato come tale nella console. Puoi scegliere uno o più segreti dal menu a discesa o creare un nuovo segreto utilizzando il link Crea un nuovo segreto.
Tipo di autenticazione client	Il tipo di autenticazione utilizzato dal proxy per le connessio ni dei client. La tua scelta si applica a tutti i segreti di Secrets Manager che associ a questo proxy. Se devi specificare un tipo di autenticazione client diverso per ogni segreto, crea il tuo proxy utilizzando invece l'API AWS CLI o l'API. Specificate questa opzione solo quando la connessione del client utilizza le credenziali del database per l'autenticazione.
Autenticazione IAM	Specificate Obbligatoria o Non consentita per l'autenticazione IAM per le connessioni al proxy. La tua scelta si applica a tutti i segreti di Secrets Manager che associ a questo proxy. Se devi specificare un'autenticazione IAM diversa per ogni segreto, crea il proxy utilizzando invece l'API AWS CLI o.

Impostazione	Description
Schema di autenticazione predefinito	Scegli il tipo di autenticazione predefinito utilizzato dal proxy per le connessioni del client al proxy e le connessioni dal proxy al database sottostante. Sono disponibili le seguenti opzioni: • Nessuno (impostazione predefinita): il proxy recupera le credenziali del database dai segreti di Secrets Manager. • Autenticazione IAM: il proxy utilizza l'autenticazione IAM per connettersi al database, abilitando l'autenticazione end-to-end IAM. Quando si seleziona l'autenticazione IAM, viene visualizzato un avviso informativo che ricorda di abilitare l'autenticazione del database IAM per i database nella configurazione del gruppo target. **Note** Questa opzione è supportata solo per le famiglie di motori MySQL, PostgreSQL e MariaDB.

Impostazione	Description
Account di database per l'autenticazione IAM	Questo campo viene visualizzato solo quando lo schema di autenticazione predefinito è impostato sull'autenticazione IAM e il ruolo di gestione delle identità e degli accessi (IAM) è impostato su Crea ruolo IAM. Assegna un nome agli account utente del database per il proxy da utilizzare con l'autenticazione IAM. Questo è un campo obbligatorio. Specificare più account mediante: • Digitare il nome utente di un database per aggiungerlo come tag • Utilizzo di nomi utente specifici del database (ad esempio`db_user`,`jane_doe`) • Utilizzo di pattern jolly per più utenti (ad esempio,`db_test_*`) Ogni account appare come un tag rimovibile che puoi eliminare facendo clic sull'icona X. La console utilizza questi valori per creare le `rds-db:connect` autorizzazioni appropriate nella policy del ruolo IAM.
Richiede Transport Layer Security	Applica TLS/SSL per tutte le connessioni client. Il proxy utilizza la stessa impostazione di crittografia quando effettua la connessione al database sottostante, indipendentemente dal fatto che la connessione client sia crittografata o meno.

Impostazione	Description
Tipo di rete della connessione di destinazione	La versione IP utilizzata dal proxy per connettersi al database di destinazione. Seleziona una delle opzioni seguenti: • IPv4— Il proxy si connette al database utilizzando IPv4 gli indirizzi. • IPv6— Il proxy si connette al database utilizzando IPv6 gli indirizzi. Il valore predefinito è IPv4. Per poter essere utilizzato IPv6, il database deve supportare la modalità dual-stack. La modalità dual-stack non è disponibile per le connessioni di destinazione.
Tipo di rete dell’endpoint	La versione IP dell’endpoint proxy utilizzato dai client per connettersi al proxy. Seleziona una delle opzioni seguenti: • IPv4— L'endpoint proxy utilizza solo indirizzi. IPv4 • IPv6— L'endpoint proxy utilizza solo IPv6 indirizzi. • Dual-stack: l'endpoint proxy supporta entrambi gli indirizzi. IPv4 IPv6 Il valore predefinito è IPv4. Per utilizzare IPv6 il dual-stack, il VPC e le sottoreti devono essere configurati per supportare il tipo di rete selezionato.
Sottoreti	Questo campo è precompilato con tutte le sottoreti associate al VPC. È possibile rimuovere tutte le sottoreti non necessarie per il proxy, ma è necessario lasciarne almeno due. Per i tipi di rete endpoint IPv6 o dual-stack, assicurati che le sottoreti selezionate supportino il tipo di rete scelto.

Impostazione	Description
Gruppo di sicurezza VPC	Scegli un gruppo di sicurezza VPC esistente o creane uno nuovo dalla Console di gestione AWS. Configura le regole in entrata per permettere alle applicazioni di accedere al proxy e le regole in uscita per consentire il traffico proveniente dalle destinazioni del database. **Note** Questo gruppo di sicurezza deve consentire le connessioni dal proxy al database. Viene utilizzato sia per l'ingresso dalle applicazioni al proxy sia per l'uscita dal proxy al database. Ad esempio, se utilizzi lo stesso gruppo di sicurezza sia per il database che per il proxy, assicurati che le risorse al suo possano comunicare tra loro. Quando ricorri a un VPC condiviso, evita di utilizzar e il gruppo di sicurezza predefinito per il VPC o uno associato a un altro account. Scegli un gruppo di sicurezza appartenente all'account. Se non ne esiste uno, crealo. Per ulteriori informazioni, consulta Work with shared. VPCs RDS implementa un proxy su più zone di disponibilità per garantire una disponibilità elevata. Per abilitare la comunicaz ione tra zone di disponibilità, la lista di controllo degli accessi (ACL) relativa alla sottorete del proxy deve consentire l'uscita sulla porta del motore e l'ingresso su tutte le porte. Per ulteriori informazioni sulla rete ACLs, consulta Controllare il traffico verso le sottoreti utilizzando la rete. ACLs Se l'ACL di rete per il proxy e la destinazione sono identici, devi aggiungere una regola di ingresso del protocollo TCP in cui Origine è impostata sul VPC CIDR. È inoltre necessario aggiungere una regola di uscita del

Impostazione	Description
	protocollo TCP specifica per la porta del motore in cui l'opzione Destinazione sia impostata su CIDR VPC.
Attiva la registrazione di log avanzata	Attiva questa impostazione per risolvere problemi di compatibi lità o prestazioni del proxy. Se abilitato, Server proxy per RDS registra informazioni dettagliate sulle prestazioni per aiutarti a eseguire il debug del comportamento SQL o delle prestazioni e della scalabilità della connessione proxy. Abilita questa impostazione solo per il debug e assicurati che siano state adottate misure di sicurezza adeguate per proteggere le informazioni sensibili nei log. Per ridurre al minimo il sovraccarico, Server proxy per RDS disattiva automatic amente questa impostazione 24 ore dopo l'attivazione. Utilizzala temporaneamente per risolvere problemi specifici.

5. Scegli Create proxy (Crea proxy).

AWS CLI

Per creare un proxy utilizzando il AWS CLI, chiamate il create-db-proxycomando con i seguenti parametri obbligatori:

- `--db-proxy-name`
- `--engine-family`
- `--role-arn`
- `--vpc-subnet-ids`

Il valore `--engine-family` prevede la distinzione tra lettere maiuscole e minuscole.

Example

Per Linux, macOS o Unix:

```
aws rds create-db-proxy \
    --db-proxy-name proxy_name \
```

```
    --engine-family { MYSQL | POSTGRESQL | SQLSERVER } \
    --role-arn iam_role \
    --vpc-subnet-ids space_separated_list \
    [--default-auth-scheme { NONE | IAM_AUTH }] \
    [--auth ProxyAuthenticationConfig_JSON_string] \
    [--vpc-security-group-ids space_separated_list] \
    [--require-tls | --no-require-tls] \
    [--idle-client-timeout value] \
    [--debug-logging | --no-debug-logging] \
    [--endpoint-network-type { IPV4 | IPV6 | DUAL }] \
    [--target-connection-network-type { IPV4 | IPV6 }] \
    [--tags comma_separated_list]
```

Per Windows:

```
aws rds create-db-proxy ^
    --db-proxy-name proxy_name ^
    --engine-family { MYSQL | POSTGRESQL | SQLSERVER } ^
    --role-arn iam_role ^
    --vpc-subnet-ids space_separated_list ^
    [--default-auth-scheme { NONE | IAM_AUTH }] ^
    [--auth ProxyAuthenticationConfig_JSON_string] ^
    [--vpc-security-group-ids space_separated_list] ^
    [--require-tls | --no-require-tls] ^
    [--idle-client-timeout value] ^
    [--debug-logging | --no-debug-logging] ^
    [--endpoint-network-type { IPV4 | IPV6 | DUAL }] ^
    [--target-connection-network-type { IPV4 | IPV6 }] ^
    [--tags comma_separated_list]
```

Di seguito è riportato un esempio di valore JSON per l'opzione `--auth`. In questo esempio viene applicato un tipo di autenticazione client diverso per ciascun segreto.

```
[
  {
    "Description": "proxy description 1",
    "AuthScheme": "SECRETS",
    "SecretArn": "arn:aws:secretsmanager:us-
west-2:123456789123:secret/1234abcd-12ab-34cd-56ef-1234567890ab",
    "IAMAuth": "DISABLED",
    "ClientPasswordAuthType": "POSTGRES_SCRAM_SHA_256"
  },
```

```
  {
    "Description": "proxy description 2",
    "AuthScheme": "SECRETS",
    "SecretArn": "arn:aws:secretsmanager:us-
west-2:111122223333:secret/1234abcd-12ab-34cd-56ef-1234567890cd",
    "IAMAuth": "DISABLED",
    "ClientPasswordAuthType": "POSTGRES_MD5"

  },

  {
    "Description": "proxy description 3",
    "AuthScheme": "SECRETS",
    "SecretArn": "arn:aws:secretsmanager:us-
west-2:111122221111:secret/1234abcd-12ab-34cd-56ef-1234567890ef",
    "IAMAuth": "REQUIRED"
  }

]
```

Il parametro `--endpoint-network-type` specifica la versione IP dell'endpoint proxy utilizzato dai client per connettersi al proxy. I valori validi sono:

- `IPV4`— L'endpoint proxy utilizza solo IPv4 indirizzi (impostazione predefinita).
- `IPV6`— L'endpoint proxy utilizza solo IPv6 indirizzi.
- `DUAL`— L'endpoint proxy supporta entrambi IPv4 gli IPv6 indirizzi.

Il parametro `--target-connection-network-type` specifica la versione IP utilizzata dal proxy per connettersi al database di destinazione. I valori validi sono:

- `IPV4`— Il proxy si connette al database utilizzando IPv4 gli indirizzi (impostazione predefinita).
- `IPV6`— Il proxy si connette al database utilizzando IPv6 gli indirizzi.

Per utilizzare tipi IPv6 di rete endpoint dual-stack, il VPC e le sottoreti devono essere configurati per supportare il tipo di rete selezionato. Per utilizzare il tipo di rete con connessione IPv6 di destinazione, il database deve supportare la modalità dual-stack.

> **Tip**
>
> Se non conosci già la sottorete IDs da utilizzare per il `--vpc-subnet-ids` parametro, consulta Configurazione dei prerequisiti di rete per Server proxy per RDS alcuni esempi su come trovarli.

> **Note**
>
> Il gruppo di protezione deve consentire l'accesso al database a cui si connette il proxy. Lo stesso gruppo di protezione viene utilizzato per l'ingresso dalle applicazioni al proxy e per l'uscita dal proxy al database. Si supponga, ad esempio, di utilizzare lo stesso gruppo di protezione per il database e il proxy. In questo caso, assicurarsi di specificare che le risorse di tale gruppo di protezione possono comunicare con altre risorse dello stesso gruppo di protezione.
>
> Quando si utilizza un VPC condiviso, non è possibile utilizzare il gruppo di sicurezza predefinito per il VPC o uno appartenente a un altro account. Scegli un gruppo di sicurezza appartenente all'account. Se non esiste, creane uno. Per ulteriori informazioni su questa limitazione, consulta Work with shared VPCs.

Per creare le associazioni corrette per il proxy, si usa anche il register-db-proxy-targetscomando. Specificare il tipo di gruppo di destinazione `default` RDS Proxy crea automaticamente un gruppo di destinazione con questo nome quando si crea ogni proxy.

```
aws rds register-db-proxy-targets
    --db-proxy-name value
    [--target-group-name target_group_name]
    [--db-instance-identifiers space_separated_list]  # rds db instances, or
    [--db-cluster-identifiers cluster_id]        # rds db cluster (all instances)
```

API RDS

Per creare un proxy RDS, chiama l'operazione API Amazon RDS Create. DBProxy Passi un parametro con la struttura dei AuthConfigdati.

RDS Proxy crea automaticamente un gruppo di destinazione denominato `default` quando si crea ogni proxy. È possibile associare un cluster Aurora al gruppo di destinazione chiamando la funzione DBProxy Register Targets.

Important

Quando si seleziona l'autenticazione IAM per lo schema di autenticazione predefinito:

- È necessario abilitare l'autenticazione del database IAM sulle istanze o sui cluster del database di destinazione prima che il proxy possa connettersi correttamente.
- Se scegli Crea il ruolo IAM, è obbligatorio il campo Database accounts for IAM authentication.
- Se selezioni un ruolo IAM esistente, la console non aggiorna automaticamente il ruolo con le autorizzazioni di connessione al database. Verifica che il ruolo disponga delle `rds-db:connect` autorizzazioni necessarie.

Visualizzazione di un proxy

Dopo aver creato uno o più proxy RDS, puoi visualizzarli e gestirli in Console di gestione AWS, AWS CLI o nell'API di RDS. È possibile esaminarne i dettagli di configurazione, monitorare le prestazioni e determinare quali proxy modificare o eliminare in base alle esigenze.

Per consentire alle applicazioni di database di indirizzare il traffico attraverso un proxy, è necessario specificare l'endpoint proxy nella stringa di connessione.

Console

Per visualizzare un proxy nella console

1. Accedi alla Console di gestione AWS e apri la console Amazon RDS all'indirizzo https://console.aws.amazon.com/rds/.
2. Nel riquadro di navigazione scegli Proxies (Proxy).
3. Seleziona il nome del proxy per visualizzare i dettagli.
4. Nella pagina dei dettagli, la sezione Gruppi di destinazione mostra come il proxy è associato a un cluster di database Aurora specifico. È possibile accedere alla pagina del gruppo di destinazione predefinito per una visione più approfondita di questa associazione, comprese le impostazioni di configurazione definite durante la creazione del proxy. Queste impostazioni includono la percentuale massima di connessione, il timeout del prestito di connessione, la famiglia di motori e i filtri di associazione delle sessioni.

CLI

Per visualizzare il proxy utilizzando la CLI, utilizzare il comando describe-db-proxies. Per impostazione predefinita, vengono visualizzati tutti i proxy di proprietà dell'account AWS. Per visualizzare i dettagli di un singolo proxy, specificarne il nome con il parametro `--db-proxy-name`.

```
aws rds describe-db-proxies [--db-proxy-name proxy_name]
```

Per visualizzare altre informazioni associate al proxy, utilizza questi comandi:

```
aws rds describe-db-proxy-target-groups  --db-proxy-name proxy_name

aws rds describe-db-proxy-targets --db-proxy-name proxy_name
```

Utilizza la seguente sequenza di comandi per visualizzare ulteriori dettagli sugli elementi associati al proxy:

1. Per ottenere un elenco di proxy, eseguire describe-db-proxies.
2. Per visualizzare i parametri di connessione, ad esempio la percentuale massima di connessioni che il proxy può utilizzare, esegui describe-db-proxy-target-groups `--db-proxy-name`. Utilizza il nome del proxy come valore del parametro.
3. Per visualizzare i dettagli del cluster di database Aurora associato al gruppo di destinazione restituito, esegui describe-db-proxy-targets.

API RDS

Per visualizzare i proxy utilizzando l'API RDS, utilizza l'operazione DescribedBProxies . Restituisce valori del tipo di dati DbProxy .

Puoi utilizzare gli identificatori proxy dal valore restituito per visualizzare i dettagli delle impostazioni di connessione per il proxy utilizzando l'operazione DescribeDBProxyTargetGroups. Restituisce valori del tipo di dati DbProxyTargetGroup .

Puoi visualizzare l'istanza RDS o il cluster di database Aurora associato al proxy utilizzando l'operazione DescribeDBProxyTargets. Restituisce valori del tipo di dati DbProxyTarget .

Connessione a un database tramite RDS Proxy

In linea generale, si stabilisce una connessione a cluster database Aurora o a un cluster che utilizza Aurora Serverless v2 tramite un proxy allo stesso modo in cui ci si connette direttamente al database. La differenza principale è che si specifica l'endpoint del proxy anziché l'endpoint del cluster. Per impostazione predefinita, tutte le connessioni proxy dispongono read/write di funzionalità e utilizzano l'istanza writer. Se normalmente utilizzi l'endpoint di lettura per connessioni di sola lettura, puoi creare un endpoint aggiuntivo di sola lettura per il proxy e usare l'endpoint allo stesso modo. Per ulteriori informazioni, consulta Panoramica degli endpoint proxy.

Argomenti

- Connessione a un database utilizzando le credenziali del database
- Connessione a un database tramite l'autenticazione IAM
- Considerazioni sulla connessione a PostgreSQL

Connessione a un database utilizzando le credenziali del database

Utilizza i seguenti passaggi per connetterti a un proxy utilizzando le credenziali del database:

1. Trova l'endpoint del proxy. In Console di gestione AWS, puoi trovare l'endpoint nella pagina dei dettagli del proxy corrispondente. Con AWS CLI, è possibile utilizzare il describe-db-proxiescomando. L'esempio seguente mostra come.

```
# Add --output text to get output as a simple tab-separated list.
$ aws rds describe-db-proxies --query '*[*].
{DBProxyName:DBProxyName,Endpoint:Endpoint}'
[
    [
        {
            "Endpoint": "the-proxy.proxy-demo.us-east-1.rds.amazonaws.com",
            "DBProxyName": "the-proxy"
        },
        {
            "Endpoint": "the-proxy-other-secret.proxy-demo.us-
east-1.rds.amazonaws.com",
            "DBProxyName": "the-proxy-other-secret"
        },
        {
```

```
            "Endpoint": "the-proxy-rds-secret.proxy-demo.us-
east-1.rds.amazonaws.com",
            "DBProxyName": "the-proxy-rds-secret"
        },
        {
            "Endpoint": "the-proxy-t3.proxy-demo.us-east-1.rds.amazonaws.com",
            "DBProxyName": "the-proxy-t3"
        }
    ]
]
```

2. Specificare l'endpoint come parametro host nella stringa di connessione per l'applicazione client. Ad esempio, specificare l'endpoint proxy come valore per l'opzione `mysql -h` o l'opzione `psql -h`.
3. Fornisci lo stesso nome utente e la stessa password del database come fai normalmente.

Connessione a un database tramite l'autenticazione IAM

Quando utilizzi l'autenticazione IAM con RDS Proxy, hai due opzioni per l'autenticazione tra il client e il proxy:

- Configura gli utenti del database in modo che si autentichino con nomi utente e password normali. RDS Proxy recupera le credenziali del nome utente e della password da Secrets Manager. La connessione da RDS Proxy al database sottostante non passa tramite IAM.
- È inoltre possibile utilizzare l'autenticazione end-to-end IAM, che si connette al database tramite il proxy utilizzando IAM senza richiedere le credenziali del database.

Per connetterti a Server proxy per RDS con l'autenticazione IAM, utilizza la stessa procedura di connessione generale utilizzata per l'autenticazione IAM con un cluster di database Aurora. Per informazioni generali sull'utilizzo di IAM, consulta Sicurezza in Amazon Aurora. Se utilizzi l'autenticazione end-to-end IAM, fornisci il plug-in di autenticazione IAM al tuo utente DB. Per informazioni, consulta Creazione di un account database tramite l'autenticazione IAM.

Le principali differenze nell'utilizzo di IAM per RDS Proxy includono le seguenti:

- Con l'autenticazione IAM standard, gli utenti del database dispongono di credenziali regolari all'interno del database. Si impostano i segreti Secrets Manager contenenti questi nomi utente e password e si autorizza RDS Proxy a recuperare le credenziali da Secrets Manager. L'autenticazione IAM si applica alla connessione tra il programma client e il proxy. Il proxy esegue

quindi l'autenticazione nel database utilizzando il nome utente e le credenziali della password recuperate da Secrets Manager.

- Con l'autenticazione end-to-end IAM, non è necessario configurare i segreti di Secrets Manager per le credenziali del database. L'autenticazione IAM si applica alla connessione tra il client al proxy e il proxy al database.
- Invece dell'istanza, del cluster o dell'endpoint dell'istanza di lettura, specifica l'endpoint del proxy. Per informazioni dettagliate sull'endpoint proxy, vedere Connessione al cluster di database tramite l'autenticazione IAM.
- Quando ci si connette a un proxy utilizzando l'autenticazione IAM è necessario utilizzare Transport Layer Security (TLS)/Secure Sockets Layer (SSL).

È possibile concedere a un utente specifico l'accesso al proxy modificando la policy IAM. Di seguito è riportato un esempio.

```
"Resource": "arn:aws:rds-db:us-east-2:1234567890:dbuser:prx-ABCDEFGHIJKL01234/db_user"
```

> **Tip**
>
> Quando configuri l'autenticazione IAM per le connessioni proxy RDS, segui queste importanti linee guida per evitare problemi di connessione:
>
> - Non concedete il `rds_iam` ruolo mantenendo l'autenticazione generale con password per lo stesso utente o ruolo del database.
> - Ricorda che mentre i client si connettono al proxy RDS utilizzando l'autenticazione IAM, il proxy RDS si connette sempre al database utilizzando l'autenticazione tramite password tramite Secrets Manager.
> - Se si verificano frequenti interruzioni e riconnessioni della connessione, rimuovi eventuali `rds_iam` concessioni esistenti dall'utente o dal ruolo e utilizza solo l'autenticazione tramite password.
> - Assicurati che la tua politica in materia di password soddisfi i requisiti relativi ai caratteri sicuri SCRAM-SHA-256.
>
> La combinazione di metodi di autenticazione IAM e password per lo stesso utente del database può causare instabilità della connessione.

Considerazioni sulla connessione a PostgreSQL

Se si crea un nuovo utente del database PostgreSQL per la connessione a Server proxy per RDS, assicurarsi di concedere all'utente il privilegio `CONNECT` sul database. Senza questo privilegio, l'utente non può stabilire una connessione. Per ulteriori informazioni, consulta the section called "Aggiungere un nuovo utente del database a un database PostgreSQL quando si utilizza Server proxy per RDS".

Quando un client avvia una connessione a un database PostgreSQL, invia un messaggio di avvio che include coppie di stringhe di nome parametro e valore. Per i dettagli, vedere i `StartupMessage` in Formati dei messaggi PostgreSQL nella documentazione di PostgreSQL.

Quando si effettua la connessione tramite un proxy RDS, il messaggio di avvio può includere i seguenti parametri attualmente riconosciuti:

- `user`
- `database`

Il messaggio di avvio può includere anche i seguenti parametri di runtime aggiuntivi:

- `application_name`
- `client_encoding`
- `DateStyle`
- `TimeZone`
- `extra_float_digits`
- `search_path`

Per ulteriori informazioni sulla messaggistica PostgreSQL, vedere Frontend/Backend Protocol nella documentazione di PostgreSQL.

Per PostgreSQL, se si utilizza JDBC, è consigliabile effettuare quanto segue per evitare il pinning:

- Impostare il parametro `assumeMinServerVersion` di connessione JDBC su almeno `9.0` per evitare il pinning. In questo modo si impedisce al driver JDBC di eseguire un ulteriore round trip durante l'avvio della connessione quando viene eseguito `SET extra_float_digits = 3`.
- Impostare il parametro di connessione JDBC `ApplicationName` su *`any/your-application-name`* per evitare il pinning. In questo modo si impedisce al driver JDBC di eseguire un ulteriore

round trip durante l'avvio della connessione quando viene eseguito `SET application_name = "PostgreSQL JDBC Driver"`. Si noti che il parametro JDBC è `ApplicationName` ma il parametro `StartupMessage` PostgreSQL è `application_name`.

Per ulteriori informazioni, consulta Evitare di effettuare il pinning di un Server proxy per RDS.
Per ulteriori informazioni sulla connessione tramite JDBC, vedere Connessione al database nella documentazione di PostgreSQL.

Gestire un RDS Proxy

Questa sezione fornisce informazioni su come gestire il funzionamento e la configurazione di Server proxy per RDS. Queste procedure consentono all'applicazione di utilizzare in modo più efficiente le connessioni al database e di ottenere il massimo riutilizzo della connessione. Più sfrutti il riutilizzo della connessione, maggiore sarà il risparmio in termini di sovraccarico di CPU e memoria. Questo a sua volta riduce la latenza per l'applicazione e consente al database di dedicare più risorse all'elaborazione delle richieste dell'applicazione.

Argomenti

- Modifica di un RDS Proxy
- Aggiunta di un nuovo utente di database quando si utilizza Server proxy per RDS
- Passaggio dall'autenticazione IAM standard all'autenticazione end-to-end IAM per RDS Proxy
- Considerazioni sulla connessione a Server proxy per RDS
- Evitare di effettuare il pinning di un Server proxy per RDS
- Eliminazione di un RDS Proxy

Modifica di un RDS Proxy

Puoi modificare specifiche impostazioni associate a un proxy dopo aver creato il proxy. Esegui questa operazione modificando il proxy stesso, il suo gruppo di destinazione associato o entrambi. Ogni proxy ha un gruppo di destinazione associato.

Console di gestione AWS

> **Important**
>
> I valori nei campi Client authentication type (Tipo di autenticazione client) e IAM authentication (autenticazione IAM) si applicano a tutti i segreti di Secrets Manager associati al proxy. Per specificare valori diversi per ogni segreto, modifica il proxy utilizzando invece l'API AWS CLI o.

Per modificare le impostazioni di un proxy

1. Accedi a Console di gestione AWS e apri la console Amazon RDS all'indirizzo https://console.aws.amazon.com/rds/.
2. Nel riquadro di navigazione scegli Proxies (Proxy).
3. Nell'elenco dei proxy, scegli il proxy di cui desideri modificare le impostazioni o passa alla relativa pagina dei dettagli.
4. Per Actions (Operazioni), scegliere Modify (Modifica).
5. Inserisci o scegli le proprietà da modificare. È possibile modificare le seguenti:

   - Identificatore Proxy: rinominare il proxy immettendo un nuovo identificatore.
   - Timeout della connessione client per inattività: immettere un periodo di tempo per il timeout della connessione client inattiva.
   - Ruolo IAM: modificare il ruolo IAM utilizzato per recuperare i segreti da Secrets Manager.

     > **Note**
     >
     > Non puoi creare un nuovo ruolo IAM se imposti lo schema di autenticazione predefinito sull'autenticazione IAM.

   - Segreti di Secrets Manager: aggiungere o rimuovere segreti di Secrets Manager. Questi segreti corrispondono a nomi utente e password del database.
   - Tipo di autenticazione client: modifica il tipo di autenticazione per le connessioni dei client al proxy.
   - Autenticazione IAM: richiedi o disabilita l'autenticazione IAM per le connessioni al proxy.

- Schema di autenticazione predefinito: modifica lo schema di autenticazione predefinito utilizzato dal proxy per le connessioni dei client al proxy e le connessioni dal proxy al database sottostante.
- Richiedi Transport Layer Security: attivare o disattivare il requisito per Transport Layer Security (TLS).
- Gruppo di sicurezza VPC: aggiungere o rimuovere gruppi di sicurezza VPC da utilizzare per il proxy.
- Abilitazione della registrazione avanzata: abilitare o disabilitare la registrazione avanzata.

6. Scegliere Modify (Modifica).

Se non sono state trovate le impostazioni elencate che si desidera modificare, attenersi alla procedura seguente per aggiornare il gruppo di destinazione per il proxy. Il gruppo di destinazione associato a un proxy controlla le impostazioni relative alle connessioni del database fisico. Ogni proxy ha un gruppo di destinazione associato denominato `default`, che viene creato automaticamente insieme al proxy. Non è possibile rinominare il gruppo di destinazione predefinito.

Puoi modificare il gruppo di destinazione solo dalla pagina dei dettagli del proxy, non dall'elenco nella pagina Proxy .

Per modificare le impostazioni di un gruppo di destinazione proxy

1. Dalla pagina Proxy, passa alla pagina dei dettagli di un proxy.
2. Per Gruppi di destinazione, scegli il collegamento di `default`. Attualmente, tutti i proxy hanno un singolo gruppo di destinazione denominato `default`.
3. Nella pagina dei dettagli del gruppo di destinazione di default, scegli Modify (Modifica).
4. Scegli nuove impostazioni per le proprietà che è possibile modificare:

   - Database: puoi scegliere un diverso cluster Aurora.
   - Connessioni massime del pool di connessioni: puoi modificare la percentuale delle connessioni massime disponibili che il proxy può utilizzare.
   - Filtri di pinning della sessione: (opzionale) scegliere un filtro di pinning della sessione. Ciò elude le misure di sicurezza predefinite per il multiplexing delle connessioni al database tra connessioni client. Attualmente, l'impostazione non è supportata per PostgreSQL. L'unica scelta è `EXCLUDE_VARIABLE_SETS`.

L'abilitazione di questa impostazione può far sì che le variabili di sessione di una connessione interessino le altre connessioni. Ciò può causare errori o problemi di correttezza se le query dipendono dai valori delle variabili di sessione impostati al di fuori della transazione corrente. Valuta l'utilizzo di questa opzione dopo aver verificato che sia sicuro per le applicazioni condividere le connessioni al database tra connessioni client.

I seguenti modelli possono essere considerati sicuri:

- Istruzioni SET in cui non viene apportata alcuna modifica al valore della variabile di sessione effettiva, ovvero non viene apportata alcuna modifica alla variabile di sessione.
- Modifichi il valore della variabile di sessione ed esegui un'istruzione nella stessa transazione.

Per ulteriori informazioni, consulta Evitare di effettuare il pinning di un Server proxy per RDS.

- Timeout del prestito di connessione: puoi regolare l'intervallo di timeout del prestito di connessione. Questa impostazione si applica quando il numero massimo di connessioni è già in uso per il proxy. In questi casi, è possibile specificare per quanto tempo il proxy attende che una connessione diventi disponibile prima di restituire un errore di timeout.
- Query di inizializzazione. (Opzionale) Aggiungi una query di inizializzazione o modifica quella corrente. Puoi specificare una o più istruzioni SQL per l'esecuzione del proxy all'apertura di ogni nuova connessione al database. L'impostazione è in genere utilizzata con le istruzioni SET per assicurarsi che ogni connessione abbia impostazioni identiche. Assicurati che la query che aggiungi sia valida. Per includere più variabili in una singola istruzione SET, utilizza la virgola come separatore. Ad esempio:

```
SET variable1=value1, variable2=value2
```

Per più istruzioni, utilizzare il punto e virgola come separatore.

Alcune proprietà, ad esempio l'identificatore del gruppo di destinazione e il motore del database, sono fisse.

5. Scegli Modify target group (Modifica gruppo di destinazione).

AWS CLI

Per modificare un proxy utilizzando il AWS CLI, usa i comandi modify-db-proxymodify-db-proxy-target-group deregister-db-proxy-targetse register-db-proxy-targets.

Con il comando `modify-db-proxy`, è possibile modificare proprietà come le seguenti:

- L'insieme di segreti Secrets Manager usati dal proxy.
- Indica se TLS è necessario.
- Il timeout del client inattivo.
- Indica se registrare ulteriori informazioni dalle istruzioni SQL per il debug.
- Il ruolo IAM utilizzato per recuperare i segreti Secrets Manager.
- I gruppi di sicurezza utilizzati dal proxy.
- Lo schema di autenticazione predefinito associato al proxy.

Nell'esempio seguente viene illustrato come rinominare un proxy esistente.

```
aws rds modify-db-proxy --db-proxy-name the-proxy --new-db-proxy-name the_new_name
```

Con il comando `modify-db-proxy-target-group`, puoi modificare le impostazioni relative alla connessione o rinominare il gruppo di destinazione. Attualmente, tutti i proxy hanno un singolo gruppo di destinazione denominato `default`. Quando utilizzi questo gruppo di destinazione, devi specificare il nome del proxy e `default` come nome del gruppo di destinazione. Non è possibile rinominare il gruppo di destinazione predefinito.

Nell'esempio seguente viene illustrato come controllare prima l'impostazione `MaxIdleConnectionsPercent` per un proxy e quindi modificarla utilizzando il gruppo di destinazione.

```
aws rds describe-db-proxy-target-groups --db-proxy-name the-proxy

{
    "TargetGroups": [
        {
            "Status": "available",
            "UpdatedDate": "2019-11-30T16:49:30.342Z",
            "ConnectionPoolConfig": {
                "MaxIdleConnectionsPercent": 50,
                "ConnectionBorrowTimeout": 120,
```

```
                "MaxConnectionsPercent": 100,
                "SessionPinningFilters": []
            },
            "TargetGroupName": "default",
            "CreatedDate": "2019-11-30T16:49:27.940Z",
            "DBProxyName": "the-proxy",
            "IsDefault": true
        }
    ]
}

aws rds modify-db-proxy-target-group --db-proxy-name the-proxy --target-group-name
 default --connection-pool-config '
{ "MaxIdleConnectionsPercent": 75 }'

{
    "DBProxyTargetGroup": {
        "Status": "available",
        "UpdatedDate": "2019-12-02T04:09:50.420Z",
        "ConnectionPoolConfig": {
            "MaxIdleConnectionsPercent": 75,
            "ConnectionBorrowTimeout": 120,
            "MaxConnectionsPercent": 100,
            "SessionPinningFilters": []
        },
        "TargetGroupName": "default",
        "CreatedDate": "2019-11-30T16:49:27.940Z",
        "DBProxyName": "the-proxy",
        "IsDefault": true
    }
}
```

Con i comandi `deregister-db-proxy-targets` e `register-db-proxy-targets`, puoi modificare i cluster di database Aurora a cui il proxy è associato tramite il relativo gruppo di destinazione. Attualmente, ogni proxy può connettersi a un cluster di database Aurora. Il gruppo di destinazione tiene traccia dei dettagli di connessione per tutte le istanze database in un cluster Aurora.

L'esempio seguente inizia con un proxy associato a un cluster Aurora MySQL denominato `cluster-56-2020-02-25-1399`. Nell'esempio viene illustrato come modificare il proxy in modo che possa connettersi a un cluster diverso denominato `provisioned-cluster`.

Quando utilizzi un cluster di database Aurora, specifichi l'opzione `--db-cluster-identifier`.

L'esempio seguente modifica un proxy Aurora MySQL. Un proxy Aurora PostgreSQL ha la porta 5432.

```
aws rds describe-db-proxy-targets --db-proxy-name the-proxy

{
    "Targets": [
        {
            "Endpoint": "instance-9814.demo.us-east-1.rds.amazonaws.com",
            "Type": "RDS_INSTANCE",
            "Port": 3306,
            "RdsResourceId": "instance-9814"
        },
        {
            "Endpoint": "instance-8898.demo.us-east-1.rds.amazonaws.com",
            "Type": "RDS_INSTANCE",
            "Port": 3306,
            "RdsResourceId": "instance-8898"
        },
        {
            "Endpoint": "instance-1018.demo.us-east-1.rds.amazonaws.com",
            "Type": "RDS_INSTANCE",
            "Port": 3306,
            "RdsResourceId": "instance-1018"
        },
        {
            "Type": "TRACKED_CLUSTER",
            "Port": 0,
            "RdsResourceId": "cluster-56-2020-02-25-1399"
        },
        {
            "Endpoint": "instance-4330.demo.us-east-1.rds.amazonaws.com",
            "Type": "RDS_INSTANCE",
            "Port": 3306,
            "RdsResourceId": "instance-4330"
        }
    ]
}

aws rds deregister-db-proxy-targets --db-proxy-name the-proxy --db-cluster-identifier
 cluster-56-2020-02-25-1399

aws rds describe-db-proxy-targets --db-proxy-name the-proxy
```

```
{
    "Targets": []
}

aws rds register-db-proxy-targets --db-proxy-name the-proxy --db-cluster-identifier
 provisioned-cluster

{
    "DBProxyTargets": [
        {
            "Type": "TRACKED_CLUSTER",
            "Port": 0,
            "RdsResourceId": "provisioned-cluster"
        },
        {
            "Endpoint": "gkldje.demo.us-east-1.rds.amazonaws.com",
            "Type": "RDS_INSTANCE",
            "Port": 3306,
            "RdsResourceId": "gkldje"
        },
        {
            "Endpoint": "provisioned-1.demo.us-east-1.rds.amazonaws.com",
            "Type": "RDS_INSTANCE",
            "Port": 3306,
            "RdsResourceId": "provisioned-1"
        }
    ]
}
```

API RDS

Per modificare un proxy utilizzando l'API RDS, si utilizzano le operazioni Modify, Modify DBProxy DBProxyTargetGroup, Deregister DBProxy Targets e DBProxyRegister Targets.

Con `ModifyDBProxy`, è possibile modificare proprietà come le seguenti:

- L'insieme di segreti Secrets Manager usati dal proxy.
- Indica se TLS è necessario.
- Il timeout del client inattivo.
- Indica se registrare ulteriori informazioni dalle istruzioni SQL per il debug.
- Il ruolo IAM utilizzato per recuperare i segreti Secrets Manager.

- I gruppi di sicurezza utilizzati dal proxy.

Con `ModifyDBProxyTargetGroup`, puoi modificare le impostazioni relative alla connessione. Attualmente, tutti i proxy hanno un singolo gruppo di destinazione denominato `default`. Quando utilizzi questo gruppo di destinazione, devi specificare il nome del proxy e `default` come nome del gruppo di destinazione. Non è possibile rinominare il gruppo di destinazione predefinito.

Con `DeregisterDBProxyTargets` e `RegisterDBProxyTargets`, puoi modificare il cluster Aurora a cui il proxy è associato tramite il relativo gruppo di destinazione. Attualmente, ogni proxy può connettersi a un cluster Aurora. Il gruppo di destinazione tiene traccia dei dettagli di connessione per le istanze database in un cluster Aurora.

Aggiunta di un nuovo utente di database quando si utilizza Server proxy per RDS

In alcuni casi, puoi aggiungere un nuovo utente di database a un cluster Aurora associato a un proxy. Procedi a seconda che tu stia utilizzando l'autenticazione standard con i segreti di Secrets Manager o l'autenticazione end-to-end IAM.

Se utilizzi l'autenticazione IAM standard, segui queste istruzioni:

1. Crea un nuovo segreto Secrets Manager, utilizzando la procedura descritta in Impostazione delle credenziali del database per RDS Proxy.
2. Aggiornare il ruolo IAM per consentire a RDS Proxy l'accesso al nuovo segreto Secrets Manager. A tale scopo, aggiornare la sezione Risorse della policy di ruolo IAM.
3. Modifica il proxy RDS per aggiungere il nuovo segreto di Secrets Manager nell'area Segreti Secrets Manager.
4. Se il nuovo utente sostituisce un utente esistente, aggiorna le credenziali memorizzate nel segreto Secrets Manager del proxy per l'utente esistente.

Se utilizzi l'autenticazione end-to-end IAM, devi creare l'utente del database e configurare le autorizzazioni IAM. Per eseguire questa operazione, segui questa procedura:

1. Crea un nuovo utente del database nel tuo database che corrisponda al nome utente o ruolo IAM che desideri utilizzare per l'autenticazione.

2. Assicurati che l'utente del database sia configurato con il plug-in di autenticazione IAM nel database. Per informazioni, consulta Creazione di un account database tramite l'autenticazione IAM.
3. Aggiorna la policy IAM per concedere l'`rds-db:connect`autorizzazione all'utente o al ruolo IAM, come descritto inCreazione di una policy IAM per l'autenticazione end-to-end IAM.
4. Assicurati che il proxy sia configurato per utilizzare l'autenticazione IAM come schema di autenticazione predefinito.

Con l'autenticazione end-to-end IAM, non è necessario gestire le credenziali del database nei segreti di Secrets Manager, poiché le credenziali IAM vengono utilizzate per l'autenticazione dal client al proxy e dal proxy al database.

Aggiungere un nuovo utente del database a un database PostgreSQL quando si utilizza Server proxy per RDS

Quando aggiungi un nuovo utente al database PostgreSQL, se hai eseguito il seguente comando:

```
REVOKE CONNECT ON DATABASE postgres FROM PUBLIC;
```

Concedi all'utente `rdsproxyadmin` il privilegio `CONNECT` in modo che possa monitorare le connessioni sul database di destinazione.

```
GRANT CONNECT ON DATABASE postgres TO rdsproxyadmin;
```

È anche possibile consentire ad altri utenti del database di destinazione di eseguire controlli dell'integrità passando `rdsproxyadmin` all'utente del database nel comando precedente.

Modifica della password per un utente di database quando si utilizza Server proxy per RDS

In alcuni casi, puoi modificare la password per un utente di database in un cluster Aurora associato a un proxy. In tal caso, aggiorna il segreto Secrets Manager corrispondente con la nuova password.

Se utilizzi l'autenticazione end-to-end IAM, non devi aggiornare alcuna password nei segreti di Secrets Manager.

Passaggio dall'autenticazione IAM standard all'autenticazione end-to-end IAM per RDS Proxy

Se attualmente utilizzi l'autenticazione IAM standard per RDS Proxy, in cui i client si autenticano sul proxy utilizzando IAM ma il proxy si connette al database tramite segreti, puoi migrare all'autenticazione end-to-end IAM in cui entrambe le client-to-proxy proxy-to-database connessioni utilizzano l'autenticazione IAM.

Per passare all'autenticazione IAM end-to-end

1. Aggiorna le autorizzazioni del ruolo IAM del proxy RDS

   Crea una politica di autorizzazione proxy aggiornata che includa sia Secrets Manager che `rds:db-connect` le autorizzazioni:

```
# Create updated proxy permission policy
cat > updated-proxy-policy.json # EOF
```

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "GetSecretsValue",
      "Action": [
        "secretsmanager:GetSecretValue"
      ],
      "Effect": "Allow",
      "Resource": [
        "arn:aws:secretsmanager:us-east-1:123456789012:secret:secretName-1234f"
      ]
    },
    {
      "Sid": "RdsDBConnect",
      "Action": [
        "rds-db:connect"
      ],
      "Effect": "Allow",
      "Resource": [
        "arn:aws:rds-db:us-east-1:123456789012:dbuser:cluster-ABCDEFGHIJKL01234/
jane_doe"
      ]
```

```
    }
  ]
}
```

Aggiorna la politica relativa al tuo ruolo:

```
aws iam put-role-policy \
            --role-name RDSProxyRole \
            --policy-name UpdatedProxyPermissions \
            --policy-document file://updated-proxy-policy.json
```

2. Modifica il tuo proxy RDS per abilitare l'autenticazione end-to-end IAM

```
aws rds modify-db-proxy \
  --db-proxy-name my-database-proxy \
  --default-auth-scheme IAM_AUTH \
  --region us-east-1
```

Verifica che lo stato del proxy RDS sia Disponibile e lo `DefaultAuthScheme` sia `IAM_AUTH` prima di procedere per garantire zero tempi di inattività durante la migrazione.

```
aws rds describe-db-proxies --db-proxy-name my-database-proxy --region us-east-1
```

Output previsto:

```
{
  "DBProxies": [
    {
      "DBProxyName": "my-database-proxy",
      "DBProxyArn": "arn:aws:rds:us-east-1:123456789012:db-
proxy:prx-0123456789abcdef",
      "Status": "available",
      ...
      "DefaultAuthScheme": "IAM_AUTH"
    }
  ]
}
```

3. Abilita l'autenticazione IAM sul database

```
aws rds modify-db-cluster \
```

```
  --db-cluster-identifier my-database-cluster \
  --enable-iam-database-authentication \
  --region us-east-1
```

4. Configura l'utente del database per l'autenticazione IAM

   Per Aurora PostgreSQL:

   ```
   GRANT rds_iam TO jane_doe;
   ```

   Per Aurora MySQL:

   ```
   ALTER USER 'jane_doe' IDENTIFIED WITH AWSAuthenticationPlugin AS 'RDS';
   ALTER USER 'jane_doe'@'%' REQUIRE SSL;
   ```

5. Non è necessario modificare il codice dell'applicazione client. Il processo di connessione rimane lo stesso:

   Per Aurora PostgreSQL:

   ```
   # Generate authentication token
   export PGPASSWORD=$(aws rds generate-db-auth-token \
     --hostname my-database-proxy.proxy-ABCDEFGHIJKL01234.us-east-1.rds.amazonaws.com
    \
     --port 5432 \
     --username jane_doe \
     --region us-east-1)

   # Connect to database through proxy
   psql "host=my-database-proxy.proxy-ABCDEFGHIJKL01234.us-east-1.rds.amazonaws.com
    port=5432 user=jane_doe dbname=postgres password=$PGPASSWORD sslmode=require
    sslrootcert=us-east-1-bundle.pem"
   ```

   Per Aurora MySQL:

   ```
   # Generate authentication token
   export MYSQL_PWD=$(aws rds generate-db-auth-token \
     --hostname my-database-proxy.proxy-ABCDEFGHIJKL01234.us-east-1.rds.amazonaws.com
    \
     --port 3306 \
     --username jane_doe \
     --region us-east-1)
   ```

```
# Connect to database through proxy
mysql -h my-database-proxy.proxy-ABCDEFGHIJKL01234.us-east-1.rds.amazonaws.com \
  -P 3306 \
  -u jane_doe \
  --ssl-ca=us-east-1-bundle.pem \
  --enable-cleartext-plugin
```

Considerazioni sulla connessione a Server proxy per RDS

Configurazione delle impostazioni di connessione

Per regolare il pooling di connessioni del proxy RDS, è possibile modificare le seguenti impostazioni:

- IdleClientTimeout
- MaxConnectionsPercent
- MaxIdleConnectionsPercent
- ConnectionBorrowTimeout

IdleClientTimeout

È possibile specificare per quanto tempo una connessione client può essere inattiva prima che il proxy la chiuda. Il valore predefinito è 1.800 secondi (30 minuti).

Una connessione client è considerata inattiva quando l'applicazione non invia una nuova richiesta entro il tempo specificato dopo il completamento della richiesta precedente. La connessione al database sottostante rimane aperta e viene restituita al pool di connessioni. Pertanto, è disponibile per essere riutilizzata per nuove connessioni client. Se si desidera che il proxy rimuova in modo proattivo le connessioni obsolete, è necessario ridurre il timeout della connessione client inattiva. Se il carico di lavoro stabilisce connessioni frequenti con il proxy, è possibile aumentare il timeout della connessione client inattiva per risparmiare sui costi di creazione delle connessioni.

Questa impostazione è rappresentata dal timeout della connessione client per inattività nella console RDS e nell'impostazione `IdleClientTimeout` nella AWS CLI e nell'API. Per informazioni su come modificare il valore del campo Idle client connection timeout (Timeout di connessione client inattivo) nella console RDS, consulta Console di gestione AWS. Per informazioni su come modificare il valore dell'impostazione `IdleClientTimeout`, consulta il comando della CLI modify-db-proxy o l'operazione API ModifyDBProxy.

MaxConnectionsPercent

Puoi limitare il numero di connessioni che un proxy RDS può stabilire con il database di destinazione. Puoi specificare il limite come percentuale delle connessioni massime disponibili per il tuo database. Questa impostazione è rappresentata dal campo Connection pool maximum connection (Connessioni massime del pool di connessioni) nella console RDS e nell'impostazione `MaxConnectionsPercent` della AWS CLI o dell'API.

Il valore `MaxConnectionsPercent` viene espresso come percentuale dell'impostazione `max_connections` per il cluster di database Aurora usatocluster dal gruppo di destinazione. Il proxy non crea tutte queste connessioni in anticipo. Questa impostazione consente al proxy di stabilire queste connessioni in base al carico di lavoro necessario.

Ad esempio, per una destinazione di database registrata con il parametro `max_connections` impostato su 1000 e il parametro `MaxConnectionsPercent` impostato su 95, RDS Proxy imposta 950 connessioni come limite massimo per le connessioni simultanee al database di destinazione specificato.

Un effetto collaterale comune del raggiungimento del numero massimo di connessioni al database consentite dal carico di lavoro è un aumento della latenza complessiva delle query e un incremento del valore della metrica `DatabaseConnectionsBorrowLatency`. È possibile monitorare le connessioni al database attualmente utilizzate e il totale consentito confrontando le metriche `DatabaseConnections` e `MaxDatabaseConnectionsAllowed`.

Se imposti questo parametri, segui le best practice riportate di seguito:

- Consenti un margine sufficiente per le connessioni per la modifica dello schema del carico di lavoro. Si consiglia di impostare il parametro su un valore superiore di almeno il 30% rispetto all'utilizzo massimo monitorato. Poiché RDS Proxy ridistribuisce le quote di connessione del database su più nodi, le modifiche alla capacità interna potrebbero richiedere un margine di almeno il 30% per connessioni aggiuntive per evitare l'incremento delle latenze di prestito.
- RDS Proxy riserva un certo numero di connessioni per il monitoraggio attivo per supportare il failover rapido, l'instradamento del traffico e le operazioni interne. Il parametro `MaxDatabaseConnectionsAllowed` non include queste connessioni riservate. Rappresenta il numero di connessioni disponibili per servire il carico di lavoro e può essere inferiore al valore derivato dall'impostazione `MaxConnectionsPercent`.

  I valori `MaxConnectionsPercent` minimi consigliati sono i seguenti:

  - db.t3.small: 100

- db.t3.medium: 55
- db.t3.large: 35
- db.r3.large o superiore: 20

Se più istanze di destinazione sono registrate con RDS Proxy come un cluster Aurora con nodi di lettura, imposta il valore minimo in base all'istanza registrata più piccola.

Per informazioni su come modificare il valore del campo Connection pool maximum connections (Numero massimo di connessioni del pool di connessioni) nella console RDS, consulta Console di gestione AWS. Per informazioni su come modificare il valore dell'impostazione `MaxConnectionsPercent`, consulta il comando della CLI modify-db-proxy-target-group o l'operazione API ModifyDBProxyTargetGroup.

> **Important**
>
> Se il cluster database fa parte di un database globale con l'inoltro di scrittura attivato, riduci il valore `MaxConnectionsPercent` del proxy con la quota assegnata all'inoltro di scrittura. La quota di inoltro di scrittura è impostata nel parametro `aurora_fwd_writer_max_connections_pct` del cluster database. Per informazioni sull'inoltro di scrittura, consulta Utilizzo dell'inoltro di scrittura in un database globale Amazon Aurora.

Per informazioni sui limiti di connessione al database, consulta Numero massimo di connessioni a un'istanza database Aurora MySQL e Numero massimo di connessioni a un'istanza database Aurora PostgreSQL.

MaxIdleConnectionsPercent

Puoi controllare il numero di connessioni al database inattive che RDS Proxy può mantenere nel pool di connessione. Per impostazione predefinita, Server proxy per RDS considera che una connessione al database nel proprio pool sia inattiva quando non c'è stata alcuna attività sulla connessione per cinque minuti.

Il valore `MaxIdleConnectionsPercent` viene espresso come percentuale dell'impostazione `max_connections` per il gruppo di destinazione dell'istanza database RDS. Il valore predefinito è 50% di `MaxConnectionsPercent` e il limite superiore è il valore di

`MaxConnectionsPercent`. Ad esempio, se `MaxConnectionsPercent` è 80, il valore predefinito di `MaxIdleConnectionsPercent` è 40.

Con un valore elevato, il proxy lascia aperta un'alta percentuale di connessioni al database inattive. Con un valore basso, il proxy chiude un'alta percentuale di connessioni al database inattive. Se i carichi di lavoro sono imprevedibili, valuta la possibilità di impostare un valore elevato per `MaxIdleConnectionsPercent`. In tal modo Server proxy per RDS può soddisfare i picchi di attività senza aprire molte nuove connessioni al database.

Questa impostazione è rappresentata dall'impostazione `MaxIdleConnectionsPercent` di `DBProxyTargetGroup` nella AWS CLI e nell'API. Per informazioni su come modificare il valore dell'impostazione `MaxIdleConnectionsPercent`, consulta il comando della CLI modify-db-proxy-target-group o l'operazione API ModifyDBProxyTargetGroup.

Per informazioni sui limiti di connessione al database, consulta Numero massimo di connessioni a un'istanza database Aurora MySQL e Numero massimo di connessioni a un'istanza database Aurora PostgreSQL.

ConnectionBorrowTimeout

Puoi specificare per quanto tempo RDS Proxy attende che una connessione di database nel pool di connessione diventi disponibile per l'uso prima di restituire un errore di timeout. Il valore predefinito è 120 secondi. Questa impostazione si applica quando il numero di connessioni è pari al massimo e quindi non sono disponibili connessioni nel pool di connessioni. Si applica anche quando non è disponibile alcuna istanza database appropriata per gestire la richiesta, ad esempio quando è in corso un'operazione di failover. Utilizzando questa impostazione, è possibile impostare il periodo di attesa migliore per l'applicazione senza modificare il timeout della query nel codice dell'applicazione.

Questa impostazione è rappresentata dal campo Connection borrow timeout (Timeout del prestito della connessione) nella console RDS o nell'impostazione `ConnectionBorrowTimeout` di `DBProxyTargetGroup` nella AWS CLI o nell'API. Per informazioni su come modificare il valore del campo Connection borrow timeout (Timeout del prestito della connessione) nella console RDS, consulta Console di gestione AWS. Per informazioni su come modificare il valore dell'impostazione `ConnectionBorrowTimeout`, consulta il comando della CLI modify-db-proxy-target-group o l'operazione API ModifyDBProxyTargetGroup.

Connessioni client e database

Le connessioni dall'applicazione a Server proxy per RDS sono note come connessioni client. Le connessioni da un proxy al database sono le connessioni database. Quando si utilizza Server proxy

per RDS, le connessioni client terminano sul proxy mentre le connessioni database vengono gestite all'interno di Server proxy per RDS.

Il pool di connessioni lato applicazione può offrire il vantaggio di ridurre la creazione di connessioni ricorrenti tra l'applicazione e Server proxy per RDS.

Considerare i seguenti aspetti di configurazione prima di implementare un pool di connessioni lato applicazione:

- Durata massima della connessione client: Server proxy per RDS applica una durata massima delle connessioni client di 24 ore. Questo valore non è configurabile. Configurare il pool con una durata massima della connessione inferiore a 24 ore per evitare interruzioni impreviste delle connessioni client.
- Timeout di inattività della connessione client: Server proxy per RDS applica un tempo di inattività massimo per le connessioni client. Configura il pool con un valore di timeout di inattività della connessione inferiore all'impostazione di timeout di inattività della connessione client per Server proxy per RDS per evitare interruzioni impreviste della connessione.

Il numero massimo di connessioni client configurate nel pool di connessioni lato applicazione non deve essere limitato all'impostazione max_connections per Server proxy per RDS.

Il pool delle connessioni dei client consente di prolungare la durata della connessione client. Se si verifica il pinning delle connessioni, il pool delle connessioni client può ridurre l'efficienza del multiplexing. Le connessioni client aggiunte ma inattive nel pool di connessioni lato applicazione continuano a mantenere la connessione al database e impediscono che venga riutilizzata da altre connessioni client. Esaminare i log del proxy per verificare se le connessioni presentano il pinning.

> **Note**
>
> RDS Proxy chiude le connessioni al database all'incirca dopo 24 ore quando non sono più in uso. Il proxy esegue questa operazione indipendentemente dal valore dell'impostazione massima delle connessioni inattive.

Evitare di effettuare il pinning di un Server proxy per RDS

Il multiplexing è più efficiente quando le richieste del database non si basano su informazioni di stato provenienti da richieste precedenti. In tal caso, RDS Proxy può riutilizzare una connessione

alla conclusione di ogni transazione. Esempi di tali informazioni sullo stato includono la maggior parte delle variabili e dei parametri di configurazione che puoi modificare attraverso le istruzioni `SET` o `SELECT`. Le transazioni SQL su una connessione client possono eseguire il multiplex tra le connessioni di database sottostanti per impostazione predefinita.

Le connessioni al proxy possono entrare in uno stato noto come pinning. Quando una connessione viene bloccata, ogni transazione successiva utilizza la stessa connessione al database sottostante fino al termine della sessione. Altre connessioni client, inoltre, non possono riutilizzare tale connessione al database fino al termine della sessione. La sessione termina quando viene interrotta la connessione client.

RDS Proxy collega automaticamente una connessione client a una specifica connessione DB quando rileva una modifica dello stato della sessione che non è appropriata per altre sessioni. Il pinning riduce l'efficacia del riutilizzo della connessione. Se tutte o quasi tutte le connessioni riscontrano il pinning, potresti modificare il codice dell'applicazione o il carico di lavoro per ridurre le condizioni che causano il blocco.

Ad esempio, l'applicazione modifica una variabile di sessione o un parametro di configurazione. In questo caso, le istruzioni successive possono basarsi sulla nuova variabile o sul nuovo parametro per essere effettive. Pertanto, quando RDS Proxy elabora le richieste di modifica delle variabili di sessione o delle impostazioni di configurazione, il medesimo effettua il pinning di tale sessione alla connessione DB. In questo modo, lo stato della sessione rimane attivo per tutte le transazioni successive nella stessa sessione.

Per alcuni motori di database, questa regola non si applica a tutti i parametri che puoi impostare. RDS Proxy tiene traccia di determinate istruzioni e variabili. Pertanto, Server proxy per RDS non applica il pinning alla sessione quando si effettuano queste modifiche. In tal caso, RDS Proxy riutilizza la connessione solo per altre sessioni con gli stessi valori per tali impostazioni. Per gli elenchi delle istruzioni e delle variabili tracciate per Aurora MySQL, consulta Istruzioni tracciate da Server proxy per RDS per database Aurora MySQL.

Istruzioni tracciate da Server proxy per RDS per database Aurora MySQL

Di seguito sono riportate le istruzioni MySQL di cui Server proxy per RDS tiene traccia:

- DROP DATABASE
- DROP SCHEMA
- USE

Di seguito sono riportate le variabili MySQL di cui Server proxy per RDS tiene traccia:

- AUTOCOMMIT
- AUTO_INCREMENT_INCREMENT
- CHARACTER SET (or CHAR SET)
- CHARACTER_SET_CLIENT
- CHARACTER_SET_DATABASE
- CHARACTER_SET_FILESYSTEM
- CHARACTER_SET_CONNECTION
- CHARACTER_SET_RESULTS
- CHARACTER_SET_SERVER
- COLLATION_CONNECTION
- COLLATION_DATABASE
- COLLATION_SERVER
- INTERACTIVE_TIMEOUT
- NAMES
- NET_WRITE_TIMEOUT
- QUERY_CACHE_TYPE
- SESSION_TRACK_SCHEMA
- SQL_MODE
- TIME_ZONE
- TRANSACTION_ISOLATION (or TX_ISOLATION)
- TRANSACTION_READ_ONLY (or TX_READ_ONLY)
- WAIT_TIMEOUT

> **Note**
>
> Server proxy per RDS tiene traccia delle modifiche alle variabili TRANSACTION_ISOLATION e TRANSACTION_READ_ONLY quando le imposti nell'ambito della sessione. Tuttavia, se le imposti nell'ambito della transazione successiva, Proxy RDS blocca le connessioni.

Questo comportamento si applica sia che si utilizzi un'istruzione `SET` o un'istruzione `SET TRANSACTION` per configurare questi valori.

Riduzione dell'associazione

L'ottimizzazione delle prestazioni di RDS Proxy comporta il tentativo di massimizzare il riutilizzo della connessione a livello di transazione (multiplexing) riducendo al minimo il pinning.

È possibile ridurre l'associazione effettuando le seguenti operazioni:

- Evitare richieste di database non necessarie che potrebbero causare il pinning.
- Impostare le variabili e le impostazioni di configurazione in modo coerente su tutte le connessioni. In questo modo, le sessioni successive hanno maggiori probabilità di riutilizzare le connessioni con quelle specifiche impostazioni.

  Tuttavia, per l'impostazione di PostgreSQL una variabile porterà al pinning della sessione.
- Per un database della famiglia di motori MySQL, applica un filtro di pinning della sessione al proxy. Puoi esentare determinati tipi di operazioni dal pinning della sessione se sai che tale operazione non influisce sul corretto funzionamento dell'applicazione.
- Vedi la frequenza con cui si verifica l'associazione monitorando la metrica Amazon CloudWatch `DatabaseConnectionsCurrentlySessionPinned`. Per informazioni su questa e altri parametri CloudWatch, consulta Monitoraggio dei parametri RDS Proxy con Amazon CloudWatch.
- Se utilizzi istruzioni `SET` per eseguire l'inizializzazione identica per ogni connessione client, puoi farlo pur mantenendo il multiplexing a livello di transazione. In questo caso, sposta le istruzioni che impostano lo stato della sessione iniziale nella query di inizializzazione utilizzata da un proxy. Questa proprietà è una stringa contenente una o più istruzioni SQL, separate da punto e virgola.

  Ad esempio, puoi definire una query di inizializzazione per un proxy che imposta determinati parametri di configurazione. Quindi, RDS Proxy applica tali impostazioni ogni volta che imposta una nuova connessione per tale proxy. Puoi rimuovere le istruzioni `SET` corrispondenti dal codice dell'applicazione, in modo che non interferiscano con il multiplexing a livello di transazione.

  Per visualizzare i parametri sulla frequenza con cui si verifica il pinning in relazione a un proxy, consulta Monitoraggio dei parametri RDS Proxy con Amazon CloudWatch.

Condizioni che causano il pinning per tutte le famiglie di motori

Il proxy effettua il pinning della sessione alla connessione corrente nelle seguenti situazioni in cui il multiplexing potrebbe causare un comportamento imprevisto:

- Qualsiasi istruzione con una dimensione del testo maggiore di 16 KB fa sì che il proxy effettui il pinning della sessione.

Condizioni che causano l'associazione per Aurora MySQL

Per PostgreSQL, anche le seguenti interazioni causano il pinning:

- Le dichiarazioni di blocco esplicito delle tabelle `LOCK TABLE`, `LOCK TABLES` o `FLUSH TABLES WITH READ LOCK` causano il pinning della sessione da parte del proxy.
- La creazione di blocchi denominati mediante `GET_LOCK` fa sì che il proxy esegua il pinning della sessione.
- L'impostazione di una variabile utente o di una variabile di sistema (con alcune eccezioni) effettua il pinning al proxy. Se ciò limita in modo significativo il riutilizzo della connessione, è possibile configurare le operazioni `SET` per evitare il pinning. A tale scopo, regolate la proprietà dei filtri di blocco della sessione. Per ulteriori informazioni, consulta Creazione di un proxy per Amazon Aurora e Modifica di un RDS Proxy.
- La creazione di una tabella temporanea fa sì che il proxy effettui il pinning della sessione. In questo modo, il contenuto della tabella temporanea viene conservato per tutta la sessione indipendentemente dai limiti delle transazioni.
- Le chiamate delle funzioni `ROW_COUNT` e `FOUND_ROWS` talvolta causano il pinning.

  Le circostanze esatte in cui queste funzioni causano il blocco potrebbero differire tra le versioni Aurora MySQL compatibili con MySQL 5.7.
- Le istruzioni preparate fanno sì che il proxy effettui il pinning della sessione. Questa regola si applica se l'istruzione preparata utilizza il testo SQL o il protocollo binario.
- RDS Proxy non blocca le connessioni quando si utilizza SET LOCAL.
- Le chiamate di stored procedure e di funzioni archiviate non causano il pinning. RDS Proxy non rileva alcuna modifica dello stato della sessione derivante da tali chiamate. Assicurati che l'applicazione non cambi lo stato della sessione all'interno delle routine memorizzate se si fa affidamento su tale stato di sessione per persistere tra le transazioni. Ad esempio, Server proxy per

RDS non è attualmente compatibile con una stored procedure crea una tabella temporanea che persiste tra tutte le transazioni.

Se disponi di un'approfondita conoscenza del comportamento dell'applicazione, puoi scegliere di ignorare il comportamento del pinning per determinate istruzioni dell'applicazione. A tale scopo, puoi selezionare l'opzione (Filtri di pinning della sessione durante la creazione del proxy. Attualmente, è possibile disattivare l'aggiunta della sessione per l'impostazione delle variabili di sessione e delle impostazioni di configurazione.

Condizioni che causano l'associazione per Aurora PostgreSQL

Per PostgreSQL, le seguenti interazioni causano anche il pinning:

- L'utilizzo dei comandi `SET`.
- Utilizzo dei comandi `PREPARE`, `DISCARD`, `DEALLOCATE` o `EXECUTE` per gestire le istruzioni preparate.
- Creazione di sequenze temporanee, tabelle o viste.
- Dichiarazione dei cursori.
- Eliminazione dello stato della sessione.
- Ascolto su un canale di notifica.
- Caricamento di un modulo di libreria come `auto_explain`.
- Manipolazione delle sequenze usando funzioni come `nextval` e `setval`.
- Interazione con i blocchi utilizzando funzioni come `pg_advisory_lock` e `pg_try_advisory_lock`.

  > **Note**
  >
  > Server proxy per RDS non prevede blocchi di consulenza a livello di transazione, in particolare `pg_advisory_xact_lock`, `pg_advisory_xact_lock_shared`, `pg_try_advisory_xact_lock` e `pg_try_advisory_xact_lock_shared`.

- Impostazione di un parametro o reimpostazione di un parametro sul valore predefinito. In particolare, utilizzo dei comandi `SET` e `set_config` per assegnare valori predefiniti alle variabili di sessione.
- Le chiamate di stored procedure e di funzioni archiviate non causano il pinning. RDS Proxy non rileva alcuna modifica dello stato della sessione derivante da tali chiamate. Assicurati che

l'applicazione non cambi lo stato della sessione all'interno delle routine memorizzate se si fa affidamento su tale stato di sessione per persistere tra le transazioni. Ad esempio, Server proxy per RDS non è attualmente compatibile con una stored procedure crea una tabella temporanea che persiste tra tutte le transazioni.

- Eliminazione dello stato della sessione. Se si utilizzano librerie di connection pooling con una query `DISCARD ALL` configurata come query di ripristino, Server proxy per RDS blocca la connessione client al momento del rilascio. Ciò riduce l'efficienza del multiplexing del proxy e potrebbe portare a risultati imprevisti perché il comando `DISCARD ALL` può interferire con la gestione dello stato della sessione.

Eliminazione di un RDS Proxy

È possibile eliminare un proxy quando non è più necessario. Oppure, è possibile eliminare un proxy se si mette fuori servizio l'istanza database o il cluster ad esso associato.

Console di gestione AWS

Per eliminare un proxy

1. Accedi alla Console di gestione AWS e apri la console Amazon RDS all'indirizzo https://console.aws.amazon.com/rds/.
2. Nel riquadro di navigazione scegli Proxies (Proxy).
3. Scegli il proxy da eliminare dall'elenco.
4. Scegli Delete Proxy (Elimina proxy).

AWS CLI

Per eliminare un proxy DB, utilizzare il comando AWS CLI delete-db-proxy. Per rimuovere le associazioni correlate, utilizzare anche il comando deregister-db-proxy-targets.

```
aws rds delete-db-proxy --name proxy_name
```

```
aws rds deregister-db-proxy-targets
    --db-proxy-name proxy_name
    [--target-group-name target_group_name]
    [--target-ids comma_separated_list]       # or
    [--db-instance-identifiers instance_id]       # or
```

```
    [--db-cluster-identifiers cluster_id]
```

API RDS

Per eliminare un proxy DB, chiama la funzione API Amazon RDS DeleteDBProxy. Per eliminare elementi e associazioni correlati, è anche possibile chiamare le funzioni DeleteDBProxyTargetGroup e DeregisterDBProxyTargets.

Utilizzo degli endpoint Amazon RDS Proxy

Gli endpoint Server proxy per RDS offrono modi flessibili ed efficienti per gestire le connessioni al database, migliorando la scalabilità, la disponibilità e la sicurezza. Con gli endpoint proxy, è possibile:

- Semplificare il monitoraggio e la risoluzione dei problemi: utilizzare più endpoint per monitorare e gestire le connessioni da diverse applicazioni in modo indipendente.
- Migliorare la scalabilità di lettura: sfruttare gli endpoint di lettura con i cluster di database Aurora per distribuire in modo efficiente il traffico di lettura, ottimizzando le prestazioni per i carichi di lavoro con un elevato numero di query.
- Abilita l'accesso tra VPC: collega i database VPCs utilizzando endpoint cross-VPC, che consentono a risorse come le istanze Amazon in un VPC di accedere ai database EC2 in un altro VPC.

Argomenti

- Panoramica degli endpoint proxy
- Limitazioni per gli endpoint proxy
- Utilizzo degli endpoint di lettura con cluster Aurora
- Accesso ai database Aurora RDS VPCs
- Creazione di un endpoint proxy
- Visualizzazione degli endpoint proxy
- Modifica di un endpoint proxy
- Eliminazione di un endpoint proxy

Panoramica degli endpoint proxy

L'uso degli endpoint Server proxy per RDS include gli stessi tipi di procedure degli endpoint di cluster di database Aurora e di lettura. Se non hai familiarità con gli endpoint Aurora, puoi consultare Connessioni degli endpoint Amazon Aurora.

Quando si utilizza Server proxy per RDS con un cluster Aurora, l'endpoint predefinito supporta operazioni di lettura e scrittura. Ciò significa che indirizza tutte le richieste all'istanza di scrittura del cluster, contribuendo al limite `max_connections`. Per distribuire meglio il traffico, puoi creare endpoint aggiuntivi read/write o di sola lettura per il tuo proxy, il che consente una gestione più efficiente del carico di lavoro e una maggiore scalabilità.

Utilizzare un endpoint di sola lettura con il proxy per gestire le query di lettura, proprio come si farebbe con un endpoint di lettura per un cluster Aurora con provisioning. Questo approccio massimizza la scalabilità di lettura di Aurora distribuendo le query su una o più istanze database di lettura. Utilizzando un endpoint di sola lettura e aggiungendo più istanze di lettura in base alle necessità, è possibile aumentare il numero di query e connessioni simultanee che il cluster può gestire.

> **Tip**
>
> Quando si crea un proxy per un cluster Aurora utilizzando Console di gestione AWS, è possibile fare in modo che RDS Proxy crei automaticamente un endpoint di lettura. Per informazioni sui vantaggi di un endpoint di lettura, consulta Utilizzo degli endpoint di lettura con cluster Aurora.

Quando si crea un endpoint proxy, è possibile associarlo a un cloud privato virtuale (VPC) diverso da quello del proxy. In questo modo è possibile connettersi al proxy da un altro VPC, ad esempio da un VPC utilizzato da un'applicazione diversa all'interno dell'organizzazione.

Per informazioni sui limiti associati agli endpoint proxy, consulta Limitazioni per gli endpoint proxy.

Nei log Server proxy per RDS, ogni voce viene preceduta dal nome dell'endpoint proxy associato. Può essere il nome specificato per un endpoint definito dall'utente o il nome speciale `default` per l'endpoint predefinito del proxy. read/write

Ogni endpoint proxy ha il proprio set di metriche. CloudWatch Monitora le metriche per tutti gli endpoint proxy, un endpoint specifico o tutti gli read/write endpoint di un proxy o di sola lettura. Per ulteriori informazioni, consulta Monitoraggio dei parametri RDS Proxy con Amazon CloudWatch.

Un endpoint proxy utilizza lo stesso meccanismo di autenticazione del proxy associato. RDS Proxy imposta automaticamente i permessi e le autorizzazioni per l'endpoint definito dall'utente, coerenti con le proprietà del proxy associato.

Per informazioni su come funzionano gli endpoint proxy per i cluster database di un database globale Aurora, consulta Come funzionano gli endpoint Server proxy per RDS con i database globali.

Limitazioni per gli endpoint proxy

Gli endpoint Server proxy per RDS hanno le seguenti limitazioni:

- L'endpoint Server proxy per RDS predefinito non può essere modificato.
- Il numero massimo di endpoint definiti dall'utente per un proxy è 20. Pertanto, un proxy può avere fino a 21 endpoint: l'endpoint predefinito più 20 creati.
- Quando associ degli endpoint aggiuntivi a un proxy, RDS Proxy determina automaticamente quali istanze database nel cluster utilizzare per ciascun endpoint. Non è possibile scegliere istanze specifiche come invece è possibile con gli endpoint personalizzati Aurora.
- Per i IPv6 tipi di rete endpoint o dual-stack, il VPC e le sottoreti devono essere configurati per supportare il tipo di rete selezionato.

Quando si crea un proxy, RDS crea automaticamente un endpoint VPC per comunicazioni sicure tra le applicazioni e il database. L'endpoint VPC è visibile e accessibile dalla console Amazon VPC.

L'aggiunta di un nuovo endpoint proxy fornisce un endpoint di interfaccia. AWS PrivateLink Se si aggiungono uno o più endpoint al proprio proxy, si incorre in costi aggiuntivi. Per ulteriori informazioni, consulta Prezzi di Server proxy per RDS.

Utilizzo degli endpoint di lettura con cluster Aurora

Puoi creare e connetterti a endpoint di sola lettura denominati endpoint di lettura quando utilizzi RDS Proxy con cluster Aurora. Questi endpoint di lettura consentono di migliorare la scalabilità di lettura delle applicazioni che richiedono un uso intensivo di query. Gli endpoint di lettura consentono inoltre di migliorare la disponibilità delle connessioni se un'istanza database di lettura nel cluster non è disponibile.

> **Note**
>
> Quando si specifica che un nuovo endpoint è di sola lettura, RDS Proxy richiede che il cluster Aurora abbia una o più istanze database di lettura. In alcuni casi, potrebbe essere necessario

modificare la destinazione del proxy in un cluster Aurora contenente solo una singola istanza di scrittura. In questo caso, qualsiasi richiesta all'endpoint di lettura ha esito negativo e restituisce un errore. Anche le richieste avranno esito negativo se la destinazione del proxy è un'istanza RDS invece di un cluster Aurora.
Se un cluster Aurora ha istanze di lettura ma tali istanze non sono disponibili, RDS Proxy attende per inviare la richiesta invece di restituire immediatamente un errore. Se nessuna istanza di lettura diventa disponibile entro il periodo di timeout di prestito della connessione, la richiesta ha esito negativo con un errore.

In che modo gli endpoint di lettura aiutano la disponibilità delle applicazioni

In alcuni casi, una o più istanze di lettura nel cluster potrebbero non essere disponibili. In questo caso, le connessioni che utilizzano un endpoint di lettura di un proxy DB possono essere ripristinate più rapidamente di quelle che utilizzano l'endpoint di lettura Aurora. RDS Proxy instrada le connessioni solo alle istanze del lettore disponibili nel cluster. Non vi è alcun ritardo dovuto alla memorizzazione nella cache DNS quando un'istanza diventa non disponibile.

Se la connessione è multiplexing, RDS Proxy indirizza le query successive a un'istanza database di lettura diversa senza alcuna interruzione dell'applicazione. Durante il passaggio automatico a una nuova istanza di lettura, RDS Proxy controlla il ritardo della replica delle istanze di lettura vecchie e nuove. RDS Proxy assicura che la nuova istanza di lettura sia aggiornata con le stesse modifiche dell'istanza di lettura precedente. In questo modo, la tua applicazione non vedrà mai dati obsoleti quando RDS Proxy passa da un'istanza database di lettura a un'altra.

Se la connessione è bloccata, la successiva query sulla connessione restituisce un errore. Tuttavia, l'applicazione può riconnettersi immediatamente allo stesso endpoint. RDS Proxy indirizza la connessione a un'istanza database del lettore diversa che si trova nello stato `available`. Se ti riconnetti manualmente, RDS Proxy non controlla il ritardo di replica tra le vecchie e le nuove istanze di lettura.

Se il tuo cluster Aurora non dispone di istanze di lettura disponibili, RDS Proxy verifica se questa condizione è temporanea o permanente. Il comportamento in ogni caso è il seguente:

- Si assuma che il cluster abbia una o più istanze database di lettura, ma nessuna di esse si trova nello stato `Available`. Ad esempio, tutte le istanze di lettura potrebbero essere riavviate o potrebbero riscontrare dei problemi. In tal caso, i tentativi di connessione a un endpoint di lettura attendono la disponibilità di un'istanza di lettura. Se nessuna istanza di lettura diventa disponibile

entro il periodo di timeout di prestito della connessione, il tentativo di connessione ha esito negativo. Se invece un'istanza di lettura diventa disponibile, il tentativo di connessione ha esito positivo.

- Si assuma che il cluster non abbia istanze database di lettura. In tal caso, RDS Proxy restituisce immediatamente un errore se si prova a connettersi a un endpoint di lettura. Per risolvere il problema, aggiungi una o più istanze di lettura al cluster prima di connetterti all'endpoint di lettura.

In che modo gli endpoint di lettura aiutano la scalabilità delle query

Gli endpoint di lettura per un proxy aiutano con la scalabilità delle query di Aurora nei seguenti modi:

- Quando aggiungi istanze di lettura al cluster Aurora, RDS Proxy può instradare nuove connessioni a qualsiasi endpoint di lettura a istanze di lettura differenti. In questo modo, le query eseguite utilizzando una connessione all'endpoint di lettura non rallentano le query eseguite utilizzando un'altra connessione all'endpoint di lettura. Le query vengono eseguite su istanze database separate. Ogni istanza database ha le proprie risorse di calcolo, cache di buffer e così via.
- Laddove possibile, RDS Proxy utilizza la stessa istanza database di lettura per tutti i problemi di query utilizzando una particolare connessione all'endpoint di lettura. In questo modo, un insieme di query correlate sulle stesse tabelle può sfruttare la memorizzazione nella cache, l'ottimizzazione del piano e così via, su una particolare istanza database.
- Se un'istanza database di lettura non è disponibile, l'effetto sull'applicazione dipende dal fatto che la sessione sia multiplexing o bloccata. Se la sessione è multiplexing, RDS Proxy indirizza tutte le query successive a un'istanza database di lettura diversa senza richiedere alcuna azione da parte tua. Se invece la sessione è bloccata, l'applicazione riceve un errore e deve riconnettersi. Puoi riconnetterti immediatamente all'endpoint di lettura e RDS Proxy indirizza la connessione a un'istanza database di lettura disponibile. Per ulteriori informazioni sul multiplexing e sul blocco per le sessioni proxy, consulta Panoramica dei concetti RDS Proxy.
- Maggiore è il numero di istanze database di lettura presenti nel cluster, maggiore sarà il numero di connessioni simultanee che è possibile effettuare utilizzando gli endpoint di lettura. Supponi, ad esempio, che il cluster disponga di quattro istanze database di lettura, ciascuna configurata per supportare 200 connessioni simultanee. Supponi anche che il proxy sia configurato per utilizzare il 50% del numero massimo di connessioni. Qui, il numero massimo di connessioni che è possibile effettuare attraverso gli endpoint di lettura nel proxy è 100 (50% di 200) per la lettura 1. È 100 anche per la lettura 2, e così via, per un totale di 400. Se raddoppi il numero di istanze database di lettura del cluster a otto, anche il numero massimo di connessioni attraverso gli endpoint di lettura raddoppia a 800.

Esempi di utilizzo degli endpoint di lettura

L'esempio seguente di Linux mostra come è possibile confermare di essere connessi a un cluster Aurora MySQL tramite un endpoint di lettura. L'impostazione di configurazione di `innodb_read_only` è abilitata. I tentativi di esecuzione di operazioni di scrittura come le istruzioni `CREATE DATABASE` non riescono e restituiscono un errore. Inoltre, puoi confermare di essere connesso a un'istanza database di lettura controllando il nome dell'istanza database utilizzando la variabile `aurora_server_id`.

> **Tip**
>
> Non affidarti solo al controllo del nome dell'istanza DB per determinare se la connessione è o di sola lettura. read/write Ricorda che le istanze database in un cluster Aurora possono cambiare ruolo tra scrittura e lettura quando si verificano i failover.

```
$ mysql -h endpoint-demo-reader.endpoint.proxy-demo.us-east-1.rds.amazonaws.com -u
 admin -p
...
mysql> select @@innodb_read_only;
+--------------------+
| @@innodb_read_only |
+--------------------+
|                  1 |
+--------------------+
mysql> create database shouldnt_work;
ERROR 1290 (HY000): The MySQL server is running with the --read-only option so it
 cannot execute this statement

mysql> select @@aurora_server_id;
+--------------------------------------+
| @@aurora_server_id                   |
+--------------------------------------+
| proxy-reader-endpoint-demo-instance-3 |
+--------------------------------------+
```

L'esempio seguente mostra come la connessione a un endpoint di lettura proxy può continuare a funzionare anche quando l'istanza database di lettura viene eliminata. In questo esempio, il cluster Aurora dispone di due istanze di lettura, `instance-5507` e `instance-7448` . La connessione all'endpoint di lettura inizia utilizzando una delle istanze di lettura. Durante l'esempio, questa istanza

di lettura viene eliminata da un comando `delete-db-instance`. RDS Proxy passa a un'istanza di lettura diversa per le query successive.

```
$ mysql -h reader-demo.endpoint.proxy-demo.us-east-1.rds.amazonaws.com
  -u my_user -p
...
mysql> select @@aurora_server_id;
+--------------------+
| @@aurora_server_id |
+--------------------+
| instance-5507      |
+--------------------+

mysql> select @@innodb_read_only;
+--------------------+
| @@innodb_read_only |
+--------------------+
|                  1 |
+--------------------+

mysql> select count(*) from information_schema.tables;
+----------+
| count(*) |
+----------+
|      328 |
+----------+
```

Mentre la sessione `mysql` è ancora in esecuzione, il comando seguente elimina l'istanza di lettura a cui è connesso l'endpoint di lettura.

```
aws rds delete-db-instance --db-instance-identifier instance-5507 --skip-final-snapshot
```

Le query nella sessione `mysql` continuano a funzionare senza la necessità di riconnettersi. RDS Proxy passa quindi automaticamente a un'istanza database di lettura diversa.

```
mysql> select @@aurora_server_id;
+--------------------+
| @@aurora_server_id |
+--------------------+
| instance-7448      |
+--------------------+
```

```
mysql> select count(*) from information_schema.TABLES;
+----------+
| count(*) |
+----------+
|      328 |
+----------+
```

Accesso ai database Aurora RDS VPCs

Per impostazione predefinita, i componenti dello stack di tecnologia di Aurora si trovano tutti nello stesso Amazon VPC. Ad esempio, supponiamo che un'applicazione in esecuzione su un' EC2 istanza Amazon si connetta a un'istanza database del cluster Aurora DB. In questo caso, il server delle applicazioni e il database devono trovarsi entrambi all'interno dello stesso VPC.

Con RDS Proxy, puoi configurare l'accesso a un'istanza del cluster Aurora in un VPC dalle risorse di un altro VPC, come le istanze. EC2 Ad esempio, l'organizzazione potrebbe avere più applicazioni che accedono alle stesse risorse del database. Ogni applicazione potrebbe trovarsi nel proprio VPC.

Per consentire l'accesso tra VPC, crea un nuovo endpoint per il proxy. Il proxy stesso risiede nello stesso VPC del cluster di database Aurora . Tuttavia, l'endpoint cross-VPC risiede nell'altro VPC, insieme alle altre risorse come le istanze. EC2 L'endpoint cross-VPC è associato a sottoreti e gruppi di sicurezza dello stesso VPC delle risorse e di altre risorse. EC2 Queste associazioni consentono di connettersi all'endpoint dalle applicazioni che altrimenti non potrebbero accedere al database a causa delle restrizioni del VPC.

Nella procedura seguente viene illustrato come creare e accedere a un endpoint tra VPC tramite RDS Proxy:

1. Creane due VPCs o scegline due VPCs che già usi per il lavoro con Aurora . Ogni VPC deve avere le proprie risorse di rete associate, ad esempio un gateway Internet, tabelle di routing, sottoreti e gruppi di sicurezza. Se si dispone di un solo VPC, è possibile consultare la procedura Nozioni di base su Amazon Aurora per configurare un altro VPC per l'utilizzo corretto di Aurora. Puoi anche esaminare il tuo VPC esistente nella EC2 console Amazon per vedere i tipi di risorse da connettere tra loro.
2. Crea un proxy database associato al cluster di database Aurora a cui desideri connetterti. Segui la procedura riportata in Creazione di un proxy per Amazon Aurora.
3. Nella pagina Dettagli del proxy nella console RDS, nella sezione Endpoint proxy seleziona Crea endpoint. Segui la procedura riportata in Creazione di un endpoint proxy.

4. Scegli se rendere l'endpoint read/write cross-VPC o di sola lettura.
5. Invece di accettare il valore predefinito dello stesso VPC del cluster di database Aurora , seleziona un VPC diverso. Questo VPC deve trovarsi nella stessa AWS regione del VPC in cui risiede il proxy.
6. Ora invece di accettare i valori predefiniti per le sottoreti e i gruppi di sicurezza dallo stesso VPC del cluster di database Aurora , effettua nuove selezioni. Le nuove selezioni dovranno essere relative alle sottoreti e ai gruppi di sicurezza del VPC scelto.
7. Non è necessario modificare alcuna delle impostazioni per i segreti di Secrets Manager. Le stesse credenziali funzionano per tutti gli endpoint del proxy, indipendentemente dal VPC in cui si trova ciascun endpoint. Allo stesso modo, quando si utilizza l'autenticazione IAM, la configurazione e le autorizzazioni IAM funzionano in modo coerente su tutti gli endpoint proxy, anche quando gli endpoint sono diversi. VPCs Non è richiesta alcuna configurazione IAM aggiuntiva per endpoint.
8. Attendi che il nuovo endpoint raggiunga lo stato Disponibile.
9. Prendi nota del nome completo dell'endpoint. Questo è il valore che termina in *`Region_name`*`.rds.amazonaws.com` fornito come parte della stringa di connessione per l'applicazione di database.
10. Accedi al nuovo endpoint da una risorsa nello stesso VPC dell'endpoint. Un modo semplice per testare questo processo è creare una nuova EC2 istanza in questo VPC. Quindi, accedi all' EC2 istanza ed `psql` esegui i comandi `mysql` or per connetterti utilizzando il valore dell'endpoint nella stringa di connessione.

Creazione di un endpoint proxy

Per creare un endpoint proxy, seguire queste istruzioni:

Console

Per creare un endpoint proxy

1. Accedi a Console di gestione AWS e apri la console Amazon RDS all'indirizzo https://console.aws.amazon.com/rds/.
2. Nel riquadro di navigazione scegli Proxies (Proxy).
3. Fai clic sul nome del proxy per il quale desideri creare un nuovo endpoint.

   Sarà visualizzata la pagina dei dettagli del proxy.
4. In Endpoint proxy, seleziona Crea endpoint proxy.

Verrà visualizzata la finestra Crea endpoint proxy.

5. Per Nome endpoint proxy, specifica un nome descrittivo a scelta.
6. Per il ruolo Target, scegli se rendere l'endpoint read/write o di sola lettura.

   Le connessioni che utilizzano gli read/write endpoint possono eseguire qualsiasi tipo di operazione, come istruzioni DDL (Data Definition Language), istruzioni DML (Data Manipulation Language) e query. Questi endpoint si connettono sempre all'istanza primaria del cluster Aurora. È possibile utilizzare gli read/write endpoint per operazioni generali sul database quando si utilizza un solo endpoint nell'applicazione. È inoltre possibile utilizzare gli read/write endpoint per operazioni amministrative, applicazioni di elaborazione delle transazioni online (OLTP) e lavori extract-transform-load (ETL).

   Le connessioni che utilizzano un endpoint di sola lettura possono soltanto eseguire query. Quando nel cluster Aurora sono presenti più istanze di lettura, Server proxy per RDS può utilizzare un'istanza di lettura diversa per ogni connessione all'endpoint. In questo modo, un'applicazione ad uso intensivo di query può sfruttare i vantaggi del clustering di Aurora. Puoi aggiungere più capacità di query al cluster aggiungendo più istanze database di lettura. Queste connessioni di sola lettura non impongono alcun sovraccarico sull'istanza primaria del cluster. In questo modo, le query di report e analisi non rallentano le operazioni di scrittura delle applicazioni OLTP.
7. Per Virtual Private Cloud (VPC), scegli l'impostazione predefinita per accedere all'endpoint dalle stesse EC2 istanze o altre risorse che normalmente vengono utilizzate per accedere al proxy o al database associato. Per impostare l'accesso tra VPC per questo proxy, scegli un VPC diverso da quello predefinito. Per ulteriori informazioni sull'accesso tra VPC, consulta Accesso ai database Aurora RDS VPCs.
8. Per il tipo di rete dell'endpoint, scegliere la versione IP per l'endpoint proxy. Le opzioni disponibili sono:

   - IPv4— L'endpoint proxy utilizza solo IPv4 indirizzi (impostazione predefinita).
   - IPv6— L'endpoint proxy utilizza solo IPv6 indirizzi.
   - Dual-stack: l'endpoint proxy supporta entrambi gli indirizzi. IPv4 IPv6

   Per utilizzare IPv6 il dual-stack, il VPC e le sottoreti devono essere configurati per supportare il tipo di rete selezionato.

9. Per Sottoreti, RDS Proxy riempie le stesse sottoreti del proxy associato per impostazione predefinita. Per limitare l'accesso all'endpoint in modo che solo una parte dell'intervallo di indirizzi del VPC possa connettersi, rimuovere una o più sottoreti.
10. Per Gruppo di sicurezza VPC puoi selezionare un gruppo di sicurezza esistente o crearne uno nuovo. Per impostazione predefinita, RDS Proxy riempie lo stesso gruppo o gruppi di sicurezza del proxy associato. Se le regole in entrata e in uscita per il proxy sono appropriate per questo endpoint, lasciare la scelta predefinita.

    Se decidi di creare un nuovo gruppo di sicurezza, specifica un nome per il gruppo di sicurezza in questa pagina. Quindi modifica le impostazioni del gruppo di sicurezza dalla console in un secondo momento. EC2
11. Scegli Crea endpoint proxy.

AWS CLI

Per creare un endpoint proxy, usa il AWS CLI create-db-proxy-endpointcomando.

Includi i parametri obbligatori seguenti:

- `--db-proxy-name` *`value`*
- `--db-proxy-endpoint-name` *`value`*
- `--vpc-subnet-ids` *`list_of_ids`*. Separa la sottorete IDs con degli spazi. Non specificare l'ID del VPC stesso.

Puoi inoltre includere i seguenti parametri facoltativi:

- `--target-role { READ_WRITE | READ_ONLY }`. Questo parametro per impostazione predefinita è `READ_WRITE`. Il valore `READ_ONLY` ha un effetto solo su cluster Aurora con provisioning che contengono una o più istanze database di lettura. Quando il proxy è associato a un cluster Aurora che contiene solo un'istanza database di scrittura, non sarà possibile specificare `READ_ONLY`. Per ulteriori informazioni sull'uso previsto degli endpoint di sola lettura con cluster Aurora, consulta Utilizzo degli endpoint di lettura con cluster Aurora .
- `--vpc-security-group-ids` *`value`*. Separa il gruppo di sicurezza IDs con degli spazi. Se ometti questo parametro, RDS Proxy utilizza il gruppo di sicurezza predefinito per il VPC. Il proxy RDS determina il VPC in base alla IDs sottorete specificata per il parametro. `--vpc-subnet-ids`

- `--endpoint-network-type { IPV4 | IPV6 | DUAL }`. Questo parametro specifica la versione IP per l'endpoint proxy. Il valore predefinito è `IPV4`. Per utilizzare `IPV6` o `DUAL`, il VPC e le sottoreti devono essere configurati per supportare il tipo di rete selezionato.

Example

Nell'esempio seguente viene creato un endpoint proxy denominato `my-endpoint`.

Per Linux, macOS o Unix:

```
aws rds create-db-proxy-endpoint \
  --db-proxy-name my-proxy \
  --db-proxy-endpoint-name my-endpoint \
  --vpc-subnet-ids subnet_id subnet_id subnet_id ... \
  --target-role READ_ONLY \
  --vpc-security-group-ids security_group_id \
  --endpoint-network-type DUAL
```

Per Windows:

```
aws rds create-db-proxy-endpoint ^
  --db-proxy-name my-proxy ^
  --db-proxy-endpoint-name my-endpoint ^
  --vpc-subnet-ids subnet_id_1 subnet_id_2 subnet_id_3 ... ^
  --target-role READ_ONLY ^
  --vpc-security-group-ids security_group_id ^
  --endpoint-network-type DUAL
```

API RDS

Per creare un endpoint proxy, utilizzare l'azione Create Endpoint dell'API RDS. DBProxy

Visualizzazione degli endpoint proxy

Per visualizzare gli endpoint proxy esistenti, seguire queste istruzioni:

Console

Per visualizzare i dettagli di un endpoint proxy

1. Accedi alla Console di gestione AWS e apri la console Amazon RDS all'indirizzo https://console.aws.amazon.com/rds/.

2. Nel riquadro di navigazione scegli Proxies (Proxy).
3. Nell'elenco scegli il proxy di cui si desidera visualizzare l'endpoint. Fai clic sul nome del proxy per visualizzarne la pagina dei dettagli.
4. Nella sezione Endpoint proxy, scegli l'endpoint che desideri visualizzare. Fai clic sul relativo nome per visualizzare la pagina dei dettagli.
5. Esamina i parametri di cui ti interessano i valori. Puoi controllare proprietà come le seguenti:
   - Se l'endpoint è di lettura/scrittura o di sola lettura.
   - L'indirizzo dell'endpoint utilizzato in una stringa di connessione al database.
   - Le sottoreti e i gruppi di sicurezza del VPC associati all'attività.

AWS CLI

Per visualizzare uno o più endpoint proxy, utilizzare il comando AWS CLI describe-db-proxy-endpoints.

Puoi includere i seguenti parametri facoltativi:

- `--db-proxy-endpoint-name`
- `--db-proxy-name`

Nell'esempio seguente viene descritto l'endpoint proxy `my-endpoint`.

Example

Per Linux, macOS o Unix:

```
aws rds describe-db-proxy-endpoints \
  --db-proxy-endpoint-name my-endpoint
```

Per Windows:

```
aws rds describe-db-proxy-endpoints ^
  --db-proxy-endpoint-name my-endpoint
```

API RDS

Per descrivere uno o più endpoint proxy, utilizza l'operazione DescribeDBProxyEndpoints dell'API RDS.

Modifica di un endpoint proxy

Per eliminare gli endpoint proxy, seguire queste istruzioni:

Console

Per modificare uno o più endpoint proxy

1. Accedi alla Console di gestione AWS e apri la console Amazon RDS all'indirizzo https://console.aws.amazon.com/rds/.
2. Nel riquadro di navigazione scegli Proxies (Proxy).
3. Nell'elenco seleziona il proxy di cui desideri modificare l'endpoint. Fai clic sul nome del proxy per visualizzarne la pagina dei dettagli.
4. Nella sezione Endpoint proxy, scegli l'endpoint che desideri modificare. Puoi selezionarlo dall'elenco oppure fare clic sul relativo nome e visualizzarne la pagina dei dettagli.
5. Nella pagina dei dettagli del proxy, sotto la sezione Endpoint proxy, seleziona Modifica. Oppure nella pagina dei dettagli dell'endpoint proxy, in Operazioni, selezionare Modifica.
6. Modificare i valori dei parametri desiderati.
7. Scegli Save changes (Salva modifiche).

AWS CLI

Per modificare un endpoint proxy, utilizzare il comando AWS CLI modify-db-proxy-endpoint con i seguenti parametri obbligatori:

- `--db-proxy-endpoint-name`

Specifica le modifiche alle proprietà dell'endpoint utilizzando uno o più dei seguenti parametri:

- `--new-db-proxy-endpoint-name`
- `--vpc-security-group-ids`. Separa gli ID dei gruppi di sicurezza con gli spazi.

Nel seguente esempio l'endpoint proxy `my-endpoint` viene ridenominato in `new-endpoint-name`.

Example

Per Linux, macOS o Unix:

```
aws rds modify-db-proxy-endpoint \
  --db-proxy-endpoint-name my-endpoint \
  --new-db-proxy-endpoint-name new-endpoint-name
```

Per Windows:

```
aws rds modify-db-proxy-endpoint ^
  --db-proxy-endpoint-name my-endpoint ^
  --new-db-proxy-endpoint-name new-endpoint-name
```

API RDS

Per modificare un endpoint proxy, utilizza l'operazione ModifyDBProxyEndpoint dell'API RDS.

Eliminazione di un endpoint proxy

Per eliminare un endpoint per il proxy, seguire queste istruzioni:

> **Note**
>
> Non è possibile eliminare l'endpoint proxy predefinito creato automaticamente da Server proxy per RDS per ogni proxy.
> Quando elimini un proxy, RDS Proxy elimina automaticamente tutti gli endpoint associati.

Console

Per eliminare un endpoint proxy utilizzando il Console di gestione AWS

1. Nel riquadro di navigazione scegli Proxies (Proxy).
2. Nell'elenco seleziona il proxy per cui desideri eliminare l'endpoint. Fai clic sul nome del proxy per visualizzarne la pagina dei dettagli.
3. Nella sezione Endpoint proxy, scegli l'endpoint che desideri eliminare. Puoi selezionare uno o più endpoint dall'elenco oppure fare clic sul nome di un singolo endpoint e visualizzarne la pagina dei dettagli.

4. Nella pagina dei dettagli del proxy, sotto la sezione Endpoint proxy, seleziona Elimina. Oppure nella pagina dei dettagli dell'endpoint proxy, in Operazioni, selezionare Elimina.

AWS CLI

Per eliminare un endpoint proxy, esegui il comando delete-db-proxy-endpoint con i seguenti parametri obbligatori:

- `--db-proxy-endpoint-name`

Il comando seguente elimina l'endpoint proxy denominato `my-endpoint`.

Per Linux, macOS o Unix:

```
aws rds delete-db-proxy-endpoint \
  --db-proxy-endpoint-name my-endpoint
```

Per Windows:

```
aws rds delete-db-proxy-endpoint ^
  --db-proxy-endpoint-name my-endpoint
```

API RDS

Per eliminare un endpoint proxy con l'API RDS, esegui l'operazione DeleteDBProxyEndpoint. Specifica il nome dell'endpoint proxy per il parametro `DBProxyEndpointName`.

Monitoraggio dei parametri RDS Proxy con Amazon CloudWatch

Puoi monitorare RDS Proxy utilizzando Amazon CloudWatch. CloudWatch raccoglie ed elabora i dati non elaborati dai proxy trasformandoli in parametri leggibili quasi in tempo reale. Per trovare questi parametri nella console CloudWatch, seleziona Metrics (Parametri), quindi seleziona RDS e scegli Per-Proxy Metrics (Parametri per proxy). Per maggiori informazioni, consulta Utilizzo dei parametri di Amazon CloudWatch nella Guida per l'utente di Amazon CloudWatch.

> **Note**
>
> RDS pubblica questi parametri per ogni istanza Amazon EC2 sottostante associata al proxy. Un singolo proxy potrebbe essere servito da più di un'istanza EC2. Utilizza le statistiche di CloudWatch per aggregare i valori di un proxy in tutte le istanze associate.
> Alcuni di questi parametri potrebbero non essere visibili fino al completamento della prima connessione mediante un proxy.

Nei log RDS Proxy, ogni voce è preceduta dal nome dell'endpoint proxy associato. Questo nome può essere il nome specificato per un endpoint definito dall'utente o il nome speciale `default` per l'endpoint predefinito di un proxy che esegue richieste di lettura/scrittura.

Tutte le metriche RDS Proxy sono nel gruppo `proxy`.

Ogni endpoint proxy ha i propri parametri CloudWatch. Puoi monitorare l'utilizzo di ciascun endpoint proxy in modo indipendente. Per ulteriori informazioni sugli endpoint proxy, consulta Utilizzo degli endpoint Amazon RDS Proxy.

Puoi aggregare i valori per ogni parametro utilizzando uno dei seguenti set di dimensioni. Ad esempio, utilizzando il metodo `ProxyName`, puoi analizzare tutto il traffico per un determinato proxy. Utilizzando gli altri set di dimensioni, puoi suddividere i parametri in modi diversi. Puoi suddividere i parametri in base ai diversi endpoint o ai database di destinazione di ciascun proxy oppure al traffico di lettura/scrittura e di sola lettura verso ciascun database.

- Set di dimensioni 1: `ProxyName`
- Set di dimensioni 2: `ProxyName`, `EndpointName`
- Set di dimensioni 3: `ProxyName`, `TargetGroup`, `Target`
- Set di dimensioni 4: `ProxyName`, `TargetGroup`, `TargetRole`

Parametro	Descrizione	Periodo valido	Set di dimensioni CloudWatch
`AvailabilityPercentage`	Percentuale di tempo durante il quale il gruppo target era disponibi	1 minuto	Dimension set 4

Parametro	Descrizione	Periodo valido	Set di dimensioni CloudWatch
	le nel ruolo indicato dalla dimension e. Questa metrica viene segnalata ogni minuto. La statistica più utile per questo parametro è Sum.		
ClientConnections	Il numero corrente di connessioni client. Questa metrica viene segnalata ogni minuto. La statistica più utile per questo parametro è Sum.	1 minuto	Dimension set 1, Dimension set 2
ClientConnectionsClosed	Il numero di connessioni client chiuse. La statistica più utile per questo parametro è Sum.	1 minuto e oltre	Dimension set 1, Dimension set 2
ClientConnectionsInSetup	Il numero corrente di connessioni client sono aperte ma la configurazione non è stata completat a. Questa metrica viene segnalata ogni minuto. La statistica più utile per questo parametro è Sum.	1 minuto	Dimension set 1, Dimension set 2

Parametro	Descrizione	Periodo valido	Set di dimensioni CloudWatch
`ClientConnectionsNoTLS`	Numero corrente di connessioni client senza Transport Layer Security (TLS). Questa metrica viene segnalata ogni minuto. La statistica più utile per questo parametro è `Sum`.	1 minuto	Dimension set 1, Dimension set 2
`ClientConnectionsReceived`	Numero di richieste di connessione client ricevute. La statistica più utile per questo parametro è `Sum`.	1 minuto e oltre	Dimension set 1, Dimension set 2
`ClientConnectionsSetupFailedAuth`	Numero di tentativi di connessione client non riusciti a causa di autenticazione errata o configurazione TLS errata. La statistica più utile per questo parametro è `Sum`.	1 minuto e oltre	Dimension set 1, Dimension set 2

Parametro	Descrizione	Periodo valido	Set di dimensioni CloudWatch
`ClientConnectionsSetupSucceeded`	Il numero di connessioni client stabilito correttamente con qualsiasi meccanismo di autenticazione con o senza TLS. La statistica più utile per questo parametro è `Sum`.	1 minuto e oltre	Dimension set 1, Dimension set 2
`ClientConnectionsTLS`	Il numero corrente di connessioni client con TLS. Questa metrica viene segnalata ogni minuto. La statistica più utile per questo parametro è `Sum`.	1 minuto	Dimension set 1, Dimension set 2
`DatabaseConnectionRequests`	Numero di richieste per creare una connessione al database. La statistica più utile per questo parametro è `Sum`.	1 minuto e oltre	Dimension set 1, Dimension set 3, Dimension set 4
`DatabaseConnectionRequestsWithTLS`	Numero di richieste per creare una connessione al database con TLS. La statistica più utile per questo parametro è `Sum`.	1 minuto e oltre	Dimension set 1, Dimension set 3, Dimension set 4

Parametro	Descrizione	Periodo valido	Set di dimensioni CloudWatch
`DatabaseConnections`	Il numero corrente di connessioni al database. Questa metrica viene segnalata ogni minuto. La statistica più utile per questo parametro è `Sum`.	1 minuto	Dimension set 1, Dimension set 3, Dimension set 4
`DatabaseConnectionsBorrowLatency`	Il tempo in microsecondi necessario per il proxy monitorato per ottenere una connessione al database. La statistica più utile per questo parametro è `Sum`.	1 minuto e oltre	Dimension set 1, Dimension set 2
`DatabaseConnectionsCurrentlyBorrowed`	Il numero corrente di connessioni al database nello stato di prestito. Questa metrica viene segnalata ogni minuto. La statistica più utile per questo parametro è `Sum`.	1 minuto	Dimension set 1, Dimension set 3, Dimension set 4

Parametro	Descrizione	Periodo valido	Set di dimensioni CloudWatch
`DatabaseConnectionsCurrentlyInTransaction`	Il numero corrente di connessioni al database in una transazione. Questa metrica viene segnalata ogni minuto. La statistica più utile per questo parametro è `Sum`.	1 minuto	Dimension set 1, Dimension set 3, Dimension set 4
`DatabaseConnectionsCurrentlySessionPinned`	Il numero corrente di connessioni al database attualmente bloccati a causa di operazioni nelle richieste client che modificano lo stato della sessione. Questa metrica viene segnalata ogni minuto. La statistica più utile per questo parametro è `Sum`.	1 minuto	Dimension set 1, Dimension set 3, Dimension set 4
`DatabaseConnectionsSetupFailed`	Il numero di richieste di connessione al database non riuscite. La statistica più utile per questo parametro è `Sum`.	1 minuto e oltre	Dimension set 1, Dimension set 3, Dimension set 4

Parametro	Descrizione	Periodo valido	Set di dimensioni CloudWatch
`DatabaseConnectionsSetupSucceeded`	Il numero di connessioni al database stabilito correttamente con o senza TLS. La statistica più utile per questo parametro è `Sum`.	1 minuto e oltre	Dimension set 1, Dimension set 3, Dimension set 4
`DatabaseConnectionsWithTLS`	Il numero corrente di connessioni al database con TLS. Questa metrica viene segnalata ogni minuto. La statistica più utile per questo parametro è `Sum`.	1 minuto	Dimension set 1, Dimension set 3, Dimension set 4
`MaxDatabaseConnectionsAllowed`	Il numero massimo di connessioni al database consentite. Questa metrica viene segnalata ogni minuto. La statistica più utile per questo parametro è `Sum`.	1 minuto	Dimension set 1, Dimension set 3, Dimension set 4
`QueryDatabaseResponseLatency`	Tempo in microsecondi impiegato dal database per rispondere alla query. La statistica più utile per questo parametro è `Average`.	1 minuto e oltre	Dimension set 1, Dimension set 2, Dimension set 3, Dimension set 4

Parametro	Descrizione	Periodo valido	Set di dimensioni CloudWatch
`QueryRequests`	Il numero di query ricevute. Una query che include più istruzioni viene conteggiata come una query. La statistica più utile per questo parametro è `Sum`.	1 minuto e oltre	Dimension set 1, Dimension set 2
`QueryRequestsNoTLS`	Numero di query ricevute da connessioni non TLS. Una query che include più istruzioni viene conteggiata come una query. La statistica più utile per questo parametro è `Sum`.	1 minuto e oltre	Dimension set 1, Dimension set 2
`QueryRequestsTLS`	Numero di query ricevute dalle connessioni TLS. Una query che include più istruzioni viene conteggiata come una query. La statistica più utile per questo parametro è `Sum`.	1 minuto e oltre	Dimension set 1, Dimension set 2

Parametro	Descrizione	Periodo valido	Set di dimensioni CloudWatch
`QueryResponseLatency`	Il tempo in microsecondi tra l'ottenimento di una richiesta di query e il proxy che risponde ad essa. La statistica più utile per questo parametro è `Average`.	1 minuto e oltre	Dimension set 1, Dimension set 2

Puoi trovare i log dell'attività di RDS Proxy in CloudWatch nella Console di gestione AWS. Ogni proxy ha una voce nella pagina Log groups (Gruppi di registro) .

> **Important**
>
> Questi registri sono destinati all'utilizzo umano per scopi di risoluzione dei problemi e non per l'accesso programmatico. Il formato e il contenuto dei registri sono soggetti a modifiche.
> In particolare, i log meno recenti non contengono prefissi che indicano l'endpoint per ogni richiesta. Nei log più recenti, ogni voce è preceduta dal nome dell'endpoint proxy associato. Questo nome può essere il nome specificato per un endpoint definito dall'utente o il nome speciale `default` per le richieste che utilizzano l'endpoint predefinito di un proxy.

Utilizzo degli eventi RDS Proxy

Un evento indica una modifica in un ambiente come un ambiente AWS, un'applicazione o un servizio di un partner software as a service (SaaS). Oppure può essere uno dei tuoi servizi o applicazioni personalizzati. Ad esempio, Amazon Aurora genera un evento quando si crea o si modifica un proxy RDS. Amazon Aurora distribuisce gli eventi ad Amazon EventBridge quasi in tempo reale. Di seguito puoi trovare un elenco di eventi RDS Proxy a cui puoi iscriverti e un esempio di evento RDS Proxy.

Per ulteriori informazioni sull'utilizzo degli eventi, consulta quanto segue:

- Per istruzioni su come visualizzare gli eventi utilizzando la Console di gestione AWS, la AWS CLI o l'API RDS, consulta Visualizzazione di eventi Amazon RDS.

- Per informazioni su come configurare Amazon Aurora per inviare eventi a EventBridge, consulta Creazione di una regola che si attiva su un evento Amazon Aurora.

Eventi RDS Proxy

La tabella seguente riporta la categoria di eventi e un elenco di eventi applicabili quando il tipo di origine è un proxy RDS.

Categoria	ID evento RDS	Messaggio	Note
modifica della configurazione	RDS-EVENT-0204	Proxy DB *nome* modificato da RDS.	Nessuno
modifica della configurazione	RDS-EVENT-0207	RDS ha modificato l'endpoint del proxy DB *nome*.	Nessuno
modifica della configurazione	RDS-EVENT-0213	RDS ha rilevato l'aggiunta dell'istanza database e l'ha aggiunta automaticamente al gruppo di destinazione del proxy DB *nome*.	Nessuno
modifica della configurazione	RDS-EVENT-0214	RDS ha rilevato l'eliminazione dell'istanza database *nome* e l'ha rimossa automaticamente dal gruppo di destinazione *nome* del proxy DB *nome*.	Nessuno
modifica della configurazione	RDS-EVENT-0215	RDS ha rilevato l'eliminazione del cluster database *nome* e l'ha rimosso automaticamente dal gruppo di destinazione *nome* del proxy DB *nome*.	Nessuno

Categoria	ID evento RDS	Messaggio	Note
creazione	RDS-EVENT-0203	RDS ha creato il proxy DB *nome*.	Nessuno
creazione	RDS-EVENT-0206	RDS ha creato l'endpoint *nome* del proxy DB *nome*.	Nessuno
eliminazione	RDS-EVENT-0205	RDS ha eliminato il proxy DB *nome*.	Nessuno
eliminazione	RDS-EVENT-0208	RDS ha eliminato l'endpoint *nome* per il proxy DB *nome*.	Nessuno
errore	RDS-EVENT-0243	RDS non è riuscito ad eseguire il provisioning della capacità per il proxy *nome* perché non ci sono sufficienti indirizzi IP disponibili nelle sottoreti : *nome*. Per risolvere il problema, assicurarsi che le sottoreti abbiano il numero minimo di indirizzi IP non utilizzati come consigliato nella documentazione di Server proxy per Amazon RDS.	Per determinare il numero consigliato per la classe di istanza, consulta Pianificazione della capacità degli indirizzi IP.
errore	RDS-EVENT-0275	RDS ha limitato la larghezza di banda di alcune connessioni al proxy database *nome*. Il numero di richieste di connessione simultanee dal client al proxy ha superato il limite.	Nessuno

Di seguito è riportato un esempio di evento di RDS Proxy in formato JSON. L'evento mostra che RDS ha modificato l'endpoint denominato `my-endpoint` del proxy RDS denominato `my-rds-proxy`. L'ID evento è RDS-EVENT-0207.

```
{
  "version": "0",
  "id": "68f6e973-1a0c-d37b-f2f2-94a7f62ffd4e",
  "detail-type": "RDS DB Proxy Event",
  "source": "aws.rds",
  "account": "123456789012",
  "time": "2018-09-27T22:36:43Z",
  "region": "us-east-1",
  "resources": [
     "arn:aws:rds:us-east-1:123456789012:db-proxy:my-rds-proxy"
  ],
  "detail": {
    "EventCategories": [
      "configuration change"
    ],
    "SourceType": "DB_PROXY",
    "SourceArn": "arn:aws:rds:us-east-1:123456789012:db-proxy:my-rds-proxy",
    "Date": "2018-09-27T22:36:43.292Z",
    "Message": "RDS modified endpoint my-endpoint of DB Proxy my-rds-proxy.",
    "SourceIdentifier": "my-endpoint",
    "EventID": "RDS-EVENT-0207"
  }
}
```

Esempi della riga di comando per RDS Proxy

Per vedere come interagiscono le combinazioni di comandi di connessione e istruzioni SQL con RDS Proxy, consulta gli esempi seguenti.

Esempi

- Preserving Connections to a MySQL Database Across a Failover
- Adjusting the max_connections Setting for an Aurora DB Cluster

Example Mantenimento delle connessioni a un database MySQL attraverso un failover

In questo esempio di MySQL viene illustrato come le connessioni aperte continuano a funzionare durante un failover, come quando un database viene riavviato o diventa non disponibile a causa di un problema. In questo esempio viene utilizzato un proxy denominato `the-proxy` e un cluster di database Aurora con istanze DB `instance-8898` e `instance-9814`. Quando il comando `failover-db-cluster` viene eseguito dalla riga di comando di Linux, l'istanza writer a cui il proxy è connesso cambia in un'istanza database diversa. Puoi vedere che l'istanza database associata al proxy cambia mentre la connessione rimane aperta.

```
$ mysql -h the-proxy.proxy-demo.us-east-1.rds.amazonaws.com -u admin_user -p
Enter password:
...

mysql> select @@aurora_server_id;
+--------------------+
| @@aurora_server_id |
+--------------------+
| instance-9814      |
+--------------------+
1 row in set (0.01 sec)

mysql>
[1]+  Stopped                 mysql -h the-proxy.proxy-demo.us-east-1.rds.amazonaws.com
 -u admin_user -p
$ # Initially, instance-9814 is the writer.
$ aws rds failover-db-cluster --db-cluster-identifier cluster-56-2019-11-14-1399
JSON output
$ # After a short time, the console shows that the failover operation is complete.
$ # Now instance-8898 is the writer.
$ fg
mysql -h the-proxy.proxy-demo.us.us-east-1.rds.amazonaws.com -u admin_user -p

mysql> select @@aurora_server_id;
+--------------------+
| @@aurora_server_id |
+--------------------+
| instance-8898      |
+--------------------+
1 row in set (0.01 sec)

mysql>
```

```
[1]+  Stopped                 mysql -h the-proxy.proxy-demo.us-east-1.rds.amazonaws.com
 -u admin_user -p
$ aws rds failover-db-cluster --db-cluster-identifier cluster-56-2019-11-14-1399
JSON output
$ # After a short time, the console shows that the failover operation is complete.
$ # Now instance-9814 is the writer again.
$ fg
mysql -h the-proxy.proxy-demo.us-east-1.rds.amazonaws.com -u admin_user -p

mysql> select @@aurora_server_id;
+--------------------+
| @@aurora_server_id |
+--------------------+
| instance-9814      |
+--------------------+
1 row in set (0.01 sec)
+---------------+---------------+
| Variable_name | Value         |
+---------------+---------------+
| hostname      | ip-10-1-3-178 |
+---------------+---------------+
1 row in set (0.02 sec)
```

Example Regolazione dell'impostazione max_connections per un cluster di database Aurora

In questo esempio viene illustrato come è possibile regolare l'impostazione `max_connections` per un cluster di database Aurora MySQL. A tale scopo, crei il gruppo di parametri del cluster di database in base alle impostazioni dei parametri predefinite per i cluster compatibili con MySQL 5.7. Specifichi un valore per l'impostazione `max_connections`, sovrascrivendo la formula che imposta il valore predefinito. Associ il gruppo di parametri del cluster di database al cluster di database.

```
export REGION=us-east-1
export CLUSTER_PARAM_GROUP=rds-proxy-mysql-57-max-connections-demo
export CLUSTER_NAME=rds-proxy-mysql-57

aws rds create-db-parameter-group --region $REGION \
  --db-parameter-group-family aurora-mysql5.7 \
  --db-parameter-group-name $CLUSTER_PARAM_GROUP \
  --description "Aurora MySQL 5.7 cluster parameter group for RDS Proxy demo."

aws rds modify-db-cluster --region $REGION \
  --db-cluster-identifier $CLUSTER_NAME \
  --db-cluster-parameter-group-name $CLUSTER_PARAM_GROUP
```

```
echo "New cluster param group is assigned to cluster:"
aws rds describe-db-clusters --region $REGION \
  --db-cluster-identifier $CLUSTER_NAME \
  --query '*[*].{DBClusterParameterGroup:DBClusterParameterGroup}'

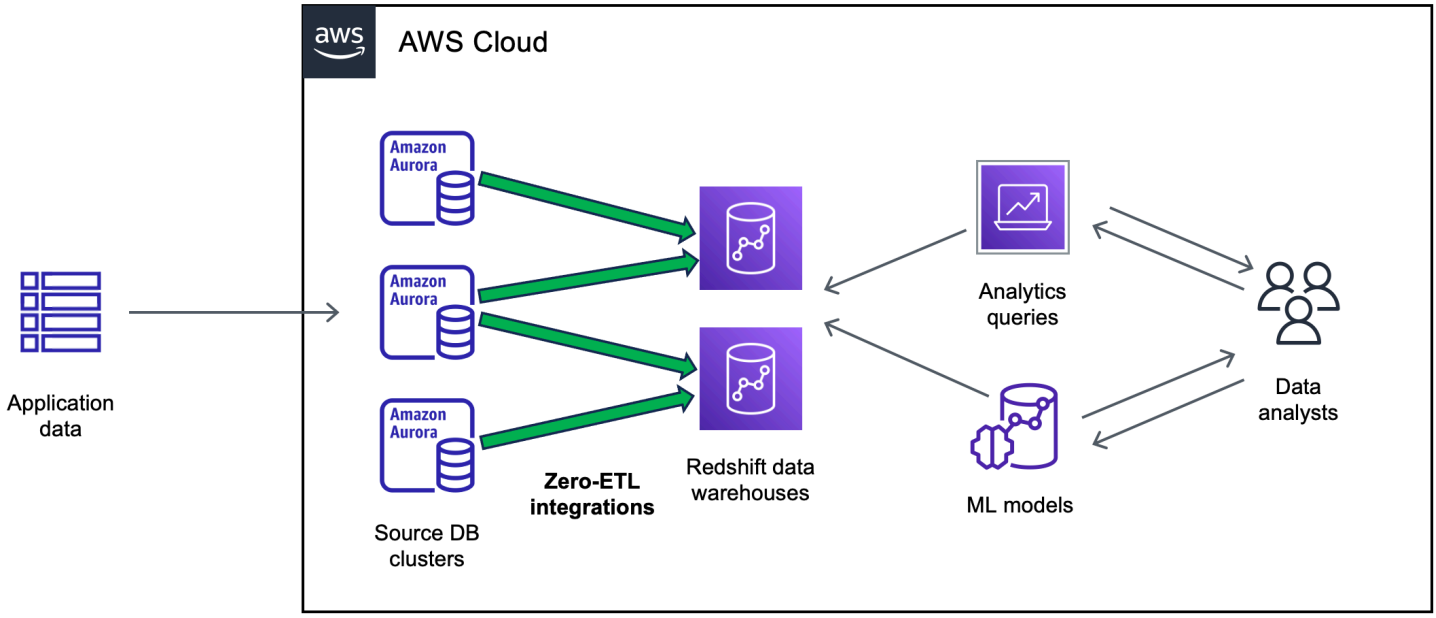
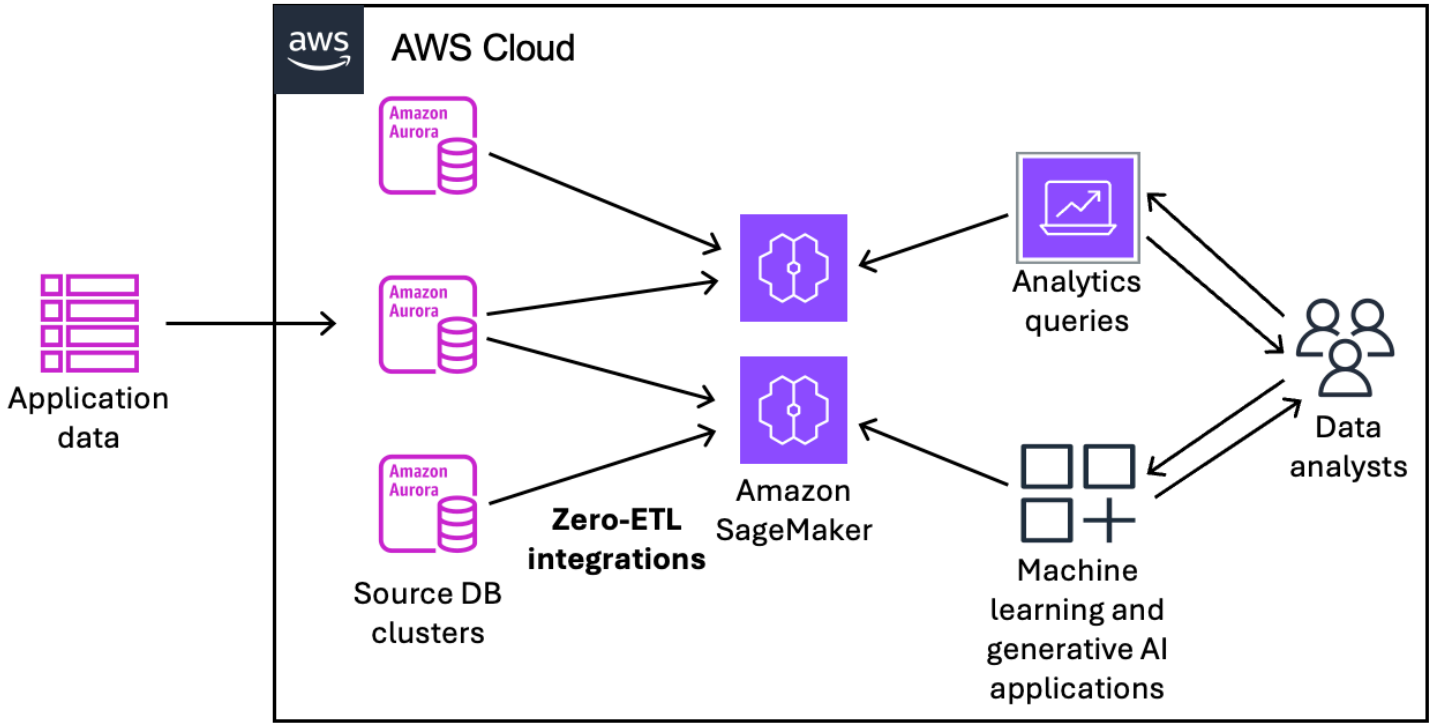
echo "Current value for max_connections:"
aws rds describe-db-cluster-parameters --region $REGION \
  --db-cluster-parameter-group-name $CLUSTER_PARAM_GROUP \
  --query '*[*].{ParameterName:ParameterName,ParameterValue:ParameterValue}' \
  --output text | grep "^max_connections"

echo -n "Enter number for max_connections setting: "
read answer

aws rds modify-db-cluster-parameter-group --region $REGION --db-cluster-parameter-
group-name $CLUSTER_PARAM_GROUP \
  --parameters "ParameterName=max_connections,ParameterValue=$
$answer,ApplyMethod=immediate"

echo "Updated value for max_connections:"
aws rds describe-db-cluster-parameters --region $REGION \
  --db-cluster-parameter-group-name $CLUSTER_PARAM_GROUP \
  --query '*[*].{ParameterName:ParameterName,ParameterValue:ParameterValue}' \
  --output text | grep "^max_connections"
```

Risoluzione dei problemi per RDS Proxy

Di seguito, sono disponibili idee per la risoluzione di alcuni problemi comuni relativi al proxy RDS e informazioni sui CloudWatch registri di RDS Proxy.

Nei log RDS Proxy, ogni voce è preceduta dal nome dell'endpoint proxy associato. Questo nome può essere quello specificato per un endpoint definito dall'utente. In alternativa, può essere il nome `default` speciale dell'endpoint predefinito di un proxy che esegue le richieste. read/write Per ulteriori informazioni sugli endpoint proxy, consulta Utilizzo degli endpoint Amazon RDS Proxy.

Argomenti

- Verifica della connettività a un proxy
- Problemi e soluzioni comuni
- Risoluzione dei problemi di Server proxy per RDS con RDS per MySQL

- Risoluzione dei problemi di Server proxy per RDS con RDS per PostgreSQL

Verifica della connettività a un proxy

Per verificare che tutti i componenti, come il proxy, il database e le istanze di calcolo nella connessione, possano comunicare tra loro, è possibile utilizzare i seguenti comandi.

Esamina il proxy stesso usando il describe-db-proxiescomando. Esamina anche il gruppo target associato utilizzando il comando describe-db-proxy-target-groups. Verifica che i dettagli delle destinazioni corrispondano al cluster Aurora che intendi associare al proxy. Utilizzare comandi come i seguenti.

```
aws rds describe-db-proxies --db-proxy-name $DB_PROXY_NAME
aws rds describe-db-proxy-target-groups --db-proxy-name $DB_PROXY_NAME
```

Per confermare che il proxy è in grado di connettersi al database sottostante, esamina le destinazioni specificate nei gruppi di destinazione utilizzando il describe-db-proxy-targetscomando. Utilizzare un comando come il seguente.

```
aws rds describe-db-proxy-targets --db-proxy-name $DB_PROXY_NAME
```

L'output del describe-db-proxy-targetscomando include un `TargetHealth` campo. È possibile esaminare i campi `State`, `Reason` e `Description` all'interno di `TargetHealth` per verificare se il proxy può comunicare con l'istanza database sottostante.

- Un valore `State` di `AVAILABLE` indica che il proxy può connettersi all'istanza database.
- Un valore `State` di `UNAVAILABLE` indica un problema di connessione temporaneo o permanente. In questo caso, esaminare i campi `Reason` e `Description`. Ad esempio, se `Reason` ha un valore pari a `PENDING_PROXY_CAPACITY`, provare a connettersi nuovamente dopo che il proxy ha terminato l'operazione di ridimensionamento. Se `Reason` ha un valore di `UNREACHABLE`, `CONNECTION_FAILED` o `AUTH_FAILURE`, utilizzare la spiegazione del campo `Description` per facilitare la diagnosi del problema.
- Il valore del campo `State` potrebbe essere `REGISTERING` per un breve periodo prima di passare a `AVAILABLE` o `UNAVAILABLE`.

Se il seguente comando Netcat (`nc`) ha esito positivo, puoi accedere all'endpoint proxy dall' EC2 istanza o da un altro sistema a cui hai effettuato l'accesso. Questo comando segnala un errore se

non ti trovi nello stesso VPC del proxy e del database associato. Potresti essere in grado di accedere direttamente al database senza essere nello stesso VPC. Tuttavia, non puoi accedere al proxy a meno che non ti trovi nello stesso VPC.

```
nc -zx MySQL_proxy_endpoint 3306

nc -zx PostgreSQL_proxy_endpoint 5432
```

Puoi usare i seguenti comandi per assicurarti che l' EC2 istanza abbia le proprietà richieste. In particolare, il VPC dell' EC2 istanza deve essere lo stesso del VPC per il cluster Aurora dell'istanza DB RDS a cui si connette il proxy.

```
aws ec2 describe-instances --instance-ids your_ec2_instance_id
```

Esamina i segreti Secrets Manager utilizzati per il proxy.

```
aws secretsmanager list-secrets
aws secretsmanager get-secret-value --secret-id your_secret_id
```

Assicurati che il campo `SecretString` visualizzato da `get-secret-value` sia codificato come stringa JSON che include i campi `username` e `password`. Nell'esempio seguente viene illustrato il formato del campo `SecretString`.

```
{
  "ARN": "some_arn",
  "Name": "some_name",
  "VersionId": "some_version_id",
  "SecretString": '{"username":"some_username","password":"some_password"}',
  "VersionStages": [ "some_stage" ],
  "CreatedDate": some_timestamp
}
```

Durante la risoluzione dei problemi di autenticazione IAM, verifica quanto segue:

- Nel database è abilitata l'autenticazione IAM.
- Il proxy è configurato con lo schema di autenticazione corretto.
- Le politiche IAM nel ruolo IAM fornite al proxy concedono le `rds-db:connect` autorizzazioni necessarie al database appropriato e al relativo nome utente.

- Per l'autenticazione end-to-end IAM, esistono utenti del database che corrispondono ai nomi degli utenti o dei ruoli IAM.
- SSL/TLS è abilitato per la connessione.

Problemi e soluzioni comuni

Questa sezione descrive alcuni problemi comuni e potenziali soluzioni quando si utilizza Server proxy per RDS.

Dopo l'esecuzione del comando `aws rds describe-db-proxy-targets` della CLI, se la descrizione `TargetHealth` riporta `Proxy does not have any registered credentials`, verifica quanto segue:

- L'utente dispone di credenziali registrate per l'accesso al proxy.
- Il ruolo IAM per l'accesso al segreto Secrets Manager utilizzato dal proxy è valido.

È possibile che si verifichino i seguenti eventi RDS durante la creazione di o la connessione a un proxy DB.

Categoria	ID evento RDS	Description
errore	RDS-EVENT-0243	RDS non è stato in grado di allocare la capacità per il proxy perché non ci sono sufficienti indirizzi IP disponibili nelle sottoreti. Per risolvere il problema, assicurati che le sottoreti abbiano il numero minimo di indirizzi IP non utilizzati. Per determinare il numero consigliato per la classe di istanza, consulta Pianificazione della capacità degli indirizzi IP.
errore	RDS-EVENT-0275	RDS ha limitato alcune connessioni al proxy DB.

Categoria	ID evento RDS	Description
		name Il numero di richieste di connessione simultanee dal client al proxy ha superato il limite.

È possibile che si verifichino i seguenti problemi durante la creazione di un nuovo proxy o la connessione a un proxy.

Errore	Cause o soluzioni alternative
`403: The security token included in the request is invalid`	Seleziona un ruolo IAM esistente invece di crearne uno nuovo.

Risoluzione dei problemi di Server proxy per RDS con RDS per MySQL

È possibile che si verifichino i seguenti problemi durante la connessione a un proxy MySQL.

Errore	Cause o soluzioni alternative
`ERROR 1040 (HY000): Connections rate limit exceeded (`*`limit_value`*` )`	La velocità di richieste di connessione dal client al proxy ha superato il limite.
`ERROR 1040 (HY000): IAM authentication rate limit exceeded`	Il numero di richieste simultanee con autenticazione IAM dal client al proxy ha superato il limite.

Errore	Cause o soluzioni alternative
`ERROR 1040 (HY000): Number simultaneous connections exceeded (`*`limit_value`*` )`	Il numero di richieste di connessione simultanee dal client al proxy ha superato il limite.
`ERROR 1045 (28000): Access denied for user '`*`DB_USER`*`'@'%' (using password: YES)`	Il segreto Secrets Manager utilizzato dal proxy non corrisponde al nome utente e alla password di un utente di database esistente. Aggiorna le credenziali nel segreto Secrets Manager o assicurati che l'utente del database esista e disponga della stessa password del segreto.
`ERROR 1105 (HY000): Unknown error`	Si è verificato un errore sconosciuto.
`ERROR 1231 (42000): Variable ''character_set_client'' can't be set to the value of `*`value`*	Il valore impostato per il parametro `character_set_client` non è valido. Ad esempio, il valore `ucs2` non è valido perché può provocare un arresto anomalo del server MySQL.
`ERROR 3159 (HY000): This RDS Proxy requires TLS connections.`	Hai abilitato l'impostazione Richiedi Transport Layer Security nel proxy ma la tua connessione includeva il parametro `ssl-mode=DISABLED` nel client MySQL. Eseguire una delle operazioni seguenti: • Disattivare l'impostazione Richiedi Transport Layer Security per il proxy. • Connettersi al database utilizzando l'impostazione minima di `ssl-mode=REQUIRED` nel client MySQL.

Errore	Cause o soluzioni alternative
`ERROR 2026 (HY000): SSL connection error: Internal Server` *`Error`*	L'handshake TLS con il proxy non è riuscito. Alcuni possibili motivi includono quanto segue: • SSL è richiesto ma il server non lo supporta. • Si è verificato un errore interno del server. • Si è verificato un handshake non valido.
`ERROR 9501 (HY000): Timed-out waiting to acquire database connection`	Il proxy è in attesa di acquisire una connessione al database. Alcuni possibili motivi includono quanto segue: • Il proxy non è in grado di stabilire una connessione al database perché sono state raggiunte le connessioni massime. • Il proxy non è in grado di stabilire una connessione al database perché il database non è disponibile.

Risoluzione dei problemi di Server proxy per RDS con RDS per PostgreSQL

È possibile che si verifichino i seguenti problemi durante la connessione a un proxy PostgreSQL.

Errore	Causa	Soluzione
`ERROR 28000: IAM authentication is allowed only with SSL connections.`	L'utente ha tentato di connettersi al database utilizzando l'autenticazione IAM con l'impostazione `sslmode=disable` nel client PostgreSQL.	L'utente deve connettersi al database utilizzando l'impostazione minima di `sslmode=require` nel client PostgreSQL. Per ulteriori informazioni, consulta la documentazione Supporto SSL PostgreSQL.
`ERROR 28000: This RDS proxy has no credentials for the role` *`role_name`* `. Check the`	Non c'è un Secrets Manager segreto per questo ruolo.	Aggiungere un Secrets Manager segreto per questo ruolo. Per ulteriori informazioni, consulta Configurazione dell'autenticazione IAM per Server proxy per RDS.

Errore	Causa	Soluzione
`credentials for this role and try again.`		
`ERROR 28000: RDS supports only IAM, MD5, or SCRAM authentication.`	Il client di database utilizzato per connettersi al proxy utilizza un meccanismo di autenticazione non attualmente supportato dal proxy.	Se non utilizzi l'autenticazione IAM, utilizza l'autenticazione tramite password MD5 o SCRAM.
`ERROR 28000: A user name is missing from the connection startup packet. Provide a user name for this connection.`	Il client di database utilizzato per connettersi al proxy non invia un nome utente quando si tenta di stabilire una connessione.	Assicurarsi di definire un nome utente quando si imposta una connessione al proxy utilizzando il client PostgreSQL di propria scelta.
`ERROR 28000: IAM is allowed only with SSL connections.`	Un client ha tentato di connettersi utilizzando l'autenticazione IAM, ma SSL non è stato abilitato.	Abilitare SSL nel client PostgreSQL.
`ERROR 28000: This RDS Proxy requires TLS connections.`	L'utente ha abilitato l'impostazione Richiedi Transport Layer Security ma ha tentato di connettersi con `sslmode=disable` nel client PostgreSQL.	Per risolvere questo errore, effettuare una delle seguenti operazioni: • Disattivare l'impostazione Richiedi Transport Layer Security del proxy. • Connettersi al database utilizzando l'impostazione minima di `sslmode=allow` nel client PostgreSQL.

Errore	Causa	Soluzione
`ERROR 28P01: IAM authentication failed for user` *`user_name`* `. Check the IAM token for this user and try again.`	Questo errore potrebbe essere dovuto ai seguenti fattori: • Il client ha fornito il nome utente IAM non corretto. • Il client ha fornito un token di autorizzazione IAM non corretto per l'utente • Il client utilizza una policy IAM che non dispone delle autorizzazioni necessarie. • Il client ha fornito un token di autorizzazione IAM scaduto per l'utente.	Per correggere questo errore, effettuare le seguenti operazioni: 1. Verificare che l'utente IAM fornito esista. 2. Verificare che il token di autorizzazione IAM appartenga all'utente IAM fornito. 3. Verificare che la policy IAM disponga di autorizzazioni adeguate per RDS. 4. Verificare la validità del token di autorizzazione IAM utilizzato.
`ERROR 28P01: The password that was provided for the role` *`role_name`* `is wrong.`	La password per questo ruolo non corrisponde al segreto Secrets Manager.	Controlla il segreto per questo ruolo in Secrets Manager per vedere se la password è uguale a quella utilizzata nel client PostgreSQL.
`ERROR 28P01: The IAM authentication failed for the role` *`role_name`* `. Check the IAM token for this role and try again.`	Si è verificato un problema con il token IAM utilizzato per l'autenticazione IAM.	Generare un nuovo token di autenticazione e utilizzarlo in una nuova connessione.

Errore	Causa	Soluzione
`ERROR 0A000: Feature not supported: RDS Proxy supports only version 3.0 of the PostgreSQL messaging protocol.`	Il client PostgreSQL utilizzato per connettersi al proxy utilizza un protocollo precedente a 3.0.	Utilizzare un client PostgreSQL più recente che supporti il protocollo di messaggistica 3.0. Se si utilizza la CLI `psql` di PostgreSQL, utilizzare una versione maggiore o uguale a 7.4.
`ERROR 0A000: Feature not supported: RDS Proxy currently doesn't support streaming replication mode.`	Il client PostgreSQL utilizzato per connettersi al proxy sta tentando di utilizzare la modalità di replica in streaming, che non è attualmente supportata dal proxy RDS.	Disattivare la modalità di replica in streaming nel client PostgreSQL utilizzato per la connessione.
`ERROR 0A000: Feature not supported: RDS Proxy currently doesn't support the option` *`option_name`* `.`	Tramite il messaggio di avvio, il client PostgreSQL utilizzato per connettersi al proxy richiede un'opzione che non è attualmente supportata dal proxy RDS.	Disattivare l'opzione visualizzata come non supportata dal messaggio precedente nel client PostgreSQL utilizzato per connettersi.
`ERROR 53300: The IAM authentication failed because of too many competing requests.`	Il numero di richieste simultanee con autenticazione IAM dal client al proxy ha superato il limite.	Ridurre la velocità con cui vengono stabilite le connessioni che utilizzano l'autenticazione IAM da un client PostgreSQL.
`ERROR 53300: The maximum number of client connections to the proxy exceeded` *`number_value`* `.`	Il numero di richieste di connessione simultanee dal client al proxy ha superato il limite.	Ridurre il numero di connessioni attive dai client PostgreSQL a questo proxy RDS.

Errore	Causa	Soluzione
`ERROR 53300: Rate of connection to proxy exceeded` *`number_value`* `.`	La velocità di richieste di connessione dal client al proxy ha superato il limite.	Ridurre la velocità con cui vengono stabilite le connessioni da un client PostgreSQL.
`ERROR XX000: Unknown error.`	Si è verificato un errore sconosciuto.	Contatta l' AWS assistenza per esaminare il problema.
`ERROR 08000: Timed-out waiting to acquire database connection.`	Il proxy è entrato in timeout durante l'attesa di acquisire una connessione al database entro la durata specificata dall'impostazione `ConnectionBorrowTimeout` . Alcuni possibili motivi includono quanto segue: • Il proxy non può stabilire una connessione al database perché sono state raggiunte le connessioni massime. • Il proxy non riesce a stabilire una connessione al database perché il database non è disponibile o se la connessione al database richiede più tempo rispetto al valore configurato per `ConnectionBorrowTimeout` .	Le possibili soluzioni sono le seguenti: • Evita di bloccare le connessioni proxy. Per informazioni, consulta Evitare di effettuare il pinning di un Server proxy per RDS. • Verifica le impostazioni di `ConnectionBorrowTimeout` e `MaxConnectionsPercent` . Per informazioni, consulta Considerazioni sulla connessione a Server proxy per RDS. • Verifica la disponibilità della destinazione. Consulta `AvailabilityPercentage` in Monitoraggio dei parametri RDS Proxy con Amazon CloudWatch.

Errore	Causa	Soluzione
`ERROR XX000: Request returned an error: database_error .`	La connessione al database stabilita dal proxy ha restituito un errore.	La soluzione dipende dall'errore specifico del database. Un esempio : `Request returned an error: database "your-database-name" does not exist`. Ciò significa che il nome del database specificato non esiste nel server di database oppure che il nome utente utilizzato come nome del database (se non è specificato un nome del database) non esiste nel server.
`ERROR 53300: The IAM authentication failed because of too many competing requests.`	Il numero di richieste simultanee con autenticazione IAM dal client al proxy ha superato il limite.	Ridurre la velocità con cui vengono stabilite le connessioni che utilizzano l'autenticazione IAM da un client PostgreSQL.
`ERROR 28000: Enable IAM authentication for the client connection to the proxy and try again.`	Il proxy RDS non può connettersi al database perché l'autenticazione IAM non è abilitata per la connessione del client al proxy. Ciò si verifica quando il `DefaultAuthScheme` parametro del proxy è impostato `IAM_AUTH` su un utente registrato, ma il client utilizza l'autenticazione tramite password anziché l'autenticazione IAM.	Abilita l'autenticazione IAM per la connessione del client al proxy e riprova.

Errore	Causa	Soluzione
`ERROR 28000: Configure IAM authentication as the DefaultAuthScheme in your proxy and try again.`	Il proxy RDS non può connettersi al database perché `DefaultAuthScheme` non è impostato `IAM_AUTH` su. Il `DefaultAuthScheme` parametro del proxy è impostato su`NONE`, ma il client sta tentando di utilizzare l'autenticazione IAM.	Imposta `DefaultAuthScheme` su `IAM_AUTH` per il tuo proxy e riprova.

Risoluzione dei problemi relativi al database **`postgres`** eliminato

Se elimini erroneamente il database `postgres` dall'istanza, devi ripristinare il database per ristabilire la connettività all'istanza. Esegui i comandi seguenti all'interno dell'istanza database:

```
CREATE DATABASE postgres;
GRANT CONNECT ON DATABASE postgres TO rdsproxyadmin;
```

Utilizzo di RDS Proxy con AWS CloudFormation

Puoi usare RDS Proxy con AWS CloudFormation. In questo modo è possibile creare gruppi di risorse correlate. Un gruppo di questo tipo può includere un proxy che può connettersi a un cluster di database Aurora appena creato. Il supporto di RDS Proxy in CloudFormation coinvolge due nuovi tipi di registro: `DBProxy` e `DBProxyTargetGroup`.

L'elenco seguente mostra un modello CloudFormation di esempio per RDS Proxy.

```
Resources:
 DBProxy:
   Type: AWS::RDS::DBProxy
   Properties:
     DBProxyName: CanaryProxy
     EngineFamily: MYSQL
     RoleArn:
      Fn::ImportValue: SecretReaderRoleArn
     Auth:
```

```
      - {AuthScheme: SECRETS, SecretArn: !ImportValue ProxySecret, IAMAuth: DISABLED}
    VpcSubnetIds:
      Fn::Split: [",", "Fn::ImportValue": SubnetIds]

ProxyTargetGroup:
  Type: AWS::RDS::DBProxyTargetGroup
  Properties:
    DBProxyName: CanaryProxy
    TargetGroupName: default
    DBInstanceIdentifiers:
      - Fn::ImportValue: DBInstanceName
  DependsOn: DBProxy
```

Per ulteriori informazioni sulle risorse di questo esempio, consulta `DBProxy` e `DBProxyTargetGroup`.

Per ulteriori informazioni sulle risorse che è possibile creare utilizzando CloudFormation, consulta RDS resource type reference.

Utilizzo del Server proxy per RDS con i database globali Aurora

Un database globale Aurora è un singolo database che comprende più Regioni AWS, consentendo letture globali a bassa latenza e ripristino di emergenza da interruzioni dell'attività a livello regionale. Fornisce tolleranza ai guasti incorporata per la distribuzione perché l'istanza database non si basa su una singola Regione AWS, ma su più Regioni e zone di disponibilità diverse. Per ulteriori informazioni, consulta Utilizzo di Database globale Amazon Aurora.

È possibile utilizzare Server proxy per RDS con qualsiasi cluster database in un database globale Aurora. Prima di iniziare a utilizzare queste funzionalità, assicurati di verificare le seguenti informazioni.

> **Important**
>
> Se il cluster database fa parte di un database globale con l'inoltro di scrittura attivato, riduci il valore `MaxConnectionsPercent` del proxy con la quota assegnata all'inoltro di scrittura. La quota di inoltro di scrittura è impostata nel parametro `aurora_fwd_writer_max_connections_pct` del cluster database. Per informazioni sull'inoltro di scrittura, consulta Utilizzo dell'inoltro di scrittura in un database globale Amazon Aurora.

Limitazioni di Server proxy per RDS con i database globali

Quando il cluster database Aurora ha l'inoltro di scrittura attivato, Server proxy per RDS non supporta il valore `SESSION` per la variabile `aurora_replica_read_consistency`, la cui impostazione può causare un comportamento imprevisto.

Come funzionano gli endpoint Server proxy per RDS con i database globali

Se comprendi come gli endpoint Server proxy per RDS funzionano con i database globali, puoi gestire meglio le tue applicazioni che utilizzano i database Aurora con entrambe queste funzionalità.

Per un proxy con il cluster primario di un database globale come destinazione registrata, gli endpoint proxy funzionano allo stesso modo che per qualsiasi cluster database Aurora. Gli endpoint di lettura/scrittura del proxy inviano tutte le richieste all'istanza di scrittura del cluster. Gli endpoint di sola lettura del proxy inviano tutte le richieste alle istanze di lettura. Se un'istanza di lettura non è disponibile mentre la connessione è aperta, Server proxy per RDS reindirizza le query successive sulla connessione a un'altra istanza di lettura. Per un proxy con un cluster secondario come destinazione registrata, le richieste inviate agli endpoint di sola lettura del proxy vengono inviate anche alle istanze di lettura. Poiché il cluster non ha istanze di scrittura, le richieste inviate agli endpoint di lettura/scrittura non riescono e viene restituito l'errore "`The target group doesn't have any associated read/write instances`".

Le operazioni di failover e switchover globale del database prevedono entrambe un cambio di ruolo tra il cluster database primario e uno dei cluster database secondari. Quando il cluster secondario selezionato diventa il nuovo cluster primario, una delle istanze di lettura viene promossa a istanza di scrittura. Questa istanza database è ora la nuova istanza di scrittura del cluster globale. Assicurati di reindirizzare le operazioni di scrittura dell'applicazione all'endpoint di lettura/scrittura appropriato del proxy associato al nuovo cluster primario. Questo endpoint del proxy può essere l'endpoint predefinito o un endpoint di lettura/scrittura personalizzato.

Server proxy per RDS mette in coda tutte le richieste tramite endpoint di lettura/scrittura e le invia all'istanza di scrittura del nuovo cluster primario non appena è disponibile. Ciò avviene indipendentemente dal completamento dell'operazione di failover o switchover. Durante il failover o lo switchover, l'endpoint predefinito del proxy per il vecchio cluster primario accetta ancora le operazioni di scrittura. Tuttavia, non appena quel cluster diventa secondario, tutte le operazioni di scrittura restituiscono esito negativo. Per informazioni su come e quando eseguire specifiche operazioni di failover o switchover globali, consulta i seguenti argomenti:

- Switchover globale del database: Esecuzione di switchover per database globali Amazon Aurora

- Failover globale del database: Ripristino di un database globale Amazon Aurora da un'interruzione non pianificata

Integrazioni Zero-ETL di Aurora

È una soluzione completamente gestita per rendere disponibili i dati transazionali nella destinazione di analisi dopo la scrittura su un cluster di database Aurora. Estrazione, trasformazione e caricamento (ETL) è il processo di combinazione di dati provenienti da più origini in un grande data warehouse centrale.

Un'integrazione Zero-ETL rende disponibili i dati del cluster di database Aurora in Amazon Redshift o in un lakehouse Amazon SageMaker AI quasi in tempo reale. Una volta che i dati sono nel data warehouse o data lake di destinazione, puoi potenziare i tuoi carichi di lavoro di analisi, ML e intelligenza artificiale utilizzando le funzionalità integrate, come machine learning, viste materializzate, condivisione dei dati, accesso federato a più data store e data lake e integrazioni con SageMaker Amazon AI, Quick Suite e altro. Servizi AWS

Per creare un'integrazione Zero-ETL, è necessario specificare un cluster di database Aurora come origine e un lakehouse o un data warehouse supportato come destinazione. L'integrazione replica i dati dal database di origine nel lakehouse o data warehouse di destinazione.

Il diagramma seguente illustra questa funzionalità per l'integrazione Zero-ETL con Amazon Redshift:



Il diagramma seguente illustra questa funzionalità per l'integrazione Zero-ETL con un Amazon SageMaker AI Lakehouse:



L'integrazione monitora lo stato della pipeline dei dati ed esegue il ripristino in caso di problemi quando possibile. Puoi creare integrazioni da più cluster di database Aurora in un unico lakehouse o data warehouse di destinazione così da ottenere informazioni dettagliate da diverse applicazioni.

Per ulteriori informazioni sul costo delle integrazioni Zero-ETL, consulta le pagine Prezzi di Amazon Aurora e Prezzi di Amazon Redshift.

Argomenti

- Vantaggi
- Concetti chiave
- Limitazioni
- Quote
- Regioni supportate
- Guida introduttiva alle integrazioni Zero-ETL di Aurora
- Creazione di integrazioni Zero-ETL di Aurora con Amazon Redshift
- Creazione di integrazioni Zero-ETL di Aurora con un lakehouse Amazon SageMaker
- Filtro dei dati per integrazioni Zero-ETL di Aurora
- Aggiunta di dati a un cluster di database Aurora di origine ed esecuzione di query al suo interno

- Visualizzazione e monitoraggio delle integrazioni Zero-ETL Aurora
- Modifica delle integrazioni Zero-ETL di Aurora
- Eliminazione delle integrazioni Zero-ETL di Aurora
- Risoluzione dei problemi delle integrazioni Zero-ETL di Aurora

Vantaggi

Le integrazioni Zero-ETL di Aurora offrono i seguenti vantaggi:

- Ti consentono di ottenere approfondimenti di tipo olistico da più origini dati.
- Eliminano la necessità di creare e gestire pipeline dei dati complesse che eseguono operazioni di estrazione, trasformazione e caricamento (ETL). Le integrazioni Zero-ETL forniscono e gestiscono le pipeline per te, eliminando le sfide legate alla loro creazione e gestione.
- Ti consentono di ridurre il carico e i costi operativi e di concentrarti sul miglioramento delle applicazioni.
- Puoi utilizzare le funzionalità di analisi e machine learning della destinazione per ricavare informazioni dettagliate da dati transazionali e di altro tipo e reagire in modo efficace in caso di eventi critici e urgenti.

Concetti chiave

Per iniziare a utilizzare le integrazioni Zero-ETL, tieni presente i seguenti concetti:

Integrazione

Una pipeline dei dati completamente gestita che replica automaticamente i dati transazionali e gli schemi da un cluster di database Aurora a un data warehouse o un catalogo.

Cluster di database di origine

Il cluster di database Aurora da cui viene eseguita la replica dei dati. È possibile specificare un cluster DB che utilizza istanze DB o istanze DB assegnate come origine. Aurora Serverless v2

Target

Si tratta del lakehouse o data warehouse in cui viene eseguita la replica dei dati. Esistono due tipi di data warehouse: un data warehouse con cluster con provisioning e un data warehouse serverless. Un data warehouse con cluster con provisioning è costituito da un insieme di risorse

di calcolo denominate nodi, strutturate in un gruppo denominato cluster. Un data warehouse serverless è composto da un gruppo di lavoro che archivia le risorse di calcolo e da un spazio dei nomi che ospita gli oggetti e gli utenti del database. Entrambi i data warehouse utilizzano un motore di analisi e contengono uno o più database.

Un lakehouse di destinazione è costituito da cataloghi, database, tabelle e viste. Per ulteriori informazioni sull'architettura lakehouse, consulta SageMaker Lakehouse components nella Guida per l'utente di Amazon SageMaker AI Unified Studio.

Più cluster di database di destinazione possono scrivere nella stessa destinazione.

Per ulteriori informazioni sui nodi principali e sui nodi di calcolo, consulta Architettura del sistema di data warehouse nella Guida per sviluppatori di database di Amazon Redshift.

Limitazioni

Le seguenti limitazioni si applicano alle integrazioni Zero-ETL di Aurora.

Argomenti

- Limitazioni generali
- Aurora MySQL
- Limitazioni di Aurora PostgreSQL
- Limitazioni di Amazon Redshift
- Amazon SageMaker AI limitazioni di Lakehouse

Limitazioni generali

- Il cluster di database di origine deve trovarsi nella stessa Regione della destinazione.
- Non è possibile rinominare un cluster di database né alcuna delle sue istanze se include integrazioni esistenti.
- Non è possibile creare più integrazioni tra gli stessi database di origine e destinazione.
- Non è possibile eliminare un cluster di database con integrazioni esistenti. Devi prima eliminare tutte le integrazioni associate.
- Se il cluster di database di origine viene arrestato, è possibile che le ultime transazioni non vengano replicate nella destinazione finché non si riprende il cluster.

- Se il cluster è l'origine di un'implementazione blu/verde, gli ambienti blu e verde non possono avere integrazioni Zero-ETL esistenti durante lo switchover. Occorre eliminare l'integrazione, eseguire lo switchover e poi ricrearla.
- Un cluster di database deve contenere almeno un'istanza database per essere l'origine di un'integrazione.
- Non è possibile creare un'integrazione per un cluster DB di origine che è un clone tra più account, come quelli condivisi utilizzando (). AWS Resource Access Manager AWS RAM
- Se il cluster di origine è il cluster database primario in un database globale Aurora e esegue il failover su uno dei relativi cluster secondari, l'integrazione diventa inattiva. È necessario eliminare e ricreare l'integrazione.
- Non è possibile creare un'integrazione per un database di origine in cui un'altra integrazione è attivamente in fase di creazione.
- Durante la fase iniziale della creazione di un'integrazione o quando una tabella viene risincronizzata, il seeding dei dati dall'origine alla destinazione può richiedere 20-25 minuti o più, a seconda delle dimensioni del database di origine. Ciò può causare un aumento del ritardo di replica.
- Alcuni tipi di dati non sono supportati. Per ulteriori informazioni, consulta the section called "Differenze dei tipi di dati".
- Le tabelle di sistema, le tabelle temporanee e le viste non vengono replicate nei warehouse di destinazione.
- L'esecuzione di comandi DDL (ad esempio `ALTER TABLE`) su una tabella di origine può attivare una risincronizzazione della tabella. Durante tale operazione non sarà possibile utilizzare la tabella per le query. Per ulteriori informazioni, consulta the section called "Una o più tabelle Amazon Redshift richiedono una risincronizzazione".

Aurora MySQL

- Il cluster di database di origine deve eseguire una versione supportata di Aurora MySQL. Per un elenco delle versioni supportate, consulta the section called "Integrazioni Zero-ETL ".
- Le integrazioni Zero-ETL si basano sui log binari MySQL (binlog) per acquisire le modifiche continue dei dati. Il filtraggio dei dati basato su binlog è sconsigliato, poiché può causare incoerenze nei dati tra il database di origine e quello di destinazione.
- Le integrazioni Zero-ETL sono supportate solo per i database configurati per l'utilizzo del motore di storage InnoDB.

- I riferimenti a chiavi esterne con aggiornamenti di tabella predefiniti non sono supportati. In particolare, le regole `ON DELETE` e `ON UPDATE` non sono supportate nelle operazioni `CASCADE`, `SET NULL` e `SET DEFAULT`. Se si tenta di creare o aggiornare una tabella con tali riferimenti a un'altra tabella, la tabella entrerà in uno stato di errore.
- Le transazioni XA eseguite sul cluster di database di origine fanno sì che l'integrazione entri in uno stato di `Syncing`.

Limitazioni di Aurora PostgreSQL

- Il cluster di database di origine deve eseguire una versione supportata di Aurora PostgreSQL. Per un elenco delle versioni supportate, consulta the section called "Integrazioni Zero-ETL ".
- Se selezioni un cluster di database di origine Aurora PostgreSQL, è necessario specificare almeno un modello di filtro dei dati. Come minimo, il modello deve includere un singolo database (*`database-name`*`.*.*`) per la replica nel warehouse di destinazione. Per ulteriori informazioni, consulta the section called "Filtro dei dati per le integrazioni Zero-ETL".
- Tutti i database creati all'interno del cluster di database di origine Aurora PostgreSQL devono utilizzare la codifica UTF-8.
- Quando utilizzi il partizionamento dichiarativo, le partizioni delle tabelle verranno replicate su Amazon Redshift. Tuttavia, la tabella partizionata stessa non viene replicata su Amazon Redshift.
- Le transazioni in due fasi non sono supportate.
- Se elimini tutte le istanze database da un cluster di database che è l'origine di un'integrazione e successivamente aggiungi di nuovo un'istanza database, la replica tra il cluster di origine e quello di destinazione si interrompe.
- Il cluster di database di origine non può utilizzare Aurora Limitless Database.
- Le chiavi primarie sono obbligatorie su tutte le tabelle presenti nel filtro dati. Tutte le tabelle senza una chiave primaria verranno messe nello stato di errore.

Limitazioni di Amazon Redshift

Per un elenco delle limitazioni di Amazon Redshift relative alle integrazioni Zero-ETL, consulta la pagina delle Considerazioni sull'utilizzo delle integrazioni Zero-ETL con Amazon Redshift nella Guida alla gestione di Amazon Redshift.

Amazon SageMaker AI limitazioni di Lakehouse

Di seguito è riportata una limitazione per le integrazioni Amazon SageMaker AI Lakehouse Zero-ETL.

- I nomi dei cataloghi hanno un limite di lunghezza di 19 caratteri.

Quote

Sul tuo account sono disponibili le seguenti quote relative alle integrazioni Zero-ETL di Aurora. Salvo dove diversamente specificato, ogni quota fa riferimento a una Regione specifica.

Nome	Predefinito	Description
Integrazioni	100	Numero totale di integrazioni all'interno di un Account AWS.
Integrazioni per destinazione	50	Numero di integrazioni che inviano dati a un unico lakehouse o data warehouse di destinazione.
Integrazioni per cluster di origine	5	Numero di integrazioni che inviano dati da un unico cluster di database di origine.

Inoltre, il warehouse di destinazione pone determinati limiti al numero di tabelle consentite in ogni istanza database o nodo del cluster. Per informazioni su quote e limiti di Amazon Redshift, consulta Quote e limiti in Amazon Redshift nella Guida alla gestione di Amazon Redshift.

Regioni supportate

Le integrazioni Aurora Zero-ETL sono disponibili in un sottoinsieme di. Regioni AWS Per un elenco delle regioni supportate, consulta the section called “Integrazioni Zero-ETL ”.

Guida introduttiva alle integrazioni Zero-ETL di Aurora

Prima di creare un'integrazione Zero-ETL, configura il cluster di database Aurora e il data warehouse con le autorizzazioni e i parametri richiesti. Durante la configurazione, dovrai completare i seguenti passaggi:

1. Creazione di un gruppo di parametri per il cluster di database personalizzato.
2. Creazione di un cluster di database di origine.
3. Crea un data warehouse di destinazione per Amazon Redshift o Crea un data warehouse di destinazione Amazon SageMaker AI Lakehouse.

Dopo aver completato queste attività, passa a the section called “Creazione di integrazioni Zero-ETL con Amazon Redshift” o the section called “Creazione di integrazioni Zero-ETL con un lakehouse Amazon SageMaker”.

Puoi utilizzare il AWS SDKs per automatizzare il processo di configurazione per te. Per ulteriori informazioni, consulta the section called “Configura un'integrazione utilizzando AWS SDKs”.

> **Tip**
>
> Puoi fare in modo che RDS completi questi passaggi di configurazione al posto tuo durante la creazione dell’integrazione, in modo da non doverli eseguire manualmente. Per iniziare subito a creare un’integrazione, consulta the section called “Creazione di integrazioni Zero-ETL con Amazon Redshift”.

Per il Passaggio 3, puoi scegliere di creare un data warehouse di destinazione (Passaggio 3a) o un lakehouse di destinazione (Passaggio 3b) a seconda delle tue esigenze:

- Scegli un data warehouse se hai bisogno di funzionalità di data warehousing tradizionali con analisi basate su SQL.
- Scegli un Amazon SageMaker AI lakehouse se hai bisogno di funzionalità di apprendimento automatico e desideri utilizzare le funzionalità di Lakehouse per flussi di lavoro di data science e ML.

Fase 1: creazione di un gruppo di parametri del cluster di database personalizzato

Le integrazioni Zero-ETL di Aurora richiedono valori specifici per i parametri del cluster di database che controllano la replica. In particolare, Aurora MySQL richiede il file di log binario avanzato (`aurora_enhanced_binlog`) e Aurora PostgreSQL richiede la replica logica avanzata (`aurora.enhanced_logical_replication`).

Per configurare la registrazione di log binari o la replica logica, è necessario creare prima un gruppo di parametri del cluster di database personalizzato e poi associarlo al cluster di database di origine.

Aurora MySQL (famiglia aurora-mysql8.0):

- `aurora_enhanced_binlog=1`
- `binlog_backup=0`
- `binlog_format=ROW`
- `binlog_replication_globaldb=0`
- `binlog_row_image=full`
- `binlog_row_metadata=full`

Inoltre, assicurati che il parametro `binlog_transaction_compression` non sia impostato su `ON` e che il parametro `binlog_row_value_options` non sia impostato su `PARTIAL_JSON`.

Per ulteriori informazioni sul file di log binario avanzato di Aurora MySQL, consulta the section called “Configurazione del file di log binario avanzato”.

Aurora PostgreSQL (famiglia aurora-postgresql16):

- `rds.logical_replication=1`
- `aurora.enhanced_logical_replication=1`
- `aurora.logical_replication_backup=0`
- `aurora.logical_replication_globaldb=0`

L’abilitazione della replica logica avanzata (`aurora.enhanced_logical_replication`) scriverà sempre tutti i valori delle colonne nel WAL (write ahead log) anche se `REPLICA IDENTITY FULL` non è abilitata. Ciò potrebbe aumentare le IOPS per il cluster di database di origine.

> **Important**
>
> Se abiliti o disabiliti il parametro del cluster di database `aurora.enhanced_logical_replication`, l’istanza database primaria invalida tutti gli slot di replica logica. Ciò interrompe la replica dall’origine alla destinazione ed è necessario ricreare gli slot di replica sull’istanza database primaria. Per evitare interruzioni, mantieni costante lo stato del parametro durante la replica.

Passaggio 2: seleziona o crea un cluster di database di origine

Dopo aver creato un gruppo di parametri personalizzato per il cluster di database, scegli o crea un cluster di database Aurora. Questo cluster sarà l'origine della replica dei dati nel data warehouse di destinazione. Puoi specificare un cluster di database che utilizza istanze database con provisioning o istanze database Aurora Serverless v2 come origine. Per istruzioni su come creare un cluster di database, consulta the section called "Creazione di un cluster di database" o the section called "Creazione di un cluster di database Aurora Serverless v2".

Il database deve eseguire una versione del motore di database supportata. Per un elenco delle versioni supportate, consulta the section called "Integrazioni Zero-ETL ".

Quando crei il database, in Configurazione aggiuntiva modifica il valore predefinito del gruppo di parametri del cluster di database impostandolo sul gruppo di parametri personalizzato creato nel passaggio precedente.

> **Note**
>
> Se associ il gruppo di parametri al cluster di database dopo che il cluster è già stato creato, è necessario riavviare l'istanza database primaria nel cluster per applicare le modifiche prima di poter creare un'integrazione Zero-ETL. Per istruzioni, consulta the section called "Riavvio di un cluster o di un'istanza Aurora DB".

Passaggio 3a: creazione di un data warehouse di destinazione

Dopo aver creato il cluster di database di origine, è necessario creare e configurare un data warehouse di destinazione. Il data warehouse deve soddisfare i seguenti requisiti:

- Utilizzando un tipo di RA3 nodo con almeno due nodi o Redshift Serverless.
- Deve essere crittografato (se si utilizza un cluster con provisioning). Per ulteriori informazioni, consulta Crittografia dei database di Amazon Redshift.

Per istruzioni su come creare un data warehouse, consulta Creazione di un cluster per i cluster con provisioning o Creazione di un gruppo di lavoro con uno spazio dei nomi per Redshift Serverless.

Abilitazione della distinzione tra maiuscole e minuscole nel data warehouse

Affinché l'integrazione venga eseguita correttamente, il parametro di distinzione tra maiuscole e minuscole (`enable_case_sensitive_identifier`) deve essere abilitato per il data warehouse. Per impostazione predefinita, la distinzione tra maiuscole e minuscole è disabilitata su tutti i cluster con provisioning e sui gruppi di lavoro Redshift serverless.

Per abilitare la distinzione tra maiuscole e minuscole, esegui i seguenti passaggi a seconda del tipo di data warehouse:

- Cluster con provisioning: per abilitare la distinzione tra maiuscole e minuscole su un cluster con provisioning, crea un gruppo di parametri personalizzato con il parametro `enable_case_sensitive_identifier` abilitato. Poi, associa il gruppo di parametri al cluster. Per istruzioni, consulta Gestione di gruppi di parametri mediante la console o Configurazione dei valori di parametro mediante AWS CLI.

  > **Note**
  >
  > Ricordati di riavviare il cluster dopo aver associato il gruppo di parametri personalizzati.

- Gruppo di lavoro serverless: per abilitare la distinzione tra maiuscole e minuscole su un gruppo di lavoro SRedshift Serverless, è necessario utilizzare la AWS CLI. La console Amazon Redshift attualmente non supporta la modifica dei valori dei parametri Redshift Serverless. Invia la seguente richiesta update-workgroup:

  ```
  aws redshift-serverless update-workgroup \
    --workgroup-name target-workgroup \
    --config-parameters
   parameterKey=enable_case_sensitive_identifier,parameterValue=true
  ```

  Non è necessario riavviare un gruppo di lavoro dopo aver modificato i valori dei parametri.

Configura l'autorizzazione per il data warehouse

Dopo aver creato un data warehouse, è necessario configurare un cluster di database Aurora di origine come origine autorizzata dell'integrazione. Per istruzioni, consulta Configurazione dell'autorizzazione per il data warehouse Amazon Redshift.

Configura un'integrazione utilizzando AWS SDKs

Invece di configurare ogni risorsa manualmente, puoi eseguire lo script Python riportato di seguito per configurare automaticamente le risorse richieste. L'esempio di codice utilizza AWS SDK per Python (Boto3) per creare un cluster di database Amazon Aurora di origine e un data warehouse di destinazione, ciascuno con i valori dei parametri richiesti. Attende quindi che i database siano disponibili prima di creare un'integrazione Zero-ETL tra di essi. Puoi commentare diverse funzioni a seconda delle risorse che è necessario configurare.

Per installare le dipendenze richieste, esegui i seguenti comandi:

```
pip install boto3
pip install time
```

All'interno dello script, è possibile modificare i nomi dei gruppi di origine, destinazione e parametri. Una volta che le risorse sono state impostate, la funzione finale crea un'integrazione che viene denominata `my-integration`.

Esempio di codice Python

Aurora MySQL

```
import boto3
import time

# Build the client using the default credential configuration.
# You can use the CLI and run 'aws configure' to set access key, secret
# key, and default Region.

rds = boto3.client('rds')
redshift = boto3.client('redshift')
sts = boto3.client('sts')

source_cluster_name = 'my-source-cluster' # A name for the source cluster
source_param_group_name = 'my-source-param-group' # A name for the source parameter
 group
target_cluster_name = 'my-target-cluster' # A name for the target cluster
target_param_group_name = 'my-target-param-group' # A name for the target parameter
 group

def create_source_cluster(*args):
    """Creates a source Aurora MySQL DB cluster"""
```

```
    response = rds.create_db_cluster_parameter_group(
        DBClusterParameterGroupName=source_param_group_name,
        DBParameterGroupFamily='aurora-mysql8.0',
        Description='For Aurora MySQL binary logging'
    )
    print('Created source parameter group: ' + response['DBClusterParameterGroup']
['DBClusterParameterGroupName'])

    response = rds.modify_db_cluster_parameter_group(
        DBClusterParameterGroupName=source_param_group_name,
        Parameters=[
            {
                'ParameterName': 'aurora_enhanced_binlog',
                'ParameterValue': '1',
                'ApplyMethod': 'pending-reboot'
            },
            {
                'ParameterName': 'binlog_backup',
                'ParameterValue': '0',
                'ApplyMethod': 'pending-reboot'
            },
            {
                'ParameterName': 'binlog_format',
                'ParameterValue': 'ROW',
                'ApplyMethod': 'pending-reboot'
            },
            {
                'ParameterName': 'binlog_replication_globaldb',
                'ParameterValue': '0',
                'ApplyMethod': 'pending-reboot'
            },
            {
                'ParameterName': 'binlog_row_image',
                'ParameterValue': 'full',
                'ApplyMethod': 'pending-reboot'
            },
            {
                'ParameterName': 'binlog_row_metadata',
                'ParameterValue': 'full',
                'ApplyMethod': 'pending-reboot'
            }
        ]
    )
```

```
    print('Modified source parameter group: ' +
 response['DBClusterParameterGroupName'])

    response = rds.create_db_cluster(
        DBClusterIdentifier=source_cluster_name,
        DBClusterParameterGroupName=source_param_group_name,
        Engine='aurora-mysql',
        EngineVersion='8.0.mysql_aurora.3.05.2',
        DatabaseName='myauroradb',
        MasterUsername='username',
        MasterUserPassword='Password01**'
    )
    print('Creating source cluster: ' + response['DBCluster']
['DBClusterIdentifier'])
    source_arn = (response['DBCluster']['DBClusterArn'])
    create_target_cluster(target_cluster_name, source_arn, target_param_group_name)

    response = rds.create_db_instance(
        DBInstanceClass='db.r6g.2xlarge',
        DBClusterIdentifier=source_cluster_name,
        DBInstanceIdentifier=source_cluster_name + '-instance',
        Engine='aurora-mysql'
    )
    return(response)

def create_target_cluster(target_cluster_name, source_arn, target_param_group_name):
    """Creates a target Redshift cluster"""

    response = redshift.create_cluster_parameter_group(
        ParameterGroupName=target_param_group_name,
        ParameterGroupFamily='redshift-1.0',
        Description='For Aurora MySQL zero-ETL integrations'
    )
    print('Created target parameter group: ' + response['ClusterParameterGroup']
['ParameterGroupName'])

    response = redshift.modify_cluster_parameter_group(
        ParameterGroupName=target_param_group_name,
        Parameters=[
            {
                'ParameterName': 'enable_case_sensitive_identifier',
                'ParameterValue': 'true'
            }
        ]
```

```
    )
    print('Modified target parameter group: ' + response['ParameterGroupName'])

    response = redshift.create_cluster(
        ClusterIdentifier=target_cluster_name,
        NodeType='ra3.4xlarge',
        NumberOfNodes=2,
        Encrypted=True,
        MasterUsername='username',
        MasterUserPassword='Password01**',
        ClusterParameterGroupName=target_param_group_name
    )
    print('Creating target cluster: ' + response['Cluster']['ClusterIdentifier'])

    # Retrieve the target cluster ARN
    response = redshift.describe_clusters(
        ClusterIdentifier=target_cluster_name
    )
    target_arn = response['Clusters'][0]['ClusterNamespaceArn']

    # Retrieve the current user's account ID
    response = sts.get_caller_identity()
    account_id = response['Account']

    # Create a resource policy specifying cluster ARN and account ID
    response = redshift.put_resource_policy(
        ResourceArn=target_arn,
        Policy='''
        {
            \"Version\":\"2012-10-17\",
            \"Statement\":[
                {\"Effect\":\"Allow\",
                \"Principal\":{
                    \"Service\":\"redshift.amazonaws.com\"
                },
                \"Action\":[\"redshift:AuthorizeInboundIntegration\"],
                \"Condition\":{
                    \"StringEquals\":{
                        \"aws:SourceArn\":\"%s\"}
                    }
                },
                {\"Effect\":\"Allow\",
                \"Principal\":{
                    \"AWS\":\"arn:aws:iam::%s:root\"},
```

```
                \"Action\":\"redshift:CreateInboundIntegration\"}
            ]
        }
        ''' % (source_arn, account_id)
    )
    return(response)

def wait_for_cluster_availability(*args):
    """Waits for both clusters to be available"""

    print('Waiting for clusters to be available...')

    response = rds.describe_db_clusters(
        DBClusterIdentifier=source_cluster_name
    )
    source_status = response['DBClusters'][0]['Status']
    source_arn = response['DBClusters'][0]['DBClusterArn']

    response = rds.describe_db_instances(
        DBInstanceIdentifier=source_cluster_name + '-instance'
    )
    source_instance_status = response['DBInstances'][0]['DBInstanceStatus']

    response = redshift.describe_clusters(
        ClusterIdentifier=target_cluster_name
    )
    target_status = response['Clusters'][0]['ClusterStatus']
    target_arn = response['Clusters'][0]['ClusterNamespaceArn']

    # Every 60 seconds, check whether the clusters are available.
    if source_status != 'available' or target_status != 'available' or
 source_instance_status != 'available':
        time.sleep(60)
        response = wait_for_cluster_availability(
            source_cluster_name, target_cluster_name)
    else:
        print('Clusters available. Ready to create zero-ETL integration.')
        create_integration(source_arn, target_arn)
        return

def create_integration(source_arn, target_arn):
    """Creates a zero-ETL integration using the source and target clusters"""

    response = rds.create_integration(
```

```
        SourceArn=source_arn,
        TargetArn=target_arn,
        IntegrationName='my-integration'
    )
    print('Creating integration: ' + response['IntegrationName'])

def main():
    """main function"""
    create_source_cluster(source_cluster_name, source_param_group_name)
    wait_for_cluster_availability(source_cluster_name, target_cluster_name)

if __name__ == "__main__":
    main()
```

Aurora PostgreSQL

```
import boto3
import time

# Build the client using the default credential configuration.
# You can use the CLI and run 'aws configure' to set access key, secret
# key, and default Region.

rds = boto3.client('rds')
redshift = boto3.client('redshift')
sts = boto3.client('sts')

source_cluster_name = 'my-source-cluster' # A name for the source cluster
source_param_group_name = 'my-source-param-group' # A name for the source parameter
 group
target_cluster_name = 'my-target-cluster' # A name for the target cluster
target_param_group_name = 'my-target-param-group' # A name for the target parameter
 group

def create_source_cluster(*args):
    """Creates a source Aurora PostgreSQL DB cluster"""

    response = rds.create_db_cluster_parameter_group(
        DBClusterParameterGroupName=source_param_group_name,
        DBParameterGroupFamily='aurora-postgresql16',
        Description='For Aurora PostgreSQL logical replication'
    )
```

```
    print('Created source parameter group: ' + response['DBClusterParameterGroup']
['DBClusterParameterGroupName'])

    response = rds.modify_db_cluster_parameter_group(
        DBClusterParameterGroupName=source_param_group_name,
        Parameters=[
            {
                'ParameterName': 'rds.logical_replication',
                'ParameterValue': '1',
                'ApplyMethod': 'pending-reboot'
            },
            {
                'ParameterName': 'aurora.enhanced_logical_replication',
                'ParameterValue': '1',
                'ApplyMethod': 'pending-reboot'
            },
            {
                'ParameterName': 'aurora.logical_replication_backup',
                'ParameterValue': '0',
                'ApplyMethod': 'pending-reboot'
            },
            {
                'ParameterName': 'aurora.logical_replication_globaldb',
                'ParameterValue': '0',
                'ApplyMethod': 'pending-reboot'
            }
        ]
    )
    print('Modified source parameter group: ' +
 response['DBClusterParameterGroupName'])

    response = rds.create_db_cluster(
        DBClusterIdentifier=source_cluster_name,
        DBClusterParameterGroupName=source_param_group_name,
        Engine='aurora-postgresql',
        EngineVersion='16.4.aurora-postgresql',
        DatabaseName='mypostgresdb',
        MasterUsername='username',
        MasterUserPassword='Password01**'
    )
    print('Creating source cluster: ' + response['DBCluster']
['DBClusterIdentifier'])
    source_arn = (response['DBCluster']['DBClusterArn'])
    create_target_cluster(target_cluster_name, source_arn, target_param_group_name)
```

```
    response = rds.create_db_instance(
        DBInstanceClass='db.r6g.2xlarge',
        DBClusterIdentifier=source_cluster_name,
        DBInstanceIdentifier=source_cluster_name + '-instance',
        Engine='aurora-postgresql'
    )
    return(response)

def create_target_cluster(target_cluster_name, source_arn, target_param_group_name):
    """Creates a target Redshift cluster"""

    response = redshift.create_cluster_parameter_group(
        ParameterGroupName=target_param_group_name,
        ParameterGroupFamily='redshift-1.0',
        Description='For Aurora PostgreSQL zero-ETL integrations'
    )
    print('Created target parameter group: ' + response['ClusterParameterGroup']
['ParameterGroupName'])

    response = redshift.modify_cluster_parameter_group(
        ParameterGroupName=target_param_group_name,
        Parameters=[
            {
                'ParameterName': 'enable_case_sensitive_identifier',
                'ParameterValue': 'true'
            }
        ]
    )
    print('Modified target parameter group: ' + response['ParameterGroupName'])

    response = redshift.create_cluster(
        ClusterIdentifier=target_cluster_name,
        NodeType='ra3.4xlarge',
        NumberOfNodes=2,
        Encrypted=True,
        MasterUsername='username',
        MasterUserPassword='Password01**',
        ClusterParameterGroupName=target_param_group_name
    )
    print('Creating target cluster: ' + response['Cluster']['ClusterIdentifier'])

    # Retrieve the target cluster ARN
    response = redshift.describe_clusters(
```

```
        ClusterIdentifier=target_cluster_name
    )
    target_arn = response['Clusters'][0]['ClusterNamespaceArn']

    # Retrieve the current user's account ID
    response = sts.get_caller_identity()
    account_id = response['Account']

    # Create a resource policy specifying cluster ARN and account ID
    response = redshift.put_resource_policy(
        ResourceArn=target_arn,
        Policy='''
        {
            \"Version\":\"2012-10-17\",
            \"Statement\":[
                {\"Effect\":\"Allow\",
                \"Principal\":{
                    \"Service\":\"redshift.amazonaws.com\"
                },
                \"Action\":[\"redshift:AuthorizeInboundIntegration\"],
                \"Condition\":{
                    \"StringEquals\":{
                        \"aws:SourceArn\":\"%s\"}
                    }
                },
                {\"Effect\":\"Allow\",
                \"Principal\":{
                    \"AWS\":\"arn:aws:iam::%s:root\"},
                \"Action\":\"redshift:CreateInboundIntegration\"}
            ]
        }
        ''' % (source_arn, account_id)
    )
    return(response)

def wait_for_cluster_availability(*args):
    """Waits for both clusters to be available"""

    print('Waiting for clusters to be available...')

    response = rds.describe_db_clusters(
        DBClusterIdentifier=source_cluster_name
    )
    source_status = response['DBClusters'][0]['Status']
```

```
    source_arn = response['DBClusters'][0]['DBClusterArn']

    response = rds.describe_db_instances(
        DBInstanceIdentifier=source_cluster_name + '-instance'
    )
    source_instance_status = response['DBInstances'][0]['DBInstanceStatus']

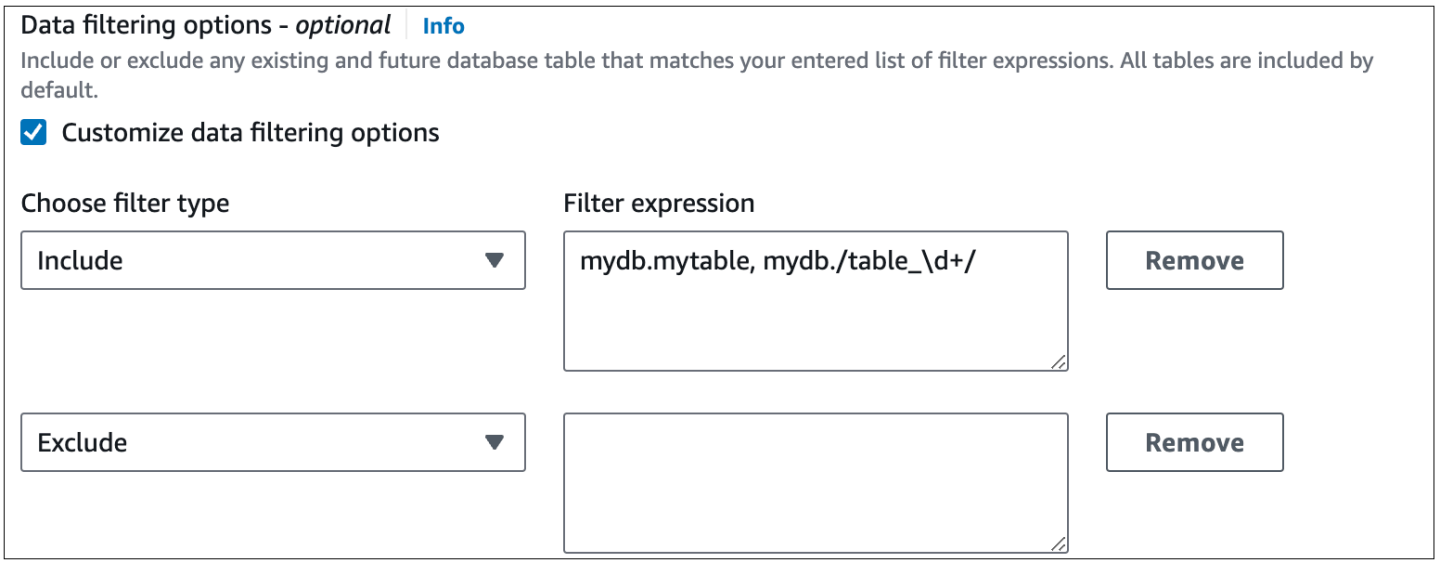
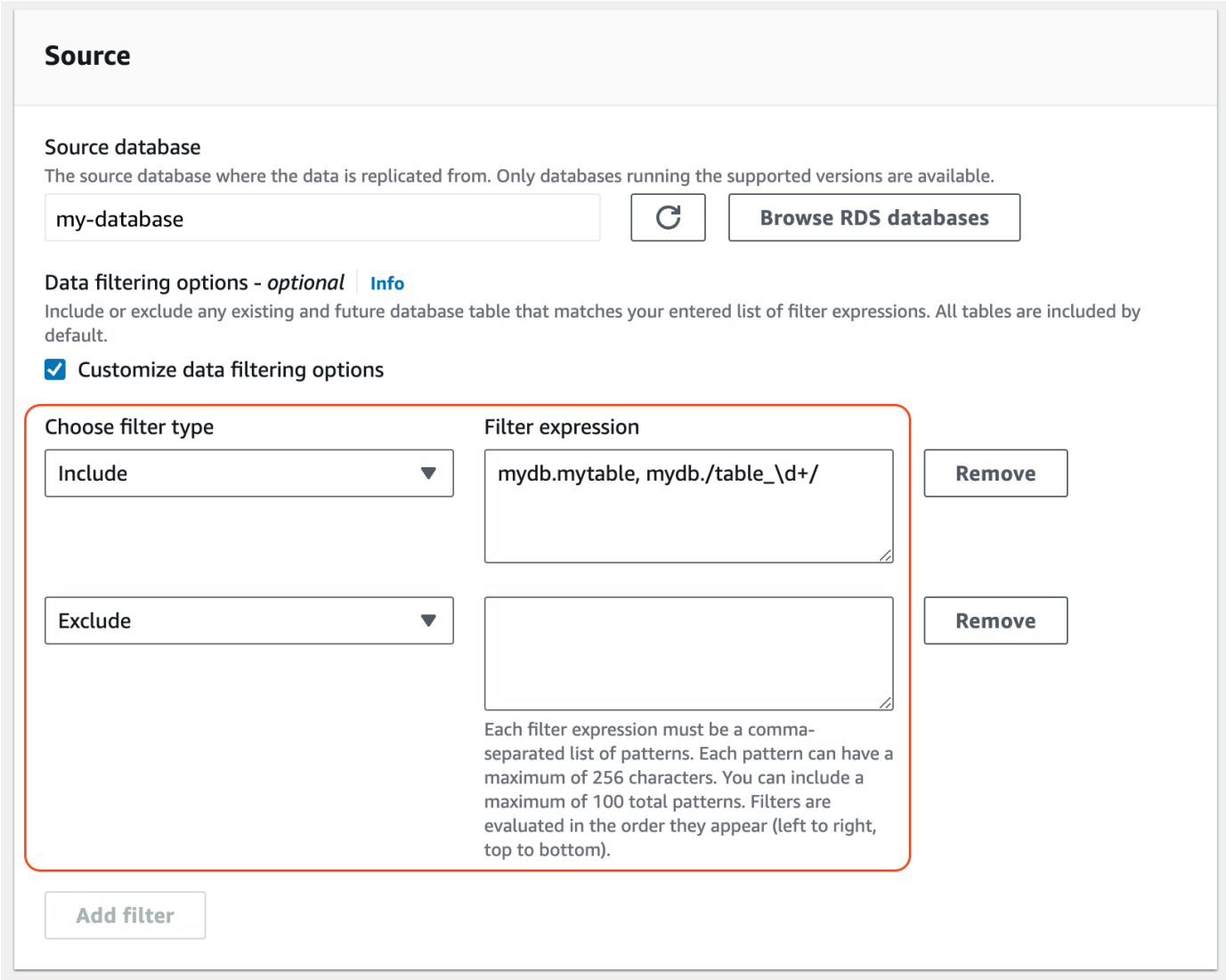
    response = redshift.describe_clusters(
        ClusterIdentifier=target_cluster_name
    )
    target_status = response['Clusters'][0]['ClusterStatus']
    target_arn = response['Clusters'][0]['ClusterNamespaceArn']

    # Every 60 seconds, check whether the clusters are available.
    if source_status != 'available' or target_status != 'available' or
 source_instance_status != 'available':
        time.sleep(60)
        response = wait_for_cluster_availability(
            source_cluster_name, target_cluster_name)
    else:
        print('Clusters available. Ready to create zero-ETL integration.')
        create_integration(source_arn, target_arn)
        return

def create_integration(source_arn, target_arn):
    """Creates a zero-ETL integration using the source and target clusters"""

    response = rds.create_integration(
        SourceArn=source_arn,
        TargetArn=target_arn,
        IntegrationName='my-integration'
    )
    print('Creating integration: ' + response['IntegrationName'])

def main():
    """main function"""
    create_source_cluster(source_cluster_name, source_param_group_name)
    wait_for_cluster_availability(source_cluster_name, target_cluster_name)

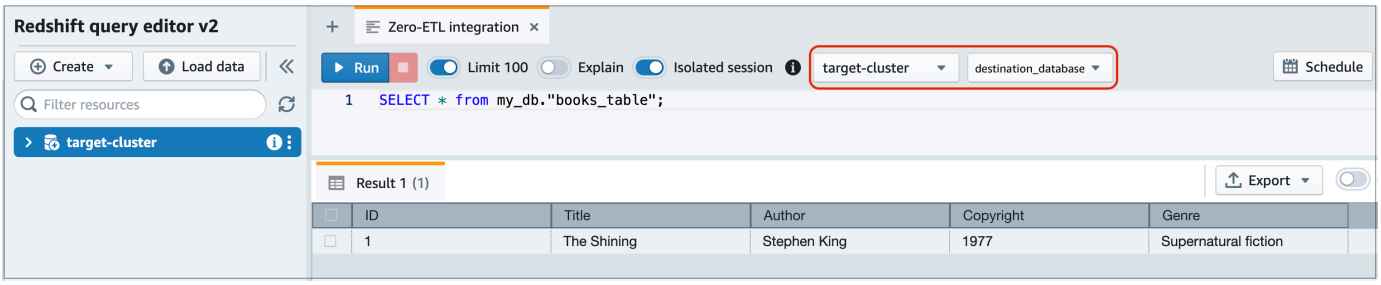
if __name__ == "__main__":
    main()
```

Fase 3b: creare un AWS Glue catalogo per l'integrazione con Amazon SageMaker AI Zero-ETL

Quando crei un'integrazione zero-ETL con un lago Amazon SageMaker AI, devi creare un catalogo gestito in. AWS Glue AWS Lake Formation Il catalogo di destinazione deve essere un catalogo gestito di Amazon Redshift. Per creare un catalogo gestito di Amazon Redshift, crea innanzitutto il ruolo collegato al servizio `AWSServiceRoleForRedshift`. Nella console Lake Formation, aggiungi `AWSServiceRoleForRedshift` come amministratore di sola lettura.

Per ulteriori informazioni sulle attività precedenti, consulta i seguenti argomenti.

- Per informazioni sulla creazione di un catalogo gestito di Amazon Redshift, consulta Creazione di un catalogo gestito di Amazon Redshift nel AWS Glue Data Catalog nella Guida per gli sviluppatori di AWS Lake Formation.
- Per informazioni sul ruolo collegato ai servizi per Amazon Redshift, consulta l'articolo Utilizzo di ruoli collegati ai servizi per Amazon Redshift nella Guida alla gestione di Amazon Redshift.
- Per informazioni sulle autorizzazioni di amministratore di sola lettura per Lake Formation, consulta l'articolo Riferimento ai personaggi di Lake Formation e alle autorizzazioni IAM nella Guida per gli sviluppatori di AWS Lake Formation.

Configura le autorizzazioni per il catalogo di destinazione AWS Glue

Prima di creare un catalogo di destinazione per l'integrazione zero-ETL, è necessario creare il ruolo di creazione del target Lake Formation e il ruolo di trasferimento dei AWS Glue dati. Usa il ruolo di creazione di destinazioni di Lake Formation per creare il catalogo di destinazioni. Quando crei il catalogo di destinazione, inserisci il ruolo di trasferimento dei dati Glue nel campo Ruolo IAM nella sezione Accesso dai motori.

Ruolo di creazione di destinazioni di Lake Formation

Il ruolo di creazione di destinazioni deve essere di un amministratore di Lake Formation e richiede le seguenti autorizzazioni.

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
```

```
        {
            "Sid": "VisualEditor0",
            "Effect": "Allow",
            "Action": "lakeformation:RegisterResource",
            "Resource": "*"
        },
        {
            "Sid": "VisualEditor1",
            "Effect": "Allow",
            "Action": [
                "s3:PutEncryptionConfiguration",
                "iam:PassRole",
                "glue:CreateCatalog",
                "glue:GetCatalog",
                "s3:PutBucketTagging",
                "s3:PutLifecycleConfiguration",
                "s3:PutBucketPolicy",
                "s3:CreateBucket",
                "redshift-serverless:CreateNamespace",
                "s3:DeleteBucket",
                "s3:PutBucketVersioning",
                "redshift-serverless:CreateWorkgroup"
            ],
            "Resource": [
                "arn:aws:glue:*:111122223333:catalog",
                "arn:aws:glue:*:111122223333:catalog/*",
                "arn:aws:s3:::*",
                "arn:aws:redshift-serverless:*:111122223333:workgroup/*",
                "arn:aws:redshift-serverless:*:111122223333:namespace/*",
                "arn:aws:iam::111122223333:role/GlueDataCatalogDataTransferRole"
            ]
        }
    ]
}
```

Il ruolo di creazione di destinazioni deve avere la seguente relazione di trust.

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
```

```
        {
            "Effect": "Allow",
            "Principal": {
                "Service": "glue.amazonaws.com"
            },
            "Action": "sts:AssumeRole"
        },
        {
          "Effect": "Allow",
          "Principal": {
            "AWS": "arn:aws:iam::111122223333:user/Username"
          },
          "Action": "sts:AssumeRole"
        }
    ]
}
```

Ruolo di trasferimento dei dati Glue

Il ruolo di trasferimento dei dati Glue è necessario per eseguire le operazioni del catalogo MySQL e deve disporre delle seguenti autorizzazioni.

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Sid": "DataTransferRolePolicy",
            "Effect": "Allow",
            "Action": [
                "kms:GenerateDataKey",
                "kms:Decrypt",
                "glue:GetCatalog",
                "glue:GetDatabase"
            ],
            "Resource": [
                "*"
            ]
        }
    ]
```

```
}
```

Il ruolo di trasferimento dei dati Glue deve avere la seguente relazione di trust.

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Principal": {
                "Service": [
                    "glue.amazonaws.com",
                    "redshift.amazonaws.com"
                ]
            },
            "Action": "sts:AssumeRole"
        }
    ]
}
```

Fasi successive

Con un di origine, un cluster Aurora DB e un data warehouse di destinazione Amazon Redshift o SageMaker Amazon AI lakehouse, puoi creare un'integrazione zero-ETL e replicare i dati. Per istruzioni, consulta the section called “Creazione di integrazioni Zero-ETL con Amazon Redshift”.

Creazione di integrazioni Zero-ETL di Aurora con Amazon Redshift

Quando crei un’integrazione Zero-ETL di Aurora, specifichi il cluster di database Aurora di origine e il data warehouse Amazon Redshift di destinazione. Puoi anche personalizzare le impostazioni di crittografia e aggiungere tag. Aurora crea un’integrazione tra il cluster di database di origine e la sua destinazione. Dopo aver attivato l’integrazione, tutti i dati che inserisci nel cluster di database di origine verranno replicati nella destinazione Amazon Redshift configurata.

Prerequisiti

Prima di creare un'integrazione Zero-ETL, devi creare un cluster di database di origine e un data warehouse Amazon Redshift di destinazione. Devi anche consentire la replica nel data warehouse aggiungendo il cluster di database Aurora come origine di integrazione autorizzata.

Per istruzioni su come completare ciascuno di questi passaggi, consulta the section called "Guida introduttiva alle integrazioni Zero-ETL".

Autorizzazioni richieste

Per creare un'integrazione Zero-ETL, occorrono alcune autorizzazioni IAM. Come requisito minimo, dovrai disporre delle autorizzazioni per eseguire le seguenti operazioni:

- Creazione di integrazioni Zero-ETL per il cluster di database Aurora di origine.
- Visualizzazione ed eliminazione di tutte le integrazioni Zero-ETL.
- Creazione di integrazioni in entrata nel data warehouse di destinazione.

Le seguenti policy di esempio mostrano le autorizzazioni con privilegi minimi richieste per creare e gestire le integrazioni. Potresti non aver bisogno di queste stesse autorizzazioni se il tuo utente o ruolo dispone di autorizzazioni più ampie, come una policy gestita di `AdministratorAccess`.

> **Note**
>
> Redshift Amazon Resource Names (ARNs) hanno il seguente formato. Nota l'uso di una barra (`/`) anziché dei due punti (`:`) prima dell'UUID dello spazio dei nomi serverless.
>
> - Cluster con provisioning: `arn:aws:redshift:{region}:{account-id}:namespace:namespace-uuid`
> - Serverless: `arn:aws:redshift-serverless:{region}:{account-id}:namespace/namespace-uuid`

Policy di esempio per la destinazione Redshift

JSON

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Sid": "CreateIntegration",
      "Effect": "Allow",
      "Action": [
        "rds:CreateIntegration"
      ],
      "Resource": [
        "arn:aws:rds:us-east-1:123456789012:db:source-db",
        "arn:aws:rds:us-east-1:123456789012:integration:*"
      ]
    },
    {
      "Sid": "DescribeIntegrationDetails",
      "Effect": "Allow",
      "Action": [
        "rds:DescribeIntegrations"
      ],
      "Resource": [
      "arn:aws:rds:us-east-1:123456789012:integration:*"
  ]
    },
    {
      "Sid": "ChangeIntegrationDetails",
      "Effect": "Allow",
      "Action": [
        "rds:DeleteIntegration",
        "rds:ModifyIntegration"
      ],
      "Resource": [
        "arn:aws:rds:us-east-1:123456789012:integration:*"
      ]
    },
    {
      "Sid": "AllowRedShiftIntegration",
      "Effect": "Allow",
      "Action": [
        "redshift:CreateInboundIntegration"
      ],
      "Resource": [
        "arn:aws:redshift:us-east-1:123456789012:namespace:namespace-uuid"
      ]
```

```
    }
  ]
}
```

Scelta di un data warehouse di destinazione in un account diverso

Se prevedi di specificare un data warehouse Amazon Redshift di destinazione che si trova in un altroAccount AWS, devi creare un ruolo che consenta agli utenti dell'account corrente di accedere alle risorse nell'account di destinazione. Per ulteriori informazioni, consulta Fornire l'accesso a un utente IAM in un altro utente Account AWS di tua proprietà.

Il ruolo deve disporre delle seguenti autorizzazioni, che consentono all'utente di visualizzare i cluster Amazon Redshift con provisioning e gli spazi dei nomi Redshift Serverless disponibili nell'account di destinazione.

Autorizzazioni richieste e policy di attendibilità

JSON

```
{
   "Version":"2012-10-17",
   "Statement":[
      {
         "Effect":"Allow",
         "Action":[
            "redshift:DescribeClusters",
            "redshift-serverless:ListNamespaces"
         ],
         "Resource":[
            "*"
         ]
      }
   ]
}
```

Il ruolo deve avere la seguente policy di attendibilità, che specifica l'ID dell'account di destinazione.

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Principal": {
                "AWS": "arn:aws:iam::111122223333:root"
            },
            "Action": "sts:AssumeRole"
        }
    ]
}
```

Per istruzioni sulla creazione del ruolo, consulta Creazione di un ruolo utilizzando policy di attendibilità personalizzate.

Creazione di integrazioni Zero-ETL

Puoi creare un'integrazione zero-ETL utilizzando l'API Console di gestione AWSAWS CLI, the o RDS.

> **Important**
>
> Le integrazioni Zero-ETL non supportano le operazioni di aggiornamento o risincronizzazione. Se riscontri problemi con un'integrazione dopo la creazione, devi eliminarla e crearne una nuova.

Console RDS

Creazione di un'integrazione Zero-ETL

1. Accedi a Console di gestione AWS e apri la console Amazon RDS all'indirizzo https://console.aws.amazon.com/rds/.
2. Nel pannello di navigazione a sinistra, scegli Interfacce di rete.
3. Scegli Crea un'integrazione Zero-ETL.

4. In Identificatore dell'integrazione, inserisci un nome per l'integrazione. Il nome può contenere fino a 63 caratteri alfanumerici e può includere trattini.

   > **Important**
   >
   > I nomi dei cataloghi hanno un limite di lunghezza di 19 caratteri. Assicurati che il tuo identificatore di integrazione soddisfi questo requisito se verrà utilizzato come nome di catalogo.

5. Scegli Next (Successivo).
6. In Origine, seleziona il cluster di database Aurora da cui provengono i dati.

   > **Note**
   >
   > RDS avvisa l'utente se i parametri del cluster DB non sono configurati correttamente. Se ricevi questo messaggio, puoi scegliere Correggi per me o configurarli manualmente. Per istruzioni su come correggerli manualmente, consulta the section called "Fase 1: creazione di un gruppo di parametri del cluster di database personalizzato".
   > La modifica dei parametri del cluster DB richiede un riavvio. Prima di poter creare l'integrazione, è necessario completare il riavvio e applicare correttamente i nuovi valori del parametro al cluster.

7. (Facoltativo) Seleziona Personalizza le opzioni di filtro dei dati e aggiungi i filtri dei dati all'integrazione. È possibile utilizzare i filtri dei dati per definire l'ambito della replica nel data warehouse di destinazione. Per ulteriori informazioni, consulta the section called "Filtro dei dati per le integrazioni Zero-ETL".
8. Una volta completata la configurazione del cluster di database di origine, scegli Avanti.
9. Per Destinazione, esegui queste operazioni:

   1. (Facoltativo) Per utilizzare un account diverso Account AWS per il target Amazon Redshift, scegli Specificare un account diverso. Quindi, inserisci l'ARN di un ruolo IAM con le autorizzazioni per visualizzare i data warehouse. Per istruzioni su come creare il ruolo IAM, consulta the section called "Scelta di un data warehouse di destinazione in un account diverso".
   2. Per Data warehouse Amazon Redshift, seleziona la destinazione per i dati replicati dal cluster di database di origine. Puoi scegliere un cluster Amazon Redshift con provisioning o uno spazio dei nomi Redshift Serverless come destinazione.

> **Note**
>
> RDS ti avvisa se le policy relative alle risorse o le impostazioni di distinzione tra maiuscole e minuscole per il data warehouse specificato non sono configurate correttamente. Se ricevi questo messaggio, puoi scegliere Correggi per me o configurarli manualmente. Per istruzioni su come correggerli manualmente, consulta Attivazione della distinzione tra maiuscole e minuscole per il data warehouse e Configurazione dell'autorizzazione per il data warehouse nella Guida alla gestione di Amazon Redshift. La modifica della distinzione tra maiuscole e minuscole per un cluster Redshift con provisioning richiede un riavvio. Prima di poter creare l'integrazione, è necessario completare il riavvio e applicare correttamente il nuovo valore del parametro al cluster. Se l'origine e la destinazione selezionate si trovano in Account AWS diversi, Amazon RDS non può correggere queste impostazioni per te. Devi accedere all'altro account e correggerle manualmente in Amazon Redshift.

10. Una volta configurato correttamente il data warehouse di destinazione, scegli Avanti.
11. (Facoltativo) In Tag, aggiungi uno o più tag all'integrazione. Per ulteriori informazioni, consulta the section called "Applicazione di tag alle risorse Aurora e RDS".
12. In Crittografia, specifica come eseguire la crittografia dell'integrazione. Per impostazione predefinita, RDS crittografa tutte le integrazioni con un. Chiave di proprietà di AWS Per scegliere invece una chiave gestita dal cliente, abilita Personalizza le impostazioni di crittografia e scegli una chiave KMS da utilizzare per la crittografia. Per ulteriori informazioni, consulta the section called "Crittografia delle risorse Amazon Aurora".

    Aggiungi un contesto di crittografia (facoltativo). Per ulteriori informazioni, consultare Contesto della crittografia nella Guida per gli sviluppatori di AWS Key Management Service.

    > **Note**
    >
    > Amazon RDS aggiunge le seguenti coppie di contesti di crittografia oltre a quelle che aggiungi tu:
    >
    > - `aws:redshift:integration:arn` - `IntegrationArn`
    > - `aws:servicename:id` - `Redshift`

Ciò riduce il numero complessivo di coppie che puoi aggiungere da 8 a 6 e contribuisce al limite complessivo di caratteri del vincolo di concessione. Per ulteriori informazioni sulle autorizzazioni, consulta Utilizzo dei vincoli di concessione nella Guida per gli sviluppatori di AWS Key Management Service.

13. Scegli Next (Successivo).
14. Rivedi le impostazioni dell'integrazione e scegli Crea integrazione Zero-ETL.

    Se la creazione ha esito negativo, consulta the section called “Non riesco a creare un'integrazione Zero-ETL” per la procedura di risoluzione dei problemi.

Lo stato dell’integrazione è `Creating` mentre l’integrazione è in fase di creazione, mentre lo stato del data warehouse Amazon Redshift di destinazione è `Modifying`. Durante questo periodo, non puoi eseguire query sul data warehouse o apportare modifiche alla configurazione.

Quando la creazione dell’integrazione viene completata correttamente, lo stato dell’integrazione e del data warehouse Amazon Redshift di destinazione cambia in `Active`.

AWS CLI

Per creare un'integrazione zero-ETL utilizzando ilAWS CLI, utilizza il comando create-integration con le seguenti opzioni:

> **Note**
>
> Ricorda che i nomi dei cataloghi hanno un limite di 19 caratteri. Scegli il nome dell’integrazione tenendo conto del fatto che potrà essere utilizzato come nome di catalogo.

- `--integration-name`: specifica un nome per l’integrazione.
- `--source-arn`: specifica l’ARN del cluster di database Aurora che sarà l’origine dell’integrazione.
- `--target-arn`: specifica l’ARN del data warehouse di Amazon Redshift che sarà la destinazione dell’integrazione.

Example

Per Linux, macOS o Unix:

```
aws rds create-integration \
    --integration-name my-integration \
    --source-arn arn:aws:rds:{region}:{account-id}:my-db \
    --target-arn arn:aws:redshift:{region}:{account-id}:namespace:namespace-uuid
```

Per Windows:

```
aws rds create-integration ^
    --integration-name my-integration ^
    --source-arn arn:aws:rds:{region}:{account-id}:my-db ^
    --target-arn arn:aws:redshift:{region}:{account-id}:namespace:namespace-uuid
```

API RDS

Per creare un'integrazione Zero-ETL con l'API Amazon RDS, utilizza l'operazione `CreateIntegration` con i seguenti parametri:

> **Note**
>
> I nomi dei cataloghi hanno un limite di 19 caratteri. Assicuratevi che il IntegrationName parametro soddisfi questo requisito se verrà utilizzato come nome di catalogo.

- `IntegrationName`: specifica un nome per l'integrazione.
- `SourceArn`: specifica l'ARN del cluster di database Aurora che sarà l'origine dell'integrazione.
- `TargetArn`: specifica l'ARN del data warehouse di Amazon Redshift che sarà la destinazione dell'integrazione.

Crittografia delle integrazioni con una chiave gestita dal cliente

Se specifichi una chiave KMS personalizzata anziché una Chiave di proprietà di AWS quando crei un'integrazione, la policy chiave deve fornire al servizio Amazon Redshift l'accesso principale all'`CreateGrant`azione. Inoltre, deve consentire all'utente corrente di eseguire le operazioni `DescribeKey` e `CreateGrant`.

La seguente policy di esempio mostra come fornire le autorizzazioni richieste nella policy della chiave. Include chiavi contestuali per ridurre ulteriormente l'ambito delle autorizzazioni.

Policy della chiave di esempio

JSON

```
{
    "Version":"2012-10-17",
    "Id": "Key policy",
    "Statement": [
        {
            "Sid": "Enables IAM user permissions",
            "Effect": "Allow",
            "Principal": {
                "AWS": "arn:aws:iam::111122223333:root"
            },
            "Action": "kms:*",
            "Resource": "*"
        },
        {
            "Sid": "Allows the Redshift service principal to add a grant to a KMS
 key",
            "Effect": "Allow",
            "Principal": {
                "Service": "redshift.amazonaws.com"
            },
            "Action": "kms:CreateGrant",
            "Resource": "*",
            "Condition": {
                "StringEquals": {
                    "kms:EncryptionContext:{context-key}": "{context-value}"
                },
                "ForAllValues:StringEquals": {
                    "kms:GrantOperations": [
                        "Decrypt",
                        "GenerateDataKey",
                        "CreateGrant"
                    ]
                }
            }
        },
        {
            "Sid": "Allows the current user or role to add a grant to a KMS key",
            "Effect": "Allow",
            "Principal": {
```

```
                "AWS": "arn:aws:iam::111122223333:role/{role-name}"
            },
            "Action": "kms:CreateGrant",
            "Resource": "*",
            "Condition": {
                "StringEquals": {
                    "kms:EncryptionContext:{context-key}": "{context-value}",
                    "kms:ViaService": "rds.us-east-1.amazonaws.com"
                },
                "ForAllValues:StringEquals": {
                    "kms:GrantOperations": [
                        "Decrypt",
                        "GenerateDataKey",
                        "CreateGrant"
                    ]
                }
            }
        },
        {
            "Sid": "Allows the current uer or role to retrieve information about
 a KMS key",
            "Effect": "Allow",
            "Principal": {
                "AWS": "arn:aws:iam::111122223333:role/{role-name}"
            },
            "Action": "kms:DescribeKey",
            "Resource": "*"
        }
    ]
}
```

Per ulteriori informazioni, consulta Creazione di una policy delle chiavi nella Guida per gli sviluppatori di AWS Key Management Service.

Fasi successive

Dopo aver creato correttamente un'integrazione Zero-ETL, devi creare un database di destinazione all'interno del cluster o del gruppo di lavoro Amazon Redshift di destinazione. Quindi, puoi iniziare ad aggiungere dati al cluster di database Aurora ed eseguirvi query in Amazon Redshift. Per istruzioni, consulta Creazione di database di destinazione in Amazon Redshift.

Creazione di integrazioni Zero-ETL di Aurora con un lakehouse Amazon SageMaker

Quando crei un'integrazione Zero-ETL di Aurora con Amazon SageMaker un lakehouse, specifichi il di origine, il cluster Aurora DB e il catalogo gestito di destinazione. AWS Glue Puoi anche personalizzare le impostazioni di crittografia e aggiungere tag. Aurora crea un'integrazione tra il cluster di database di origine e la sua destinazione. Dopo aver attivato l'integrazione, tutti i dati che inserisci nel cluster di database di origine verranno replicati nella destinazione configurata.

Prerequisiti

AWS Glue Devi anche consentire la replica nel catalogo aggiungendo il cluster di database come origine di integrazione autorizzata.

Per istruzioni su come completare ciascuno di questi passaggi, consulta the section called "Guida introduttiva alle integrazioni Zero-ETL".

Autorizzazioni richieste

Per creare un'integrazione Zero-ETL, occorrono alcune autorizzazioni IAM con un lakehouse Amazon SageMaker. Come requisito minimo, dovrai disporre delle autorizzazioni per eseguire le seguenti operazioni:

- Creazione di integrazioni Zero-ETL per il cluster di database Aurora di origine.
- Visualizzazione ed eliminazione di tutte le integrazioni Zero-ETL.
- Crea integrazioni in entrata nel catalogo gestito di destinazione. AWS Glue
- Accedi ai bucket Amazon S3 utilizzati dal catalogo gestito. AWS Glue
- Usa AWS KMS le chiavi per la crittografia se è configurata una crittografia personalizzata.
- Registra le risorse con Lake Formation.
- Inserisci la politica delle risorse nel catalogo AWS Glue gestito per autorizzare le integrazioni in entrata.

La seguente policy di esempio mostra le autorizzazioni con privilegi minimi richieste per creare e gestire le integrazioni con un lakehouse Amazon SageMaker. Potresti non aver bisogno di queste autorizzazioni esatte se il tuo utente o ruolo dispone di autorizzazioni più ampie, come una policy gestita `AdministratorAccess`.

Inoltre, è necessario configurare una politica delle risorse sul catalogo AWS Glue gestito di destinazione per autorizzare le integrazioni in entrata. Utilizza il AWS CLI comando seguente per applicare la politica delle risorse.

AWS CLIComando di esempio per autorizzare le integrazioni in entrata sul catalogo di destinazione

```
aws glue put-resource-policy \
      --policy-in-json  '{
    "Version": "2012-10-17",
    "Statement": [{
        "Effect": "Allow",
        "Principal": {
            "Service": "glue.amazonaws.com"
        },
        "Action": [
            "glue:AuthorizeInboundIntegration"
        ],
        "Resource": ["arn:aws:glue:region:account_id:catalog/catalog_name"],
        "Condition": {
            "StringEquals": {
                "aws:SourceArn": "arn:aws:rds:region:account_id:db:source_name"
            }
        }
    },
    {
        "Effect": "Allow",
        "Principal": {
            "AWS": "account_id"
        },
        "Action": ["glue:CreateInboundIntegration"],
        "Resource": ["arn:aws:glue:region:account_id:catalog/catalog_name"]
    }
    ]
}' \
      --region region
```

> **Note**
>
> Il catalogo Glue Amazon Resource Names (ARNs) ha il seguente formato:
>
> - Catalogo Glue — `arn:aws:glue:{region}:{account-id}:catalog/catalog-name`

Scelta di un catalogo AWS Glue gestito di destinazione in un account diverso

Se prevedi di specificare un catalogo AWS Glue gestito di destinazione che si trova in un altroAccount AWS, devi creare un ruolo che consenta agli utenti dell'account corrente di accedere alle risorse nell'account di destinazione. Per ulteriori informazioni, consulta Fornire l'accesso a un utente IAM in un altro Account AWS utente di tua proprietà.

Il ruolo deve disporre delle seguenti autorizzazioni, che consentono all'utente di visualizzare AWS Glue i cataloghi disponibili nell'account di destinazione.

Autorizzazioni richieste e policy di attendibilità

JSON

```
{
   "Version":"2012-10-17",
   "Statement":[
      {
         "Effect":"Allow",
         "Action":[
            "glue:GetCatalog"
         ],
         "Resource":[
            "*"
         ]
      }
   ]
}
```

Il ruolo deve avere la seguente policy di attendibilità, che specifica l'ID dell'account di destinazione.

JSON

```
{
   "Version":"2012-10-17",
   "Statement":[
      {
         "Effect":"Allow",
         "Principal":{
            "AWS": "arn:aws:iam::111122223333:root"
```

```
            },
            "Action":"sts:AssumeRole"
        }
    ]
}
```

Per istruzioni sulla creazione del ruolo, consulta Creazione di un ruolo utilizzando policy di attendibilità personalizzate.

Creazione di integrazioni Zero-ETL con un lakehouse Amazon SageMaker

Puoi creare un'integrazione zero-ETL con un Amazon SageMaker lakehouse utilizzando l'APIConsole di gestione AWS, the o RDS. AWS CLI

> **Important**
>
> Le integrazioni Zero-ETL con un lakehouse Amazon SageMaker non supportano le operazioni di aggiornamento o risincronizzazione. Se riscontri problemi con un'integrazione dopo la creazione, devi eliminarla e crearne una nuova.

Console RDS

Per creare un'integrazione Zero-ETL con un lakehouse Amazon SageMaker

1. Accedi a Console di gestione AWS e apri la console Amazon RDS all'indirizzo https://console.aws.amazon.com/rds/.
2. Nel pannello di navigazione a sinistra, scegli Interfacce di rete.
3. Scegli Crea un'integrazione Zero-ETL.
4. In Identificatore dell'integrazione, inserisci un nome per l'integrazione. Il nome può contenere fino a 63 caratteri alfanumerici e può includere trattini.
5. Scegli Next (Successivo).
6. In Origine, seleziona il cluster di database Aurora da cui provengono i dati.

> **Note**
>
> RDS avvisa l'utente se i parametri del cluster DB non sono configurati correttamente. Se ricevi questo messaggio, puoi scegliere Correggi per me o configurarli manualmente.

Per istruzioni su come correggerli manualmente, consulta the section called “Fase 1: creazione di un gruppo di parametri del cluster di database personalizzato”.
La modifica dei parametri del cluster DB richiede un riavvio. Prima di poter creare l’integrazione, è necessario completare il riavvio e applicare correttamente i nuovi valori del parametro al cluster.

7. (Facoltativo) Seleziona Personalizza le opzioni di filtro dei dati e aggiungi i filtri dei dati all’integrazione. È possibile utilizzare i filtri dei dati per definire l’ambito della replica al lakehouse Amazon SageMaker di destinazione. Per ulteriori informazioni, consulta the section called “Filtro dei dati per le integrazioni Zero-ETL”.
8. Una volta completata la configurazione del cluster di database di origine, scegli Avanti.
9. Per Destinazione, esegui queste operazioni:

   1. (Facoltativo) Per utilizzarne uno diverso Account AWS per Amazon SageMaker Lakehouse Target, scegli Specificare un account diverso. Quindi, inserisci l'ARN di un ruolo IAM con le autorizzazioni per visualizzare i tuoi cataloghi. AWS Glue Per istruzioni su come creare il ruolo IAM, consulta the section called “Scelta di un catalogo AWS Glue gestito di destinazione in un account diverso”.
   2. Per Catalogo AWS Glue, seleziona la destinazione per i dati replicati dal cluster di database di origine. Puoi selezionare un catalogo AWS Glue gestito esistente come destinazione.
   3. Il ruolo IAM di destinazione deve descrivere le autorizzazioni nel catalogo di destinazione e deve disporre delle seguenti autorizzazioni:

      JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Sid": "VisualEditor0",
            "Effect": "Allow",
            "Action": "glue:GetCatalog",
            "Resource": [
                "arn:aws:glue:us-east-1:111122223333:catalog/*",
                "arn:aws:glue:us-east-1:111122223333:catalog"
            ]
        }
    ]
}
```

Il ruolo IAM deve avere la relazione di trust seguente:

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Principal": {
                "Service": "glue.amazonaws.com"
            },
            "Action": "sts:AssumeRole"
        }
    ]
}
```

4. È necessario concedere al ruolo IAM di destinazione le autorizzazioni di descrizione del ruolo per il catalogo AWS Glue gestito di destinazione con il ruolo di amministratore Lake Formation creato inFase 3b: creare un AWS Glue catalogo per l'integrazione con Amazon SageMaker AI Zero-ETL.

> **Note**
>
> RDS ti avvisa se la politica delle risorse o le impostazioni di configurazione per il catalogo AWS Glue gestito specificato non sono configurate correttamente. Se ricevi questo messaggio, puoi scegliere Correggi per me o configurarli manualmente.
> Se l'origine e la destinazione selezionate si trovano in Account AWS diversi, Amazon RDS non può correggere queste impostazioni per te. Devi accedere all'altro account e correggerli manualmente in SageMaker Unified Studio.

10. Una volta configurato correttamente il catalogo AWS Glue gestito di destinazione, scegli Avanti.
11. (Facoltativo) In Tag, aggiungi uno o più tag all'integrazione. Per ulteriori informazioni, consulta the section called "Applicazione di tag alle risorse Aurora e RDS".
12. In Crittografia, specifica come eseguire la crittografia dell'integrazione. Per impostazione predefinita, RDS crittografa tutte le integrazioni con un. Chiave di proprietà di AWS Per scegliere invece una chiave gestita dal cliente, abilita Personalizza le impostazioni di crittografia e scegli

una chiave KMS da utilizzare per la crittografia. Per ulteriori informazioni, consulta the section called “Crittografia delle risorse Amazon Aurora”.

Aggiungi un contesto di crittografia (facoltativo). Per ulteriori informazioni, consultare Contesto della crittografia nella Guida per gli sviluppatori di AWS Key Management Service.

> **Note**
>
> Amazon RDS aggiunge le seguenti coppie di contesti di crittografia oltre a quelle che aggiungi tu:
>
> - `aws:glue:integration:arn` - `IntegrationArn`
> - `aws:servicename:id` - `glue`
>
> Ciò riduce il numero complessivo di coppie che puoi aggiungere da 8 a 6 e contribuisce al limite complessivo di caratteri del vincolo di concessione. Per ulteriori informazioni sulle autorizzazioni, consulta Utilizzo dei vincoli di concessione nella Guida per gli sviluppatori di AWS Key Management Service.

13. Scegli Next (Successivo).
14. Rivedi le impostazioni dell'integrazione e scegli Crea integrazione Zero-ETL.

    Se la creazione ha esito negativo, consulta the section called “Risoluzione dei problemi delle integrazioni Zero-ETL” per la procedura di risoluzione dei problemi.

Lo stato dell’integrazione è `Creating` mentre l’integrazione è in fase di creazione, mentre il lakehouse Amazon SageMaker di destinazione ha uno stato di `Modifying`. Durante questo periodo, non puoi eseguire query sul catalogo o apportare modifiche alla configurazione.

Quando la creazione dell’integrazione viene completata correttamente, lo stato dell’integrazione e del lakehouse Amazon SageMaker di destinazione cambia in `Active`.

AWS CLI

Per preparare un catalogo AWS Glue gestito di destinazione per l'integrazione zero-ETL utilizzando ilAWS CLI, è necessario innanzitutto utilizzare il comando con le create-integration-resource-propertyseguenti opzioni:

- `--resource-arn`— Specificare l'ARN del catalogo AWS Glue gestito che sarà la destinazione per l'integrazione.
- `--target-processing-properties`— Specificare l'ARN del ruolo IAM per accedere al catalogo gestito di destinazione AWS Glue

```
aws glue create-integration-resource-property --region us-east-1
 --resource-arn arn:aws:glue:region:account_id:catalog/catalog_name \
 --target-processing-properties '{"RoleArn" : "arn:aws:iam::account_id:role/
TargetIamRole"}'
```

Per creare un'integrazione zero-ETL con un Amazon SageMaker lakehouse utilizzando ilAWS CLI, utilizza il comando create-integration con le seguenti opzioni:

- `--integration-name`: specifica un nome per l'integrazione.
- `--source-arn`: specificare l'ARN del cluster di database Aurora che sarà l'origine dell'integrazione.
- `--target-arn`— Specificare l'ARN del catalogo AWS Glue gestito che sarà la destinazione per l'integrazione.

Example

Per Linux, macOS o Unix:

```
aws rds create-integration \
    --integration-name my-sagemaker-integration \
    --source-arn arn:aws:rds:{region}:{account-id}:cluster:my-db \
    --target-arn arn:aws:glue:{region}:{account-id}:catalog/catalog-name
```

Per Windows:

```
aws rds create-integration ^
    --integration-name my-sagemaker-integration ^
    --source-arn arn:aws:rds:{region}:{account-id}:cluster:my-db ^
    --target-arn arn:aws:glue:{region}:{account-id}:catalog/catalog-name
```

API RDS

Per creare un'integrazione Zero-ETL con Amazon SageMaker mediante l'API Amazon RDS, utilizza l'operazione `CreateIntegration` con i seguenti parametri:

> **Note**
>
> I nomi dei cataloghi sono limitati a 19 caratteri. Assicurati che il IntegrationName parametro soddisfi questo requisito se verrà utilizzato come nome di catalogo.

- `IntegrationName`: specifica un nome per l'integrazione.
- `SourceArn`: specificare l'ARN del cluster di database Aurora che sarà l'origine dell'integrazione.
- `TargetArn`— Specificare l'ARN del catalogo AWS Glue gestito che sarà la destinazione per l'integrazione.

Crittografia delle integrazioni con una chiave gestita dal cliente

Se si specifica una chiave KMS personalizzata anziché una chiave Chiave di proprietà di AWS quando si crea un'integrazione conAmazon SageMaker, la politica chiave deve fornire al SageMaker Unified Studio servizio principale l'accesso all'azione. `CreateGrant` Inoltre, deve consentire all'utente corrente di eseguire le operazioni `DescribeKey` e `CreateGrant`.

La seguente policy di esempio mostra come fornire le autorizzazioni richieste nella policy della chiave. Include chiavi contestuali per ridurre ulteriormente l'ambito delle autorizzazioni.

Policy della chiave di esempio

JSON

```
{
    "Version":"2012-10-17",
    "Id": "Key policy",
    "Statement": [
        {
            "Sid": "EnablesIAMUserPermissions",
            "Effect": "Allow",
            "Principal": {
            "AWS": "arn:aws:iam::111122223333:root"
```

```
            },
            "Action": "kms:*",
            "Resource": "*"
        },
        {
            "Sid": "GlueServicePrincipalAddGrant",
            "Effect": "Allow",
            "Principal": {
                "Service": "glue.amazonaws.com"
            },
            "Action": "kms:CreateGrant",
            "Resource": "*",
            "Condition": {
                "StringEquals": {
                    "kms:EncryptionContext:{context-key}":"{context-value}"
                },
                "ForAllValues:StringEquals": {
                    "kms:GrantOperations": [
                        "Decrypt",
                        "GenerateDataKey",
                        "CreateGrant"
                    ]
                }
            }
        },
        {
            "Sid": "AllowsCurrentUserRoleAddGrantKMSKey",
            "Effect": "Allow",
            "Principal": {
            "AWS": "arn:aws:iam::111122223333:role/{role-name}"
            },
            "Action": "kms:CreateGrant",
            "Resource": "*",
            "Condition": {
                "StringEquals": {
                    "kms:EncryptionContext:{context-key}":"{context-value}",
                    "kms:ViaService": "rds.us-east-1.amazonaws.com"
                },
                "ForAllValues:StringEquals": {
                    "kms:GrantOperations": [
                        "Decrypt",
                        "GenerateDataKey",
                        "CreateGrant"
                    ]
```

```
                }
            }
        },
        {
            "Sid": "AllowsCurrentUserRoleRetrieveKMSKeyInformation",
            "Effect": "Allow",
            "Principal": {
            "AWS": "arn:aws:iam::111122223333:role/{role-name}"
            },
            "Action": "kms:DescribeKey",
            "Resource": "*"
        }
    ]
}
```

Per ulteriori informazioni, consulta Creazione di una policy delle chiavi nella Guida per gli sviluppatori di AWS Key Management Service.

Fasi successive

Dopo aver creato con successo un'integrazione Zero-ETL con Amazon SageMaker, puoi iniziare ad aggiungere dati al cluster di database Aurora di origine ed eseguire query su di essi nel tuo lakehouse Amazon SageMaker. I dati verranno replicati automaticamente e resi disponibili per analisi e carichi di lavoro di machine learning.

Filtro dei dati per integrazioni Zero-ETL di Aurora

Le integrazioni Zero-ETL di Aurora supportano il filtro dei dati, che consente di controllare quali dati vengono replicati dal cluster di database Aurora di origine al data warehouse di destinazione. Invece di replicare l'intero database, puoi applicare uno o più filtri per includere o escludere selettivamente delle tabelle specifiche. Ciò consente di ottimizzare le prestazioni di archiviazione e query assicurando che vengano trasferiti solo i dati pertinenti. Attualmente, il filtro è limitato a livello di database e tabella. Il filtro a livello di colonna e riga non è supportato.

Il filtro dei dati può essere utile se desideri:

- Unire determinate tabelle da due o più cluster di origine diversi e non hai bisogno dei dati completi da ciascun cluster.

- Risparmiare sui costi eseguendo analisi che utilizzano solo un sottoinsieme di tabelle anziché un intero parco di database.
- Filtrare le informazioni sensibili, come numeri di telefono, indirizzi o dettagli delle carte di credito, da determinate tabelle.

Puoi aggiungere filtri di dati a un'integrazione zero-ETL utilizzando Console di gestione AWS, the AWS Command Line Interface (AWS CLI) o l'API Amazon RDS.

Se l'integrazione ha come destinazione un cluster con provisioning, per utilizzare il filtro dei dati è necessario che il cluster utilizzi la patch 180 o una versione successiva.

Argomenti

- Formato di un filtro dei dati
- Logica dei filtri
- Precedenza dei filtri
- Esempi di Aurora MySQL
- Esempi di Aurora PostgreSQL
- Aggiunta di filtri dei dati a un'integrazione
- Rimozione di filtri dei dati da un'integrazione

Formato di un filtro dei dati

Puoi definire diversi filtri per una singola integrazione. Ogni filtro include o esclude tutte le tabelle di database esistenti e future che corrispondono a uno dei modelli nell'espressione di filtro. Le integrazioni Zero-ETL di Aurora utilizzano la sintassi del filtro Maxwell per il filtro dei dati.

Ogni filtro contiene i seguenti elementi:

Elemento	Description
Tipo di filtro	Un tipo di filtro `Include` include tutte le tabelle che corrispondono a uno dei modelli nell'espressione di filtro. Un tipo di filtro `Exclude` esclude tutte le tabelle che corrispondono a uno dei modelli.

Elemento	Description
Espressione filtro	Un elenco di modelli separato da virgole. Le espressioni devono utilizzare la sintassi del filtro Maxwell.
Pattern	Un modello di filtro nel formato *database*.*table* per Aurora MySQL oppure nel formato *database*.*schema*.*table* per Aurora PostgreSQL. Puoi specificare nomi letterali o definire espressioni regolari. **Note** Per Aurora MySQL, le espressioni regolari sono supportate sia nel nome del database sia in quello della tabella. Per Aurora PostgreSQL, le espressioni regolari sono supportate solo nel nome dello schema e della tabella, non in quello del database. Non è possibile includere filtri o elenco elementi non consentiti a livello di colonna. Una singola integrazione può avere un massimo di 99 modelli totali. Nella console, è possibile inserire modelli all'interno di una singola espressione di filtro o distribuirli tra diverse espressioni. Un singolo modello non può superare i 256 caratteri.

> **Important**
>
> Se selezioni un cluster di database di origine Aurora PostgreSQL, è necessario specificare almeno un modello di filtro dei dati. Come minimo, il modello deve includere un singolo database (*`database-name`*`.*.*`) per la replica nel data warehouse di destinazione.

L'immagine seguente mostra la struttura dei filtri dei dati di Aurora MySQL nella console:



> **Important**
>
> Non includere informazioni riservate, sensibili o di identificazione personale nei modelli di filtro.

Filtri di dati in AWS CLI

Quando si utilizza AWS CLI per aggiungere un filtro dati, la sintassi è leggermente diversa da quella della console. È necessario assegnare un tipo di filtro (`Include` o `Exclude`) a ciascun modello singolarmente, in modo da non poter raggruppare diversi modelli in un unico tipo di filtro.

Ad esempio, nella console puoi raggruppare i seguenti modelli separati da virgola in un'unica istruzione `Include`:

Aurora MySQL

```
mydb.mytable, mydb./table_\d+/
```

Aurora PostgreSQL

```
mydb.myschema.mytable, mydb.myschema./table_\d+/
```

Tuttavia, quando si utilizza il AWS CLI, lo stesso filtro dati deve avere il seguente formato:

Aurora MySQL

```
'include: mydb.mytable, include: mydb./table_\d+/'
```

Aurora PostgreSQL

```
'include: mydb.myschema.mytable, include: mydb.myschema./table_\d+/'
```

Logica dei filtri

Se non specifichi alcun filtro dei dati nell'integrazione, Aurora presuppone un filtro predefinito `include:*.*`, che replica tutte le tabelle nel data warehouse di destinazione. Tuttavia, se aggiungi almeno un filtro, la logica predefinita passa a `exclude:*.*`, che esclude tutte le tabelle per impostazione predefinita. Ciò consente di definire in modo esplicito quali database e tabelle includere nella replica.

Ad esempio, se definisci il filtro seguente:

```
'include: db.table1, include: db.table2'
```

Aurora valuta il filtro come indicato di seguito:

```
'exclude:*.*, include: db.table1, include: db.table2'
```

Pertanto, Aurora esegue la replica di `table1` e `table2` dal database denominato db al data warehouse di destinazione.

Precedenza dei filtri

Aurora valuta i filtri dei dati nell'ordine specificato. In Console di gestione AWS, elabora le espressioni di filtro da sinistra a destra e dall'alto verso il basso. Un secondo filtro o un modello individuale che segue il primo può sovrascriverlo.

Ad esempio, se il primo filtro è `Include books.stephenking`, include solo la tabella `stephenking` del database `books`. Tuttavia, se aggiungi un secondo filtro `Exclude books.*`, questo sostituisce il primo. Ciò impedisce che le tabelle dell'indice `books` vengano replicate nel data warehouse di destinazione.

Quando si specifica almeno un filtro, la logica parte presupponendo `exclude:*.*` per impostazione predefinita, che esclude automaticamente tutte le tabelle dalla replica. Come best practice, definisci i filtri dal più ampio al più specifico. Inizia con una o più istruzioni `Include` per specificare i dati da replicare, quindi aggiungi i filtri `Exclude` per rimuovere selettivamente determinate tabelle.

Lo stesso principio si applica ai filtri definiti utilizzando AWS CLI. Aurora valuta questi modelli di filtro nell'ordine in cui li specifichi, quindi un modello potrebbe sovrascrivere quello specificato in precedenza.

Esempi di Aurora MySQL

Gli esempi seguenti mostrano come funziona il filtro dei dati per le integrazioni Zero-ETL di Aurora MySQL:

- Includi tutti i database e tutte le tabelle:

```
'include: *.*'
```

- Includi tutte le tabelle all'interno del database `books`:

```
'include: books.*'
```

- Escludi tutte le tabelle denominate `mystery`:

```
'include: *.*, exclude: *.mystery'
```

- Includi due tabelle specifiche all'interno del database `books`:

```
'include: books.stephen_king, include: books.carolyn_keene'
```

- Includi tutte le tabelle del database `books`, ad eccezione di quelle contenenti la sottostringa `mystery`:

```
'include: books.*, exclude: books./.*mystery.*/'
```

- Includi tutte le tabelle del database `books`, ad eccezione di quelle che iniziano con `mystery`:

```
'include: books.*, exclude: books./mystery.*/'
```

- Includi tutte le tabelle del database `books`, ad eccezione di quelle che terminano con `mystery`:

```
'include: books.*, exclude: books./.*mystery/'
```

- Includi tutte le tabelle del database `books` che iniziano con `table_`, ad eccezione di quella denominata `table_stephen_king`. Ad esempio, `table_movies` o `table_books` verrebbe replicato, ma `table_stephen_king` non verrebbe replicato.

```
'include: books./table_.*/, exclude: books.table_stephen_king'
```

Esempi di Aurora PostgreSQL

Gli esempi seguenti mostrano come funziona il filtro dei dati per le integrazioni Zero-ETL di Aurora PostgreSQL:

- Includi tutte le tabelle all'interno del database `books`:

```
'include: books.*.*'
```

- Escludi tutte le tabelle denominate `mystery` nel database `books`:

```
'include: books.*.*, exclude: books.*.mystery'
```

- Includi una tabella all'interno del database `books` nello schema `mystery` e una tabella nel database `employee` nello schema `finance`:

```
'include: books.mystery.stephen_king, include: employee.finance.benefits'
```

- Includi tutte le tabelle del database `books` e dello schema `science_fiction`, ad eccezione di quelle contenenti la sottostringa `king`:

```
'include: books.science_fiction.*, exclude: books.*./.*king.*/
```

- Includi tutte le tabelle del database `books`, ad eccezione di quelle con un nome di schema che inizia con`sci`:

```
'include: books.*.*, exclude: books./sci.*/.*'
```

- Includi tutte le tabelle del database `books`, ad eccezione di quelle nello schema `mystery` che termina con `king`:

```
'include: books.*.*, exclude: books.mystery./.*king/'
```

- Includi tutte le tabelle del database `books` che iniziano con `table_`, ad eccezione di quella denominata `table_stephen_king`. Ad esempio, `table_movies` nello schema `fiction` e `table_books` nello schema `mystery` vengono replicati, ma `table_stephen_king` non verrà replicato in nessuno dei due schemi:

```
'include: books.*./table_.*/, exclude: books.*.table_stephen_king'
```

Aggiunta di filtri dei dati a un'integrazione

Puoi configurare il filtraggio dei dati utilizzando Console di gestione AWS l'API AWS CLI Amazon RDS o la.

> **Important**
>
> Se aggiungi un filtro dopo aver creato un'integrazione, Aurora lo considera come se fosse sempre esistito. Rimuove tutti i dati nel data warehouse di destinazione che non corrispondono ai nuovi criteri di filtro e risincronizza tutte le tabelle interessate.

Console RDS

Per aggiungere i filtri dei dati a un'integrazione Zero-ETL

1. Accedi a Console di gestione AWS e apri la console Amazon RDS all'indirizzo https://console.aws.amazon.com/rds/.

2. Nel pannello di navigazione, scegli Integrazioni zero-ETL. Seleziona l'integrazione a cui desideri aggiungere i filtri dei dati, quindi scegli Modifica.
3. In Origine, aggiungi una o più istruzioni `Include` e `Exclude`.

   La seguente immagina mostra un esempio di filtri dei dati per un'integrazione MySQL:



4. Una volta completate le modifiche, scegli Continua, quindi Salva modifiche.

AWS CLI

Per aggiungere filtri di dati a un'integrazione zero-ETL utilizzando AWS CLI, chiama il comando modify-integration. Oltre all'identificatore di integrazione, specifica il parametro `--data-filter` con un elenco separato da virgola di filtri Maxwell `Include` e `Exclude`.

Example

Nell'esempio seguente si aggiungono modelli di filtro a `my-integration`.

Per Linux, macOS o Unix:

```
aws rds modify-integration \
    --integration-identifier my-integration \
    --data-filter 'include: foodb.*, exclude: foodb.tbl, exclude: foodb./table_\d+/'
```

Per Windows:

```
aws rds modify-integration ^
    --integration-identifier my-integration ^
    --data-filter 'include: foodb.*, exclude: foodb.tbl, exclude: foodb./table_\d+/'
```

API RDS

Per modificare un'integrazione zero-ETL utilizzando l'API RDS, chiama l'operazione. ModifyIntegration Specifica l'identificatore di integrazione e fornisci un elenco separato da virgola di modelli di filtro.

Rimozione di filtri dei dati da un'integrazione

Quando rimuovi un filtro dei dati da un'integrazione, Aurora rivaluta i filtri rimanenti come se il filtro rimosso non fosse mai esistito. Successivamente, replica tutti i dati precedentemente esclusi che ora soddisfano i criteri nel data warehouse di destinazione. Ciò attiva una risincronizzazione di tutte le tabelle interessate.

Aggiunta di dati a un cluster di database Aurora di origine ed esecuzione di query al suo interno

Per completare la creazione di un'integrazione Zero-ETL che replichi i dati da Amazon Aurora in Amazon Redshift, devi creare un database nella destinazione.

Per le connessioni con Amazon Redshift, connettiti al cluster o al gruppo di lavoro Amazon Redshift e crea un database con un riferimento al tuo identificatore di integrazione. Quindi, puoi aggiungere dati al cluster di database Aurora e vederli replicati in Amazon Redshift o Amazon SageMaker.

Argomenti

- Creazione di un database di destinazione
- Aggiunta di dati al cluster di database di origine
- Interrogazione dei dati Aurora in Amazon Redshift
- Differenze tra i tipi di dati tra i database Aurora e Amazon Redshift
- Operazioni DDL per Aurora PostgreSQL

Creazione di un database di destinazione

Prima di poter iniziare a replicare i dati in Amazon Redshift dopo aver creato un'integrazione, devi creare un database nel data warehouse di destinazione. Questo database deve includere un riferimento all'identificatore di integrazione. Puoi utilizzare la console Amazon Redshift o Editor di query v2 per creare il database.

Per istruzioni sulla creazione di un database di destinazione, consulta Creazione di un database di destinazione in Amazon Redshift.

Aggiunta di dati al cluster di database di origine

Dopo aver configurato l'integrazione, puoi popolare il cluster di database Aurora di origine con i dati che desideri replicare nel data warehouse.

> **Note**
>
> Esistono differenze tra i tipi di dati in Amazon Aurora e il warehouse di analisi di destinazione. Per una tabella di mappature dei tipi di dati, consulta the section called "Differenze dei tipi di dati".

Innanzitutto, connettiti al cluster di database di origine utilizzando il client MySQL o PostgreSQL di tua scelta. Per istruzioni, consulta the section called "Connessione a un cluster database".

Quindi, crea una tabella e inserisci una riga di dati di esempio.

> **Important**
>
> Assicurati che la tabella abbia una chiave primaria. Altrimenti, non può essere replicata nel data warehouse di destinazione.

Le utilità PostgreSQL pg_dump e pg_restore inizialmente creano tabelle senza una chiave primaria e poi l'aggiungono in seguito. Se utilizzi una di queste utilità, ti consigliamo di creare prima uno schema e poi caricare i dati in un comando separato.

MySQL

L'esempio seguente utilizza l'utilità MySQL Workbench.

```
CREATE DATABASE my_db;

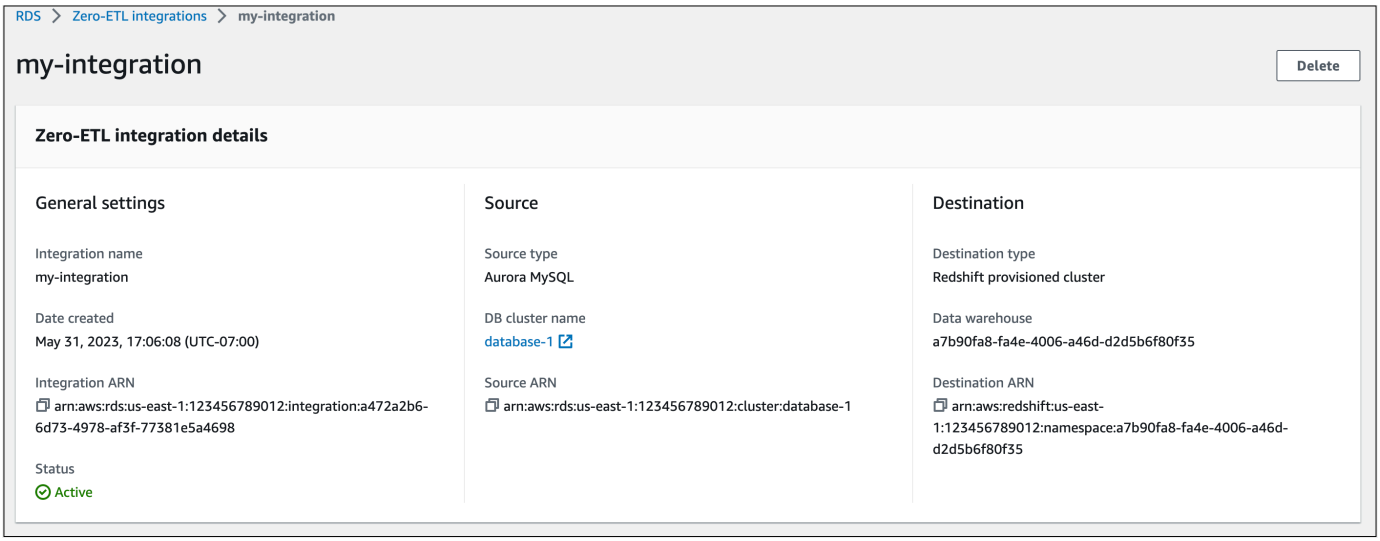
USE my_db;

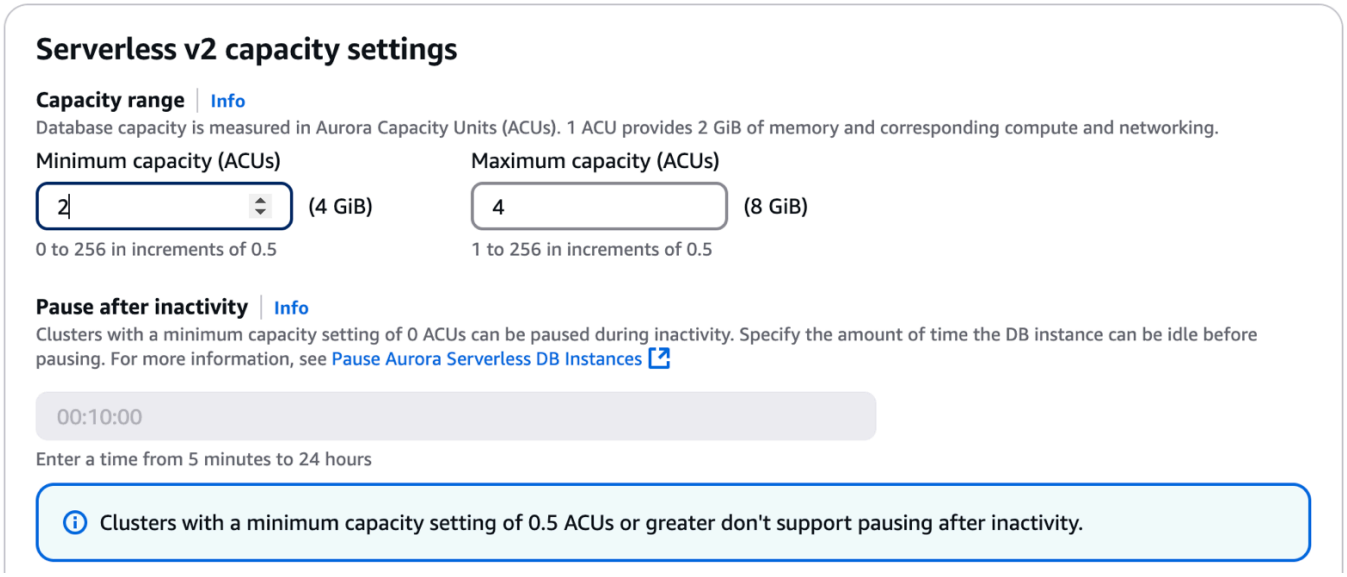
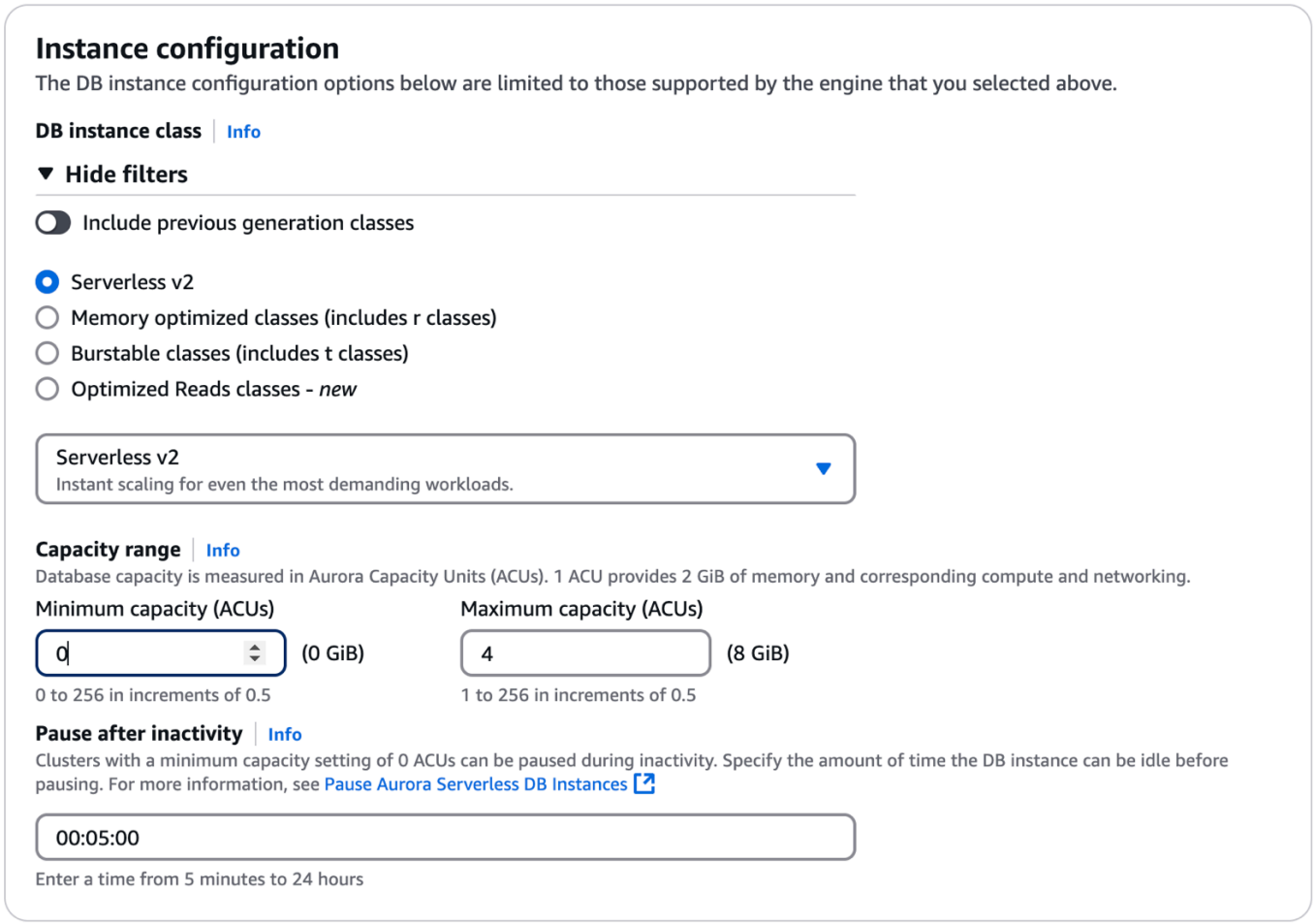
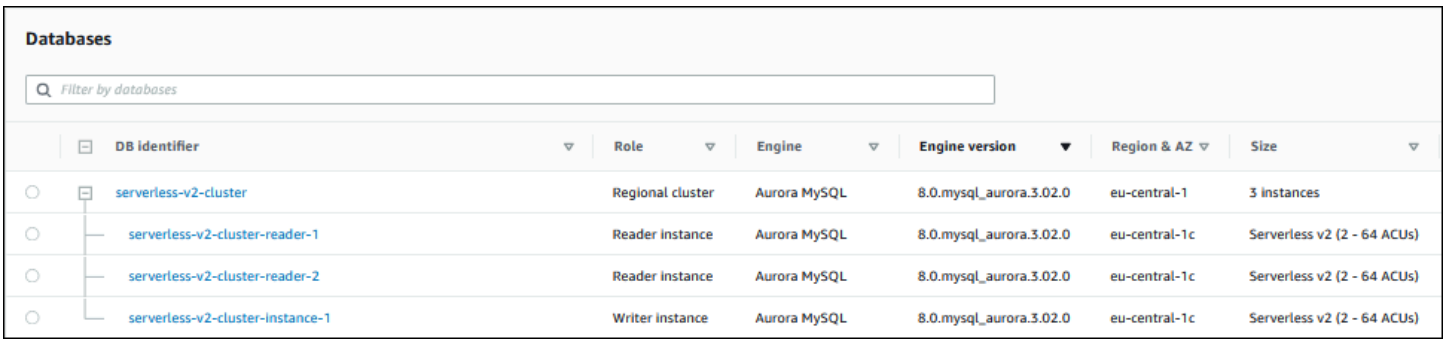
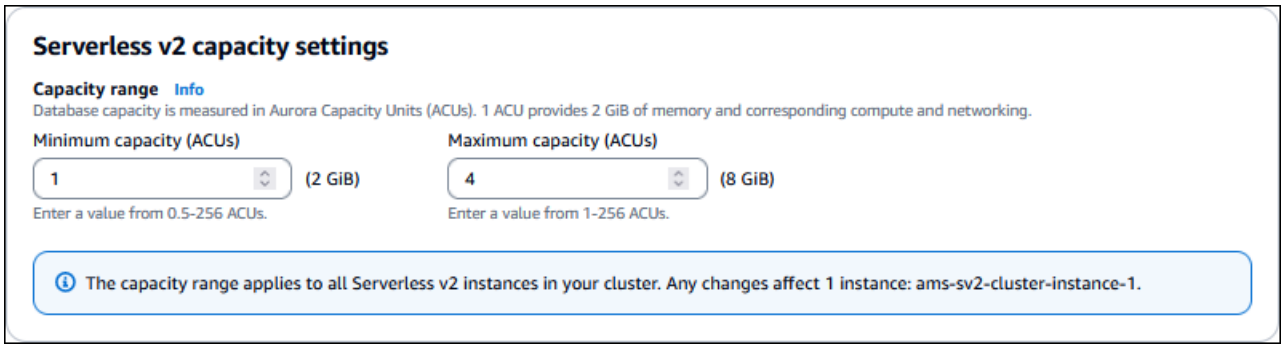
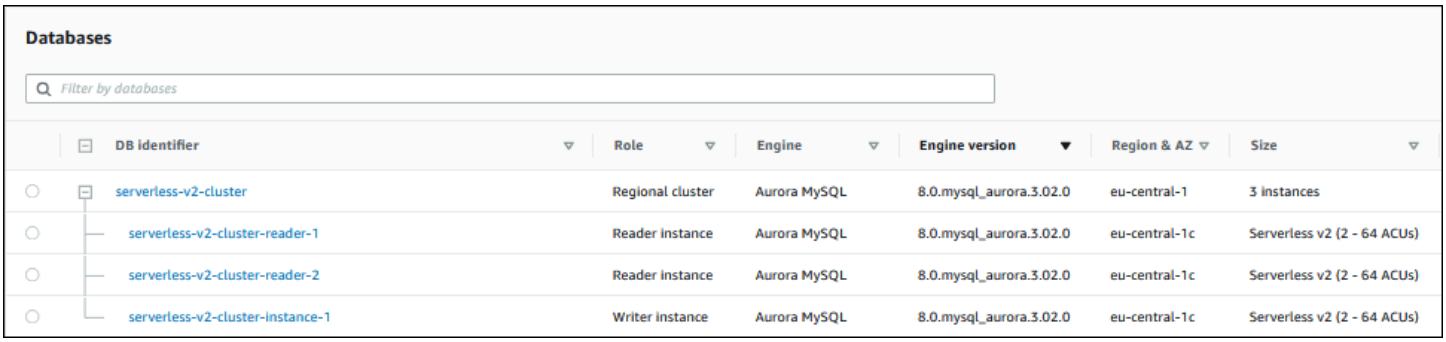
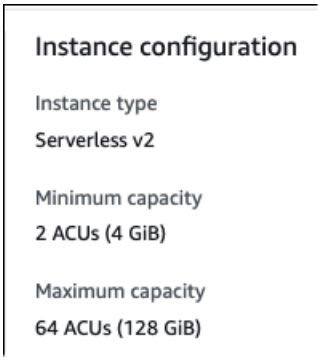
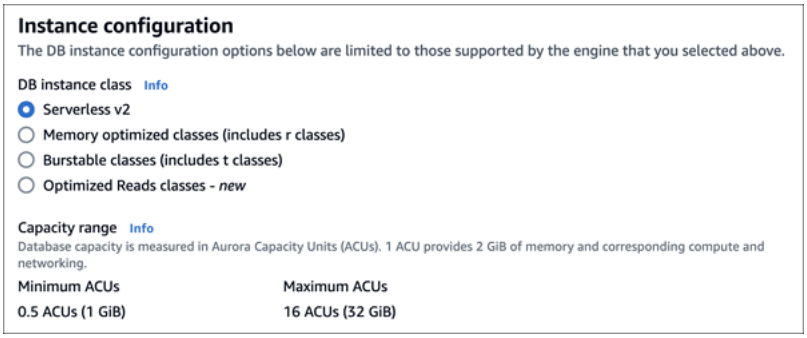
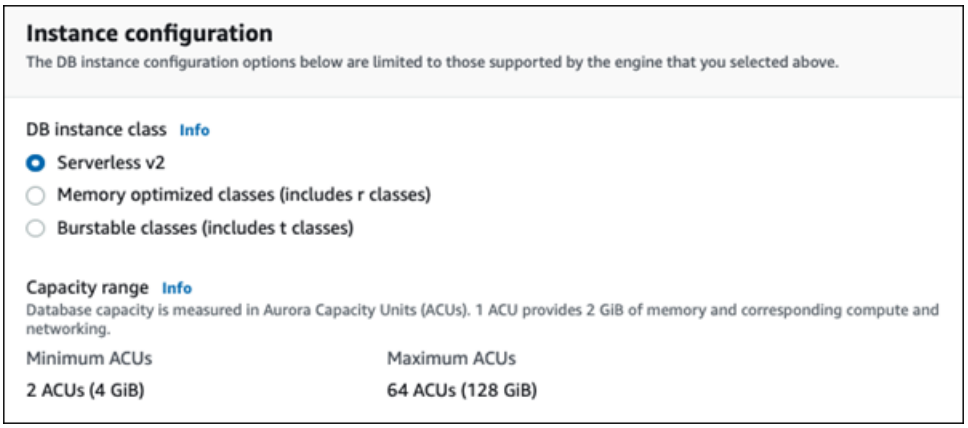
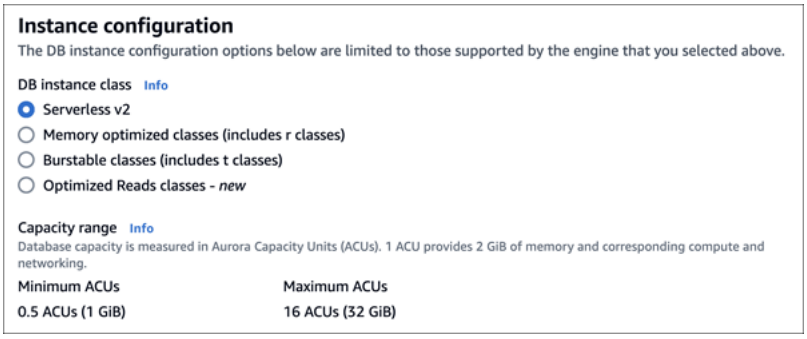
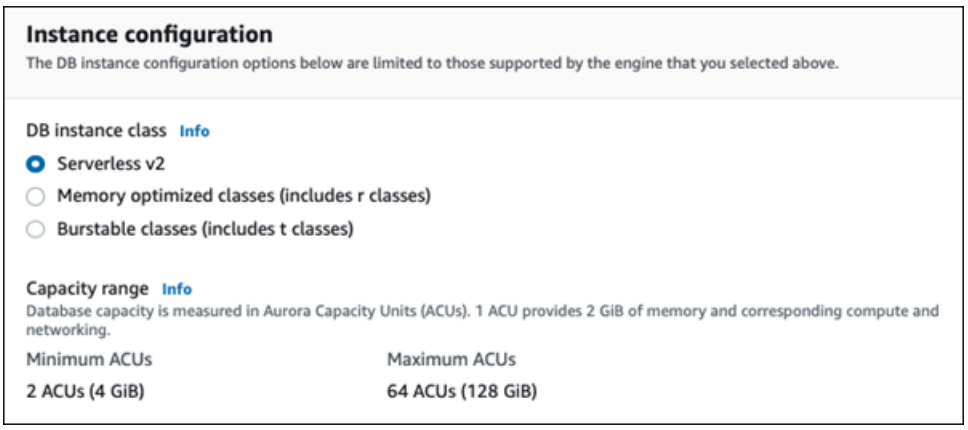
CREATE TABLE books_table (ID int NOT NULL, Title VARCHAR(50) NOT NULL, Author
 VARCHAR(50) NOT NULL,
Copyright INT NOT NULL, Genre VARCHAR(50) NOT NULL, PRIMARY KEY (ID));

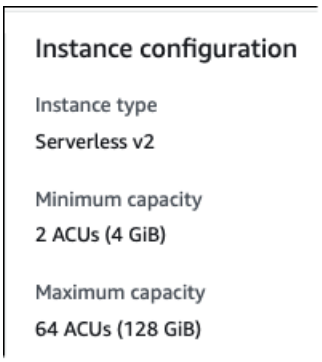
INSERT INTO books_table VALUES (1, 'The Shining', 'Stephen King', 1977, 'Supernatural
 fiction');
```

PostgreSQL

L'esempio seguente utilizza il terminale interattivo PostgreSQL `psql`. Quando ti connetti al cluster, includi il database che hai specificato durante la creazione dell'integrazione.

```
psql -h mycluster.cluster-123456789012.us-east-2.rds.amazonaws.com -p 5432 -U username
 -d named_db;

named_db=> CREATE TABLE books_table (ID int NOT NULL, Title VARCHAR(50) NOT NULL,
 Author VARCHAR(50) NOT NULL,
Copyright INT NOT NULL, Genre VARCHAR(50) NOT NULL, PRIMARY KEY (ID));

named_db=> INSERT INTO books_table VALUES (1, 'The Shining', 'Stephen King', 1977,
 'Supernatural fiction');
```

Interrogazione dei dati Aurora in Amazon Redshift

Dopo aver aggiunto i dati al cluster di database Aurora, questi vengono replicati nel database di destinazione e sono pronti per essere sottoposti a query.

Esecuzione di query sui dati replicati

1. Vai alla console Amazon Redshift e scegli Editor di query v2 nel riquadro di navigazione a sinistra.
2. Connettiti al cluster o al gruppo di lavoro e scegli il database di destinazione (creato dall'integrazione) dal menu a tendina (destination_database in questo esempio). Per istruzioni sulla creazione di un database di destinazione, consulta Creazione di un database di destinazione in Amazon Redshift.
3. Utilizzo di un'istruzione SELECT per eseguire query sui dati. In questo esempio, puoi eseguire il comando seguente per selezionare tutti i dati dalla tabella creata nel cluster di database Aurora di origine:

```
SELECT * from my_db."books_table";
```



- *my_db* è il nome dello schema del database Aurora. Questa opzione è necessaria solo per i database MySQL.
- *books_table* è il nome della tabella Aurora.

Puoi anche eseguire query sui dati utilizzando il client della riga di comando. Ad esempio:

```
destination_database=# select * from my_db."books_table";

 ID |       Title |        Author |   Copyright |                 Genre |  txn_seq |
 txn_id
----+-------------+---------------+-------------+-----------------------+----------
+--------+
  1 | The Shining |  Stephen King |        1977 |   Supernatural fiction |        2 |
 12192
```

> **Note**
>
> Per distinguere tra maiuscole e minuscole, usa le virgolette doppie (" ") per i nomi di schemi, tabelle e colonne. Per ulteriori informazioni, consulta enable_case_sensitive_identifier.

Differenze tra i tipi di dati tra i database Aurora e Amazon Redshift

Di seguito, le tabelle mostrano le mappature dei tipi di dati Aurora MySQL e Aurora PostgreSQL ai tipi di dati di destinazione corrispondenti. Amazon Aurora attualmente supporta solo questi tipi di dati per le integrazioni Zero-ETL.

Se una tabella nel cluster di database di origine include un tipo di dati non supportato, la tabella non viene sincronizzata e non è utilizzabile dalla destinazione. Lo streaming dall'origine alla destinazione va avanti, ma la tabella con il tipo di dati non supportato non è disponibile. Per correggere la tabella e renderla disponibile nella destinazione, devi annullare manualmente la modifica iniziale e aggiornare l'integrazione eseguendo `ALTER DATABASE...INTEGRATION REFRESH`.

> **Note**
>
> Non puoi aggiornare le integrazioni Zero-ETL con Amazon SageMaker Lakehouse. Piuttosto, elimina e prova a creare nuovamente l'integrazione.

Argomenti

- Aurora MySQL
- Best practice di Aurora PostgreSQL

Aurora MySQL

Tipo di dati Aurora MySQL	Tipo di dati di destinazione	Descrizione	Limitazioni
INT	INTEGER	Intero a quattro byte firmato	Nessuna

Tipo di dati Aurora MySQL	Tipo di dati di destinazione	Descrizione	Limitazioni
SMALLINT	SMALLINT	Intero a due byte firmato	Nessuno
TINYINT	SMALLINT	Intero a due byte firmato	Nessuno
MEDIUMINT	INTEGER	Intero a quattro byte firmato	Nessuno
BIGINT	BIGINT	Intero a otto byte firmato	Nessuno
INT UNSIGNED	BIGINT	Intero a otto byte firmato	Nessuno
TINYINT UNSIGNED	SMALLINT	Intero a due byte firmato	Nessuno
MEDIUMINT UNSIGNED	INTEGER	Intero a quattro byte firmato	Nessuno
BIGINT UNSIGNED	DECIMAL(20,0)	Numerico esatto di precisione selezionabile	Nessuno
DECIMAL(p,s) = NUMERIC(p,s)	DECIMAL(p,s)	Numerico esatto di precisione selezionabile	Precisione superiore a 38 e scala superiore a 37 non supportate
DECIMAL(p,s) UNSIGNED = NUMERIC(p,s) UNSIGNED	DECIMAL(p,s)	Numerico esatto di precisione selezionabile	Precisione superiore a 38 e scala superiore a 37 non supportate

Tipo di dati Aurora MySQL	Tipo di dati di destinazione	Descrizione	Limitazioni
FLOAT4/REAL	REAL	Numero in virgola mobile a precisione singola	Nessuno
FLOAT4/REAL UNSIGNED	REAL	Numero in virgola mobile a precisione singola	Nessuno
DOUBLE/REAL/FLOAT8	DOUBLE PRECISION	Numero in virgola mobile a precisione doppia	Nessuno
DOUBLE/REAL/FLOAT8 UNSIGNED	DOUBLE PRECISION	Numero in virgola mobile a precisione doppia	Nessuno
BIT(n)	VARBYTE(8)	Valore binario a lunghezza variabile	Nessuno
BINARY(n)	VARBYTE(n)	Valore binario a lunghezza variabile	Nessuno
VARBINARY(n)	VARBYTE(n)	Valore binario a lunghezza variabile	Nessuno
CHAR(n)	VARCHAR(n)	Valore di stringa di lunghezza variabile	Nessuno

Tipo di dati Aurora MySQL	Tipo di dati di destinazione	Descrizione	Limitazioni
VARCHAR(n)	VARCHAR(n)	Valore di stringa di lunghezza variabile	Nessuno
TEXT	VARCHAR(65535)	Valore di stringa di lunghezza variabile fino a 65.535 caratteri	Nessuno
TINYTEXT	VARCHAR(255)	Valore di stringa di lunghezza variabile fino a 255 caratteri	Nessuno
MEDIUMTEXT	VARCHAR(65535)	Valore di stringa di lunghezza variabile fino a 65.535 caratteri	Nessuno
LONGTEXT	VARCHAR(65535)	Valore di stringa di lunghezza variabile fino a 65.535 caratteri	Nessuno
ENUM	VARCHAR(1020)	Valore di stringa di lunghezza variabile fino a 1.020 caratteri	Nessuno
SET	VARCHAR(1020)	Valore di stringa di lunghezza variabile fino a 1.020 caratteri	Nessuno

Tipo di dati Aurora MySQL	Tipo di dati di destinazione	Descrizione	Limitazioni
DATE	DATE	Data di calendario (anno, mese, giorno)	Nessuno
DATETIME	TIMESTAMP	Data e ora (senza fuso orario)	Nessuno
TIMESTAMP(p)	TIMESTAMP	Data e ora (senza fuso orario)	Nessuno
TIME	VARCHAR(18)	Valore di stringa di lunghezza variabile fino a 18 caratteri	Nessuno
ANNO	VARCHAR(4)	Valore di stringa di lunghezza variabile fino a 4 caratteri	Nessuno
JSON	SUPER	Dati o documenti semistrutturati come valori	Nessuno

Best practice di Aurora PostgreSQL

Le integrazioni Zero-ETL per Aurora PostgreSQL non supportano tipi di dati personalizzati né tipi di dati creati da estensioni.

Tipo di dati Aurora PostgreSQL	Tipo di dati di Amazon Redshift	Descrizione	Limitazioni
array	SUPER	Dati o documenti semistrutturati come valori	Nessuno
bigint	BIGINT	Intero a otto byte firmato	Nessuno
bigserial	BIGINT	Intero a otto byte firmato	Nessuno
bit varying(n)	VARBYTE(n)	Valore binario di lunghezza variabile fino a 16.777.216 byte	Nessuno
bit(n)	VARBYTE(n)	Valore binario di lunghezza variabile fino a 16.777.216 byte	Nessuno
bit, bit varying	VARBYTE(16777216)	Valore binario di lunghezza variabile fino a 16.777.216 byte	Nessuno
booleano	BOOLEAN	Booleani logici (true/false)	Nessuno
bytea	VARBYTE(16777216)	Valore binario di lunghezza variabile fino a 16.777.216 byte	Nessuno
char(n)	CHAR(n)	Valore di stringa di caratteri a lunghezza fissa	Nessuno

Tipo di dati Aurora PostgreSQL	Tipo di dati di Amazon Redshift	Descrizione	Limitazioni
		fino a 65.535 byte	
char varying(n)	VARCHAR(65535)	Valore di stringa di caratteri a lunghezza variabile fino a 65.535 caratteri	Nessuno
cid	BIGINT	Intero a otto byte firmato	Nessuno
cidr	VARCHAR(19)	Valore di stringa di lunghezza variabile fino a 19 caratteri	Nessuno
date	DATE	Data di calendario (anno, mese, giorno)	I valori superiori a 294.276 A.D. non sono supportati
double precision	DOUBLE PRECISION	Numeri in virgola mobile a precisione doppia	Valori subnormali non completamente supportati
gtsvector	VARCHAR(65535)	Valore di stringa di lunghezza variabile fino a 65.535 caratteri	Nessuno
inet	VARCHAR(19)	Valore di stringa di lunghezza variabile fino a 19 caratteri	Nessuno

Tipo di dati Aurora PostgreSQL	Tipo di dati di Amazon Redshift	Descrizione	Limitazioni
integer	INTEGER	Intero a quattro byte firmato	Nessuno
int2vector	SUPER	Dati o documenti semistrutturati come valori.	Nessuno
intervallo	INTERVAL	Durata del tempo	Sono supportati solo i tipi INTERVAL che specificano un qualificatore da anno a mese o da giorno a secondo.
json	SUPER	Dati o documenti semistrutturati come valori	Nessuno
jsonb	SUPER	Dati o documenti semistrutturati come valori	Nessuno
jsonpath	VARCHAR(65535)	Valore di stringa di lunghezza variabile fino a 65.535 caratteri	Nessuno
macaddr	VARCHAR(17)	Valore di stringa di lunghezza variabile fino a 17 caratteri	Nessuno

Tipo di dati Aurora PostgreSQL	Tipo di dati di Amazon Redshift	Descrizione	Limitazioni
macaddr8	VARCHAR(23)	Valore di stringa di lunghezza variabile fino a 23 caratteri	Nessuno
money	DECIMAL(20,3)	Importi	Nessuno
nome	VARCHAR(64)	Valore di stringa di lunghezza variabile fino a 64 caratteri	Nessuno
numeric(p,s)	DECIMAL(p,s)	Valore di precisione fisso definito dall'utente	• Valori `NaN` non supportati • La precisione e la scala devono essere definite in modo esplicito e non devono essere superiori a 38 (precisione) e 37 (scala) • Scala negativa non supportata
oid	BIGINT	Intero a otto byte firmato	Nessuno
oidvector	SUPER	Dati o documenti semistrutturati come valori.	Nessuno

Tipo di dati Aurora PostgreSQL	Tipo di dati di Amazon Redshift	Descrizione	Limitazioni
pg_brin_bloom_summary	VARCHAR(65535)	Valore di stringa di lunghezza variabile fino a 65.535 caratteri	Nessuno
pg_dependencies	VARCHAR(65535)	Valore di stringa di lunghezza variabile fino a 65.535 caratteri	Nessuno
pg_lsn	VARCHAR(17)	Valore di stringa di lunghezza variabile fino a 17 caratteri	Nessuno
pg_mcv_list	VARCHAR(65535)	Valore di stringa di lunghezza variabile fino a 65.535 caratteri	Nessuno
pg_ndistinct	VARCHAR(65535)	Valore di stringa di lunghezza variabile fino a 65.535 caratteri	Nessuno
pg_node_tree	VARCHAR(65535)	Valore di stringa di lunghezza variabile fino a 65.535 caratteri	Nessuno
pg_snapshot	VARCHAR(65535)	Valore di stringa di lunghezza variabile fino a 65.535 caratteri	Nessuno

Tipo di dati Aurora PostgreSQL	Tipo di dati di Amazon Redshift	Descrizione	Limitazioni
real	REAL	Numero in virgola mobile a precisione singola	Valori subnormali non completamente supportati
refcursor	VARCHAR(65535)	Valore di stringa di lunghezza variabile fino a 65.535 caratteri	Nessuno
smallint	SMALLINT	Intero a due byte firmato	Nessuno
smallserial	SMALLINT	Intero a due byte firmato	Nessuno
seriale	INTEGER	Intero a quattro byte firmato	Nessuno
text	VARCHAR(65535)	Valore di stringa di lunghezza variabile fino a 65.535 caratteri	Nessuno
tid	VARCHAR(23)	Valore di stringa di lunghezza variabile fino a 23 caratteri	Nessuno
time [(p)] without time zone	VARCHAR(19)	Valore di stringa di lunghezza variabile fino a 19 caratteri	Valori `Infinity` e `-Infinity` non supportati

Tipo di dati Aurora PostgreSQL	Tipo di dati di Amazon Redshift	Descrizione	Limitazioni
time [(p)] with time zone	VARCHAR(22)	Valore di stringa di lunghezza variabile fino a 22 caratteri	Valori `Infinity` e `-Infinity` non supportati
timestamp [(p)] without time zone	TIMESTAMP	Data e ora (senza fuso orario)	• Valori `Infinity` e `-Infinity` non supportati • Valori maggiori di `9999-12-31` non supportati • Valori B.C. non supportati
timestamp [(p)] with time zone	TIMESTAMPTZ	Data e ora (con fuso orario)	• Valori `Infinity` e `-Infinity` non supportati • Valori maggiori di `9999-12-31` non supportati • Valori B.C. non supportati
tsquery	VARCHAR(65535)	Valore di stringa di lunghezza variabile fino a 65.535 caratteri	Nessuno

Tipo di dati Aurora PostgreSQL	Tipo di dati di Amazon Redshift	Descrizione	Limitazioni
tsvector	VARCHAR(65535)	Valore di stringa di lunghezza variabile fino a 65.535 caratteri	Nessuno
txid_snapshot	VARCHAR(65535)	Valore di stringa di lunghezza variabile fino a 65.535 caratteri	Nessuno
uuid	VARCHAR(36)	Stringa di 36 caratteri di lunghezza variabile	Nessuno
xid	BIGINT	Intero a otto byte firmato	Nessuno
xid8	DECIMAL(20, 0)	Decimale a precisione fissa	Nessuno
xml	VARCHAR(65535)	Valore di stringa di lunghezza variabile fino a 65.535 caratteri	Nessuno

Operazioni DDL per Aurora PostgreSQL

Amazon Redshift è derivato da PostgreSQL, quindi condivide diverse funzionalità con Aurora PostgreSQL a causa dell'architettura comune di PostgreSQL. Le integrazioni Zero-ETL sfruttano queste somiglianze per semplificare la replica dei dati da Aurora PostgreSQL ad Amazon Redshift, la mappatura di database per nome e l'utilizzo della struttura di tabella, schema e database condivisa.

Considera i seguenti punti quando gestisci le integrazioni Zero-ETL di Aurora PostgreSQL:

- L’isolamento è gestito a livello di database.
- La replica avviene a livello di database.
- I database Aurora PostgreSQL sono mappati ai database Amazon Redshift per nome, con i dati che fluiscono verso il corrispondente database Redshift rinominato se l’originale viene rinominato.

Nonostante le loro somiglianze, Amazon Redshift e Aurora PostgreSQL presentano differenze importanti. Le seguenti sezioni descrivono le risposte del sistema Amazon Redshift per le operazioni DDL comuni.

Argomenti

- Operazioni del database
- Operazioni dello schema
- Operazioni sulle tabelle

Operazioni del database

La tabella seguente mostra le risposte del sistema per le operazioni DDL del database.

Operazioni DDL	Risposta del sistema Redshift
`CREATE DATABASE`	Nessuna operazione
`DROP DATABASE`	Amazon Redshift elimina tutti i dati nel database Redshift di destinazione.
`RENAME DATABASE`	Amazon Redshift elimina tutti i dati nel database di destinazione originale e risincron izza i dati nel nuovo database di destinazi one. Se il nuovo database non esiste, devi crearlo manualmente. Per istruzioni, consulta Creazione di database di destinazione in Amazon Redshift.

Operazioni dello schema

La tabella seguente mostra le risposte del sistema per le operazioni DDL dello schema.

Operazioni DDL	Risposta del sistema Redshift
`CREATE SCHEMA`	Nessuna operazione
`DROP SCHEMA`	Amazon Redshift elimina lo schema originale.
`RENAME SCHEMA`	Amazon Redshift elimina lo schema originale, quindi risincronizza i dati nel nuovo schema.

Operazioni sulle tabelle

La tabella seguente mostra le risposte del sistema per le operazioni DDL della tabella.

Operazioni DDL	Risposta del sistema Redshift
`CREATE TABLE`	Amazon Redshift crea la tabella. Alcune operazioni impediscono la creazione della tabella, ad esempio la creazione di una tabella senza una chiave primaria o l'esecuzione del partizionamento dichiarativo. Per ulteriori informazioni, consulta the section called "Limitazioni" e the section called "Risoluzione dei problemi delle integrazioni Zero-ETL".
`DROP TABLE`	Amazon Redshift elimina la tabella.
`TRUNCATE TABLE`	Amazon Redshift tronca la tabella.
`ALTER TABLE (RENAME... )`	Amazon Redshift rinomina la tabella o la colonna.
`ALTER TABLE (SET SCHEMA)`	Amazon Redshift elimina la tabella nello schema originale e risincronizza la tabella nel nuovo schema.
`ALTER TABLE (ADD PRIMARY KEY)`	Amazon Redshift aggiunge una chiave primaria e risincronizza la tabella.

Operazioni DDL	Risposta del sistema Redshift
`ALTER TABLE (ADD COLUMN)`	Amazon Redshift aggiunge una colonna alla tabella.
`ALTER TABLE (DROP COLUMN)`	Amazon Redshift elimina la colonna se non è una colonna di chiave primaria. Altrimenti, risincronizza la tabella.
`ALTER TABLE (SET LOGGED/UNLOGGED )`	Se modifichi la tabella in registrata, Amazon Redshift risincronizza la tabella. Se modifichi la tabella in non registrata, Amazon Redshift elimina la tabella.

Visualizzazione e monitoraggio delle integrazioni Zero-ETL Aurora

Puoi visualizzare i dettagli di un'integrazione Zero-ETL di Amazon Aurora per visualizzarne le informazioni di configurazione e lo stato attuale. Puoi anche monitorare lo stato della tua integrazione eseguendo query sulle viste di sistema specifiche in Amazon Redshift. Inoltre, Amazon Redshift pubblica alcuni parametri relativi all'integrazione su Amazon, che puoi visualizzare all'interno della CloudWatch console Amazon Redshift.

Argomenti

- Visualizzazione delle integrazioni
- Monitoraggio delle integrazioni tramite tabelle di sistema per Amazon Redshift
- Monitoraggio delle integrazioni con Amazon EventBridge per Amazon Redshift

Visualizzazione delle integrazioni

Puoi visualizzare le integrazioni zero-ETL di Aurora utilizzando Console di gestione AWS l'API RDS o la AWS CLI.

Console

Visualizzazione dei dettagli di un'integrazione Zero-ETL

1. Accedi a Console di gestione AWS e apri la console Amazon RDS all'indirizzo https://console.aws.amazon.com/rds/.
2. Nel riquadro di navigazione a sinistra, scegli Integrazioni Zero-ETL.
3. Seleziona un'integrazione per visualizzarne ulteriori dettagli, ad esempio il cluster di database di origine e il data warehouse di destinazione.



Un'integrazione può avere i seguenti stati:

- `Creating`: l'integrazione è in fase di creazione.
- `Active`: l'integrazione sta inviando dati transazionali al data warehouse di destinazione.
- `Syncing`: l'integrazione ha rilevato un errore recuperabile e deve reimpostare i dati. Le tabelle interessate non sono disponibili per l'esecuzione di query fino al termine della risincronizzazione.
- `Needs attention`: l'integrazione ha rilevato un evento o un errore che richiede un intervento manuale per la risoluzione. Per correggere il problema, segui le istruzioni nel messaggio di errore nella pagina dei dettagli dell'integrazione.
- `Failed`: l'integrazione ha rilevato un evento o un errore irreversibile che non può essere risolto. È necessario eliminare e ricreare l'integrazione.
- `Deleting`: l'integrazione è in fase di eliminazione.

AWS CLI

Per visualizzare tutte le integrazioni zero-ETL nell'account corrente utilizzando AWS CLI, usa il comando describe-integrations e specifica l'opzione. `--integration-identifier`

Example

Per Linux, macOS o Unix:

```
aws rds describe-integrations \
    --integration-identifier ee605691-6c47-48e8-8622-83f99b1af374
```

Per Windows:

```
aws rds describe-integrations ^
    --integration-identifier ee605691-6c47-48e8-8622-83f99b1af374
```

API RDS

Per visualizzare l'integrazione Zero-ETL utilizzando l'API Amazon RDS, usa l'operazione `DescribeIntegrations` con il parametro `IntegrationIdentifier`.

Monitoraggio delle integrazioni tramite tabelle di sistema per Amazon Redshift

Amazon Redshift include tabelle e viste di sistema che contengono informazioni sul funzionamento del sistema. Puoi eseguire delle query su queste tabelle e viste esattamente come faresti con qualsiasi altra tabella di database. Per ulteriori informazioni sulle tabelle e viste di sistema in Amazon Redshift, consulta Riferimento di tabelle e viste di sistema nella Guida per sviluppatori di database di Amazon Redshift.

Puoi eseguire query sulle seguenti viste e tabelle di sistema per ottenere informazioni sulle integrazioni Zero-ETL Aurora:

- SVV_INTEGRATION: fornisce i dettagli relativi alla configurazione delle integrazioni.
- SVV_INTEGRATION_TABLE_STATE: descrive lo stato di ogni tabella all'interno di un'integrazione.
- SYS_INTEGRATION_TABLE_STATE_CHANGE: visualizza i log delle modifiche dello stato della tabella per un'integrazione.
- SYS_INTEGRATION_ACTIVITY: fornisce informazioni sulle esecuzioni completate delle integrazioni.

Tutte le metriche Amazon relative all'integrazione provengono da CloudWatch Amazon Redshift. Per ulteriori informazioni, consulta Metriche per le integrazioni Zero-ETL nella Guida alla gestione di Amazon Redshift. Attualmente, Amazon Aurora non pubblica alcun parametro di integrazione su. CloudWatch

Monitoraggio delle integrazioni con Amazon EventBridge per Amazon Redshift

Amazon Redshift invia eventi relativi all'integrazione ad Amazon. EventBridge Per un elenco di eventi e l'evento corrispondente IDs, consulta Notifiche degli eventi di integrazione Zero-ETL con Amazon EventBridge nella Amazon Redshift Management Guide.

Modifica delle integrazioni Zero-ETL di Aurora

È possibile modificare solo il nome, la descrizione e le opzioni di filtro dei dati per un'integrazione Zero-ETL in un data warehouse supportato. Non è possibile modificare la chiave AWS KMS utilizzata per crittografare l'integrazione o i database di origine o di destinazione.

Se aggiungi un filtro dati a un'integrazione esistente, Aurora rivaluta il filtro come se fosse sempre esistito. Rimuove tutti i dati attualmente presenti nel data warehouse di destinazione che non corrispondono ai nuovi criteri di filtro. Se rimuovi un filtro dati da un'integrazione, verranno replicati tutti i dati che in precedenza non corrispondevano ai criteri di filtro (ma che ora lo fanno) nel data warehouse di destinazione. Per ulteriori informazioni, consulta the section called "Filtro dei dati per le integrazioni Zero-ETL".

Puoi modificare un'integrazione Zero-ETL utilizzando la Console di gestione AWS, la AWS CLI o l'API Amazon RDS.

Console RDS

Per modificare un'integrazione Zero-ETL

1. Accedi alla Console di gestione AWS e apri la console Amazon RDS all'indirizzo https://console.aws.amazon.com/rds/.
2. Nel pannello di navigazione, scegli Integrazioni Zero-ETL, quindi seleziona l'integrazione che desideri modificare.
3. Scegli Modifica e apporta modifiche a tutte le impostazioni disponibili.
4. Una volta apportate tutte le modifiche, seleziona Modifica.

AWS CLI

Per modificare un'integrazione Zero-ETL utilizzando la AWS CLI, chiama il comando modify-integration. Oltre a `--integration-identifier`, specifica una delle seguenti opzioni:

- `--integration-name`: specifica un nuovo nome per l'integrazione.
- `--description`: specifica una nuova descrizione per l'integrazione.
- `--data-filter`: specifica le opzioni di filtro dei dati per l'integrazione. Per ulteriori informazioni, consulta the section called "Filtro dei dati per le integrazioni Zero-ETL".

Example

La richiesta seguente modifica un'integrazione esistente.

Per Linux, macOS o Unix:

```
aws rds modify-integration \
    --integration-identifier ee605691-6c47-48e8-8622-83f99b1af374 \
    --integration-name my-renamed-integration
```

Per Windows:

```
aws rds modify-integration ^
    --integration-identifier ee605691-6c47-48e8-8622-83f99b1af374 ^
    --integration-name my-renamed-integration
```

API RDS

Per modificare un'integrazione Zero-ETL utilizzando l'API RDS, chiama l'operazione ModifyIntegration. Specifica l'identificatore di integrazione e i parametri che desideri modificare.

Eliminazione delle integrazioni Zero-ETL di Aurora

Quando elimini un'integrazione Zero-ETL, Amazon Aurora la rimuove dal cluster di database Aurora di origine. I dati transazionali non vengono eliminati da Amazon Aurora o dalla destinazione di analisi, ma Aurora, ma non invia nuovi dati ad Amazon Redshift o Amazon SageMaker.

È possibile eliminare un'integrazione solo quando ha lo stato di `Active`, `Failed`, `Syncing` o `Needs attention`.

È possibile eliminare le integrazioni Zero-ETL utilizzando la Console di gestione AWS, la AWS CLI o l'API RDS.

Console

Eliminazione di un'integrazione Zero-ETL

1. Accedi alla Console di gestione AWS e apri la console Amazon RDS all'indirizzo https://console.aws.amazon.com/rds/.
2. Nel riquadro di navigazione a sinistra, scegli Integrazioni Zero-ETL.
3. Seleziona l'integrazione Zero-ETL che desideri eliminare.
4. Scegli Operazioni, Elimina dominio, quindi conferma l'eliminazione.

AWS CLI

Per eliminare un'integrazione Zero-ETL, usa il comando delete-integration e specifica l'opzione `--integration-identifier`.

Example

Per Linux, macOS o Unix:

```
aws rds delete-integration \
    --integration-identifier ee605691-6c47-48e8-8622-83f99b1af374
```

Per Windows:

```
aws rds delete-integration ^
    --integration-identifier ee605691-6c47-48e8-8622-83f99b1af374
```

API RDS

Per eliminare un'integrazione Zero-ETL utilizzando l'API Amazon RDS, usa l'operazione `DeleteIntegration` con il parametro `IntegrationIdentifier`.

Risoluzione dei problemi delle integrazioni Zero-ETL di Aurora

Puoi verificare lo stato di un'integrazione Zero-ETL eseguendo query sulla tabella di sistema SVV_INTEGRATION nella destinazione di analisi. Se la colonna `state` include il valore

`ErrorState`, significa che si è verificato un problema. Per ulteriori informazioni, consulta the section called “Monitoraggio tramite tabelle di sistema”.

Usa le seguenti informazioni per risolvere i problemi comuni con le integrazioni Zero-ETL di Aurora.

> **Important**
>
> Le operazioni di risincronizzazione e aggiornamento non sono disponibili per le integrazioni Zero-ETL con un Amazon SageMaker AI Lakehouse. In caso di problemi con un'integrazione, è necessario eliminare l'integrazione e crearne una nuova. Non è possibile aggiornare né risincronizzare un'integrazione esistente.

Argomenti

- Non riesco a creare un'integrazione Zero-ETL
- La mia integrazione è bloccata in uno stato di Syncing
- Le mie tabelle non si replicano su Amazon Redshift
- Una o più tabelle Amazon Redshift richiedono una risincronizzazione
- Problemi di integrazione non riuscita per le integrazioni Zero-ETL di Amazon SageMaker AI Lakehouse
- Le modifiche DDL vengono apportate in Amazon Redshift prima del completamento della transazione DDL per Aurora PostgreSQL

Non riesco a creare un'integrazione Zero-ETL

Se non riesci a creare un'integrazione Zero-ETL, assicurati che quanto segue sia corretto per il database di origine:

- Il database di origine deve eseguire una versione del motore di database supportata. Per un elenco delle versioni supportate, consulta the section called “Integrazioni Zero-ETL ”.
- I parametri del database sono stati configurati correttamente. Se i parametri obbligatori sono impostati in modo errato o non sono associati al database, la creazione ha esito negativo. Per informazioni, consulta the section called “Fase 1: creazione di un gruppo di parametri del cluster di database personalizzato”.

Inoltre, assicurati che quanto segue sia corretto per il data warehouse di destinazione:

- È abilitata la distinzione tra maiuscole e minuscole. Consulta Attivazione della distinzione tra maiuscole e minuscole per il data warehouse.
- Hai aggiunto il principale autorizzato e l'origine dell'integrazione corretti. Consulta l'articolo Configurazione dell'autorizzazione per il data warehouse Amazon Redshift.
- Il data warehouse è crittografato (se si tratta di un cluster con provisioning). Consulta l'articolo Crittografia dei database di Amazon Redshift.

La mia integrazione è bloccata in uno stato di **`Syncing`**

L'integrazione potrebbe mostrare costantemente uno stato di `Syncing` se modifichi il valore di uno dei parametri richiesti per il database.

Per risolvere questo problema, controlla i valori dei parametri nel gruppo di parametri associato al cluster di database di origine e assicurati che corrispondano ai valori richiesti. Per ulteriori informazioni, consulta the section called "Fase 1: creazione di un gruppo di parametri del cluster di database personalizzato".

Se modifichi qualsiasi parametro, assicurati di riavviare il cluster di database per applicare le modifiche apportate.

Le mie tabelle non si replicano su Amazon Redshift

Se una o più tabelle non sono riflesse in Amazon Redshift, è possibile eseguire il comando riportato di seguito per risincronizzarle:

```
ALTER DATABASE dbname INTEGRATION REFRESH TABLES table1, table2;
```

Per ulteriori informazioni, consulta ALTER DATABASE nella Referenza SQL di Amazon Redshift.

È possibile che i dati non vengano replicati perché una o più tabelle di origine non dispongono di una chiave primaria. La dashboard di monitoraggio in Amazon Redshift mostra lo stato di queste tabelle come `Failed` e lo stato dell'integrazione Zero-ETL complessiva cambia in `Needs attention`. Per risolvere questo problema, puoi identificare una chiave esistente nella tabella che può diventare una chiave primaria oppure puoi aggiungere una chiave primaria sintetica. Per soluzioni dettagliate, consulta le seguenti risorse:

- Handle tables without primary keys while creating Amazon Aurora MySQL or Amazon RDS per MySQL zero-ETL integrations with Amazon Redshift

- Handle tables without primary keys while creating Amazon Aurora PostgreSQL zero-ETL integrations with Amazon Redshift

Una o più tabelle Amazon Redshift richiedono una risincronizzazione

L'esecuzione di determinati comandi sul database di origine potrebbe richiedere la risincronizzazione delle tabelle. In questi casi, la vista di sistema SVV_INTEGRATION_TABLE_STATE mostra un valore di `table_state` pari a `ResyncRequired`, il che significa che l'integrazione deve ricaricare completamente i dati di quella tabella specifica da MySQL in Amazon Redshift.

Quando viene avviata la risincronizzazione della tabella, lo stato diventa `Syncing`. Non è necessario eseguire alcuna azione manuale per risincronizzare una tabella. Non potrai accedere ai dati della tabella in Amazon Redshift mentre è in esecuzione la loro risincronizzazione.

Di seguito sono riportati alcuni esempi di operazioni che possono modificare lo stato di una tabella in `ResyncRequired` e le possibili alternative da considerare.

Operazione	Esempio	In alternativa
Aggiunta di una colonna in una posizione specifica	`ALTER TABLE table_name ADD COLUMN column_name INTEGER NOT NULL first;`	Amazon Redshift non supporta l'aggiunta di colonne in posizioni specifiche utilizzando le parole chiave `first` o `after`. Se l'ordine delle colonne nella tabella di destinazione non è rilevante, aggiungi

Operazione	Esempio	In alternativa
		la colonna alla fine della tabella utilizzando un comando più semplice: `ALTER TABLE table_name ADD COLUMN column_name column_type ;`

Operazione	Esempio	In alternativa

Aggiunta di una colonna timestamp con il valore predefinito to `CURRENT_TIMESTAMP`	`ALTER TABLE table_name`
	`  ADD COLUMN column_name  TIMESTAMP`
	`  NOT NULL DEFAULT CURRENT_TIMESTAMP;`
		Il valore `CURRENT_TIMESTAMP` per le righe tabelle esistenti viene calcolato da Aurora MySQL e non può essere simulato in Amazon Redshift senza eseguire la risincronizzazione completa dei dati della tabella.
		
		Se possibile, converti il valore predefinito in una costante letterale, ad esempio `2023-01-01 00:00:15`, per evitare la latenza a livello di

Operazione	Esempio	In alternativa
		disponibilità della tabella.
Esecuzione di operazioni su più colonne all'interno di un unico comando	`ALTER TABLE table_name ADD COLUMN column_1, RENAME COLUMN column_2 TO column_3;`	Prendi in considerazione la possibilità di suddividere il comando in due operazioni distinte, `ADD` e `RENAME`, che non richiedono la risincronizzazione.

Problemi di integrazione non riuscita per le integrazioni Zero-ETL di Amazon SageMaker AI Lakehouse

Se riscontri problemi con un'integrazione Zero-ETL esistente con un Amazon SageMaker AI Lakehouse, l'unica soluzione è eliminare l'integrazione e crearne una nuova. A differenza di altri servizi AWS, le integrazioni Zero-ETL non supportano le operazioni di aggiornamento né quelle di risincronizzazione.

Per risolvere i problemi di integrazione:

1. Elimina l'integrazione Zero-ETL problematica utilizzando la console, l'interfaccia a riga di comando o l'API.
2. Verifica che le configurazioni del database di origine e del data warehouse di destinazione siano corrette.
3. Crea una nuova integrazione Zero-ETL con la stessa configurazione o con una configurazione aggiornata.

Questo processo comporterà una reinizializzazione completa della pipeline di dati, che potrebbe richiedere del tempo a seconda delle dimensioni del database di origine.

Le modifiche DDL vengono apportate in Amazon Redshift prima del completamento della transazione DDL per Aurora PostgreSQL

Le modifiche DDL possono apparire in Amazon Redshift prima del termine di un'operazione DDL nelle integrazioni Zero-ETL di Aurora PostgreSQL. Per ulteriori informazioni, consulta Operazioni DDL per Aurora PostgreSQL.

Uso di Aurora Serverless v2

Aurora Serverless v2 è una configurazione con scalabilità automatica on demand di Amazon Aurora.Aurora Serverless v2 aiuta ad automatizzare i processi di monitoraggio del carico di lavoro e di calibrazione della capacità dei database. La capacità viene regolata automaticamente in base alle necessità dall'applicazione. Ti vengono addebitate solo le risorse utilizzate dai cluster database. Quindi, con Aurora Serverless v2 puoi rientrare nel budget ed evitare di pagare per le risorse informatiche che non utilizzi.

Questo tipo di automazione è particolarmente utile per database multi-tenant, database distribuiti, sistemi di sviluppo e di test e altri ambienti con carichi di lavoro fortemente variabili e non prevedibili.

Argomenti

- Casi d'uso di Aurora Serverless v2
- Vantaggi di Aurora Serverless v2
- Funzionamento di Aurora Serverless v2
- Requisiti e limitazioni per Aurora Serverless v2
- Creazione di un cluster di database che utilizza Aurora Serverless v2
- Gestione di cluster di database Aurora Serverless v2
- Prestazioni e dimensionamento per Aurora Serverless v2
- Dimensionamento a zero ACU con pausa e ripresa automatiche per Aurora Serverless v2
- Migrazione a Aurora Serverless v2

Casi d'uso di Aurora Serverless v2

Aurora Serverless v2 supporta molti tipi di carichi di lavoro di database. Questi variano da ambienti di sviluppo e test, a siti Web e applicazioni con carichi di lavoro imprevedibili, fino alle applicazioni business-critical più esigenti che richiedono scalabilità e disponibilità elevate.

Aurora Serverless v2 è particolarmente utile per i seguenti casi d'uso:

- Carichi di lavoro variabili. Esegui carichi di lavoro con aumenti improvvisi e non prevedibili dell'attività. Un esempio può essere quello di un sito che offre informazioni sulla viabilità, che riscontra un forte aumento dell'attività quando inizia a piovere. Un altro è un sito di e-commerce che aumenta il proprio traffico durante un'offerta o una vendita promozionale. Con Aurora

Serverless v2, il database dimensiona automaticamente la capacità per soddisfare le esigenze del picco di carico dell'applicazione e la riduce quando il picco è terminato. Grazie a Aurora Serverless v2, non dovrai più effettuare il provisioning per la capacità media o di picco. È possibile specificare un limite massimo di capacità per gestire la peggiore situazione possibile, che viene raggiunto solo in caso di necessità.

La granularità del dimensionamento di Aurora Serverless v2 consente di variare precisamente la capacità in base alle esigenze del database. Per un cluster con provisioning, il dimensionamento richiede l'aggiunta di un'istanza database completamente nuova. Per eseguire il dimensionamento verso l'alto di un cluster, Aurora Serverless v1 raddoppia il numero di unità di capacità Aurora (ACU), ad esempio aumentandole da 16 a 32 o da 32 a 64. Al contrario, quando è necessario solo un piccolo aumento di capacità, Aurora Serverless v2 è in grado di scalare di mezza ACU. Può aggiungere 0,5, 1, 1,5, 2 ACU o eseguire incrementi ulteriori di mezza ACU in base alla capacità necessaria per affrontare l'aumento del carico di lavoro. E può eliminare 0,5, 1, 1,5, 2 ACU o eseguire decrementi ulteriori di mezza ACU quando il carico di lavoro diminuisce e tale capacità non è più necessaria.

- Applicazioni multi-tenant. Grazie a Aurora Serverless v2, non devi gestire individualmente la capacità del database per ogni applicazione utilizzata nel parco istanze. Aurora Serverless v2 gestisce la capacità del database a livello individuale per tuo conto.

  È possibile creare un cluster per ciascun tenant. Ciò ti permette di utilizzare funzionalità come la clonazione, il ripristino delle snapshot e i database globali Aurora per potenziare l'elevata disponibilità e il ripristino di emergenza in base alle esigenze di ciascun tenant.

  Ogni tenant potrebbe avere periodi di attività e inattività specifici, che variano a seconda dell'ora, del periodo dell'anno, degli eventi promozionali e così via. Ogni cluster è dotato di un'ampia gamma di capacità. In questo modo, dovrai affrontare costi minimi per le istanze database dei cluster con bassa attività. Tutti i cluster sono in grado di scalare rapidamente per gestire periodi di attività elevata.
- Nuove applicazioni. Supponiamo che desideri distribuire una nuova applicazione e hai dei dubbi riguardo alle dimensioni dell'istanza da utilizzare. Grazie a Aurora Serverless v2, puoi configurare un cluster con una o più istanze database e far sì che il database venga dimensionato automaticamente in base ai requisiti di capacità dell'applicazione.
- Applicazioni ad uso misto. Supponiamo che tu abbia un'applicazione OLTP (Online Transaction Processing), ma che si verifichino periodicamente picchi nel traffico di query. In questo caso, potresti specificare livelli di promozione per le istanze database Aurora Serverless v2 in un cluster e configurare quest'ultimo in modo che le istanze database di lettura e di scrittura scalino in

maniera indipendente in base al carico. Quando il picco di utilizzo diminuisce, le istanze database di lettura si ridimensionano adeguandosi alla capacità dell'istanza database di scrittura.

- Pianificazione della capacità. Poniamo l'esempio che tu debba regolare la capacità del database o verificare la capacità ottimale del database in base al tuo carico di lavoro. Per farlo, dovresti modificare le classi di tutte le istanze database in un cluster. Aurora Serverless v2 ti permette di evitare questo sovraccarico amministrativo. È possibile determinare la capacità minima e massima adeguata eseguendo il carico di lavoro e verificando la scalabilità effettiva delle istanze database.

  È possibile modificare le istanze database con provisioning esistenti in Aurora Serverless v2 oppure da Aurora Serverless v2 a istanze con provisioning. In questi casi, non è necessario creare un nuovo cluster o una nuova istanza database.

  Con un database globale Aurora, il bisogno di capacità dei cluster secondari potrebbe essere minore rispetto a quella del cluster primario. Utilizza istanze database Aurora Serverless v2 nei cluster secondari. In questo modo, la capacità del cluster può aumentare se una regione secondaria viene promossa e assume il carico di lavoro dell'applicazione.
- Sviluppo e test: oltre a eseguire le applicazioni più esigenti, è possibile anche utilizzare Aurora Serverless v2 per ambienti di sviluppo e test. Con Aurora Serverless v2, è possibile creare istanze database con una capacità minima bassa anziché utilizzare le classi di istanza database db.t*. È possibile impostare una capacità massima sufficientemente elevata da consentire a tali istanze database di eseguire carichi di lavoro considerevoli senza esaurire la memoria. Quando il database non è in uso, tutte le istanze database vengono ridotte per evitare costi inutili.

  > **Tip**
  >
  > Per facilitare la creazione di un nuovo cluster con Aurora Serverless v2 negli ambienti di sviluppo e test, la Console di gestione AWS fornisce il collegamento Easy create (Creazione rapida). Se scegli l'opzione Dev/Test, Aurora crea un cluster con un'istanza database Aurora Serverless v2 e un intervallo di capacità tipico di un sistema di sviluppo e test.

Utilizzo di Aurora Serverless v2 per carichi di lavoro con provisioning esistenti

Supponiamo che tu disponga già di un'applicazione Aurora in esecuzione su un cluster con provisioning. È possibile verificare come funzionerebbe l'applicazione con Aurora Serverless v2

aggiungendo una o più istanze database Aurora Serverless v2 di lettura nel cluster esistente. È possibile verificare la frequenza con cui le istanze di lettura scalano verso l'alto e verso il basso. È possibile utilizzare il meccanismo di failover Aurora per promuovere un'istanza database Aurora Serverless v2 rendendola di scrittura e controllare come gestisce il carico di lavoro di lettura/scrittura. Questo passaggio viene eseguito con tempi di inattività minimi e senza modificare l'endpoint utilizzato dalle applicazioni client. Per informazioni dettagliate sulla procedura per convertire i cluster esistenti inAurora Serverless v2, consulta Migrazione a Aurora Serverless v2.

Vantaggi di Aurora Serverless v2

Aurora Serverless v2 è adatto per i carichi di lavoro variabili o tendenti ai picchi. A causa dei carichi di lavoro non prevedibili, potresti avere difficoltà a pianificare le modifiche della capacità del database. Inoltre, modificare la capacità tramite le procedure comuni, come aggiungendo istanze database o modificando le classi delle istanze database, potrebbe richiedere troppo tempo. In tali casi d'uso, Aurora Serverless v2 fornisce i seguenti vantaggi:

- Gestione della capacità più semplice rispetto al provisioning.Aurora Serverless v2 Riduce lo sforzo necessario per la pianificazione delle dimensioni delle istanze database e per il loro dimensionamento in base alla variazione del carico di lavoro. Inoltre, riduce lo sforzo per mantenere la capacità coerente per tutte le istanze database in un cluster.
- Scalabilità più rapida e semplice durante i periodi di attività intensa. Aurora Serverless v2 dimensiona la capacità di elaborazione e memoria in base alle necessità, senza alcuna interruzione delle transazioni client o del carico di lavoro complessivo. Utilizzando Aurora Serverless v2 insieme alle istanze database di lettura, puoi sfruttare sia il dimensionamento orizzontale sia quello verticale. I database globali Aurora, inoltre, ti permettono di distribuire il carico di lavoro in lettura di Aurora Serverless v2 su più Regioni AWS. Questa funzionalità offre una praticità maggiore rispetto ai meccanismi di dimensionamento dei cluster con provisioning. È anche più veloce e maggiormente granulare rispetto alle funzionalità di diminsionamento di Aurora Serverless v1.
- Economico durante i periodi di bassa attività. Aurora Serverless v2 ti aiuta a evitare l'overprovisioning delle istanze database. Aurora Serverless v2 aggiunge risorse con incrementi granulari quando le istanze database scalano verso l'alto. Paghi soltanto per le risorse del database che utilizzi. Il costo delle risorse Aurora Serverless v2 utilizzate viene calcolato al secondo. In questo modo, quando un'istanza database scala verso il basso, la riduzione dell'utilizzo delle risorse viene registrata immediatamente.

- Parità delle funzionalità migliore. Aurora Serverless v2 fornisce numerose funzionalità Aurora non disponibili in Aurora Serverless v1. Tra le funzionalità offerte da Aurora Serverless v2 ci sono istanze database di lettura, database globali, Autenticazione del database AWS Identity and Access Management (IAM) e Performance Insights. Hai anche a diposizione molti più parametri di configurazione rispetto a quelli presenti in Aurora Serverless v1.

  In particolare, Aurora Serverless v2 permette di usufruire delle seguenti funzionalità dei cluster con provisioning:

  - Istanze database di lettura: con Aurora Serverless v2 puoi sfruttare il dimensionamento orizzontale grazie alle istanze database di lettura. Quando un cluster contiene una o più istanze database di lettura, può eseguire immediatamente il failover se si verificano problemi con l'istanza database di scrittura. Questa funzionalità non è disponibile con Aurora Serverless v1.
  - Cluster multi-AZ: permettono di distribuire le istanze database Aurora Serverless v2 di un cluster in più zone di disponibilità (AZ). La configurazione di un cluster Multi-AZ contribuisce a garantire la continuità aziendale anche nel raro caso che si verifichino problemi che coinvolgono l'intera AZ. Questa funzionalità non è disponibile con Aurora Serverless v1.
  - Database globali: utilizza Aurora Serverless v2 in combinazione con i database globali Aurora per creare copie aggiuntive del cluster di sola lettura in altre Regioni AWS da utilizzare per il ripristino di emergenza.
  - RDS Proxy: puoi utilizzare Amazon RDS Proxy per consentire alle applicazioni di eseguire il pool e condividere connessioni di database per migliorare la loro capacità di dimensionamento.
- Dimensionamento più veloce, più granulare e meno dirompente rispetto a Aurora Serverless v1: Aurora Serverless v2 può scalare verso l'alto e verso il basso più velocemente. Permette di effettuare modifiche di capacità di 0,5 ACU, anziché raddoppiare o dimezzare il numero di ACU. Il dimensionamento avviene in genere senza alcuna interruzione dell'elaborazione. Contrariamente a quanto avviene con Aurora Serverless v1, il dimensionamento non è un evento del quale devi essere consapevole. Esso può avvenire mentre le istruzioni SQL sono in esecuzione e le transazioni sono aperte, senza dover attendere un punto silenzioso.

Funzionamento di Aurora Serverless v2

La seguente panoramica descrive il funzionamento di Aurora Serverless v2.

Argomenti

- Panoramica di Aurora Serverless v2

- Configurazioni per i cluster di database Aurora
- Capacità di Aurora Serverless v2
- Dimensionamento di Aurora Serverless v2
- Aurora Serverless v2 ed elevata disponibilità
- Aurora Serverless v2 e spazio di archiviazione
- Parametri di configurazione per i cluster Aurora

Panoramica di Aurora Serverless v2

Amazon Aurora Serverless v2 è adatto per i carichi di lavoro più impegnativi e altamente variabili. Ad esempio, l'uso del database potrebbe essere pesante per un breve periodo di tempo, seguito da lunghi periodi di attività leggera o nessuna attività. Alcuni esempi sono siti web per la vendita al dettaglio, i giochi o sportivi con eventi promozionali periodici e database in grado di generare report quando necessario. Altri sono ambienti di sviluppo e test e nuove applicazioni in cui l'utilizzo potrebbe aumentare rapidamente. In casi come questi e molti altri, non sempre è possibile configurare correttamente la capacità in anticipo con il modello fornito. Ciò può anche comportare costi più elevati se si esegue il provisioning eccessivo e si dispone di capacità che poi non viene utilizzata.

Al contrario, i cluster di Aurora con provisioning sono adatti per carichi di lavoro stabili. Con i cluster con provisioning, puoi scegliere una classe di istanza DB con una quantità predefinita di memoria, potenza della CPU, larghezza di banda I/O e così via. Se il carico di lavoro cambia, modifichi manualmente la classe di istanza dello scrittore e dei lettori. Il modello soggetto a provisioning funziona bene quando è possibile regolare la capacità in anticipo rispetto ai modelli di consumo previsti ed è accettabile soffrire di brevi interruzioni mentre si modifica la classe di istanza dello scrittore e dei lettori all'interno del cluster.

Aurora Serverless v2 è progettato da zero per supportare cluster DB serverless che risultino istantaneamente scalabili. Aurora Serverless v2 è progettato per garantire lo stesso grado di sicurezza e isolamento degli scrittori e dei lettori con provisioning. Questi aspetti sono cruciali negli ambienti cloud serverless multitenant. Il meccanismo di dimensionamento dinamico impone un overhead molto ridotto in modo da poter rispondere rapidamente alle modifiche del carico di lavoro del database. È anche abbastanza potente da soddisfare i considerevoli aumenti della domanda di elaborazione.

Utilizzando Aurora Serverless v2 puoi creare un cluster Aurora DB senza essere legato a una capacità di database specifica per ogni scrittore e lettore. Puoi specificare l'intervallo minimo e massimo per la capacità. Aurora bilancia ciascuno scrittore o lettore Aurora Serverless v2 nel cluster

all'interno di tale intervallo di capacità. Utilizzando un cluster Multi-AZ in cui ogni scrittore o lettore può dimensionarsi dinamicamente puoi sfruttare la scalabilità dinamica e l'elevata disponibilità.

Aurora Serverless v2 dimensiona le risorse del database automaticamente in base alle specifiche di capacità minima e massima. La scalabilità è rapida perché la maggior parte delle operazioni legate agli eventi di dimensionamento mantiene lo scrittore o il lettore sullo stesso host. Nei rari casi in cui uno scrittore o un lettore Aurora Serverless v2 debbano essere spostati da un host all'altro, Aurora Serverless v2 gestisce automaticamente le connessioni. Non è necessario modificare il codice dell'applicazione client del database o le stringhe di connessione al database.

Con Aurora Serverless v2, come per i cluster con provisioning, la capacità di archiviazione e la capacità di calcolo sono indipendenti. Quando ci riferiamo a capacità e scalabilità di Aurora Serverless v2, facciamo sempre riferimento alla capacità di calcolo che aumenta o diminuisce. Pertanto, il cluster può contenere molti terabyte di dati anche quando la CPU e la capacità di memoria si dimensionano verso il basso.

Anziché effettuare il provisioning e gestire i server di database, puoi indicare la capacità del database. Per informazioni dettagliate sulle capacità di Aurora Serverless v2, consulta Capacità di Aurora Serverless v2. La capacità effettiva di ciascuno scrittore o lettore Aurora Serverless v2 varia nel tempo, a seconda del carico di lavoro. Per i dettagli su questi meccanismi, consulta Dimensionamento di Aurora Serverless v2.

> **Important**
>
> Con Aurora Serverless v1, il cluster dispone di un'unica misura della capacità di calcolo in grado di dimensionarsi tra i valori di capacità minima e massima. Con Aurora Serverless v2, il cluster può contenere dei lettori oltre allo scrittore. Ogni scrittore e lettore di Aurora Serverless v2 può dimensionarsi tra i valori di capacità minima e massima. Pertanto, la capacità totale del cluster Aurora Serverless v2 dipende sia dall'intervallo di capacità definito per il cluster di database, sia dal numero di scrittori e lettori contenuti nel cluster. In qualunque momento specifico, paghi soltanto i costi della capacità di Aurora Serverless v2 che viene effettivamente utilizzata nel cluster di database Aurora.

Configurazioni per i cluster di database Aurora

Per ciascuno dei cluster di Aurora DB è possibile scegliere qualsiasi combinazione di capacità, capacità con provisioning o entrambi di Aurora Serverless v2.

Puoi impostare un cluster che contiene entrambi sia Aurora Serverless v2 che capacità con provisioning, indicato con la denominazione cluster a configurazione mista. Ad esempio, supponi di aver bisogno di più capacità di lettura/scrittura rispetto a quella disponibile per uno scrittore Aurora Serverless v2. In questo caso puoi configurare il cluster con uno scrittore con provisioning di dimensioni molto ampie. In tal caso, puoi ancora utilizzare Aurora Serverless v2 per i lettori. Oppure supponi che il carico di lavoro in scrittura per il cluster vari ma che il carico di lavoro in lettura sia costante. In questo caso, puoi configurare il cluster con uno scrittore Aurora Serverless v2 e uno o più lettori con provisioning.

Puoi anche impostare un cluster di database in tutta la capacità è gestita da Aurora Serverless v2. Per fare ciò, puoi creare un nuovo cluster e utilizzare Aurora Serverless v2 fin dall'inizio. In alternativa, puoi sostituire tutta la capacità soggetta a provisioning in un cluster esistente con Aurora Serverless v2. Ad esempio, alcuni percorsi di aggiornamento delle versioni precedenti del motore richiedono di iniziare con uno scrittore con provisioning e sostituirlo con uno scrittore Aurora Serverless v2. Per le procedure per creare un nuovo cluster di database con Aurora Serverless v2 o per passare da un cluster di database esistente a Aurora Serverless v2, consulta Creazione di un cluster di database Aurora Serverless v2 e Passaggio di un cluster con provisioning ad Aurora Serverless v2.

Se non utilizzi per nulla Aurora Serverless v2 in un cluster di database, tutti gli scrittori e i lettori del cluster di database sono con provisioning. Questo è il tipo di cluster di database più vecchio e più comune con cui la maggior parte degli utenti ha familiarità. Infatti, prima dell'arrivo di Aurora Serverless, non c'era un nome speciale per questo tipo di cluster di Aurora DB. La capacità fornita è costante. Le tariffe sono relativamente semplici da prevedere. Tuttavia, è necessario prevedere in anticipo quanta capacità è necessaria. In alcuni casi le previsioni potrebbero essere imprecise o le esigenze di capacità potrebbero cambiare. In questi casi, il cluster di database può rivelarsi soggetto a provisioning in modo insufficiente (più lento di quello desiderato) o provisioning in modo eccessivo (più costoso di quello desiderato).

Capacità di Aurora Serverless v2

L'unità di misura per Aurora Serverless v2 è l'Unità di capacità Aurora (ACU). La capacità di Aurora Serverless v2 non è legata alle classi dell'istanza DB utilizzate per i cluster sottoposti a provisioning.

Ogni ACU è una combinazione di circa 2 gigabyte (GiB) di memoria, CPU corrispondente e rete. Puoi specificare l'intervallo di capacità del database utilizzando questa unità di misura. I parametri `ServerlessDatabaseCapacity` e `ACUUtilization` consentono di determinare la capacità effettivamente utilizzata dal database e se tale capacità rientra nell'intervallo specificato.

In qualsiasi momento, a ogni scrittore o lettore del database Aurora Serverless v2 è associata una capacità. La capacità è rappresentata da un numero a virgola mobile che rappresenta le ACU. La capacità aumenta o diminuisce ogni volta che lo scrittore o il lettore si dimensionano. Questo valore viene misurato ogni secondo. Per ogni cluster di database in cui intendi utilizzare Aurora Serverless v2, definisci un intervallo di capacità: i valori minimo e massimo dell'intervallo di capacità all'interno del quale ogni scrittore o lettore di Aurora Serverless v2 può dimensionarsi. L'intervallo di capacità è lo stesso per ciascuno scrittore o lettore di Aurora Serverless v2 in un cluster di database. Ogni scrittore o lettore di Aurora Serverless v2 presenta una propria capacità che ricade in qualche modo in tale intervallo.

La tabella seguente mostra gli intervalli di capacità Aurora Serverless v2 e il supporto delle versioni del motore per Aurora MySQL e Aurora PostgreSQL.

Intervallo di capacità (ACU)	Versioni supportate di Aurora MySQL	Versioni supportate di Aurora PostgreSQL
0,5–128	3.02.0 e versioni successive	13.6 e versioni successive, 14.3 e versioni successive, 15.2 e versioni successive, 16.1 e versioni successive
0,5–256	3.06.0 e versioni successive	13.13 e versioni successive, 14.10 e versioni successive, 15.5 e versioni successive, 16.1 e versioni successive
0–256	3.08.0 e versioni successive	13.15 e versioni successive, 14.12 e versioni successive, 15.7 e versioni successive, 16.3 e versioni successive

La tabella seguente mostra le versioni della piattaforma Aurora Serverless v2 con i relativi intervalli ACU e le caratteristiche delle prestazioni.

Versione della piattaforma Aurora Serverless v2	Intervallo ACU	Prestazioni
1	0–128	Prestazioni di base
2	0–256	Prestazioni di base
3	0–256	Prestazioni migliorate fino al 30% rispetto alla versione della piattaforma 2

> **Note**
>
> L'intervallo di dimensionamento disponibile per uno specifico cluster viene determinato sia in base alla versione del motore che alla versione della piattaforma. È possibile eseguire una versione del motore con prestazioni più elevate su una versione della piattaforma con prestazioni più basse e viceversa. L'intervallo di dimensionamento viene determinato in base alla versione del motore o della piattaforma con le prestazioni più basse.

È possibile determinare su quale versione della piattaforma è in esecuzione il cluster nella sezione Configurazione dell'istanza di Console di gestione AWS o tramite l'API, visualizzando `ServerlessV2PlatformVersion` per un DBCluster.

La capacità minima Aurora Serverless v2 che è possibile definire è 0 ACU, per le versioni Aurora Serverless v2 che supportano la funzionalità di pausa automatica. Puoi specificare un numero superiore se inferiore o uguale al valore di capacità massima. L'impostazione della capacità minima su un numero ridotto consente ai cluster DB con carichi leggeri di consumare risorse di calcolo minime. Allo stesso tempo, rimangono pronti ad accettare immediatamente le connessioni e a dimensionarsi quando diventano impegnati.

Si consiglia di impostare il minimo su un valore che consenta a ciascuno scrittore o lettore del database DB di mantenere il set di lavoro dell'applicazione nel buffer pool. In questo modo, il contenuto del buffer pool non viene scartato durante i periodi di inattività. Per tutte le considerazioni nella scelta del valore massimo della capacità, consulta Scelta dell'impostazione Aurora Serverless v2 minima della capacità per un cluster. Per tutte le considerazioni nella scelta del valore massimo

della capacità, consulta Scelta dell'impostazione Aurora Serverless v2 massima della capacità per un cluster.

A seconda di come si configurano le istanze di lettura in un'implementazione Multi-AZ, le loro capacità possono essere collegate alla capacità dell'istanza di scrittura o rimanere indipendenti. Per i dettagli su come eseguire queste operazioni, consulta Dimensionamento di Aurora Serverless v2.

Il monitoraggio di Aurora Serverless v2 prevede la misurazione dei valori di capacità per lo scrittore e i lettori all'interno del cluster di database nel tempo. Se il database non viene si dimensiona verso il basso fino alla capacità minima, puoi intraprendere azioni come la regolazione del minimo e l'ottimizzazione dell'applicazione database. Se il database raggiunge costantemente la sua capacità massima, puoi intraprendere operazioni come l'aumento di tale vincolo. Puoi inoltre ottimizzare l'applicazione di database e distribuire il carico di query su più lettori.

Gli addebito per la capacità di Aurora Serverless v2 sono misurati in termini di ore ACU. Per informazioni su come sono calcolati gli addebiti di Aurora Serverless v2, consulta la Pagina dei prezzi di Aurora.

Supponiamo che il numero totale di istanze di scrittura e di lettura nel cluster sia *n*. In tal caso, il cluster consuma approssimativamente *n* x *minimum ACUs* quando non si esegue alcuna operazione di database. Aurora stessa potrebbe eseguire operazioni di monitoraggio o manutenzione che causano una piccola quantità di carico. Nel momento in cui il database è in esecuzione a piena capacità, quel cluster non consuma più di *n* x *maximum ACUs*.

Per ulteriori informazioni sulla scelta dei valori minimi e massimi appropriati di ACU, consulta Scelta dell'intervallo di capacità di Aurora Serverless v2 per un cluster Aurora. I valori ACU minimo e massimo specificati influiscono anche sul modo in cui funzionano alcuni parametri di configurazione di Aurora per Aurora Serverless v2. Per informazioni dettagliate sull'interazione tra l'intervallo di capacità e i parametri di configurazione, consulta Uso di gruppi di parametri per Aurora Serverless v2.

Dimensionamento di Aurora Serverless v2

Per ogni scrittore o lettore di Aurora Serverless v2, Aurora tiene costantemente traccia dell'utilizzo di risorse come CPU, memoria e rete. Queste misurazioni sono chiamate collettivamente carico. Il carico include le operazioni di database eseguite dall'applicazione. Include anche l'elaborazione in background per il server di database e le attività amministrative di Aurora. Quando la capacità è vincolata da una di queste opzioni, Aurora Serverless v2 esegue un dimensionamento verso l'alto. Aurora Serverless v2 esegue un dimensionamento verso l'alto anche quando rileva problemi di prestazioni che possono essere risolti in questo modo. Puoi monitorare l'utilizzo delle risorse e come

questo influisce sul dimensionamento di Aurora Serverless v2 utilizzando le procedure in Parametri di Amazon CloudWatch importanti per Aurora Serverless v2 e Monitoraggio delle prestazioni di Aurora Serverless v2 con Approfondimenti sulle prestazioni.

Il carico può variare tra lo scrittore e i lettori del cluster di database. Lo scrittore gestisce tutte le istruzioni DDL (Data Definition Language), come `CREATE TABLE`,`ALTER TABLE` e `DROP TABLE`. Lo scrittore gestisce anche tutte le istruzioni DML (Data Manipulation Language), come ad esempio `INSERT` e `UPDATE`. I lettori possono elaborare istruzioni di sola lettura, come ad esempio le query `SELECT`.

Il dimensionamento è l'operazione che incrementa o riduce le capacità di Aurora Serverless v2 per il tuo database. Con Aurora Serverless v2, ogni scrittore e lettore ha il proprio valore di capacità attuale, misurato in ACU. Aurora Serverless v2 dimensiona uno scrittore o un lettore a una capacità superiore quando la sua capacità attuale è troppo bassa per gestire il carico. Ridimensiona lo scrittore o il lettore a una capacità inferiore quando la sua capacità corrente è superiore a quella necessaria.

A differenza di Aurora Serverless v1, che si dimensiona raddoppiando la capacità ogni volta che il cluster di database raggiunge una specifica soglia, Aurora Serverless v2 può aumentare la capacità in modo incrementale. Quando la domanda del carico di lavoro inizia a raggiungere la capacità del database corrente di uno scrittore o un lettore, Aurora Serverless v2 incrementa il numero di ACU per lo scrittore o il lettore. Aurora Serverless v2 dimensiona la capacità secondo gli incrementi necessari per fornire le migliori prestazioni per le risorse consumate. Il dimensionamento avviene con incrementi ridotti fino a 0,5 ACU. Maggiore è la capacità attuale, maggiore è l'incremento nel dimensionamento e quindi più velocemente può essere garantito il dimensionamento.

Poiché dimensionamento di Aurora Serverless v2 è così frequente, granulare e senza interruzioni, non genera eventi discreti nella Console di gestione AWS come accade invece per Aurora Serverless v1. Puoi invece misurare le metriche di Amazon CloudWatch come `ServerlessDatabaseCapacity` e `ACUUtilization` e tenere traccia dei valori minimo, massimo e medio nel tempo. Per maggiori informazioni sui parametri di Aurora, consulta Monitoraggio dei parametri in un cluster di database Amazon Aurora. Per suggerimenti sul monitoraggio di Aurora Serverless v2, consulta Parametri di Amazon CloudWatch importanti per Aurora Serverless v2.

Puoi scegliere di creare un dimensionamento del lettore in modo contemporaneo allo scrittore associato o indipendentemente da questo. Tale obiettivo si raggiunge specificando il livello di promozione per quel lettore.

- I lettori associati ai livelli di promozione 0 e 1 si dimensionano contemporaneamente allo scrittore. Questo comportamento di dimensionamento rende i lettori con livelli prioritari 0 e 1 ideali per la disponibilità. Questo perché sono sempre dimensionati alla capacità giusta per assumere il carico di lavoro dallo scrittore in caso di failover.
- I lettori nei livelli di promozione da 2 a 15 si dimensionano indipendentemente dallo scrittore. Ogni lettore si mantiene all'interno dei valori ACU minimo e massimo specificati per il cluster. Quando un lettore si dimensiona indipendentemente dallo scrittore del database associato, può diventare inattivo e dimensionarsi verso il basso mentre lo scrittore continua a elaborare un volume elevato di transazioni. Se nessun altro lettore è disponibile in livelli di promozione inferiori rimane ancora disponibile come target di failover. Tuttavia, se viene promosso al ruolo di scrittore, potrebbe essere necessario un dimensionamento verso l'alto per gestire l'intero carico di lavoro dello scrittore.

Per i dettagli sui livelli di promozione, consulta Scelta del livello di promozione per un'istanza Aurora Serverless v2 di lettura.

Le nozioni di punti di dimensionamento e periodi di timeout associati presenti in Aurora Serverless v1 non si applicano a Aurora Serverless v2. Il dimensionamento di Aurora Serverless v2 può avvenire mentre le connessioni al database sono aperte, mentre le transazioni SQL sono in corso, mentre le tabelle sono bloccate e mentre le tabelle temporanee sono in uso. Aurora Serverless v2 non attende un momento di scarso utilizzo per avviare il dimensionamento. Il dimensionamento non interrompe le operazioni del database in corso.

Se il carico di lavoro richiede una capacità di lettura superiore a quella disponibile con un singolo scrittore e un singolo lettore, è possibile aggiungere più Aurora Serverless v2 lettori al cluster. Ogni lettore Aurora Serverless v2 può dimensionarsi all'interno dei valori di capacità minimo e massimo specificati per il cluster di database. Puoi utilizzare l'endpoint di lettura del cluster per gestire le sessioni di sola lettura attraverso i lettori e ridurre il carico sullo scrittore.

L'esecuzione del dimensionamento da parte di Aurora Serverless v2 e la velocità di dimensionamento dopo l'avvio dipendono anche dalle impostazioni ACU minima e massima per il cluster. Inoltre, dipendono dal fatto che un lettore sia configurato per dimensionarsi contemporaneamente allo scrittore o indipendentemente da esso. Per i dettagli sui fattori che influenzano il dimensionamento di Aurora Serverless v2, consulta Prestazioni e dimensionamento per Aurora Serverless v2.

Dimensionamento a Zero

Nelle versioni recenti di Aurora MySQL e Aurora PostgreSQL, le istanze di lettura e scrittura Aurora Serverless v2 possono scalare fino a zero ACU. Questa capacità viene indicata come pausa e ripresa automatiche o pausa automatica. È possibile scegliere se consentire questo comportamento specificando un valore zero o diverso da zero per la capacità minima. È anche possibile scegliere quanto tempo aspettare prima che un'istanza Aurora Serverless v2 entri in pausa. Per informazioni sulle versioni dotate di questa capacità, consulta Capacità di Aurora Serverless v2. Per informazioni su come utilizzare questa capacità in modo efficace, consulta Dimensionamento a zero ACU con pausa e ripresa automatiche per Aurora Serverless v2.

Nelle versioni precedenti di Aurora MySQL e Aurora PostgreSQL, le istanze di scrittura e lettura Aurora Serverless v2 inattive possono essere ridotte verticalmente fino al valore ACU minimo specificato per il cluster, ma non fino a zero ACU. In questo caso, zero ACU non è un'opzione disponibile per l'impostazione dell'intervallo di capacità. Questo comportamento è diverso da quello di Aurora Serverless v1, che può sospendere i processi dopo un periodo di inattività ma richiede un po' di tempo per la ripartenza quando si apre una nuova connessione.

Quando il cluster di database con capacità Aurora Serverless v2 non è necessario per un certo periodo di tempo, è possibile arrestare e avviare l'intero cluster in modo simile a come accade con i cluster di database con provisioning. Questa tecnica è particolarmente adatta per i sistemi di sviluppo e test, che potrebbero non essere necessari per molte ore consecutive e per i quali la velocità di ripresa del cluster non è essenziale. La funzionalità di arresto/avvio del cluster è disponibile per tutte le versioni Aurora Serverless v2. Per ulteriori informazioni su questa funzionalità, consulta Avvio e arresto di un cluster di database Amazon Aurora.

Aurora Serverless v2 ed elevata disponibilità

Il modo per garantire un'elevata disponibilità di un cluster Aurora DB consiste nel renderlo un cluster di database Multi-AZ. Un cluster di database Aurora Multi-AZ garantisce capacità di calcolo disponibile in ogni momento in più di una zona di disponibilità (AZ). Tale configurazione mantiene il database attivo e funzionante anche in caso di rilevanti malfunzionamenti. Aurora esegue un failover automatico in caso di problemi che riguardano lo scrittore o persino l'intera AZ. Con Aurora Serverless v2 puoi scegliere la capacità di calcolo in stand-by da dimensionare verso l'alto e verso il basso insieme alla capacità dello scrittore. In questo modo, la capacità di calcolo nella seconda AZ è pronta a rilevare il carico di lavoro corrente in qualsiasi momento. Allo stesso tempo, la capacità di calcolo in tutte le AZ può dimensionarsi verso il basso quando il database è inattivo. Per i dettagli su

come funziona Aurora con le Regioni AWS e sulle zone di disponibilità, consulta Architetture ad alta disponibilità per istanze database Aurora.

La funzionalità Multi-AZ di Aurora Serverless v2 utilizza dei lettori in aggiunta allo scrittore. Il supporto per i lettori è nuovo in Aurora Serverless v2 rispetto a quanto offerto da Aurora Serverless v1. In un cluster di database Aurora puoi aggiungere fino a 15 lettori Aurora Serverless v2 distribuiti su 3 AZ.

Per le applicazioni business-critical che devono rimanere disponibili anche in caso di problemi che riguardano l'intero cluster o l'intera regione AWS, puoi impostare un database globale di Aurora. Puoi utilizzare le funzionalità di Aurora Serverless v2 nei cluster secondari in modo che siano pronti a subentrare durante un eventuale ripristino di emergenza. Possono anche dimensionarsi verso il basso quando il database non è impegnato. Per informazioni dettagliate sui database globali di Aurora, consulta Utilizzo di Database globale Amazon Aurora.

Aurora Serverless v2 funziona come un database con provisioning per quanto riguarda il failover e altre funzionalità ad alta disponibilità. Per ulteriori informazioni, consulta Elevata disponibilità di Amazon Aurora.

Supponiamo tu voglia garantire la massima disponibilità per il tuo cluster Aurora Serverless v2. Puoi un lettore in aggiunta allo scrittore. Se assegni il lettore al livello di promozione 0 o 1, qualsiasi dimensionamento avvenga per lo scrittore esso viene replicato anche per il lettore. In questo modo, un lettore con capacità identica è sempre pronto a prendere il posto dello scrittore in caso di failover.

Supponi di voler eseguire report trimestrali per la tua azienda nello stesso momento in cui il cluster continua a elaborare le transazioni. Se aggiungi un lettore Aurora Serverless v2 al cluster e lo assegni a un livello di promozione da 2 a 15, puoi connetterti direttamente a quel lettore per eseguire i report. A seconda di quanto le query di reporting sono esigenti in termini di memoria e di utilizzo della CPU, il lettore può dimensionarsi verso l'alto per adattarsi al carico di lavoro. Può successivamente dimensionarsi di nuovo verso il basso una volta terminata la generazione dei report.

Aurora Serverless v2 e spazio di archiviazione

Lo spazio di archiviazione per ciascun cluster di database Aurora è costituito da sei copie di tutti i dati, distribuite su tre AZ. Questa replica dei dati integrata si applica indipendentemente dal fatto che il cluster di database includa dei lettori in aggiunta allo scrittore. In questo modo i dati sono al sicuro anche da problemi che influiscono sulla capacità di calcolo del cluster.

Lo spazio di archiviazione di Aurora Serverless v2 ha le stesse caratteristiche di affidabilità e durata descritte in Archiviazione Amazon Aurora. Questo perché lo spazio di archiviazione per i cluster

DB Aurora funziona allo stesso modo, indipendentemente dal fatto che la capacità di calcolo utilizzi Aurora Serverless v2 o un database con provisioning.

Parametri di configurazione per i cluster Aurora

Puoi regolare tutti gli stessi parametri di configurazione del cluster e del database per i cluster con funzionalità Aurora Serverless v2 così come faresti per i cluster DB con provisioning. Tuttavia, alcuni parametri relativi alla capacità sono gestiti in modo diverso da Aurora Serverless v2. In un cluster a configurazione mista, i valori specificati per i parametri relativi alla capacità si applicano a tutti gli scrittori e i lettori con provisioning.

Quasi tutti i parametri funzionano allo stesso modo per gli scrittori e i lettori Aurora Serverless v2 in modo analogo a quanto accade a quelli con provisioning. Le eccezioni riguardano alcuni parametri che Aurora regola automaticamente durante il dimensionamento e alcuni parametri che Aurora mantiene a valori fissi dipendenti dall'impostazione della capacità massima.

Ad esempio, la quantità di memoria riservata alla cache del buffer aumenta man mano che uno scrittore o un lettore si dimensionano verso l'alto e diminuisce man mano che si dimensionano verso il basso. In questo modo, la memoria può essere rilasciata quando il database non è impegnato. Al contrario, Aurora imposta automaticamente il numero massimo di connessioni su un valore appropriato in base all'impostazione della capacità massima. In questo modo, le connessioni attive non vengono interrotte se il carico diminuisce e Aurora Serverless v2 si ridimensiona verso il basso. Per informazioni sui come Aurora Serverless v2 gestisce parametri specifici, consulta Uso di gruppi di parametri per Aurora Serverless v2.

Requisiti e limitazioni per Aurora Serverless v2

Quando crei un cluster da utilizzare con le istanze database Aurora Serverless v2, presta attenzione ai requisiti e alle limitazioni seguenti:

Argomenti

- Disponibilità di regioni e versioni
- I cluster che utilizzano Aurora Serverless v2 devono avere un intervallo di capacità specificato
- Aurora Serverless v2 non suppporta alcune caratteristiche di provisioning
- Aurora Serverless v2 presenta alcuni aspetti diversi rispetto a Aurora Serverless v1

Disponibilità di regioni e versioni

La disponibilità e il supporto della funzionalità varia tra le versioni specifiche di ciascun motore di database Aurora e tra Regioni AWS. Per ulteriori informazioni sulla disponibilità di versioni e regioni con Aurora e Aurora Serverless v2, consultare Regioni e motori di database Aurora supportati per Aurora Serverless v2.

L'esempio seguente illustra i comandi AWS CLI per confermare i valori esatti del motore database con cui è possibile utilizzare Aurora Serverless v2 in una Regione AWS specifica. Il parametro `--db-instance-class` per Aurora Serverless v2 è sempre `db.serverless`. Il parametro `--engine` può essere `aurora-mysql` o `aurora-postgresql`. Sostituisci i valori `--region` e `--engine` appropriati per verificare i valori `--engine-version` che puoi utilizzare. Se il comando non produce output, Aurora Serverless v2 non è disponibile per tale combinazione di Regione AWS e motore database.

```
aws rds describe-orderable-db-instance-options --engine aurora-mysql --db-instance-
class db.serverless \
  --region my_region --query 'OrderableDBInstanceOptions[].[EngineVersion]' --output
 text

aws rds describe-orderable-db-instance-options --engine aurora-postgresql --db-
instance-class db.serverless \
  --region my_region --query 'OrderableDBInstanceOptions[].[EngineVersion]' --output
 text
```

I cluster che utilizzano Aurora Serverless v2 devono avere un intervallo di capacità specificato

Un cluster Aurora deve disporre di un attributo `ServerlessV2ScalingConfiguration` prima di poter aggiungere eventuali istanze database che utilizzano la classe dell'istanza database `db.serverless`. Questo attributo specifica l'intervallo di capacità. La capacità di Aurora Serverless v2 varia da un minimo di 0 unità di capacità Aurora (ACU) a un massimo di 256 ACU, con incrementi di 0,5 ACU. Il valore minimo consentito dipende dalla versione di Aurora. Ogni ACU fornisce l'equivalente di circa 2 gibibyte (GiB) di RAM, CPU corrispondente e rete. Per informazioni dettagliate sul modo in cui Aurora Serverless v2 utilizza le impostazioni dell'intervallo di capacità, consulta Funzionamento di Aurora Serverless v2.

Per gli intervalli di capacità consentiti per le varie versioni del motore di database e le varie versioni della piattaforma, consulta Capacità di Aurora Serverless v2. L'intervallo di dimensionamento

disponibile per uno specifico cluster è influenzato sia dalla versione del motore che dall'hardware (versione della piattaforma).

Puoi specificare i valori ACU minimi e massimi nella Console di gestione AWS quando crei un cluster e l'istanza database Aurora Serverless v2 associata. È anche possibile specificare l'opzione `--serverless-v2-scaling-configuration` nella AWS CLI. In alternativa, puoi specificare il parametro `ServerlessV2ScalingConfiguration` con l'API Amazon RDS. Puoi specificare questo attributo quando crei un cluster o modifichi un cluster esistente. Per le procedure di impostazione dell'intervallo di capacità, consulta Impostazione dell'intervallo di capacità di Aurora Serverless v2 per un cluster. Per una discussione dettagliata su come scegliere i valori di capacità minima e massima e su come tali impostazioni influiscono su alcuni parametri del database, vedi Scelta dell'intervallo di capacità di Aurora Serverless v2 per un cluster Aurora.

Aurora Serverless v2 non suppporta alcune caratteristiche di provisioning

Al momento, le seguenti caratteristiche delle istanze database con provisioning Aurora non sono disponibili per Amazon Aurora Serverless v2:

- Flussi di attività di database (DAS)
- Gestione della cache del cluster in Aurora PostgreSQL. Il parametro di configurazione `apg_ccm_enabled` non si applica alle istanze database Aurora Serverless v2.

Alcune caratteristiche di Aurora, pur funzionando con Aurora Serverless v2, potrebbero causare problemi se l'intervallo di capacità è inferiore a quello richiesto affinché la memoria supporti tali funzionalità con il tuo carico di lavoro specifico. In tal caso, il database potrebbe non funzionare come al solito o potrebbero verificarsi errori dovuti all'esaurimento della memoria. Per consigli sull'impostazione dell'intervallo di capacità appropriato, consulta Scelta dell'intervallo di capacità di Aurora Serverless v2 per un cluster Aurora. Per informazioni sulla risoluzione dei problemi quando il database rileva errori di memoria insufficiente causati da un intervallo di capacità configurato in modo errato, consulta Esclusione di errori dovuti all'esaurimento della memoria.

Il dimensionamento automatico in Aurora non è supportato. Questo tipo di dimensionamento aggiunge nuove istanze di lettura per gestire ulteriori carichi di lavoro ad uso intensivo di lettura in base all'utilizzo della CPU. Tuttavia, il dimensionamento basato sull'utilizzo della CPU non è significativo per Aurora Serverless v2. In alternativa, puoi creare anticipatamente le istanze database di lettura Aurora Serverless v2 e lasciarle in condizione di bassa capacità. È un modo più rapido e meno dirompente per scalare la capacità di lettura di un cluster rispetto all'aggiunta dinamica di nuove istanze database.

Aurora Serverless v2 presenta alcuni aspetti diversi rispetto a Aurora Serverless v1

Se sei utente di Aurora Serverless v1 e questa è la prima volta che utilizzi Aurora Serverless v2, consulta le differenze di requisiti tra Aurora Serverless v2 e Aurora Serverless v1 per avere un'idea più chiara dei diversi requisiti richiesti da Aurora Serverless v1 e Aurora Serverless v2.

Creazione di un cluster di database che utilizza Aurora Serverless v2

Per creare un cluster Aurora in cui è possibile aggiungere istanze database Aurora Serverless v2, è necessario seguire la stessa procedura valida per Creazione di un cluster database Amazon Aurora. In Aurora Serverless v2, i cluster sono intercambiabili con i cluster con provisioning. Possono infatti essere presenti cluster in cui alcune istanze database usano Aurora Serverless v2 e altre sono istanze con provisioning.

Argomenti

- Impostazioni per cluster di database Aurora Serverless v2
- Creazione di un cluster di database Aurora Serverless v2
- Creazione di un'istanza database di scrittura Aurora Serverless v2

Impostazioni per cluster di database Aurora Serverless v2

Verificare che le impostazioni iniziali del cluster soddisfino i requisiti elencati in Requisiti e limitazioni per Aurora Serverless v2. Specifica le impostazioni seguenti per verificare se è possibile aggiungere istanze database Aurora Serverless v2 al cluster:

Regione AWS

Crea il cluster in una Regione AWS dove sono disponibili istanze DB Aurora Serverless v2. Per i dettagli sulle regioni disponibili, consulta Regioni e motori di database Aurora supportati per Aurora Serverless v2.

DB engine version (Versione motore del database

Scegli una versione del motore compatibile con Aurora Serverless v2. Per informazioni sui requisiti di versione per Aurora Serverless v2, consulta Requisiti e limitazioni per Aurora Serverless v2.

DB instance class (Classe istanza database)

Se crei un cluster tramite la Console di gestione AWS, puoi contemporaneamente scegliere anche la classe di istanza database per l'istanza DB di scrittura. Scegli Serverless come classe di istanza DB. Quando sceglie questa classe di istanza DB, devi specificare anche l'intervallo di capacità per l'istanza DB di scrittura. Lo stesso intervallo di capacità si applica a tutti le altre istanze database Aurora Serverless v2 aggiunte al cluster.

Se l'opzione Serverless per la classe di istanza database non è disponibile, verificare di aver scelto una versione del motore database compatibile con Regioni e motori di database Aurora supportati per Aurora Serverless v2.

Quando si utilizza AWS CLI o l'API Amazon RDS, il parametro specificato per la classe di istanza database è `db.serverless`.

Intervallo di capacità

Inserisci i valori ACU (Aurora Capacity Unit) minimi e massimi da applicare a tutte le istanze DB nel cluster. Questa opzione è disponibile sia nella pagina Crea cluster che nella pagina Aggiungi lettore della console quando scegli Serverless come classe dell'istanza database.

Per gli intervalli di capacità per le varie versioni del motore database, consulta Capacità di Aurora Serverless v2.

Se i campi relativi ai valori ACU minimi e massimi non sono visualizzati, verificare di aver scelto la classe di istanza database Serverless per l'istanza database di scrittura.

Se inizialmente crei il cluster con un'istanza databse con provisioning, non devi specificare i valori ACU minimi e massimi. In tal caso è possibile modificare il cluster in seguito per aggiungere tale impostazione. Al cluster puoi anche aggiungere un'istanza DB Aurora Serverless v2 di lettura. Puoi specificare l'intervallo di capacità come parte del processo.

Puoi aggiungere altre istanze DB Aurora Serverless v2 al cluster mediante AWS CLI o l'API RDS solo dopo aver specificato l'intervallo di capacità per il cluster. Se tenti di aggiungere un'istanza DB Aurora Serverless v2, verrà restituito un errore. Nelle procedure per AWS CLI per l'API RDS, l'intervallo di capacità è rappresentato dall'attributo `ServerlessV2ScalingConfiguration`.

Per i cluster contenenti più di un'istanza DB di lettura, la priorità di failover di ciascuna istanza DB Aurora Serverless v2 di lettura ha un ruolo importante nel modo in cui il dimensionamento verso

l'alto e la riduzione vengono applicati all'istanza DB. Non è possibile specificare la priorità nella fase iniziale di creazione del cluster. Tieni presente questa proprietà quando al cluster aggiungi una seconda istanza DB di lettura o altre istanze di questo tipo. Per ulteriori informazioni, consulta Scelta del livello di promozione per un'istanza Aurora Serverless v2 di lettura.

Creazione di un cluster di database Aurora Serverless v2

È possibile utilizzare la Console di gestione AWS, l'AWS CLI o l'API RDS per creare un cluster di database Aurora Serverless v2.

Console

Per creare un cluster con un'istanza Aurora Serverless v2 di scrittura

1. Accedi alla Console di gestione AWS e apri la console Amazon RDS all'indirizzo https://console.aws.amazon.com/rds/.
2. Nel riquadro di navigazione, scegliere Databases (Database).
3. Scegliere Create database (Crea database). Nella pagina visualizzata, scegliere le opzioni seguenti:
   - Per Tipo di motore, scegli Aurora (compatibile con MySQL) o Aurora (compatibile con PostgreSQL).
   - Per Versione, scegliere una delle versioni supportate per Regioni e motori di database Aurora supportati per Aurora Serverless v2.
4. Per Classe di istanza database scegliere Serverless v2.
5. In Intervallo di capacità è possibile accettare l'intervallo predefinito. In alternativa, puoi scegliere altri valori per le unità di capacità minima e massima. È possibile scegliere da un valore minimo pari a 0 ACU fino a un valore massimo pari a 256 ACU, con incrementi di 0,5 ACU.

   Per ulteriori informazioni sulle unità di capacità Aurora Serverless v2, consulta Capacità di Aurora Serverless v2 e Prestazioni e dimensionamento per Aurora Serverless v2.

   A seconda del motore e della versione scelti, il limite superiore potrebbe essere di 128 ACU, il limite inferiore di 0,5 ACU o entrambi. Per informazioni dettagliate sul limite per ogni combinazione di motore e versione Aurora, consulta Capacità di Aurora Serverless v2.



La scelta di una capacità minima di 0 ACU abilita la capacità di pausa e ripresa automatiche Aurora Serverless v2. In questo caso, è anche possibile indicare il tempo di attesa delle istanze database Aurora Serverless v2 prima della pausa automatica in assenza di connessioni al database. Per informazioni sulla capacità di pausa e ripresa automatiche, consulta Dimensionamento a zero ACU con pausa e ripresa automatiche per Aurora Serverless v2.



6. Scegliere qualsiasi altra impostazione del cluster di database, come descritto in Impostazioni per cluster di database Aurora.
7. Scegliere Crea database per creare il cluster di database Aurora con un'istanza database Aurora Serverless v2 come istanza di scrittura, nota anche come istanza database primaria.

CLI

Per creare un cluster DB compatibile con le istanze database Aurora Serverless v2 che utilizzano AWS CLI, seguire la procedura descritta in Creazione di un cluster database Amazon Aurora. Includi i seguenti parametri nel comando `create-db-cluster`:

- --region *AWS_Region_where_Aurora Serverless v2_instances_are_available*
- --engine-version *serverless_v2_compatible_engine_version*
- --serverless-v2-scaling-configuration MinCapacity=*minimum_capacity*,MaxCapacity=*maximum_capacity*

L'esempio seguente mostra la creazione di un cluster database Aurora Serverless v2.

```
aws rds create-db-cluster \
    --db-cluster-identifier my-serverless-v2-cluster \
    --region eu-central-1 \
    --engine aurora-mysql \
    --engine-version 8.0.mysql_aurora.3.04.1 \
    --serverless-v2-scaling-configuration MinCapacity=1,MaxCapacity=4 \
    --master-username myuser \
    --manage-master-user-password
```

> **Note**
>
> Quando crei un cluster database Aurora Serverless v2 mediante la AWS CLI, la modalità motore viene visualizzata nell'output come `provisioned` anziché `serverless`. La modalità motore `serverless` si riferisce a Aurora Serverless v1.

In questo esempio è specificata l'opzione `--manage-master-user-password` per generare la password amministrativa e gestirla in Secrets Manager. Per ulteriori informazioni, consulta Gestione delle password con Amazon Aurora e Gestione dei segreti AWS. In alternativa, puoi utilizzare l'opzione `--master-password` per specificare e gestire personalmente la password.

Per informazioni sui requisiti di versione per Aurora Serverless v2, consulta Requisiti e limitazioni per Aurora Serverless v2. Per informazioni sui numeri consentiti per l'intervallo di capacità e su cosa rappresentano questi numeri, consulta Capacità di Aurora Serverless v2 e Prestazioni e dimensionamento per Aurora Serverless v2.

Per verificare se per un cluster esistente sono state specificate le impostazioni di capacità, controlla l'output del comando `describe-db-clusters` per l'attributo `ServerlessV2ScalingConfiguration`. Questo attributo è simile a quello riportato di seguito.

```
"ServerlessV2ScalingConfiguration": {
    "MinCapacity": 1.5,
    "MaxCapacity": 24.0
}
```

> **Tip**
>
> Se non sono stati specificati valori ACU minimi e massimi durante la creazione del cluster, per aggiungere tale impostazione è possibile utilizzare il comando `modify-db-cluster` in un secondo momento. Finché non esegui tale operazione non potrai aggiungere altre istanze DB Aurora Serverless v2 al cluster. Se tenti di aggiungere un'istanza DB `db.serverless`, verrà restituito un errore.

API

Per creare un cluster database compatibile con le istanze DB Aurora Serverless v2 che utilizzano l'API RDS, segui la procedura API descritta in Creazione di un cluster database Amazon Aurora. Scegli le impostazioni descritte di seguito. Assicurati che l'operazione `CreateDBCluster` includa i parametri seguenti:

```
EngineVersion serverless_v2_compatible_engine_version
ServerlessV2ScalingConfiguration with MinCapacity=minimum_capacity and
 MaxCapacity=maximum_capacity
```

Per informazioni sui requisiti di versione per Aurora Serverless v2, consulta Requisiti e limitazioni per Aurora Serverless v2. Per informazioni sui numeri consentiti per l'intervallo di capacità e su cosa rappresentano questi numeri, consulta Capacità di Aurora Serverless v2 e Prestazioni e dimensionamento per Aurora Serverless v2.

Per verificare se per un cluster esistente sono state specificate le impostazioni di capacità, controlla l'output dell'operazione `DescribeDBClusters` per l'attributo `ServerlessV2ScalingConfiguration`. Questo attributo è simile a quello riportato di seguito.

```
"ServerlessV2ScalingConfiguration": {
    "MinCapacity": 1.5,
    "MaxCapacity": 24.0
}
```

> **Tip**
>
> Se non sono stati specificati valori ACU minimi e massimi durante la creazione del cluster, per aggiungere tale impostazione è possibile utilizzare l'operazione `ModifyDBCluster` in un

secondo momento. Finché non esegui tale operazione non potrai aggiungere altre istanze DB Aurora Serverless v2 al cluster. Se tenti di aggiungere un'istanza database `db.serverless`, verrà restituito un errore.

Creazione di un'istanza database di scrittura Aurora Serverless v2

Anche se si specifica l'intervallo di capacità Aurora Serverless v2 quando si crea un cluster Aurora, è possibile scegliere se utilizzare Aurora Serverless v2 o un cluster con provisioning per ogni istanza database del cluster. Per iniziare a utilizzare Aurora Serverless v2 immediatamente nel cluster di database, aggiungere un'istanza database di scrittura che utilizza la classe di istanza `db.serverless`. Nella console, in genere questa opzione viene scelta nel contesto del flusso di lavoro per creare il cluster di database. Pertanto, questa procedura si applica principalmente quando si esegue la configurazione tramite l'interfaccia della riga di comando.

Console

Quando si crea un cluster con la Console di gestione AWS, vengono specificate contemporaneamente le proprietà dell'istanza database di scrittura. Affinché l'istanza database di scrittura possa utilizzare Aurora Serverless v2, scegli la classe di istanza database Serverless.

Impostare quindi l'intervallo di capacità per il cluster specificando i valori dell'unità di capacità Aurora (ACU) minimi e massimi. I valori minimi e massimi vengono applicati a ogni istanza database Aurora Serverless v2 nel cluster. Per la procedura e il significato delle impostazioni di capacità minima e massima, consulta Creazione di un cluster di database Aurora Serverless v2.

Se non crei un'istanza database Aurora Serverless v2 quando crei il cluster, puoi aggiungere una o più istanze database Aurora Serverless v2 in un secondo momento. A tale scopo, seguire la procedura descritta in Aggiunta di un'istanza Aurora Serverless v2 di lettura e Conversione di un'istanza di lettura o scrittura con provisioning per Aurora Serverless v2. Puoi specificare l'intervallo di capacità quando aggiungi la prima istanza DB Aurora Serverless v2 al cluster. È possibile modificare l'intervallo di capacità in un secondo momento seguendo la procedura descritta in Impostazione dell'intervallo di capacità di Aurora Serverless v2 per un cluster.

CLI

Quando crei un cluster database Aurora Serverless v2 tramite la AWS CLI, aggiungi in maniera esplicita l'istanza DB di scrittura utilizzando il comando create-db-instance. Includere il seguente parametro:

- `--db-instance-class db.serverless`

L'esempio seguente mostra la creazione di un'istanza database di scrittura Aurora Serverless v2.

```
aws rds create-db-instance \
    --db-cluster-identifier my-serverless-v2-cluster \
    --db-instance-identifier my-serverless-v2-instance \
    --db-instance-class db.serverless \
    --engine aurora-mysql
```

Gestione di cluster di database Aurora Serverless v2

In Aurora Serverless v2, i cluster sono intercambiabili con i cluster con provisioning. Le proprietà di Aurora Serverless v2 vengono applicate a una o più istanze database all'interno di un cluster di database. Pertanto, le procedure per la creazione e la modifica di cluster, per la creazione e il ripristino di snapshot e così via sono sostanzialmente le stesse procedure usate per gli altri tipi di cluster Aurora. Per le procedure generali per la gestione di cluster Aurora e istanze database, consulta Gestione di un cluster DB Amazon Aurora.

Gli argomenti successivi offrono considerazioni sulla gestione dei cluster che contengono istanze database Aurora Serverless v2.

Argomenti

- Impostazione dell'intervallo di capacità di Aurora Serverless v2 per un cluster
- Verifica dell'intervallo di capacità per Aurora Serverless v2
- Aggiunta di un'istanza Aurora Serverless v2 di lettura
- Conversione di un'istanza di lettura o scrittura con provisioning per Aurora Serverless v2
- Conversione di un'istanza Aurora Serverless v2 di scrittura o lettura in istanza con provisioning
- Messa in pausa delle istanze database Aurora Serverless v2
- Scelta del livello di promozione per un'istanza Aurora Serverless v2 di lettura
- Utilizzo di TLS/SSL con Aurora Serverless v2
- Visualizzazione di istanze Aurora Serverless v2 di scrittura e lettura
- Registrazione per Aurora Serverless v2

Impostazione dell'intervallo di capacità di Aurora Serverless v2 per un cluster

Per modificare i parametri di configurazione o altre impostazioni per i cluster contenenti istanze DB Aurora Serverless v2 o per le istanze DB stesse, seguire le stesse procedure generali valide per i cluster con provisioning. Per informazioni dettagliate, consultare Modifica di un cluster database Amazon Aurora.

L'impostazione più importante specifica per Aurora Serverless v2 è l'intervallo di capacità. Dopo aver impostato i valori minimi e massimi dell'unità di capacità Aurora (ACU) per un cluster Aurora, non è necessario regolare attivamente la capacità delle istanze DB Aurora Serverless v2 nel cluster. Aurora esegue automaticamente questa operazione. Questa impostazione è gestita a livello di cluster. Gli stessi valori ACU minimi e massimi si applicano a ciascuna istanza DB Aurora Serverless v2 nel cluster.

Puoi impostare i seguenti valori specifici:

- Minimum ACUs (Valore ACU minimo): l'istanza DB Aurora Serverless v2 può ridurre la capacità fino a questo numero di ACU.
- Maximum ACUs (Valore ACU massimo): l'istanza database Aurora Serverless v2 può aumentare la capacità fino a questo numero di ACU.

L’intervallo di capacità disponibile per Aurora Serverless v2 è una funzione sia della versione del motore database che della versione della piattaforma. Le versioni più recenti del motore database consentono una capacità massima di 256 ACU, una capacità minima di 0 ACU o entrambe. Per gli intervalli di capacità per le varie versioni del motore database, consulta Capacità di Aurora Serverless v2.

Per la capacità di pausa e ripresa automatiche, abilitata impostando la capacità minima su 0 ACU, consulta Dimensionamento a zero ACU con pausa e ripresa automatiche per Aurora Serverless v2.

Per ulteriori informazioni su come assicurarsi che i cluster di database Aurora Serverless v2 possano aumentare verticalmente fino a 256 ACU, consulta Aggiornamento del cluster di database Aurora Serverless v2 per consentire il dimensionamento fino a 256 ACU.

> **Note**
>
> Quando si modifica l'intervallo di capacità per un cluster database Aurora Serverless v2, la modifica viene implementata subito, indipendentemente dal fatto che si scelga di applicarla immediatamente o durante la successiva finestra di manutenzione pianificata.

In Aurora Serverless v2, non è possibile impostare direttamente la capacità corrente senza modificare l'intervallo di capacità, perché la capacità si regola dinamicamente in base al carico di lavoro all'interno dell'intervallo specificato. Tuttavia, è possibile influenzare la capacità in modo indiretto nei modi seguenti:

- Regola la capacità minima: riduce temporaneamente la capacità minima per limitare la capacità di base quando i carichi di lavoro sono leggeri.
- Sospendi temporaneamente il dimensionamento: per fissare la capacità su un valore specifico, impostare la capacità minima e massima allo stesso livello.
- Scala in modo proattivo per gestire i picchi di utilizzo: se si prevede un aumento esponenziale dei carichi di lavoro, aumentare preventivamente la capacità minima per disporre di una capacità di base più elevata durante i periodi di intensa attività.

Per informazioni dettagliate sugli effetti dell'intervallo di capacità e su come monitorarlo e ottimizzarlo, consulta Parametri di Amazon CloudWatch importanti per Aurora Serverless v2 e Prestazioni e dimensionamento per Aurora Serverless v2. L'obiettivo è far sì che la capacità massima per il cluster sia sufficientemente elevata da essere in grado di gestire i picchi del carico di lavoro e che la capacità minima sia sufficientemente bassa da ridurre al minimo i costi quando il cluster non è occupato.

Supponiamo che in base al monitoraggio effettuato si determini che l'intervallo ACU per il cluster debba essere più alto, più basso, meno limitativo o più limitativo. Puoi impostare la capacità di un cluster Aurora su un intervallo specifico di ACU mediante Console di gestione AWS, AWS CLI o l'API Amazon RDS. Questo intervallo di capacità si applica a tutte le istanze DB Aurora Serverless v2 incluse nel cluster.

Si supponga, ad esempio, che il cluster disponga di un intervallo di capacità di 1-16 ACU e che contenga due istanze DB Aurora Serverless v2. Il cluster nel suo complesso consumerà quindi tra 2 ACU (quando è inattivo) e 32 ACU (quando completamente utilizzato). Se si modifica l'intervallo di capacità da 8 a 40,5 ACU, il cluster consumerà 16 ACU quando è inattivo e fino a 81 ACU quando è completamente utilizzato.

Aurora imposta automaticamente determinati parametri per le istanze database Aurora Serverless v2 su valori che dipendono dal valore ACU massimo nell'intervallo di capacità. Per l'elenco completo dei parametri, consulta Numero massimo connessioni per Aurora Serverless v2. Per i parametri statici che si basano su questo tipo di calcolo, il valore viene nuovamente valutato al riavvio dell'istanza DB. Pertanto, è possibile aggiornare il valore di tali parametri riavviando l'istanza DB dopo aver modificato l'intervallo di capacità. Per verificare se è necessario riavviare l'istanza DB per acquisire le modifiche apportate ai parametri, controlla l'attributo `ParameterApplyStatus` dell'istanza DB. Il valore `pending-reboot` indica che il riavvio applicherà le modifiche ad alcuni valori di parametro.

Console

Puoi impostare l'intervallo di capacità di un cluster contenente istanze DB Aurora Serverless v2 mediante Console di gestione AWS.

Quando utilizzi la console, imposti l'intervallo di capacità per il cluster quando aggiungi la prima istanza DB Aurora Serverless v2 a tale cluster. Questa operazione può essere eseguita quando scegli la classe di istanza DB Serverless v2 per l'istanza DB di scrittura durante la creazione del cluster. In alternativa, puoi eseguire questa operazione quando scegli la classe di istanza DB Serverless durante l'aggiunta di un'istanza DB Aurora Serverless v2 di lettura al cluster. In alternativa, puoi eseguire questa operazione quando converti un'istanza database con provisioning esistente nel cluster in un'istanza database di classe Serverless. Per le versioni complete di queste procedure, consulta Creazione di un'istanza database di scrittura Aurora Serverless v2, Aggiunta di un'istanza Aurora Serverless v2 di lettura eConversione di un'istanza di lettura o scrittura con provisioning per Aurora Serverless v2.

Qualsiasi impostazione dell'intervallo di capacità a livello di cluster si applica anche a tutte le istanze database Aurora Serverless v2 nel cluster. L'immagine seguente mostra un cluster con più istanze DB Aurora Serverless v2 di lettura. Ciascuna istanza ha un intervallo di capacità identico pari a 2-64 ACU.



Per modificare l'intervallo di capacità di un cluster Aurora Serverless v2

1. Apri la console Amazon RDS all'indirizzo https://console.aws.amazon.com/rds/.
2. Nel riquadro di navigazione, scegliere Databases (Database).
3. Scegliere il cluster contenente le istanze DB Aurora Serverless v2 nell'elenco. Il cluster deve già contenere almeno un'istanza DB Aurora Serverless v2. In caso contrario, Aurora non visualizza la sezione Capacity range (Intervallo capacità).
4. Per Operazioni, scegli Modifica.
5. Nella sezione Capacity range (Intervallo capacità), scegliere quanto segue:

   a. Immettere un valore in Minimum ACUs (Valore ACU minimo). La console mostra l'intervallo di valori consentito. È possibile scegliere una capacità minima da 0 a 256 ACU. È possibile scegliere una capacità massima da 1 a 256 ACU. È possibile regolare i valori di capacità con incrementi di 0,5 ACU. L'intervallo di capacità disponibile dipende sia dalla versione del motore database che dalla versione della piattaforma.

   b. Immettere un valore in Maximum ACUs (Valore ACU massimo). Questo valore deve essere maggiore o uguale al valore specificato in Minimum ACUs (Valore ACU minimo). La console mostra l'intervallo di valori consentito. La figura seguente mostra questa selezione.



6. Scegli Continue (Continua). Viene visualizzata la pagina Riepilogo delle modifiche .
7. Scegliere Apply immediately (Applica immediatamente).

   La modifica della capacità viene implementata subito, indipendentemente dal fatto che si scelga di applicarla immediatamente o durante la successiva finestra di manutenzione pianificata.
8. Seleziona Modifica cluster per accettare il riepilogo delle modifiche. Puoi inoltre scegliere Indietro per modificare le modifiche o Annulla per ignorarle.

AWS CLI

Per impostare la capacità di un cluster in cui si intende utilizzare istanze DB Aurora Serverless v2 mediante AWS CLI, eseguire il comando AWS CLI modify-db-cluster. Specifica l'opzione `--serverless-v2-scaling-configuration`. Il cluster potrebbe già contenere una o più istanze DB Aurora Serverless v2 oppure è possibile aggiungere istanze DB in un secondo momento. I valori validi per i campi `MinCapacity` e `MaxCapacity` includono quelli riportati di seguito:

- `0`, `0.5`, `1`, `1.5`, 2 e così via, con incrementi di 0,5, fino a un massimo di 256. Il valore ACU minimo pari a 0 è disponibile solo per le versioni Aurora che supportano la funzionalità di pausa automatica.

La capacità massima disponibile dipende sia dalla versione del motore database che dalla versione della piattaforma.

In questo esempio, viene impostato l'intervallo ACU di un cluster database Aurora denominato `sample-cluster` sul valore minimo `48.5` e sul valore massimo 64.

```
aws rds modify-db-cluster --db-cluster-identifier sample-cluster \
  --serverless-v2-scaling-configuration MinCapacity=48.5,MaxCapacity=64
```

La modifica della capacità viene implementata subito, indipendentemente dal fatto che si scelga di applicarla immediatamente o durante la successiva finestra di manutenzione pianificata.

A questo punto, è possibile aggiungere istanze database Aurora Serverless v2 al cluster e ogni nuova istanza database può essere dimensionata in base a valori compresi tra 48,5 e 64 ACU. Il nuovo intervallo di capacità si applica anche a qualsiasi istanza DB Aurora Serverless v2 già presente nel cluster. Se necessario, le istanze DB vengono aumentate o ridotte in modo che i relativi valori siano compresi nel nuovo intervallo di capacità.

Per ulteriori esempi di impostazione dell'intervallo di capacità utilizzando la CLI, consulta Scelta dell'intervallo di capacità di Aurora Serverless v2 per un cluster Aurora.

Per modificare la configurazione di dimensionamento di un cluster di database Aurora Serverless tramite AWS CLI, emetti il comando modify-db-cluster AWS CLI. Specifica l'opzione `--serverless-v2-scaling-configuration` per configurare la capacità minima e la capacità massima. I valori di capacità validi includono quanto segue:

- Aurora MySQL: `0`, `0.5`, `1`, `1.5`, 2 e così via, con incrementi di 0,5 ACU, fino a un massimo di 256.

- Aurora PostgreSQL: `0`, `0.5`, `1`, `1.5`, 2 e così via, con incrementi di 0,5 ACU fino a un massimo di 256.
- Il valore ACU minimo pari a 0 è disponibile solo per le versioni Aurora che supportano la funzionalità di pausa automatica.

Nell'esempio seguente viene modificata la configurazione di dimensionamento di un'istanza database Aurora Serverless v2 denominata `sample-instance`, che fa parte di un cluster denominato `sample-cluster`.

Per Linux, macOS o Unix:

```
aws rds modify-db-cluster --db-cluster-identifier sample-cluster \
--serverless-v2-scaling-configuration MinCapacity=8,MaxCapacity=64
```

Per Windows:

```
aws rds modify-db-cluster --db-cluster-identifier sample-cluster ^
--serverless-v2-scaling-configuration MinCapacity=8,MaxCapacity=64
```

API RDS

Puoi impostare le capacità di un cluster database Aurora mediante l'operazione API ModifyDBCluster. Specifica il parametro `ServerlessV2ScalingConfiguration`. I valori validi per i campi `MinCapacity` e `MaxCapacity` includono quelli riportati di seguito:

- `0`, `0.5`, `1`, `1.5`, 2 e così via, con incrementi di 0,5, fino a un massimo di 256. Il valore ACU minimo pari a 0 è disponibile solo per le versioni Aurora che supportano la funzionalità di pausa automatica.

Puoi modificare la configurazione di dimensionamento di un cluster contenente istanze database Aurora Serverless v2 mediante l'operazione API ModifyDBCluster. Specifica il parametro `ServerlessV2ScalingConfiguration` per configurare la capacità minima e la capacità massima. I valori di capacità validi includono quanto segue:

- Aurora MySQL: `0`, `0.5`, `1`, `1.5`, 2 e così via, con incrementi di 0,5 ACU, fino a un massimo di 256.
- Aurora PostgreSQL: `0`, `0.5`, `1`, `1.5`, 2 e così via, con incrementi di 0,5 ACU fino a un massimo di 256.

- Il valore ACU minimo pari a 0 è disponibile solo per le versioni Aurora che supportano la funzionalità di pausa automatica.

La capacità massima disponibile dipende sia dalla versione del motore database che dalla versione della piattaforma.

La modifica della capacità viene implementata subito, indipendentemente dal fatto che si scelga di applicarla immediatamente o durante la successiva finestra di manutenzione pianificata.

Aggiornamento del cluster di database Aurora Serverless v2 per consentire il dimensionamento fino a 256 ACU

In alcuni casi, quando si prova a scalare il cluster di database Aurora Serverless v2 a capacità superiori a 128 ACU, viene visualizzato un messaggio di errore. Il messaggio di errore indica quali istanze sono incompatibili con il nuovo intervallo di dimensionamento. Questo aspetto evidenzia l'importante relazione tra l'intervallo di capacità, la versione del motore database e la versione della piattaforma.

L'impossibilità di scalare il cluster fino a una capacità superiore a 128 ACU può essere dovuta a uno dei due motivi seguenti:

- Versione precedente del motore database: aggiornare la versione del motore database a una che supporti 256 ACU. Per ulteriori informazioni, consulta Capacità di Aurora Serverless v2.
- Versione precedente della piattaforma: aggiornare la piattaforma per il cluster di database Aurora Serverless v2. Questa operazione può essere eseguita in uno dei seguenti modi:
  - Arrestare e riavviare il cluster di database. Al riavvio, il cluster sarà eseguito sull'ultima versione della piattaforma, che potrebbe supportare un valore ACU massimo più elevato.

    Tuttavia, l'arresto e l'avvio di un cluster di database comportano un certo tempo di inattività, in genere di diversi minuti. Per ulteriori informazioni, consulta Avvio e arresto di un cluster di database Amazon Aurora.
  - Utilizzare l'implementazione blu/verde. Per ulteriori informazioni, consulta Panoramica delle implementazioni di Amazon Aurora Blue/Green .
    1. Creare un'implementazione blu/verde. Per ulteriori informazioni, consulta Creazione di una blue/green distribuzione in .
    2. Verificare di poter impostare la capacità massima per l'ambiente di staging (verde) su 256 ACU.

3. Passare all'ambiente verde. Per ulteriori informazioni, consulta Cambiare una blue/green distribuzione in .
4. Eliminare il cluster di database originale.

> **Note**
>
> Se si conservano le credenziali del database inGestione dei segreti AWS, non è possibile utilizzare le implementazioni blu/verdi.
> Il Database globale Aurora non supporta le implementazioni blu/verdi.

Verifica dell'intervallo di capacità per Aurora Serverless v2

La procedura di verifica dell'intervallo di capacità del cluster Aurora Serverless v2 richiede innanzitutto l'impostazione di un intervallo di capacità. Se questa verifica non è ancora stata eseguita, segui la procedura descritta in Impostazione dell'intervallo di capacità di Aurora Serverless v2 per un cluster.

Qualsiasi impostazione dell'intervallo di capacità a livello di cluster si applica anche a tutte le istanze DB Aurora Serverless v2 nel cluster. L'immagine seguente mostra un cluster con più istanze DB Aurora Serverless v2. Ogni istanza ha un intervallo di capacità identico.



Puoi anche visualizzare la pagina dei dettagli di qualsiasi istanza database Aurora Serverless v2 nel cluster. L'intervallo di capacità delle istanze DB è visuaizzato nella scheda Configurazione.



È inoltre possibile visualizzare l'intervallo di capacità corrente per il cluster nella pagina Modifica del cluster. A questo punto, è possibile modificare l'intervallo di capacità. Per informazioni su come impostare o modificare l'intervallo di capacità, consulta Impostazione dell'intervallo di capacità di Aurora Serverless v2 per un cluster.

Verifica dell'intervallo di capacità corrente per un cluster Aurora

È possibile verificare l'intervallo di capacità configurato per le istanze DB Aurora Serverless v2 in un cluster esaminando l'attributo `ServerlessV2ScalingConfiguration` del cluster. Il seguente esempio in AWS CLI mostra un cluster con una capacità minima di 0,5 unità di capacità Aurora (ACU) e una capacità massima di 16 ACU.

```
$ aws rds describe-db-clusters --db-cluster-identifier serverless-v2-64-acu-cluster \
  --query 'DBClusters[*].[ServerlessV2ScalingConfiguration]'
[
    [
        {
            "MinCapacity": 0.5,
            "MaxCapacity": 16.0
        }
    ]
]
```

Aggiunta di un'istanza Aurora Serverless v2 di lettura

Per aggiungere un'istanza Aurora Serverless v2 di lettura al cluster, segui la stessa procedura generale descritta in Aggiunta di repliche di Aurora a un cluster di database. Seleziona Serverless v2 come classe di istanza per la nuova istanza DB.

Se l'istanza database di lettura è la prima istanza database Aurora Serverless v2 nel cluster, è anche possibile scegliere l'intervallo di capacità. Questa impostazione si applica all'istanza database di

lettura e a qualsiasi altra istanza database Aurora Serverless v2 aggiunta al cluster. Ogni istanza database Aurora Serverless v2 può subire un dimensionamento compreso tra i valori ACU minimo e massimo.

Se nel cluster esistono già altre istanze Aurora Serverless v2, la finestra di dialogo Aggiungi lettore mostra l'intervallo di capacità corrente per il cluster. In tal caso, la capacità non può essere modificata. L'immagine seguente mostra il report sulla capacità corrente del cluster.



Se hai già aggiunto istanze database Aurora Serverless v2 al cluster, l'aggiunta di un'altra istanza database Aurora Serverless v2 di lettura visualizza l'intervallo di capacità corrente. L'immagine seguente mostra i controlli di sola lettura.



Se vuoi modificare l'intervallo di capacità per il cluster, segui la procedura descritta in Impostazione dell'intervallo di capacità di Aurora Serverless v2 per un cluster.

Per i cluster contenenti più di un'istanza DB di lettura, la priorità di failover di ciascuna istanza DB Aurora Serverless v2 di lettura ha un ruolo importante nel modo in cui il dimensionamento verso l'alto e la riduzione vengono applicati all'istanza DB. Non è possibile specificare la priorità nella fase iniziale di creazione del cluster. Tieni presente questa proprietà quando al cluster aggiungi una seconda istanza DB di lettura o altre istanze di questo tipo. Per ulteriori informazioni, consulta Scelta del livello di promozione per un'istanza Aurora Serverless v2 di lettura.

Per altri modi in cui è possibile visualizzare l'intervallo di capacità corrente per un cluster, consulta Verifica dell'intervallo di capacità per Aurora Serverless v2.

Conversione di un'istanza di lettura o scrittura con provisioning per Aurora Serverless v2

Puoi convertire un'istanza DB con provisioning in modo che utilizzi Aurora Serverless v2. A tale scopo, segui la procedura descritta in Modifica di un'istanza database in un cluster database. Il cluster deve soddisfare i requisiti riportati in Requisiti e limitazioni per Aurora Serverless v2. Ad esempio, le istanze DB Aurora Serverless v2 prevedono che il cluster esegua alcune versioni minime del motore.

Supponiamo che si stia convertendo un cluster con provisioning in esecuzione per poter sfruttare le caratteristiche di Aurora Serverless v2. In questo caso, è possibile ridurre al minimo i tempi di inattività convertendo un'istanza DB in Aurora Serverless v2 come primo passo del processo di passaggio. Per la procedura completa, consulta Passaggio di un cluster con provisioning ad Aurora Serverless v2.

Se l'istanza DB convertita è la prima istanza DB Aurora Serverless v2 nel cluster, è possibile scegliere l'intervallo di capacità per il cluster nell'ambito dell'operazione Modifica. Questo intervallo di capacità si applica a tutte le istanze DB Aurora Serverless v2 aggiunte al cluster. L'immagine seguente mostra la pagina in cui specificare le unità di capacità Aurora (ACU) minima e massima.



Per ulteriori informazioni sull'importanza dell'intervallo di capacità, consulta Capacità di Aurora Serverless v2.

Se il cluster contiene già una o più istanze DB Aurora Serverless v2, viene visualizzato l'intervallo di capacità esistente durante l'operazione Modifica. L'immagine seguente mostra un esempio del pannello informativo.



Se vuoi modificare l'intervallo di capacità per il cluster dopo avere aggiunto altre istanze DB Aurora Serverless v2, segui la procedura descritta in Impostazione dell'intervallo di capacità di Aurora Serverless v2 per un cluster.

Conversione di un'istanza Aurora Serverless v2 di scrittura o lettura in istanza con provisioning

È possibile convertire un'istanza DB Aurora Serverless v2 in un'istanza DB con provisioning. A tale scopo, segui la procedura descritta in Modifica di un'istanza database in un cluster database. Scegli una classe di istanza DB diversa da Serverless.

Potresti convertire un'istanza DB Aurora Serverless v2 in istanza DB con provisioning se deve disporre di una capacità maggiore rispetto a quella specificata dalle unità di capacità Aurora (ACU) massime di un'istanza DB Aurora Serverless v2. Ad esempio, le classi di istanza DB db.r5 e db.r6g più grandi hanno una capacità di memoria maggiore rispetto a quella in base alla quale può essere dimensionata un'istanza DB Aurora Serverless v2.

> **Tip**
>
> Alcune delle classi di istanze DB meno recenti, ad esempio db.r3 e db.t2, non sono disponibili per le versioni Aurora utilizzate con Aurora Serverless v2. Per vedere quali classi di istanze DB è possibile utilizzare durante la conversione di un'istanza DB Aurora Serverless v2 in un'istanza con provisioning, consulta Motori di database supportati per classi di istanza database.

Se stai convertendo l'istanza DB di scrittura del cluster da Aurora Serverless v2 in istanza con provisioning, è possibile seguire la procedura descritta in Passaggio di un cluster con provisioning

ad Aurora Serverless v2 ma in ordine inverso. Cambia una delle istanze DB di lettura nel cluster da Aurora Serverless v2 in istanza con provisioning. Esegui quindi un failover per convertire l'istanza DB con provisioning in istanza di scrittura.

Qualsiasi intervallo di capacità specificato in precedenza per il cluster rimane in vigore, anche se tutte le istanze DB Aurora Serverless v2 vengono rimosse dal cluster. Se vuoi modificare l'intervallo di capacità, puoi modificare il cluster, come spiegato in Impostazione dell'intervallo di capacità di Aurora Serverless v2 per un cluster.

Messa in pausa delle istanze database Aurora Serverless v2

È possibile configurare i cluster Aurora per mettere in pausa e riprendere automaticamente le istanze database Aurora Serverless v2. Per ulteriori informazioni, consulta Dimensionamento a zero ACU con pausa e ripresa automatiche per Aurora Serverless v2.

Scelta del livello di promozione per un'istanza Aurora Serverless v2 di lettura

Per cluster contenenti più istanze DB Aurora Serverless v2 o una combinazione di istanze con provisioning e istanze DB Aurora Serverless v2, presta attenzione all'impostazione del livello di promozione per ciascuna istanza DB Aurora Serverless v2. Questa impostazione ha un controllo sul comportamento delle istanze DB Aurora Serverless v2 maggiore rispetto a quanto avviene per istanze DB con provisioning.

In Console di gestione AWS, specifica questa impostazione utilizzando l'opzione Priorità failover disponibile nell'area Configurazione aggiuntiva nelle pagine Crea database, Modifica istanza e Aggiungi lettore. Questa proprietà viene visualizzata per le istanze DB esistenti nella colonna facoltativa Livello di priorità nella pagina Database. È inoltre possibile visualizzare questa proprietà nella pagina dei dettagli di un cluster di database o un'istanza DB.

Per le istanze DB con provisioning, la scelta del livello 0-15 determina solo l'ordine in base al quale Aurora sceglie l'istanza DB di lettura da promuovere a istanza di scrittura durante un'operazione di failover. Per le istanze DB Aurora Serverless v2 di lettura, il numero di livello determina anche se l'istanza DB viene aumentata fino alla capacità dell'istanza DB di scrittura o viene dimensionata in modo indipendente in base al proprio carico di lavoro. Le istanze DB Aurora Serverless v2 di lettura di livello 0 o 1 sono mantenute a una capacità minima almeno pari a quella dell'istanza DB di scrittura. In questo modo, tali istanze sono pronte a prendere il controllo dall'istanza DB di scrittura in caso di failover. Se l'istanza DB di scrittura è un'istanza DB con provisioning, Aurora calcola la stima di una

capacità Aurora Serverless v2 equivalente. Utilizza tale stima come capacità minima per l'istanza DB Aurora Serverless v2 di lettura.

Le istanze DB Aurora Serverless v2 di lettura nei livelli 2-15 non hanno lo stesso vincolo relativamente alla capacità minima. Quando sono inattive, possono essere ridotte fino al valore minimo dell'unità di capacità Aurora (ACU) specificato nell'intervallo di capacità del cluster.

Il seguente esempio in AWS CLI per Linux mostra come esaminare i livelli di promozione di tutte le istanze DB nel cluster. Il campo finale include il valore `True` per l'istanza DB di scrittura e il valore `False` per tutte le istanze DB di lettura.

```
$ aws rds describe-db-clusters --db-cluster-identifier promotion-tier-demo \
  --query 'DBClusters[*].DBClusterMembers[*].
[PromotionTier,DBInstanceIdentifier,IsClusterWriter]' \
  --output text

1   instance-192  True
1   tier-01-4840  False
10  tier-10-7425  False
15  tier-15-6694  False
```

Il seguente esempio in AWS CLI per Linux mostra come modificare il livello di promozione di un'istanza DB specifica nel cluster. I comandi modificano innanzitutto l'istanza DB in base a un nuovo livello di promozione. Attendono quindi che l'istanza DB ridiventi disponibile e ne confermano il nuovo livello di promozione.

```
$ aws rds modify-db-instance --db-instance-identifier instance-192 --promotion-tier 0
$ aws rds wait db-instance-available --db-instance-identifier instance-192
$ aws rds describe-db-instances --db-instance-identifier instance-192 \
  --query '*[].[PromotionTier]' --output text
0
```

Per ulteriori informazioni su come specificare i livelli di promozione per i diversi casi d'uso, consulta Dimensionamento di Aurora Serverless v2.

Utilizzo di TLS/SSL con Aurora Serverless v2

Aurora Serverless v2 utilizza il protocollo TLS/SSL (Transport Layer Security/Secure Sockets Layer) per crittografare le comunicazioni tra i client e il cluster database Aurora Serverless v2. Supporta

TLS/SSL versione 1.0, 1.1 e 1.2. Per informazioni generali sull'utilizzo del protocollo TLS/SSL con Aurora, consulta Connessioni TLS a cluster di database Aurora MySQL.

Per maggiori informazioni sulla connessione al database Aurora MySQL con il client MySQL, consulta Connessione a un'istanza DB che esegue il motore del database MySQL.

Aurora Serverless v2 supporta tutte le modalità TLS/SSL disponibili per il client MySQL (`mysql`) e il client PostgreSQL (`psql`), incluse quelle elencate nella tabella seguente.

Descrizione della modalità TLS/SSL	mysql	psql
Connettiti senza utilizzare TLS/SSL.	DISABLED	disattiva
Prova prima la connessione utilizzando TLS/SSL, ma se necessario, torna all'uso senza SSL.	PREFERRED	prefer (impostazione predefinita)
Applica l'uso di TLS/SSL.	REQUIRED	require
Applicare TLS/SSL e verificare l'autorità di certificazione (CA).	VERIFY_CA	verify-ca
Applica TLS/SSL, verifica la CA e verifica il nome host della CA.	VERIFY_IDENTITY	verify-full

Aurora Serverless v2 utilizza certificati con caratteri jolly. Se specifichi l'opzione "verifica CA" o "verifica CA e nome host CA" quando utilizzi TLS/SSL, scarica innanzitutto il trust store CA 1 radice di Amazon da Amazon Trust Services. Dopo aver fatto ciò, puoi identificare questo file in formato PEMA nel comando client. Per eseguire questa operazione utilizzando il client PostgreSQL, procedi come segue.

Per Linux, macOS o Unix:

```
psql 'host=endpoint user=user sslmode=require sslrootcert=amazon-root-CA-1.pem
 dbname=db-name'
```

Per ulteriori informazioni sull'utilizzo del database Aurora PostgreSQL utilizzando il client Postgres, consulta Connessione a un'istanza database che esegue il modulo di motore del database PostgreSQL.

Per informazioni generali sulla connessione ai cluster database Aurora, consulta Connessione a un cluster database Amazon Aurora.

Suite di crittografia supportate per connessioni a cluster DB Aurora Serverless v2

Utilizzando suite di cifratura configurabili, è possibile avere maggiore controllo sulla sicurezza delle connessioni al database. È possibile specificare un elenco di suite di crittografia che si desidera abilitare per proteggere le connessioni TLS/SSL client al database. Con le suite di cifratura, è possibile controllare la crittografia di connessione accettata dal server di database. In questo modo si impedisce l'uso di sistemi di crittografia non sicuri o obsoleti.

I cluster DB Aurora Serverless v2 basati su Aurora MySQL supportano le stesse suite di crittografia dei cluster DB con provisioning di Aurora MySQL. Per informazioni su queste suite di crittografia, consulta Configurazione di suite di cifratura per connessioni ai cluster di database Aurora MySQL.

I cluster DB Aurora Serverless v2 basati su Aurora PostgreSQL supportano le stesse suite di cifratura dei cluster DB con provisioning di Aurora PostgreSQL. Per informazioni su queste suite di crittografia, consulta Configurazione di suite di cifratura per connessioni ai cluster di database Aurora PostgreSQL.

Visualizzazione di istanze Aurora Serverless v2 di scrittura e lettura

Puoi visualizzare i dettagli delle istanze DB Aurora Serverless v2 nello stesso modo in cui vengono visualizzati i dettagli delle istanze DB con provisioning. A tale scopo, segui la procedura generale descritta in Visualizzazione di un cluster di database Amazon Aurora. Un cluster potrebbe contenere solo istanze DB Aurora Serverless v2, solo istanze DB con provisioning o una combinazione di entrambe.

Dopo aver creato una o più istanze DB Aurora Serverless v2, puoi visualizzare quali istanze DB sono di tipo Serverless e quali di tipo Istanza. È inoltre possibile visualizzare le unità di capacità Aurora (ACU) minima e massima che l'istanza DB Aurora Serverless v2 può utilizzare. Ogni ACU è la combinazione di capacità di calcolo (CPU) e memoria (RAM). Questo intervallo di capacità si applica a tutte le istanze DB Aurora Serverless v2 nel cluster. Per la procedura di controllo dell'intervallo di capacità di un cluster o di qualsiasi altra istanza DB Aurora Serverless v2 nel cluster, consulta Verifica dell'intervallo di capacità per Aurora Serverless v2.

In Console di gestione AWS, le istanze DB Aurora Serverless v2 sono contrassegnate nella colonna Dimensioni nella pagina Database. Le istanze DB con provisioning riportano il nome di una classe di istanza DB, ad esempio r6g.xlarge. Le istanze DB Aurora Serverless riportano Serverless come classe di istanza DB, asieme alla relativa capacità minima e massima. Ad esempio, è possibile che vengano visualizzati dettagli quali Serverless v2 (4-64 ACU) o Serverless v2 (1-40 ACU).

Le stesse informazioni sono disponibili nella scheda per ciascuna istanza di database Aurora Serverless v2 nella console. Ad esempio, è possibile che venga visualizzata una sezione Tipo di istanza simile a quella illustrata di seguito. In questa scheda, il valore Tipo di istanza è Serverless v2, il valore Capacità minima è 2 ACU (4 GiB) e il valore Capacità massima è 64 ACU (128 GiB).



È possibile monitorare la capacità di ogni istanza DB Aurora Serverless v2 nel tempo. In questo modo, puoi verificare i valori ACU minimi, massimi e medi consumati da ciascuna istanza DB. È inoltre possibile verificare la prossimità dell'istanza DB alla relativa capacità minima o massima. Per vedere questi dettagli in Console di gestione AWS, analizza i grafici dei parametri di Amazon CloudWatch nella scheda Monitoraggio relativa all'istanza DB specifica. Per ulteriori informazioni su come visualizzare e interpretare i parametri, consulta Parametri di Amazon CloudWatch importanti per Aurora Serverless v2.

Registrazione per Aurora Serverless v2

Per attivare la registrazione del database, specifica i registri da abilitare mediante i parametri di configurazione nel gruppo di parametri personalizzati.

Per Aurora MySQL, puoi abilitare i seguenti registri.

Aurora MySQL	Descrizione
`general_log`	Crea il log generale. Imposta su 1 per attivare questa opzione. Il valore predefinito è disattiva to (0).
`log_queries_not_using_indexes`	Registra tutte le query nel log delle query lente che non utilizzano un indice. Il valore predefini to è disattivato (0). Imposta su 1 per attivare questo log.
`long_query_time`	Impedisce che le query in esecuzione rapida vengano registrate nel log delle query lente. Può essere impostato su un valore variabile compreso tra 0 e 31536000. Il valore predefini to è 0 (non attivo).
`server_audit_events`	L'elenco degli eventi da catturare nei log. I valori supportati sono `CONNECT`, `QUERY`, `QUERY_DCL` , `QUERY_DDL` , `QUERY_DML` e `TABLE`.
`server_audit_logging`	Imposta su 1 per attivare la registrazione di controllo del server. Se attivi questa opzione, puoi specificare gli eventi di controllo da inviare a CloudWatch riportandoli nel parametro `server_audit_events` .
`slow_query_log`	Crea un log di query lente. Imposta su 1 per attivare il log di query lente. Il valore predefinito è disattivato (0).

Per ulteriori informazioni, consulta Utilizzo dell'audit avanzato con un cluster di database Amazon Aurora MySQL.

Per Aurora PostgreSQL, puoi abilitare i seguenti registri nelle istanze DB Aurora Serverless v2.

Aurora PostgreSQL	Descrizione
`log_connections`	Registra ogni connessione riuscita.
`log_disconnections`	Registra la fine di una sessione, compresa la durata.
`log_lock_waits`	Il valore predefinito è 0 (disattivato). Imposta su 1 per registrare le attese di blocco.
`log_min_duration_statement`	La durata minima (in millisecondi) per l'esecuzione di un'istruzione prima della registrazione.
`log_min_messages`	Imposta i livelli dei messaggi che vengono registrati. I valori supportati sono `debug5`, `debug4`, `debug3`, `debug2`, `debug1`, `info`, `notice`, `warning`, `error`, `log`, `fatal`, `panic`. Per registrare i dati delle prestazioni nel log `postgres`, imposta il valore su `debug1`.
`log_temp_files`	Registra l'utilizzo di file temporanei che si trovano al di sopra dei kilobyte (kB) specificati.
`log_statement`	Controlla le istruzioni SQL specifiche che vengono registrate. I valori supportati sono `none`, `ddl`, `mod` e `all`. Il valore predefinito è `none`.

Argomenti

- Registrazione con Amazon CloudWatch
- Visualizzazione dei registri Aurora Serverless v2 in Amazon CloudWatch
- Monitoraggio della capacità con Amazon CloudWatch
- Monitoraggio delle attività di pausa e ripresa Aurora Serverless v2

Registrazione con Amazon CloudWatch

Dopo aver utilizzato la procedura descritta in Registrazione per Aurora Serverless v2 per scegliere i registri del database da attivare, puoi scegliere i registri da caricare ("pubblicare") in Amazon CloudWatch.

Puoi utilizzare Amazon CloudWatch puoi analizzare i dati dei registri, creare allarmi e visualizzare i parametri. Per impostazione di default, i registri degli errori per Aurora Serverless v2 sono abilitati e caricati automaticamente in CloudWatch. Puoi anche caricare altri registri dalle istanze database Aurora Serverless v2 in CloudWatch.

Scegli quindi quale di questi log caricare in CloudWatch utilizzando le impostazioni Log exports (Esportazioni del log) nella Console di gestione AWS oppure l'opzione `--enable-cloudwatch-logs-exports` disponibile in AWS CLI.

Puoi scegliere quale registro Aurora Serverless v2 caricare in CloudWatch. Per ulteriori informazioni, consulta Utilizzo dell'audit avanzato con un cluster di database Amazon Aurora MySQL.

Come per qualsiasi tipo di cluster database Aurora, non è possibile modificare il gruppo di parametri del cluster database predefinito. Crea invece il tuo gruppo di parametri del cluster database in base a un parametro predefinito per il cluster database e il tipo di motore. Si consiglia di creare il gruppo di parametri del cluster database personalizzato prima di creare il cluster database Aurora Serverless v2, in modo che sia possibile scegliere quando creare un database nella console.

> **Note**
>
> Per Aurora Serverless v2, è possibile creare sia cluster database che gruppi di parametri database. Questo è in contrasto con Aurora Serverless v1, dove è possibile solo creare gruppi di parametri cluster database.

Visualizzazione dei registri Aurora Serverless v2 in Amazon CloudWatch

Dopo aver utilizzato la procedura descritta in Registrazione con Amazon CloudWatch per scegliere i registri di database attivare, è possibile visualizzare il contenuto di tali registri.

Per ulteriori informazioni sull'uso di CloudWatch con i registri di Aurora MySQL, consulta Monitoraggio degli eventi di log in Amazon CloudWatch e Pubblicazione di log Aurora PostgreSQL in Amazon CloudWatch Logs.

Per visualizzare i log per il cluster di database Aurora Serverless v2

1. Aprire la console CloudWatch all'indirizzo https://console.aws.amazon.com/cloudwatch/.
2. Scegli il tuo Regione AWS.
3. Scegli Log groups (Gruppi di log).
4. Seleziona il log del cluster di database Aurora Serverless v2 dall'elenco. Il modello di denominazione dei log è il seguente.

```
/aws/rds/cluster/cluster-name/log_type
```

> **Note**
>
> Per i cluster DB Aurora Serverless v2 compatibili con Aurora MySQL, il log degli errori include gli eventi di dimensionamento del pool di buffer anche in assenza di errori.

Monitoraggio della capacità con Amazon CloudWatch

In Aurora Serverless v2, è possibile utilizzare CloudWatch per monitorare la capacità e l'utilizzo di tutte le istanze DB Aurora Serverless v2 nel cluster. È possibile visualizzare i parametri a livello di istanza per verificare la capacità di ciascuna istanza DB Aurora Serverless v2 durante il dimensionamento verso l'alto e la riduzione verticale. Puoi inoltre confrontare i parametri a livello di capacità con altri parametri per verificare l'impatto delle modifiche dei carichi di lavoro sul consumo di risorse. Ad esempio, puoi confrontare `ServerlessDatabaseCapacity` con `DatabaseUsedMemory`, `DatabaseConnections` e `DMLThroughput` per valutare la risposta del cluster di database durante le operazioni. Per informazioni dettagliate sui parametri relativi alla capacità applicati a Aurora Serverless v2, consulta Parametri di Amazon CloudWatch importanti per Aurora Serverless v2.

Per monitorare la capacità del cluster database Aurora Serverless v2

1. Apri la console CloudWatch all'indirizzo https://console.aws.amazon.com/cloudwatch/.
2. Seleziona Parametri. Tutti i parametri disponibili vengono visualizzati come schede nella console, raggruppate in base al nome del servizio.
3. Seleziona RDS.

4. (Facoltativo) Usa la casella Cerca per trovare i parametri particolarmente importanti per Aurora Serverless v2: `ServerlessDatabaseCapacity`, `ACUUtilization`, `CPUUtilization` e `FreeableMemory`.

Si consiglia di configurare un pannello di controllo di CloudWatch per monitorare la capacità del cluster di database Aurora Serverless v2 utilizzando i parametri relativi alla capacità. Per informazioni sulla procedura, consulta Creazione di pannelli di controllo con CloudWatch.

Per ulteriori informazioni sull'uso di Amazon CloudWatch con Amazon Aurora, consulta Pubblicazione di log Amazon Aurora MySQL in Amazon CloudWatch Logs.

Monitoraggio delle attività di pausa e ripresa Aurora Serverless v2

Aurora scrive un file di log separato per le istanze database Aurora Serverless v2 con la funzionalità di pausa automatica abilitata. Aurora riporta nel log ogni intervallo di 10 minuti durante il quale l'istanza non è in pausa. Aurora conserva fino a sette di questi log, ruotandoli ogni giorno. Il file di log corrente viene denominato `instance.log`, mentre i nomi dei log meno recenti vengono assegnati utilizzando il modello `instance.`*`YYYY-MM-DD`*`.`*`N`*`.log`.

Questo log è abilitato per impostazione predefinita per le istanze database Aurora Serverless v2 con la funzionalità di pausa automatica abilitata. È possibile visualizzare il contenuto di questo log in Console di gestione AWS o utilizzando l'AWS CLI o l'API RDSP. Al momento, questo log non può essere caricato su CloudWatch.

Gli eventi elencati in Eventi di istanza database forniscono una panoramica generale delle attività di pausa e ripresa, ad esempio:

- Inizio e fine della pausa dell'istanza.
- Inizio e fine della ripresa dell'istanza.
- Tentativi dell'istanza di mettersi in pausa, impediti da determinate condizioni.

`instance.log` fornisce informazioni più dettagliate sui motivi per cui un'istanza Aurora Serverless v2 riesce o meno a mettersi in pausa.

Il log può indicare i vari motivi per cui un'istanza viene ripresa:

- Attività dell'utente: una richiesta di connessione al database. Potrebbe provenire da una sessione client interattiva, da una chiamata API dati RDS o da una richiesta di download dei log dall'istanza.

- Attività in background: un'ampia categoria che include tutti i motivi per cui Aurora riprende un'istanza. Ad esempio, quando una richiesta di connessione a un'istanza di lettura causa la ripresa dell'istanza di scrittura, il log dell'istanza di lettura indica l'attività dell'utente, mentre il log dell'istanza di scrittura classifica tale richiesta di ripresa come attività in background. Per scoprire perché Aurora potrebbe riprendere un'istanza diversa rispetto a una richiesta di connessione utente, consulta Ripresa di un'istanza Aurora Serverless v2 in pausa automatica.

Quando Aurora non rileva alcuna condizione che impedisce la messa in pausa dell'istanza alla scadenza dell'intervallo di pausa automatica, scrive periodicamente un messaggio informativo nel log. Per i cluster con una sola istanza database, il log contiene questo messaggio:

```
[INFO] No auto-pause blockers registered since time
```

Per i cluster con più istanze database, il messaggio è leggermente diverso. Questo dipende dal fatto che un'istanza di scrittura potrebbe non riuscire ad attivare la pausa a causa di attività su una qualsiasi istanza di lettura. Se l'attività sull'istanza di lettura termina prima della scadenza dell'intervallo di pausa automatica sull'istanza di scrittura, quest'ultima riesce ad attivare la pausa all'ora prevista.

```
[INFO] No auto-pause blockers registered since time.
Database might be required to maintain compute capacity for high availability.
```

Se un'operazione di messa in pausa viene avviata, ma arriva una nuova richiesta di connessione al database prima che l'istanza completi la sospensione, il log conterrà il seguente messaggio:

```
[INFO] Unable to pause database due to a new database activity
```

Se Aurora rileva condizioni che impediscono completamente la messa in pausa dell'istanza, il log conterrà il seguente messaggio in cui vengono elencate tutte le condizioni:

```
[INFO] Auto-pause blockers registered since time: list_of_conditions
```

In questo modo, Aurora non impedisce di attivare la replica, l'integrazione Zero-ETL, il Database globale Aurora e altro ancora insieme alla funzionalità di pausa automatica. Il log informa l'utente quando l'uso di tali funzionalità potrebbe impedire l'attivazione della pausa automatica.

Di seguito sono riportati i motivi per cui un'istanza Aurora Serverless v2 potrebbe superare l'intervallo di timeout della pausa automatica senza riuscire a entrare in pausa:

- Attività del database prima del timeout della pausa automatica: l'istanza database ha ricevuto una richiesta di connessione prima della scadenza dell'intervallo di timeout.
- Membro del database globale: se il cluster di database fa parte di un Database globale Aurora, le istanze Aurora Serverless v2 nel cluster non entrano in pausa. Un cluster autonomo può diventare un cluster di database globale. Di conseguenza, le istanze che in precedenza erano entrate in pausa automaticamente potrebbero riprendere, riportando questo motivo nel log. Una volta che un cluster diventa membro di un database globale, deve essere scollegato esplicitamente per poter tornare a essere un cluster autonomo. Il cluster primario è ancora considerato parte del database globale anche se si scollegano tutti i cluster secondari.
- Capacità di replica configurata: l'istanza database di scrittura ha abilitato la replica specifica del motore, la replica binlog per MySQL o la replica logica per PostgreSQL. Questa condizione potrebbe essere causata anche dall'utilizzo di altre funzionalità di Aurora che richiedono l'attivazione della replica, ad esempio le integrazioni Zero-ETL o i Flussi di attività del database (DAS).
- Ritardo di backup continuo: se il sistema di archiviazione Aurora non ha completato l'applicazione delle modifiche all'archiviazione fino al momento corrente, l'istanza di scrittura non si ferma finché non recupera il ritardo. Questa condizione riguarda solo l'istanza di scrittura e dovrebbe essere relativamente di breve durata.
- Intervento di assistenza o manutenzione da parte del cliente: se viene avviata un'operazione di manutenzione, l'istanza database non verrà rimessa in pausa fino al termine dell'operazione. Questa condizione include un'ampia varietà di operazioni che potrebbero essere avviate dall'utente o da Aurora, come aggiornamenti, clonazione, modifica delle impostazioni di configurazione, download di file di log e così via. Questo evento si verifica anche quando si richiede di eliminare un'istanza e Aurora riprende brevemente l'istanza nel contesto del meccanismo di eliminazione.
- Problema di comunicazione transitorio: se Aurora non riesce a determinare se la configurazione di dimensionamento attuale ha un'impostazione di capacità minima pari a zero ACU, non mette in pausa l'istanza. Si tratta di un evento in genere molto raro.

Prestazioni e dimensionamento per Aurora Serverless v2

Le seguenti procedure ed esempi mostrano come impostare l'intervallo di capacità per i cluster Aurora Serverless v2 e le istanze database associate. È inoltre possibile utilizzare le procedure seguenti per monitorare il livello di utilizzo delle istanze database. Sarà quindi possibile utilizzare i risultati per determinare se è necessario aumentare o ridurre l'intervallo di capacità.

Prima di utilizzare queste procedure, assicurati di conoscere bene il funzionamento del dimensionamento in Aurora Serverless v2. Il funzionamento del dimensionamento è diverso da quello in Aurora Serverless v1. Per informazioni dettagliate, consultare Dimensionamento di Aurora Serverless v2.

Indice

- Scelta dell'intervallo di capacità di Aurora Serverless v2 per un cluster Aurora
  - Scelta dell'impostazione Aurora Serverless v2 minima della capacità per un cluster
  - Scelta dell'impostazione Aurora Serverless v2 massima della capacità per un cluster
  - Esempio: modificare l'intervallo di capacità Aurora Serverless v2 di un cluster Aurora MySQL
  - Esempio: modificare l'Intervallo di capacità Aurora Serverless v2 di un cluster Aurora PostgreSQL
- Uso di gruppi di parametri per Aurora Serverless v2
  - Valori di parametro predefiniti
  - Numero massimo connessioni per Aurora Serverless v2
  - Parametri che Aurora adegua in caso di riduzione verticale e dimensionamento verso l'alto di Aurora Serverless v2
  - Parametri calcolati da Aurora in base alla capacità massima di Aurora Serverless v2
- Esclusione di errori dovuti all'esaurimento della memoria
- Parametri di Amazon CloudWatch importanti per Aurora Serverless v2
  - Modalità di applicazione dei parametri Aurora Serverless v2 alla fattura AWS
  - Esempi di comandi di CloudWatch per i parametri di Aurora Serverless v2
- Monitoraggio delle prestazioni di Aurora Serverless v2 con Approfondimenti sulle prestazioni
- Risoluzione dei problemi di capacità di Aurora Serverless v2

Scelta dell'intervallo di capacità di Aurora Serverless v2 per un cluster Aurora

Con le istanze database Aurora Serverless v2, è possibile impostare l'intervallo di capacità applicabile a tutte le istanze database nel cluster di database mentre aggiungi la prima Istanza database Aurora Serverless v2 al cluster di database. Per eseguire questa procedura, consulta Impostazione dell'intervallo di capacità di Aurora Serverless v2 per un cluster.

È inoltre possibile modificare l'intervallo di capacità per un cluster esistente. Nelle sezioni seguenti viene descritto più dettagliatamente come scegliere i valori minimi e massimi appropriati e cosa succede quando si modifica l'intervallo di capacità. Ad esempio, la modifica dell'intervallo di capacità può modificare i valori di default di alcuni parametri di configurazione. L'applicazione di tutte le modifiche ai parametri può richiedere il riavvio di ciascuna istanza database Aurora Serverless v2.

Argomenti

- Scelta dell'impostazione Aurora Serverless v2 minima della capacità per un cluster
- Scelta dell'impostazione Aurora Serverless v2 massima della capacità per un cluster
- Esempio: modificare l'intervallo di capacità Aurora Serverless v2 di un cluster Aurora MySQL
- Esempio: modificare l'Intervallo di capacità Aurora Serverless v2 di un cluster Aurora PostgreSQL

Scelta dell'impostazione Aurora Serverless v2 minima della capacità per un cluster

Si potrebbe essere portati a scegliere sempre 0,5 come impostazione minima della capacità per Aurora Serverless v2. Questo valore consente infatti la riduzione verticale massima della capacità dell'istanza database quando è completamente inattiva, pur lasciandola attiva. È inoltre possibile abilitare il comportamento di pausa automatica specificando una capacità minima di 0 ACU, come spiegato in Dimensionamento a zero ACU con pausa e ripresa automatiche per Aurora Serverless v2. Tuttavia, a seconda del modo in cui si utilizza il cluster e delle altre impostazioni configurate, una capacità minima diversa potrebbe essere la più efficace. Considera i seguenti fattori nella scelta dell'impostazione della capacità minima:

- La velocità di dimensionamento per un'istanza database Aurora Serverless v2 dipende dalla sua capacità attuale. Maggiore è la capacità attuale, più veloce sarà il relativo dimensionamento verso l'alto. Se è necessario disporre di un rapido dimensionamento verso l'alto in modo che l'istanza database raggiunga capacità molto elevate, è consigliabile impostare la capacità minima su un valore in cui la velocità di dimensionamento soddisfi le esigenze specifiche.
- Se in genere si modifica la classe delle istanze database in previsione di un carico di lavoro particolarmente elevato o ridotto, è possibile utilizzare tale esperienza per calcolare una stima approssimativa dell'equivalente intervallo di capacità per Aurora Serverless v2. Per determinare la dimensione della memoria da utilizzare in periodi di traffico ridotto, consulta Specifiche hardware per le classi di istanza database per Aurora.

  Ad esempio, supponiamo di utilizzare la classe di istanza database db.r6g.xlarge quando il cluster ha un carico di lavoro ridotto. Questa classe di istanza database dispone di 32 GiB di

memoria. Pertanto, è possibile specificare un'impostazione minima dell'unità di capacità Aurora (ACU) pari a 16 per impostare un'istanza database Aurora Serverless v2 in grado di supportare un dimensionamento verticale avente all'incirca la stessa capacità. Questo perché ogni ACU corrisponde a circa 2 GiB di memoria. È possibile specificare un valore leggermente inferiore per consentire all'istanza database un'ulteriore riduzione verticale in caso di sottoutilizzo dell'istanza databse db.r6g.xlarge.

- Se l'applicazione funziona in modo più efficiente quando le istanze database contengono una certa quantità di dati nella cache del buffer, valuta la possibilità di specificare un'impostazione ACU minima in cui la memoria è sufficientemente grande da contenere i dati a cui si accede più di frequente. In caso contrario, alcuni dati vengono cancellati dalla cache del buffer quando le istanze database Aurora Serverless v2 vengono ridotte verticalmente a una dimensione di memoria inferiore. Quando le istanze database vengono successivamente dimensionate verso l'alto, le informazioni vengono lette nuovamente nella cache del buffer. Se la quantità di I/O per riportare i dati nella cache del buffer è rilevante, potrebbe essere più efficace scegliere un valore ACU minimo più alto.
- Se per la maggior parte del tempo le istanze database Aurora Serverless v2 vengono eseguite a una determinata capacità, valuta la possibilità di specificare un'impostazione di capacità minima inferiore, ma non di molto, a quella di base. Le istanze database Aurora Serverless v2 possono calcolare in modo più efficace una stima della quantità e velocità del dimensionamento verso l'alto (aumento) quando la capacità corrente non è sensibilmente inferiore alla capacità richiesta.
- Se il carico di lavoro con provisioning ha requisiti di memoria troppo elevati per classi di istanze database di piccole dimensioni come T3 o T4g, scegli un'impostazione ACU minima che fornisca memoria paragonabile a un'istanza database R5 o R6g.

  In particolare, raccomandiamo la seguente capacità minima per l'utilizzo con le caratteristiche specificate (queste raccomandazioni sono soggette a modifiche):

  - Approfondimenti sulle prestazioni: 2 ACU
  - Database globali Aurora: 8 ACU (si applica solo alla Regione AWS principale)
- In Aurora, la replica avviene a livello di archiviazione, quindi la capacità in lettura non influisce direttamente sulla replica. Tuttavia, per le istanze database di lettura Aurora Serverless v2 con dimensionamento indipendente, assicurati che la capacità minima sia sufficiente per gestire i carichi di lavoro durante i periodi di intensa attività di scrittura per evitare la latenza delle query. Se le istanze database di lettura che si trovano nei livelli di promozione 2-15 presentano problemi di prestazioni, è consigliabile aumentare la capacità minima del cluster. Per informazioni dettagliate in merito alla decisione se le istanze database di lettura vengono dimensionate assieme alle istanze

di scrittura oppure in modo indipendente, consulta Scelta del livello di promozione per un'istanza Aurora Serverless v2 di lettura.

- Se disponi di un cluster di database con istanze database di lettura Aurora Serverless v2, le istanze di lettura non si dimensionano insieme all'istanza database di scrittura quando il livello di promozione delle istanze di lettura non è 0 o 1. In tal caso, l'impostazione di una capacità minima ridotta può comportare un eccessivo ritardo nella replica. Ciò è dovuto al fatto che le istanze di lettura potrebbero non avere capacità sufficiente per applicare le modifiche dell'istanza di scrittura quando il database è occupato. Si consiglia di impostare la capacità minima su un valore che rappresenti una quantità comparabile di memoria e CPU rispetto all'istanza database di scrittura.
- Il valore del parametro `max_connections` per le istanze database Aurora Serverless v2 si basa sulla dimensione della memoria derivata dal numero massimo di ACU. Se specifichi una capacità minima di 0 o 0,5 ACU per le istanze database compatibili con PostgreSQL, il valore massimo di `max_connections` è limitato a 2.000.

  Se vuoi utilizzare il cluster Aurora PostgreSQL per un carico di lavoro con un numero elevato di connessioni, è consigliabile utilizzare un'impostazione ACU minima pari a 1 o superiore. Per informazioni dettagliate su come Aurora Serverless v2 gestisce il parametro di configurazione `max_connections`, consulta Numero massimo connessioni per Aurora Serverless v2.
- Il tempo necessario a un'istanza database Aurora Serverless v2 per il dimensionamento dalla capacità minima alla capacità massima dipende dalla differenza tra i valori ACU minimi e massimi. Quando la capacità corrente dell'istanza database è elevata, il dimensionamento verso l'alto di Aurora Serverless v2 avviene con incrementi più grandi rispetto a quando l'istanza database inizia da una capacità ridotta. Pertanto, se specifichi una capacità massima relativamente grande e l'istanza database rimane per la maggior parte del tempo vicino a tale capacità, è consigliabile aumentare l'impostazione ACU minima. In questo modo, un'istanza database inattiva può usufruire del dimensionamento verso l'alto e raggiungere più rapidamente la capacità massima.

Scelta dell'impostazione Aurora Serverless v2 massima della capacità per un cluster

Si potrebbe essere tentati di scegliere sempre un valore elevato per l'impostazione della capacità Aurora Serverless v2 massima. Un valore elevato per la capacità massima consente all'istanza database un dimensionamento massimo verso l'alto quando è in esecuzione un carico di lavoro con un uso intensivo di risorse. Un valore basso evita il rischio di dover sostenere di addebiti imprevisti. A seconda di come si utilizza il cluster e delle altre impostazioni configurate, il valore più efficace potrebbe essere maggiore o minore basso rispetto a quello originariamente individuato. Considera i seguenti fattori nella scelta dell'impostazione della capacità massima:

- La capacità massima deve essere almeno pari alla capacità minima. Puoi impostare lo stesso valore per la capacità minima e la capacità massima. Tuttavia, in questo caso la capacità non aumenta né diminuisce mai. Pertanto, a parte gli scenari di test, l'uso di valori identici per la capacità minima e massima non è appropriato.
- La capacità massima deve essere maggiore di 0,5 ACU. Nella maggior parte dei casi è possibile impostare la capacità minima e massima su un valore identico. Tuttavia, non è possibile specificare 0,5 sia per la capacità minima che per la la capacità massima. Utilizza un valore pari o superiore a 1 per la capacità massima.
- Se in genere modifichi la classe delle istanze database in previsione di un carico di lavoro particolarmente elevato o ridotto, è possibile utilizzare tale esperienza per calcolare la stima dell'equivalente intervallo di capacità Aurora Serverless v2. Per determinare la dimensione della memoria da utilizzare in periodi di traffico elevato, consulta Specifiche hardware per le classi di istanza database per Aurora.

  Ad esempio, supponiamo di utilizzare la classe di istanza database db.r6g.4xlarge quando il cluster ha un carico di lavoro elevato. Questa classe di istanza database dispone di 128 GiB di memoria. Pertanto, è possibile specificare un'impostazione ACU massima di 64 per impostare un'istanza database Aurora Serverless v2 in grado di eseguire un dimensionamento verso l'alto fino a circa la stessa capacità. Questo perché ogni ACU corrisponde a circa 2 GiB di memoria. È possibile specificare un valore leggermente più alto per consentire all'istanza database un ulteriore dimensionamento verso l'alto nel caso in cui l'istanza database db.r6g.4xlarge a volte non abbia capacità sufficiente per gestire efficacemente il carico di lavoro.
- Se hai un limite di budget per l'utilizzo del database, scegli un valore compreso in tale limite anche se tutte le istanze database Aurora Serverless v2 vengono eseguite sempre alla capacità massima. Ricorda che quando hain istanze database Aurora Serverless v2 nel cluster, la capacità Aurora Serverless v2 massima teorica che il cluster può consumare in qualsiasi momento è nvolte l'impostazione ACU massima per il cluster. La quantità effettiva consumata potrebbe essere inferiore, ad esempio se alcune istanze di lettura vengono dimensionate in modo indipendente rispetto all'istanza di scrittura.
- Se si utilizzano istanze database Aurora Serverless v2 di lettura per alleggerire parte del carico di lavoro di sola lettura dell'istanza database di scrittura, è possibile scegliere un'impostazione di capacità massima inferiore. Si procede in questo senso se il dimensionamento di ogni istanza database di lettura non deve raggiungere un valore così alto, come se il cluster contenga solo un'unica istanza database.

- Supponiamo di voler evitare un uso eccessivo a causa di parametri del database configurati in modo errato o di query inefficienti nell'applicazione. In tal caso, è possibile evitare un uso eccessivo accidentale scegliendo un'impostazione di capacità massima inferiore a quella più alta possibile.
- Se i picchi dovuti alla reale attività degli utenti sono rari ma si verificano, è possibile tenerne conto quando si sceglie l'impostazione della capacità massima. Se è prioritario che l'applicazione continui a funzionare con prestazioni e scalabilità complete, è possibile specificare un'impostazione di capacità massima superiore a quella osservata durante il normale utilizzo. Se l'applicazione può essere eseguita con una capacità di throughput effettiva ridotta durante picchi di attività molto estremi, è possibile scegliere un'impostazione di capacità massima leggermente inferiore. Assicurati di scegliere un'impostazione che disponga ancora di memoria e risorse CPU sufficienti per mantenere l'applicazione in esecuzione.
- Se attivi impostazioni nel cluster che aumentano l'utilizzo della memoria per ogni istanza database, considera tale capacità di memoria quando decidi il valore ACU massimo. Tali impostazioni includono quelle per Approfondimenti sulle prestazioni, le query parallele Aurora MySQL, lo schema di prestazioni Aurora MySQL e la replica del registro binario Aurora MySQL. Assicurati che il valore ACU massimo consenta un dimensionamento verso l'alto delle istanze di database Aurora Serverless v2 tale da gestire il carico di lavoro quando vengono utilizzate queste funzionalità. Per informazioni sulla risoluzione dei problemi causati dalla combinazione di un'impostazione ACU massima bassa e le funzionalità Aurora che impongono il sovraccarico della memoria, consulta Esclusione di errori dovuti all'esaurimento della memoria.

Esempio: modificare l'intervallo di capacità Aurora Serverless v2 di un cluster Aurora MySQL

Il seguente esempio per AWS CLI illustra come aggiornare l'intervallo ACU per le istanze database Aurora Serverless v2 in un cluster Aurora MySQL esistente. Inizialmente, l'intervallo di capacità per il cluster è compreso tra 8 e 32 ACU.

```
aws rds describe-db-clusters --db-cluster-identifier serverless-v2-cluster \
  --query 'DBClusters[*].ServerlessV2ScalingConfiguration|[0]'
{
    "MinCapacity": 8.0,
    "MaxCapacity": 32.0
}
```

L'istanza database è inattiva e ridotta fino a 8 ACU. A questo punto, le seguenti impostazioni relative alla capacità si applicano all'istanza database. Per rappresentare la dimensione del pool di buffer in

unità facilmente leggibili, si divide questo valore per 2 elevato alla 30, ottenendo in questo modo una misurazione in gibibyte (GiB). Ciò è dovuto al fatto che le misurazioni relative alla memoria per Aurora utilizzano unità basate su potenze di 2, non potenze di 10.

```
mysql> select @@max_connections;
+-------------------+
| @@max_connections |
+-------------------+
|              3000 |
+-------------------+
1 row in set (0.00 sec)

mysql> select @@innodb_buffer_pool_size;
+---------------------------+
| @@innodb_buffer_pool_size |
+---------------------------+
|                9294577664 |
+---------------------------+
1 row in set (0.00 sec)

mysql> select @@innodb_buffer_pool_size / pow(2,30) as gibibytes;
+-----------+
| gibibytes |
+-----------+
|   8.65625 |
+-----------+
1 row in set (0.00 sec)
```

Successivamente, si modifica l'intervallo di capacità per il cluster. Al termine dell'esecuzione del comando `modify-db-cluster`, l'intervallo ACU per il cluster è 12.5-80.

```
aws rds modify-db-cluster --db-cluster-identifier serverless-v2-cluster \
  --serverless-v2-scaling-configuration MinCapacity=12.5,MaxCapacity=80

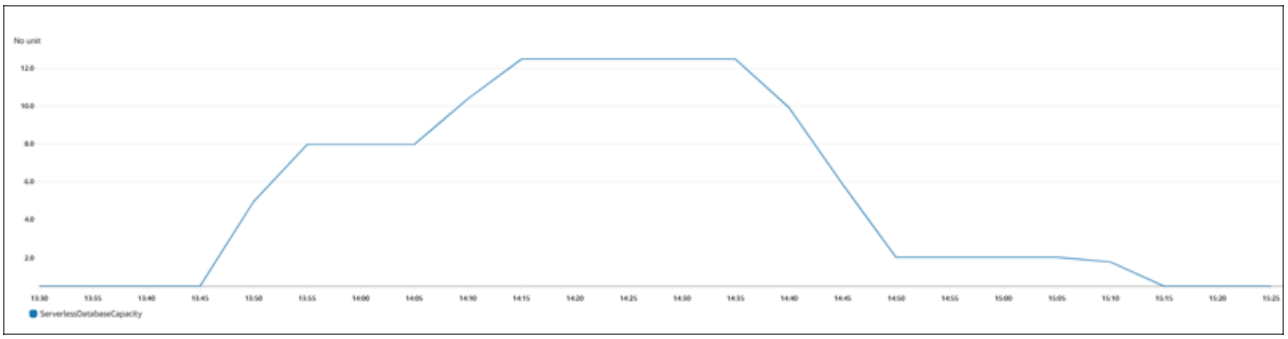
aws rds describe-db-clusters --db-cluster-identifier serverless-v2-cluster \
  --query 'DBClusters[*].ServerlessV2ScalingConfiguration|[0]'
{
    "MinCapacity": 12.5,
    "MaxCapacity": 80.0
}
```

La modifica dell'intervallo di capacità ha causato modifiche ai valori di default di alcuni parametri di configurazione. Aurora può applicare immediatamente alcuni di questi nuovi valori di default. Tuttavia, alcune modifiche ai parametri hanno effetto solo dopo il riavvio. Lo stato `pending-reboot` indica che è necessario un riavvio per applicare alcune modifiche ai parametri.

```
aws rds describe-db-clusters --db-cluster-identifier serverless-v2-cluster \
  --query '*[].{DBClusterMembers:DBClusterMembers[*].
{DBInstanceIdentifier:DBInstanceIdentifier,DBClusterParameterGroupStatus:DBClusterParameterGrou
[0]'
{
    "DBClusterMembers": [
        {
            "DBInstanceIdentifier": "serverless-v2-instance-1",
            "DBClusterParameterGroupStatus": "pending-reboot"
        }
    ]
}
```

A questo punto, il cluster è inattivo e l'istanza database `serverless-v2-instance-1` sta consumando 12,5 ACU. Il parametro `innodb_buffer_pool_size` è già modificato in base alla capacità corrente dell'istanza database. Il parametro `max_connections` riflette ancora il valore della capacità massima precedente. La reimpostazione di questo valore richiede il riavvio dell'istanza database.

> **Note**
>
> Se imposti il parametro `max_connections` direttamente in un gruppo di parametri database personalizzato, non è necessario il riavvio.

```
mysql> select @@max_connections;
+-------------------+
| @@max_connections |
+-------------------+
|              3000 |
+-------------------+
1 row in set (0.00 sec)

mysql> select @@innodb_buffer_pool_size;
+---------------------------+
| @@innodb_buffer_pool_size |
```

```
+---------------------------+
|               15572402176 |
+---------------------------+
1 row in set (0.00 sec)

mysql> select @@innodb_buffer_pool_size / pow(2,30) as gibibytes;
+---------------+
| gibibytes     |
+---------------+
| 14.5029296875 |
+---------------+
1 row in set (0.00 sec)
```

L'istanza viene riavviata e si attende che torni di nuovo disponibile.

```
aws rds reboot-db-instance --db-instance-identifier serverless-v2-instance-1
{
  "DBInstanceIdentifier": "serverless-v2-instance-1",
  "DBInstanceStatus": "rebooting"
}

aws rds wait db-instance-available --db-instance-identifier serverless-v2-instance-1
```

Lo stato `pending-reboot` viene cancellato. Il valore `in-sync` conferma che Aurora ha applicato tutte le modifiche dei parametri in sospeso.

```
aws rds describe-db-clusters --db-cluster-identifier serverless-v2-cluster \
  --query '*[].{DBClusterMembers:DBClusterMembers[*].
{DBInstanceIdentifier:DBInstanceIdentifier,DBClusterParameterGroupStatus:DBClusterParameterGrou
[0]'
{
    "DBClusterMembers": [
        {
            "DBInstanceIdentifier": "serverless-v2-instance-1",
            "DBClusterParameterGroupStatus": "in-sync"
        }
    ]
}
```

Il parametro `innodb_buffer_pool_size` è aumentato fino alla sua dimensione finale per un'istanza database inattiva. Il parametro `max_connections` è aumentato in base a un valore

derivato dal valore ACU massimo. La formula usata da Aurora per `max_connections` comporta un aumento di 1.000 quando la dimensione della memoria raddoppia.

```
mysql> select @@innodb_buffer_pool_size;
+---------------------------+
| @@innodb_buffer_pool_size |
+---------------------------+
|               16139681792 |
+---------------------------+
1 row in set (0.00 sec)

mysql> select @@innodb_buffer_pool_size / pow(2,30) as gibibytes;
+-----------+
| gibibytes |
+-----------+
|  15.03125 |
+-----------+
1 row in set (0.00 sec)

mysql> select @@max_connections;
+-------------------+
| @@max_connections |
+-------------------+
|              4000 |
+-------------------+
1 row in set (0.00 sec)
```

L’intervallo di capacità impostato è compreso tra 0,5 e 128 ACU e l’istanza database viene riavviata. A questo punto, l'istanza database inattiva ha una dimensione della cache del buffer inferiore a 1 GiB, quindi la misurazione viene eseguita in mebibyte (MiB). Il valore di 5000 del parametro `max_connections` è derivato dalla dimensione della memoria dell'impostazione di capacità massima.

```
mysql> select @@innodb_buffer_pool_size / pow(2,20) as mebibytes, @@max_connections;
+-----------+-------------------+
| mebibytes | @@max_connections |
+-----------+-------------------+
|       672 |              5000 |
+-----------+-------------------+
1 row in set (0.00 sec)
```

Esempio: modificare l'Intervallo di capacità Aurora Serverless v2 di un cluster Aurora PostgreSQL

L'esempio per l'interfaccia della linea di comando seguente illustra come aggiornare l'intervallo ACU per le istanze database Aurora Serverless v2 in un cluster Aurora PostgreSQL esistente.

1. L'intervallo di capacità del cluster iniziale è compreso tra 0,5 e 1 ACU.
2. Cambia l'intervallo di capacità compreso tra 8 e 32 ACU.
3. Cambia l'intervallo di capacità compreso tra 12,5 e 80 ACU.
4. Cambia l'intervallo di capacità compreso tra 0,5 e 128 ACU.
5. Riporta la capacità all'intervallo iniziale compreso tra 0,5 e 1 ACU.

La figura seguente mostra le modifiche della capacità in Amazon CloudWatch.



L'istanza database è inattiva e ridotta fino a 0,5 ACU. A questo punto, le seguenti impostazioni relative alla capacità si applicano all'istanza database.

```
postgres=> show max_connections;
 max_connections
-----------------
 189
(1 row)

postgres=> show shared_buffers;
 shared_buffers
----------------
 16384
(1 row)
```

Successivamente, si modifica l'intervallo di capacità per il cluster. Al termine dell'esecuzione del comando `modify-db-cluster`, l'intervallo ACU per il cluster è 8.0-32.

```
aws rds describe-db-clusters --db-cluster-identifier serverless-v2-cluster \
  --query 'DBClusters[*].ServerlessV2ScalingConfiguration|[0]'
{
    "MinCapacity": 8.0,
    "MaxCapacity": 32.0
}
```

La modifica dell'intervallo di capacità comporta modifiche ai valori di default di alcuni parametri di configurazione. Aurora può applicare immediatamente alcuni di questi nuovi valori di default. Tuttavia, alcune modifiche ai parametri hanno effetto solo dopo il riavvio. Lo stato `pending-reboot` indica che è necessario un riavvio per applicare le modifiche ad alcuni parametri.

```
aws rds describe-db-clusters --db-cluster-identifier serverless-v2-cluster \
  --query '*[].{DBClusterMembers:DBClusterMembers[*].
{DBInstanceIdentifier:DBInstanceIdentifier,DBClusterParameterGroupStatus:DBClusterParameterGrou
[0]'
{
    "DBClusterMembers": [
        {
            "DBInstanceIdentifier": "serverless-v2-instance-1",
            "DBClusterParameterGroupStatus": "pending-reboot"
        }
    ]
}
```

A questo punto, il cluster è inattivo e l'istanza database `serverless-v2-instance-1` sta consumando 8,0 ACU. Il parametro `shared_buffers` è già modificato in base alla capacità corrente dell'istanza database. Il parametro `max_connections` riflette ancora il valore della capacità massima precedente. La reimpostazione di questo valore richiede il riavvio dell'istanza database.

> **Note**
>
> Se imposti il parametro `max_connections` direttamente in un gruppo di parametri database personalizzato, non è necessario il riavvio.

```
postgres=> show max_connections;
 max_connections
-----------------
```

```
 189
(1 row)

postgres=> show shared_buffers;
 shared_buffers
----------------
 1425408
(1 row)
```

L'istanza database viene riavviata e si attende che torni di nuovo disponibile.

```
aws rds reboot-db-instance --db-instance-identifier serverless-v2-instance-1
{
  "DBInstanceIdentifier": "serverless-v2-instance-1",
  "DBInstanceStatus": "rebooting"
}

aws rds wait db-instance-available --db-instance-identifier serverless-v2-instance-1
```

Dopo il riavvio dell'istanza, lo stato `pending-reboot` è cancellato. Il valore `in-sync` conferma che Aurora ha applicato tutte le modifiche dei parametri in sospeso.

```
aws rds describe-db-clusters --db-cluster-identifier serverless-v2-cluster \
  --query '*[].{DBClusterMembers:DBClusterMembers[*].
{DBInstanceIdentifier:DBInstanceIdentifier,DBClusterParameterGroupStatus:DBClusterParameterGrou
[0]'
{
    "DBClusterMembers": [
        {
            "DBInstanceIdentifier": "serverless-v2-instance-1",
            "DBClusterParameterGroupStatus": "in-sync"
        }
    ]
}
```

Dopo il riavvio, `max_connections` mostra il valore della nuova capacità massima.

```
postgres=> show max_connections;
 max_connections
-----------------
 5000
```

```
(1 row)

postgres=> show shared_buffers;
 shared_buffers
----------------
 1425408
(1 row)
```

Successivamente, si imposta l'intervallo di capacità per il cluster compreso tra 12,5 e 80 ACU.

```
aws rds modify-db-cluster --db-cluster-identifier serverless-v2-cluster \
  --serverless-v2-scaling-configuration MinCapacity=12.5,MaxCapacity=80

aws rds describe-db-clusters --db-cluster-identifier serverless-v2-cluster \
  --query 'DBClusters[*].ServerlessV2ScalingConfiguration|[0]'
{
    "MinCapacity": 12.5,
    "MaxCapacity": 80.0
}
```

A questo punto, il cluster è inattivo e l'istanza database `serverless-v2-instance-1` sta consumando 12,5 ACU. Il parametro `shared_buffers` è già modificato in base alla capacità corrente dell'istanza database. Il valore `max_connections` è ancora 5000.

```
postgres=> show max_connections;
 max_connections
-----------------
 5000
(1 row)

postgres=> show shared_buffers;
 shared_buffers
----------------
 2211840
(1 row)
```

Si esegue nuovamente il riavvio, ma i valori del parametro rimangono invariati. Questo perché `max_connections` ha un valore massimo di 5000 per un cluster di database Aurora Serverless v2 che esegue Aurora PostgreSQL.

```
postgres=> show max_connections;
```

```
 max_connections
-----------------
 5000
(1 row)

postgres=> show shared_buffers;
 shared_buffers
----------------
 2211840
(1 row)
```

Ora impostiamo l'intervallo di capacità compreso tra 0,5 e 128 ACU. Il cluster di database viene ridotto a 10 ACU e quindi a 2. Riavviamo l'istanza database.

```
postgres=> show max_connections;
 max_connections
-----------------
 2000
(1 row)

postgres=> show shared_buffers;
 shared_buffers
----------------
 16384
(1 row)
```

Il valore di `max_connections` per le istanze database Aurora Serverless v2 si basa sulla dimensione della memoria derivata dal numero massimo di ACU. Se specifichi una capacità minima di 0 o 0,5 ACU per le istanze database compatibili con PostgreSQL, il valore massimo di `max_connections` è limitato a 2.000.

Ora riportiamo la capacità all'intervallo iniziale compreso tra 0,5 e 1 ACU e riavviamo l'istanza database. Il parametro `max_connections` è tornato al suo valore originale.

```
postgres=> show max_connections;
 max_connections
-----------------
 189
(1 row)

postgres=> show shared_buffers;
 shared_buffers
```

```
----------------
 16384
(1 row)
```

Uso di gruppi di parametri per Aurora Serverless v2

Quando crei il tuo cluster di database Aurora Serverless v2, è possibile scegliere un motore del database Aurora specifico e un gruppo di parametri del cluster di database associato. Se non si ha familiarità con il modo in cui Aurora utilizza i gruppi di parametri per applicare le impostazioni di configurazione in modo coerente tra cluster, consulta Gruppi di parametri per Amazon Aurora. Tutte queste procedure per la creazione, la modifica, l'applicazione e altre azioni per i gruppi di parametri sono valide per Aurora Serverless v2.

La funzione del gruppo di parametri funziona in genere allo stesso modo tra cluster con provisioning e cluster contenenti istanze database Aurora Serverless v2:

- Il gruppo di parametri del cluster di database include anche i valori di default per tutti i parametri a livello di istanza.
- È possibile sovrascrivere alcuni parametri per istanze database specifiche specificando un gruppo personalizzato di parametri del database per tali istanze database. È possibile eseguire questa operazione durante il debug o l'ottimizzazione delle prestazioni per istanze database specifiche. Ad esempio, supponiamo di avere un cluster contenente alcune istanze database Aurora Serverless v2 e alcune istanze database con provisioning. In questo caso, è possibile specificare alcuni parametri diversi per le istanze database con provisioning utilizzando un gruppo personalizzato di parametri del database.
- Per Aurora Serverless v2, è possibile utilizzare tutti i parametri con il valore `provisioned` nell'attributo `SupportedEngineModes` nel gruppo di parametri. In Aurora Serverless v1, è possibile utilizzare solo il sottoinsieme di parametri con `serverless` nellì'attributo `SupportedEngineModes`.

Argomenti

- Valori di parametro predefiniti
- Numero massimo connessioni per Aurora Serverless v2
- Parametri che Aurora adegua in caso di riduzione verticale e dimensionamento verso l'alto di Aurora Serverless v2
- Parametri calcolati da Aurora in base alla capacità massima di Aurora Serverless v2

Valori di parametro predefiniti

La sostanziale differenza tra le istanze database con provisioning e le istanze database Aurora Serverless v2 risiede nel fatto che Aurora sovrascrive tutti i valori dei parametri personalizzati per determinati parametri correlati alla capacità dell'istanza database. I valori dei parametri personalizzati si applicano ancora a tutte le istanze database con provisioning nel cluster. Per ulteriori informazioni su come le istanze database Aurora Serverless v2 interpretano i parametri dei gruppi di parametri Aurora, consulta Parametri di configurazione per i cluster Aurora. Per i parametri specifici sostituiti da Aurora Serverless v2, consulta Parametri che Aurora adegua in caso di riduzione verticale e dimensionamento verso l'alto di Aurora Serverless v2 e Parametri calcolati da Aurora in base alla capacità massima di Aurora Serverless v2.

È possibile ottenere un elenco di valori di default per i gruppi di parametri di default per i vari motori database Aurora utilizzando il comando della CLI describe-engine-default-cluster-parameters e interrogando la regione Regione AWS. Di seguito sono riportati i valori che è possibile utilizzare per le opzioni `--db-parameter-group-family` e `-db-parameter-group-name` per le versioni del motore compatibili con Aurora Serverless v2.

Motore del database e versione	Famiglia di gruppi di parametri	Nome del gruppo di parametri di default
Aurora MySQL versione 3	`aurora-mysql8.0`	`default.aurora-mysql8.0`
Aurora PostgreSQL versione 13.x	`aurora-postgresql13`	`default.aurora-postgresql13`
Aurora PostgreSQL versione 14.x	`aurora-postgresql14`	`default.aurora-postgresql14`
Aurora PostgreSQL versione 15.x	`aurora-postgresql15`	`default.aurora-postgresql15`
Aurora PostgreSQL versione 16.x	`aurora-postgresql16`	`default.aurora-postgresql16`
Aurora PostgreSQL versione 17.x	`aurora-postgresql17`	`default.aurora-postgresql17`

Nell'esempio seguente viene recuperato un elenco di parametri dal gruppo di cluster di database di default per Aurora MySQL versione 3 e Aurora PostgreSQL 13. Queste sono le versioni di Aurora MySQL e Aurora PostgreSQL usate con Aurora Serverless v2.

Per Linux, macOS o Unix:

```
aws rds describe-db-cluster-parameters \
  --db-cluster-parameter-group-name default.aurora-mysql8.0 \
  --query 'Parameters[*].
{ParameterName:ParameterName,SupportedEngineModes:SupportedEngineModes} |
    [?contains(SupportedEngineModes, `provisioned`) == `true`] | [*].[ParameterName]' \
  --output text

aws rds describe-db-cluster-parameters \
  --db-cluster-parameter-group-name default.aurora-postgresql13 \
  --query 'Parameters[*].
{ParameterName:ParameterName,SupportedEngineModes:SupportedEngineModes} |
    [?contains(SupportedEngineModes, `provisioned`) == `true`] | [*].[ParameterName]' \
  --output text
```

Per Windows:

```
aws rds describe-db-cluster-parameters ^
  --db-cluster-parameter-group-name default.aurora-mysql8.0 ^
  --query 'Parameters[*].
{ParameterName:ParameterName,SupportedEngineModes:SupportedEngineModes} |
    [?contains(SupportedEngineModes, `provisioned`) == `true`] | [*].[ParameterName]' ^
  --output text

aws rds describe-db-cluster-parameters ^
  --db-cluster-parameter-group-name default.aurora-postgresql13 ^
  --query 'Parameters[*].
{ParameterName:ParameterName,SupportedEngineModes:SupportedEngineModes} |
    [?contains(SupportedEngineModes, `provisioned`) == `true`] | [*].[ParameterName]' ^
  --output text
```

Numero massimo connessioni per Aurora Serverless v2

Per Aurora MySQL e Aurora PostgreSQL, le istanze database Aurora Serverless v2 contengono la costante del parametro `max_connections` in modo che le connessioni non vengano interrotte in caso di riduzione verticale dell'istanza database. Il valore predefinito per questo parametro è derivato

da una formula che si basa sulle dimensioni della memoria dell'istanza database. Per informazioni dettagliate sulla formula e sui valori di default per le classi di istanza database con provisioning, consulta Numero massimo di connessioni a un'istanza database Aurora MySQL e Numero massimo di connessioni a un'istanza database Aurora PostgreSQL.

Quando valuta la formula, Aurora Serverless v2 utilizza la dimensione della memoria in base alle unità di capacità Aurora (ACU) massime per l'istanza database e non il valore ACU corrente. Se modifichi il valore di default, si consiglia di utilizzare una variante della formula anziché specificare un valore costante. In questo modo, Aurora Serverless v2 può utilizzare un'impostazione adeguata in base alla capacità massima.

Quando si modifica la capacità massima di un cluster di database Aurora Serverless v2, è necessario riavviare le istanze database Aurora Serverless v2 per aggiornare il valore `max_connections`. Questo perché `max_connections` è un parametro statico per Aurora Serverless v2.

La tabella seguente mostra i valori predefiniti di `max_connections` per Aurora Serverless v2 in base al valore ACU massimo.

Numero massimo di ACU	Numero massimo di connessioni predefinito in Aurora MySQL	Numero massimo di connessioni predefinito in Aurora PostgreSQL
1	90	189
4	135	823
8	1.000	1.669
16	2.000	3.360
32	3.000	5.000
64	4.000	5.000
128	5.000	5.000
192	6.000	5.000
256	6.000	5.000

> **Note**
>
> Il valore di `max_connections` per le istanze database Aurora Serverless v2 si basa sulla dimensione della memoria derivata dal numero massimo di ACU. Se specifichi una capacità minima di 0 o 0,5 ACU per le istanze database compatibili con PostgreSQL, il valore massimo di `max_connections` è limitato a 2.000.

Per esempi specifici che mostrano come `max_connections` cambia con il valore ACU massimo, consulta Esempio: modificare l'intervallo di capacità Aurora Serverless v2 di un cluster Aurora MySQL e Esempio: modificare l'Intervallo di capacità Aurora Serverless v2 di un cluster Aurora PostgreSQL.

Parametri che Aurora adegua in caso di riduzione verticale e dimensionamento verso l'alto di Aurora Serverless v2

Quando viene eseguita la scalabilità automatica, Aurora Serverless v2 deve essere in grado di modificare i parametri affinché ciascuna istanza database funzioni in modo ottimale in base all'incremento o al decremento della capacità. Pertanto, non è possibile sovrascrivere alcuni parametri relativi alla capacità. Per alcuni parametri che è possibile sovrascrivere, evita di codificare valori fissi. Le seguenti considerazioni si applicano a queste impostazioni relative alla capacità.

Per Aurora MySQL, Aurora Serverless v2 esegue il dimensionamento dinamico di alcuni parametri durante il dimensionamento. Per i seguenti parametri, Aurora Serverless v2 non utilizza i valori di parametro personalizzati specificati:

- `innodb_buffer_pool_size`
- `innodb_purge_threads`
- `table_definition_cache`
- `table_open_cache`

Per Aurora PostgreSQL, Aurora Serverless v2 esegue il dimensionamento dinamico del seguente parametro durante il dimensionamento. Per i seguenti parametri, Aurora Serverless v2 non utilizza i valori di parametro personalizzati specificati:

- `shared_buffers`

Per tutti i parametri diversi da quelli elencati qui, le istanze database Aurora Serverless v2 hanno lo stesso funzionamento delle istanze database con provisioning. Il valore di default del parametro viene ereditato dal gruppo di parametri del cluster. È possibile modificare il valore di default per l'intero cluster utilizzando un gruppo di parametri personalizzato per il cluster. In alternativa, è possibile modificare il valore di default per alcune istanze database utilizzando un gruppo di parametri personalizzato per il database. I parametri dinamici vengono aggiornati immediatamente. Le modifiche apportate ai parametri statici hanno effetto solo dopo il riavvio dell'istanza database.

Parametri calcolati da Aurora in base alla capacità massima di Aurora Serverless v2

Per i seguenti parametri, Aurora PostgreSQL utilizza i valori predefiniti derivati dalle dimensioni della memoria in base all'impostazione ACU massima, in modo analogo a quanto avviene con `max_connections`:

- `autovacuum_max_workers`
- `autovacuum_vacuum_cost_limit`
- `autovacuum_work_mem`
- `effective_cache_size`
- `maintenance_work_mem`

Esclusione di errori dovuti all'esaurimento della memoria

Se una delle istanze database Aurora Serverless v2 raggiunge costantemente il limite della capacità massima, Aurora indica questa condizione impostando l'istanza datanase sullo stato `incompatible-parameters`. Se lo stato dell'istanza database è `incompatible-parameters`, alcune operazioni sono bloccate. Ad esempio, non è possibile aggiornare la versione del motore.

In genere, l'istanza database entra in questo stato quando si riavvia frequentemente a causa di errori di memoria insufficiente. Aurora registra un evento quando si verifica questo tipo di riavvio. È possibile visualizzare l'evento seguendo la procedura descritta in Visualizzazione di eventi Amazon RDS. Un utilizzo della memoria insolitamente elevato può verificarsi a causa del sovraccarico dovuto all'attivazione di impostazioni come Approfondimenti sulle prestazioni e l'autenticazione IAM. Può anche essere causato da un carico di lavoro rilevante sull'istanza database o dalla gestione dei metadati associati a un gran numero di oggetti dello schema.

Se il sovraccarico sulla memoria diminuisce e pertanto l'istanza database non raggiunge la sua capacità massima molto spesso, Aurora imposta automaticamente lo stato dell'istanza database su `available`.

Per gestire in modo ottimale questa situazione, è possibile eseguire alcune o tutte le seguenti azioni:

- Aumentare il limite inferiore della capacità per le istanze database Aurora Serverless v2 modificando il valore minimo dell'unità di capacità Aurora (ACU) per il cluster. In questo modo si evitano i problemi in cui un database inattivo viene ridotto a una capacità di memoria inferiore a quella necessaria per le funzionalità attivate nel cluster. Dopo aver modificato le impostazioni ACU per il cluster, riavviare l'istanza database Aurora Serverless v2. In questo modo Aurora valuta se è in grado di ripristinare lo stato `available`.
- Aumentare il limite superiore della capacità per le istanze database Aurora Serverless v2 modificando il valore ACU massimo per il cluster. Ciò consente di evitare problemi nel caso in cui un database occupato non sia in grado di eseguire il dimensionamento verso l'alto fino a una capacità con memoria sufficiente per le funzionalità attivate nel cluster e nel carico di lavoro del database. Dopo aver modificato le impostazioni ACU per il cluster, riavviare l'istanza database Aurora Serverless v2. In questo modo Aurora valuta se è in grado di ripristinare lo stato `available`.
- Disattivare le impostazioni di configurazione che richiedono un sovraccarico di memoria. Ad esempio, supponiamo che funzionalità quali, ad esempio, AWS Identity and Access Management (IAM), Approfondimenti sulle prestazioni o la replica dei registri binari di Aurora MySQL, siano attivate ma non vengano utilizzate. In questo caso, è possibile disattivarle. In alternativa, è possibile adeguare i valori di capacità minima e massima per il cluster in modo da considerare la memoria utilizzata da tali funzionalità. Per le linee guida sulla scelta delle impostazioni di capacità minima e massima, consulta Scelta dell'intervallo di capacità di Aurora Serverless v2 per un cluster Aurora.
- Ridurre il carico di lavoro sull'istanza database. Ad esempio, è possibile aggiungere istanze database di lettura al cluster per distribuire il carico dalle query di sola lettura su più istanze database.
- Ottimizzare il codice SQL utilizzato dall'applicazione in modo da usare meno risorse. Ad esempio, è possibile esaminare i piani di query, controllare il registro delle query caratterizzate da un'esecuzione lenta o modificare gli indici nelle tabelle. È inoltre possibile eseguire altri tipi comuni di sintonizzazione SQL.

Parametri di Amazon CloudWatch importanti per Aurora Serverless v2

Per iniziare a utilizzare Amazon CloudWatch per un'istanza database Aurora Serverless v2, consulta Visualizzazione dei registri Aurora Serverless v2 in Amazon CloudWatch. Per ulteriori informazioni su come monitorare i cluster di database Aurora attraverso CloudWatch, consulta Monitoraggio degli eventi di log in Amazon CloudWatch.

Puoi visualizzare le istanze database Aurora Serverless v2 in CloudWatch per monitorare la capacità consumata da ciascuna istanza mediante il parametro `ServerlessDatabaseCapacity`. Inoltre, puoi monitorare tutti i parametri standard di Aurora CloudWatch, ad esempio `DatabaseConnections` e `Queries`. Per l'elenco completo dei parametri di CloudWatch che puoi monitorare per Aurora, consulta CloudWatch Parametri Amazon per Amazon Aurora. I parametri metriche sono suddivise in parametri a livello di cluster e parametri a livello di istanza in Parametri a livello di cluster per Amazon Aurora e Parametri a livello di istanza per Amazon Aurora.

I seguuenti parametri di CloudWatch a livello di istanza sono importanti da monitorare per capire come vengono eseguiti il dimensionamento verso l'alto e la riduzione delle istanze database Aurora Serverless v2. Tutti questi parametri vengono calcolati ogni secondo. In questo modo, puoi monitorare lo stato corrente delle istanze database Aurora Serverless v2. È possibile impostare allarmi per avvisarti se i parametri relativi alla capacità di una delle istanze database Aurora Serverless v2 stanno raggiungendo la soglia limite. È possibile determinare se le impostazioni di capacità minima e massima sono appropriate o se è necessario adeguarle. È possibile individuare le aree da valutare con maggiore attenzione per ottimizzare l'efficienza del database.

- `ServerlessDatabaseCapacity`. Come parametro a livello di istanza, restituisce il numero di ACU rappresentato dalla capacità corrente dell'istanza database. Come parametro a livello di cluster, restituisce la media dei valori di `ServerlessDatabaseCapacity` di tutte le istanze database Aurora Serverless v2 nel cluster. Questo parametro è un parametro solo a livello di cluster in Aurora Serverless v1. In Aurora Serverless v2, è disponibile a livello di istanza database e a livello di cluster.
- `ACUUtilization`. Questo parametro è nuovo in Aurora Serverless v2. Questo valore è rappresentato come percentuale. Viene calcolato come il valore del parametro `ServerlessDatabaseCapacity` diviso per il valore ACU massimo del cluster di database. Considera le seguenti linee guida per l'interpretazione di questo parametro e per valutare quale azione eseguire:
  - Se questo parametro si avvicina al valore `100.0`, il dimensionamento verso l'alto dell'istanza database ha raggiunto il valore massimo possibile. Valuta la possibilità di aumentare

l'impostazione ACU massima per il cluster. In questo modo, sia le istanze database di scrittura che quelle di lettura possono essere dimensionate in base a una capacità superiore.

- Supponiamo che un carico di lavoro di sola lettura faccia sì che il valore del parametro `ACUUtilization` di un'istanza database di lettura si avvicini `100.0`, mentre l'istanza database di scrittura non raggiunge la relativa capacità massima. In questo caso, considera di aggiungere istanze database di lettura al cluster. In questo modo, puoi distribuire la parte di sola lettura del carico di lavoro su più istanze database e ridurre conseguentemente il carico su ogni istanza database di lettura.
- Supponiamo di eseguire un'applicazione di produzione, in cui prestazioni e scalabilità sono fattori di primaria importanza. In questo caso, puoi impostare il valore ACU massimo per il cluster su un numero elevato. Il tuo obiettivo è avere il parametro `ACUUtilization` su un valore sempre inferiore a `100.0`. Con un valore ACU massimo elevato, puoi essere certo di disporre di spazio sufficiente in caso di picchi imprevisti a livello di attività del database. Dovrai sostenere solo i costi relativi al consumo effettivo di capacità del database.

- `CPUUtilization`. Questo parametro viene interpretato in modo diverso in Aurora Serverless v2 rispetto a quanto avviene per le istanze database con provisioning. Per Aurora Serverless v2, questo valore è una percentuale calcolata come la quantità di CPU attualmente utilizzata divisa per la capacità della CPU disponibile sotto il valore ACU massimo del cluster di database. Aurora monitora automaticamente questo valore e dimensiona verso l'alto l'istanza database Aurora Serverless v2 quando questa utilizza costantemente una percentuale elevata della capacità della CPU.

  Se il valore di questo parametro si avvicina a `100.0`, l'istanza database ha raggiunto la capacità massima della CPU. Valuta la possibilità di aumentare l'impostazione ACU massima per il cluster. Se il valore di questo parametro si avvicina a `100.0` per un'istanza database di lettura, valuta la possibilità di aggiungere altre istanze database di lettura al cluster. In questo modo, puoi distribuire la parte di sola lettura del carico di lavoro su più istanze database e ridurre conseguentemente il carico su ogni istanza database di lettura.
- `FreeableMemory`. Questo valore rappresenta la quantità di memoria inutilizzata disponibile quando viene eseguito il dimensionamento dell'istanza database Aurora Serverless v2 fino alla sua capacità massima. Per ogni ACU la cui capacità corrente è inferiore alla capacità massima, questo valore aumenta di circa 2 GiB. Pertanto, questo parametro non si avvicina a zero finché non viene eseguito il dimensionamento verso l'alto dell'istanza database fino al valore più alto possibile.

  Se il valore di questo parametro si avvicina a `0`, viene eseguito il dimensionamento verso l'alto dell'istanza database al valore massimo possibile, ovvero al valore più prossimo al limite di

memoria disponibile. Valuta la possibilità di aumentare l'impostazione ACU massima per il cluster. Se il valore di questo parametro si avvicina a `0` per un'istanza database di lettura, valuta la possibilità di aggiungere altre istanze database di lettura al cluster. In questo modo, puoi distribuire la parte di sola lettura del carico di lavoro su più istanze database e ridurre conseguentemente l'utilizzo della memoria su ogni istanza database di lettura.

- `TempStorageIOPS`. Il numero di IOPS eseguiti sull'archiviazione locale collegata all'istanza database. Include gli IOPS per le operazioni sia di letture che di scrittura. Questo parametro rappresenta un conteggio e viene misurato una volta al secondo. Questo è un nuovo parametro per Aurora Serverless v2. Per informazioni dettagliate, consultare Parametri a livello di istanza per Amazon Aurora.
- `TempStorageThroughput`. Quantità di dati trasferiti da e verso l'archiviazione locale associata all'istanza database. Questo parametro rappresenta i byte e viene misurato una volta al secondo. Questo è un nuovo parametro per Aurora Serverless v2. Per informazioni dettagliate, consultare Parametri a livello di istanza per Amazon Aurora.

In genere, la maggior parte del dimensionamento verso l'alto delle istanze database Aurora Serverless v2 è causato dall'utilizzo della memoria e dall'attività della CPU. I parametri `TempStorageIOPS` e `TempStorageThroughput` possono aiutarti a diagnosticare i rari casi in cui l'attività di rete relativa ai trasferimenti tra l'istanza database e i dispositivi di archiviazione locale è responsabile di aumenti imprevisti della capacità. Per monitorare altre attività di rete, puoi utilizzare i seguenti parametri:

- `NetworkReceiveThroughput`
- `NetworkThroughput`
- `NetworkTransmitThroughput`
- `StorageNetworkReceiveThroughput`
- `StorageNetworkThroughput`
- `StorageNetworkTransmitThroughput`

Puoi fare in modo che Aurora pubblichi alcuni o tutti i log di database su Amazon CloudWatch Logs. Per le istruzioni, consulta quanto segue, a seconda del motore di database in uso:

- the section called “Pubblicazione di log Aurora PostgreSQL in CloudWatch Logs”
- the section called “Pubblicazione di log Aurora MySQL in CloudWatch Logs”

Modalità di applicazione dei parametri Aurora Serverless v2 alla fattura AWS

I costi relativi ad Aurora Serverless v2 contenuti nella fattura AWS vengono calcolati sulla base dello stesso parametro `ServerlessDatabaseCapacity` usato per il monitoraggio. Il meccanismo di fatturazione può differire dalla media calcolata di CloudWatch per questo parametro nei casi in cui utilizzi la capacità di Aurora Serverless v2 solo per frazioni di un'ora. Può anche essere diversa se eventuali problemi a livello di sistema rendono il parametro di CloudWatch non disponibile per brevi periodi. Pertanto, sulla fattura potresti vedere un valore di ore ACU leggermente diverso rispetto al calcolo fatto sulla base del valore medio del parametro `ServerlessDatabaseCapacity`.

Esempi di comandi di CloudWatch per i parametri di Aurora Serverless v2

I seguenti esempi per AWS CLI mostrano come è possibile monitorare i parametri di CloudWatch più importanti relativi a Aurora Serverless v2. In ogni caso, sostituire la stringa `Value=` per il parametro `--dimensions` con l'identificativo dell'istanza database Aurora Serverless v2 in uso.

Nell'esempio Linux seguente vengono visualizzati i valori di capacità minima, massima e media per un'istanza database, misurati ogni 10 minuti ogni ora. Il comando Linux `date` specifica l'ora di inizio e l'ora di fine rispetto alla data e all'ora correnti. La funzione `sort_by` nel parametro `--query` ordina i risultati cronologicamente in base al campo `Timestamp`.

```
aws cloudwatch get-metric-statistics --metric-name "ServerlessDatabaseCapacity" \
  --start-time "$(date -d '1 hour ago')" --end-time "$(date -d 'now')" --period 600 \
  --namespace "AWS/RDS" --statistics Minimum Maximum Average \
  --dimensions Name=DBInstanceIdentifier,Value=my_instance \
  --query 'sort_by(Datapoints[*].
{min:Minimum,max:Maximum,avg:Average,ts:Timestamp},&ts)' --output table
```

Gli esempi di Linux riportati di seguito illustrano il monitoraggio della capacità di ogni istanza database in un cluster. Misurano l'utilizzo minimo, massimo e medio della capacità di ciascuna istanza database. Le misurazioni vengono effettuate una volta ogni ora per un periodo di tre ore. Questi esempi utilizzano il parametro `ACUUtilization` che rappresenta una percentuale del limite superiore delle ACU, mentre `ServerlessDatabaseCapacity` rappresenta un numero fisso di ACU. In questo modo, non è necessario conoscere i numeri effettivi per i valori ACU minimi e massimi nell'intervallo di capacità. Puoi visualizzare percentuali che vanno da 0 a 100.

```
aws cloudwatch get-metric-statistics --metric-name "ACUUtilization" \
  --start-time "$(date -d '3 hours ago')" --end-time "$(date -d 'now')" --period 3600 \
  --namespace "AWS/RDS" --statistics Minimum Maximum Average \
```

```
  --dimensions Name=DBInstanceIdentifier,Value=my_writer_instance \
  --query 'sort_by(Datapoints[*].
{min:Minimum,max:Maximum,avg:Average,ts:Timestamp},&ts)' --output table

aws cloudwatch get-metric-statistics --metric-name "ACUUtilization" \
  --start-time "$(date -d '3 hours ago')" --end-time "$(date -d 'now')" --period 3600 \
  --namespace "AWS/RDS" --statistics Minimum Maximum Average \
  --dimensions Name=DBInstanceIdentifier,Value=my_reader_instance \
  --query 'sort_by(Datapoints[*].
{min:Minimum,max:Maximum,avg:Average,ts:Timestamp},&ts)' --output table
```

Il seguente esempio Linux esegue misurazioni simili a quelle precedenti. In questo caso, le misure fanno riferimento al parametro `CPUUtilization`. Le misurazioni vengono effettuate ogni 10 minuti per un periodo di 1 ora. I numeri rappresentano la percentuale di CPU disponibile utilizzata, in base alle risorse della CPU disponibili per l'impostazione di capacità massima per l'istanza database.

```
aws cloudwatch get-metric-statistics --metric-name "CPUUtilization" \
  --start-time "$(date -d '1 hour ago')" --end-time "$(date -d 'now')" --period 600 \
  --namespace "AWS/RDS" --statistics Minimum Maximum Average \
  --dimensions Name=DBInstanceIdentifier,Value=my_instance \
  --query 'sort_by(Datapoints[*].
{min:Minimum,max:Maximum,avg:Average,ts:Timestamp},&ts)' --output table
```

Il seguente esempio Linux esegue misurazioni simili a quelle precedenti. In questo caso, le misure fanno riferimento al parametro `FreeableMemory`. Le misurazioni vengono effettuate ogni 10 minuti per un periodo di 1 ora.

```
aws cloudwatch get-metric-statistics --metric-name "FreeableMemory" \
  --start-time "$(date -d '1 hour ago')" --end-time "$(date -d 'now')" --period 600 \
  --namespace "AWS/RDS" --statistics Minimum Maximum Average \
  --dimensions Name=DBInstanceIdentifier,Value=my_instance \
  --query 'sort_by(Datapoints[*].
{min:Minimum,max:Maximum,avg:Average,ts:Timestamp},&ts)' --output table
```

Monitoraggio delle prestazioni di Aurora Serverless v2 con Approfondimenti sulle prestazioni

Puoi utilizzare Approfondimenti sulle prestazioni per monitorare le prestazioni di istanze database Aurora Serverless v2. Per le procedure di Approfondimenti sulle prestazioni, consulta Monitoraggio del carico DB con Performance Insights su Amazon Aurora.

I seguenti nuovi contatori di Approfondimenti sulle prestazioni si applicano alle istanze database Aurora Serverless v2:

- `os.general.serverlessDatabaseCapacity`: capacità attuale, in ACU, dell'istanza database. Il valore corrisponde al parametro `ServerlessDatabaseCapacity` di CloudWatch per l'istanza database.
- `os.general.acuUtilization`: percentuale di capacità attuale oltre la capacità massima configurata. Il valore corrisponde al parametro `ACUUtilization` di CloudWatch per l'istanza database.
- `os.general.maxConfiguredAcu`: capacità massima configurata per l'istanza database Aurora Serverless v2 corrente. Viene misurata in ACU.
- `os.general.minConfiguredAcu`: capacità minima configurata per l'istanza database Aurora Serverless v2 corrente. Viene misurata in ACU

Per l'elenco completo dei contatori di Approfondimenti sulle prestazioni, consulta Metriche contatore di Performance Insights.

Quando in Approfondimenti sulle prestazioni vengono visualizzati i valori `vCPU` per un'istanza database Aurora Serverless v2, tali valori rappresentano stime basate sul valore ACU per l'istanza database. Per intervalli di default di un minuto, tutti i valori vCPU frazionari vengono arrotondati al numero intero più vicino. Per intervalli più lunghi, il valore vCPU visualizzato è la media dei valori vCPU interi per ogni minuto.

Risoluzione dei problemi di capacità di Aurora Serverless v2

In alcuni casi, Aurora Serverless v2 non viene ridotto fino alla capacità minima, anche in assenza di carico sul database. Questo può accadere per i seguenti motivi:

- Alcune funzionalità possono aumentare l'utilizzo delle risorse e impedire il dimensionamento del database fino alla capacità minima. Le caratteristiche principali comprendono:
  - Database globali di Aurora
  - Esportazione dei file di log Amazon CloudWatch
  - Abilitazione di `pg_audit` nei cluster database compatibili con Aurora PostgreSQL
  - Monitoraggio avanzato
  - Approfondimenti sulle prestazioni

Per ulteriori informazioni, consulta Scelta dell'impostazione Aurora Serverless v2 minima della capacità per un cluster.

- Se un'istanza di lettura non si riduce fino alla capacità minima e mantiene la capacità uguale o superiore dell'istanza di scrittura, controlla il livello di priorità dell'istanza di scrittura. Le istanze database di lettura Aurora Serverless v2 di livello 0 o 1 vengono mantenute a una capacità minima pari ad almeno quella dell'istanza database di scrittura. Imposta il livello di priorità dell'istanza di lettura almeno su 2 in modo che possa essere aumentata o ridotta indipendentemente dall'istanza di scrittura. Per ulteriori informazioni, consulta Scelta del livello di promozione per un'istanza Aurora Serverless v2 di lettura.
- Imposta i valori predefiniti dei parametri del database che influiscono sulla dimensione della memoria condivisa. L'impostazione di un valore superiore a quello predefinito aumenta la richiesta di memoria condivisa e impedisce la riduzione del database fino alla capacità minima. Alcuni esempi sono `max_connections` e `max_locks_per_transaction`.

  > **Note**
  >
  > L'aggiornamento dei parametri della memoria condivisa richiede il riavvio del database per rendere effettive le modifiche.

- I carichi di lavoro di database gravosi possono aumentare l'utilizzo delle risorse.
- I volumi di database di grandi dimensioni possono aumentare l'utilizzo delle risorse.

  Amazon Aurora utilizza risorse di memoria e CPU per la gestione dei cluster database. Aurora richiede più CPU e memoria per gestire cluster database con volumi di database più grandi. Se la capacità minima del cluster è inferiore a quella minima richiesta per la gestione dei cluster, il cluster non verrà ridotto fino alla capacità minima.
- Anche i processi in background, come l'eliminazione, possono aumentare l'utilizzo delle risorse.
- Le limitazioni della versione della piattaforma possono influire sulle funzionalità di dimensionamento. L'intervallo di dimensionamento disponibile per uno specifico cluster è influenzato sia dalla versione del motore che dall'hardware (versione della piattaforma). È possibile eseguire una versione del motore con prestazioni più elevate su una versione della piattaforma con prestazioni più basse e viceversa.

Se il database non riesce ancora a ridurre fino la capacità minima configurata, interrompi e riavvia il database per recuperare eventuali frammenti di memoria che potrebbero essersi accumulati nel

tempo. L'arresto e l'avvio di un database comportano tempi di inattività, quindi consigliamo di farlo raramente.

Dimensionamento a zero ACU con pausa e ripresa automatiche per Aurora Serverless v2

È possibile specificare che le istanze database Aurora Serverless v2 vengano ridotte verticalmente fino a zero ACU e entrino in pausa automaticamente, se non vengono avviate connessioni dalle attività dell'utente entro un periodo di tempo specificato. A tale scopo, è necessario specificare un valore ACU minimo pari a zero per il cluster di database. Quando un'istanza è in stato di pausa, non viene addebitato alcun costo per la sua capacità. L'abilitazione della funzionalità di pausa e ripresa automatiche (pausa automatica) per i cluster Aurora poco utilizzati o con lunghi periodi di inattività permette di contenere i costi del parco di database.

> **Note**
>
> La funzionalità di pausa automatica è disponibile per Aurora Serverless v2 con Aurora PostgreSQL e Aurora MySQL. Potrebbe essere necessario aggiornare la versione del motore di database Aurora per sfruttare questa funzionalità. Per le versioni del motore in cui è disponibile una capacità minima di 0 ACU, consulta Capacità di Aurora Serverless v2.

Argomenti

- Panoramica della funzionalità di pausa automatica Aurora Serverless v2
- Prerequisiti e limitazioni per la funzionalità di pausa automatica Aurora Serverless v2
- Attivazione e disattivazione della funzionalità di pausa automatica
- Come funziona la funzionalità di pausa automatica Aurora Serverless v2
- Come funziona la pausa automatica Aurora Serverless v2 per i diversi tipi di cluster Aurora
- Monitoraggio dei cluster Aurora che utilizzano la pausa automatica
- Risoluzione dei problemi relativi alla pausa automatica Aurora Serverless v2
- Considerazioni sulla progettazione delle applicazioni per la funzionalità di pausa automatica Aurora Serverless v2

Panoramica della funzionalità di pausa automatica Aurora Serverless v2

Le istanze database Aurora Serverless v2 possono entrare automaticamente in pausa in assenza di connessioni utente per un determinato periodo di tempo e riprendere automaticamente all'arrivo di

una richiesta di connessione. La funzionalità di pausa e ripresa automatiche Aurora Serverless v2 aiuta a contenere i costi dei sistemi che non hanno un obiettivo del livello di servizio (SLO) rigido. Ad esempio, è possibile abilitare questa funzionalità per i cluster utilizzati per lo sviluppo e il test o per le applicazioni interne in cui è accettabile una breve pausa mentre il database riprende l'attività. Se il carico di lavoro presenta periodi di inattività e può tollerare lievi ritardi nella connessione mentre l'istanza riprende, valutare la possibilità di utilizzare la pausa automatica con le istanze Aurora Serverless v2 per ridurre i costi.

Per controllare questo comportamento, occorre specificare se le istanze database Aurora Serverless v2 in un cluster possono o meno entrare in pausa automaticamente e quanto deve durare lo stato di inattività di ciascuna istanza prima della messa in pausa. Per abilitare il comportamento di pausa automatica per tutte le istanze database Aurora Serverless v2 in un cluster Aurora, impostare il valore di capacità minima per il cluster su zero ACU.

Se in passato è stata utilizzata la funzionalità Aurora Serverless v1 che impostava il valore a zero ACU dopo uno specifico periodo di inattività, è possibile passare alla funzionalità di pausa automatica Aurora Serverless v2corrispondente.

I vantaggi in termini di riduzione dei costi della funzionalità di pausa automatica sono analoghi a quelli della funzionalità di arresto/avvio del cluster. La pausa automatica per Aurora Serverless v2 offre vantaggi aggiuntivi, come la ripresa più rapida rispetto all'avvio di un cluster arrestato e la determinazione automatica del momento in cui mettere in pausa e riprendere ciascuna istanza database.

La funzionalità di pausa automatica fornisce inoltre una maggiore granularità nel controllo dei costi per le risorse di elaborazione all'interno del cluster. È possibile abilitare la pausa di alcune istanze di lettura anche lasciando sempre attivi l'istanza di scrittura e le altre istanze di lettura del cluster. Per farlo, assegnare alle istanze di lettura che possono entrare in pausa indipendentemente dalle altre istanze una priorità di failover compresa tra 2 e 15.

Le istanza di scrittura e tutte le istanze di lettura con priorità di failover 0 e 1 vengono sempre messe in pausa e riprese contemporaneamente. Pertanto, le istanze di questo gruppo entrano in pausa solo dopo il verificarsi di una specifica condizione: nessuna istanza deve avere connessioni attive per l'intervallo di tempo specificato.

I cluster Aurora database possono contenere una combinazione di istanze database di scrittura e di lettura e di istanze database Aurora Serverless v2 e con provisioning. Pertanto, per utilizzare questa funzionalità in modo efficace, è utile comprendere i seguenti aspetti del meccanismo di pausa automatica:

- Le circostanze in cui un'istanza database potrebbe mettersi in pausa automaticamente.
- Quando può essere impedita la messa in pausa di un'istanza database. Ad esempio, l'abilitazione di alcune funzionalità di Aurora o l'esecuzione di determinati tipi di operazioni sul cluster potrebbe impedire la messa in pausa delle istanze, anche se non è presente alcuna connessione a tali istanze.
- Le conseguenze della messa in pausa di un'istanza per il monitoraggio e la fatturazione.
- Quali azioni comportano la ripresa dell'elaborazione di un'istanza database.
- Come cambia la capacità di un'istanza database quando si verificano gli eventi di pausa e ripresa.
- Come controllare l'intervallo di inattività prima della pausa di un'istanza database.
- Come codificare la logica dell'applicazione per gestire il periodo di tempo durante il quale un'istanza database riprende l'elaborazione.

Prerequisiti e limitazioni per la funzionalità di pausa automatica Aurora Serverless v2

Prima di utilizzare la funzionalità di pausa automatica, controllare quali versioni del motore utilizzare. Inoltre, verificare se la pausa automatica funziona in combinazione con le altre funzionalità di Aurora che si intende utilizzare. Non è possibile attivare la pausa automatica se si utilizza una versione del motore che non la supporta. Per le funzionalità incompatibili, non viene generato alcun errore in caso di utilizzo in combinazione con la pausa automatica. Se il cluster utilizza funzionalità o impostazioni incompatibili, le istanze Aurora Serverless v2 non entreranno in pausa automaticamente.

- Se si utilizza Aurora PostgreSQL, il motore di database deve eseguire almeno la versione 16.3, 15.7, 14.12 o 13.15.
- Se si utilizza Aurora MySQL, il motore di database deve eseguire la versione 3.08.0 o successiva.
- Per l'elenco completo delle versioni del motore e delle Regioni AWS in cui è disponibile questa funzionalità, consulta Regioni e motori di database Aurora supportati per Aurora Serverless v2.
- Quando un'istanza Aurora Serverless v2 viene ripresa, la sua capacità potrebbe essere inferiore rispetto a quando è entrata in pausa. Per informazioni dettagliate, consultare Differenze nel comportamento della pausa automatica tra Aurora Serverless v2 e Aurora Serverless v1.

Alcune condizioni o impostazioni impediscono la messa in pausa automatica delle istanze Aurora Serverless v2. Per ulteriori informazioni, consulta Situazioni in cui Aurora Serverless v2 non viene messo in pausa automaticamente.

Attivazione e disattivazione della funzionalità di pausa automatica

È possibile attivare e disattivare la funzionalità di pausa automatica a livello di cluster. Per farlo, seguire le stesse procedure utilizzate per regolare la capacità minima e massima del cluster. La funzionalità di pausa automatica corrisponde a una capacità minima di 0 ACU.

Argomenti

- Attivazione della pausa automatica per le istanze Aurora Serverless v2 in un cluster
- Intervallo di timeout della pausa automatica Aurora Serverless v2 configurabile
- Ripresa di un'istanza Aurora Serverless v2 in pausa automatica
- Disattivazione della pausa automatica per le istanze Aurora Serverless v2 in un cluster

Attivazione della pausa automatica per le istanze Aurora Serverless v2 in un cluster

Segui la procedura riportata in Impostazione dell'intervallo di capacità di Aurora Serverless v2 per un cluster. Per la capacità minima, scegliere 0 ACU. Quando si sceglie una capacità minima di 0 ACU, è possibile anche specificare il periodo di inattività dell'istanza prima che venga messa automaticamente in pausa.

L'esempio dell'interfaccia della riga di comando seguente mostra come creare un cluster Aurora con la funzionalità di pausa automatica abilitata e l'intervallo di pausa automatica impostato su dieci minuti (600 secondi).

```
aws rds create-db-cluster \
    --db-cluster-identifier my-serverless-v2-cluster \
    --region eu-central-1 \
    --engine aurora-mysql \
    --engine-version 8.0 \
    --serverless-v2-scaling-configuration
 MinCapacity=0,MaxCapacity=4,SecondsUntilAutoPause=600 \
    --master-username myuser \
    --manage-master-user-password
```

L'esempio dell'interfaccia della riga di comando seguente mostra come attivare la funzionalità di pausa automatica per un cluster Aurora esistente. Questo esempio imposta l'intervallo di pausa automatica su un'ora (3.600 secondi).

```
aws rds modify-db-cluster --db-cluster-identifier serverless-v2-cluster \
```

```
  --serverless-v2-scaling-configuration
 MinCapacity=0,MaxCapacity=80,SecondsUntilAutoPause=3600

aws rds describe-db-clusters --db-cluster-identifier serverless-v2-cluster \
  --query 'DBClusters[*].ServerlessV2ScalingConfiguration|[0]'
{
    "MinCapacity": 0,
    "MaxCapacity": 80.0,
    "SecondsUntilAutoPause": 3600
}
```

Intervallo di timeout della pausa automatica Aurora Serverless v2 configurabile

L'intervallo di timeout è mostrato nell'attributo `ServerlessV2ScalingConfiguration` del cluster Aurora. È possibile specificare questo intervallo nella Console di gestione AWS durante la creazione o la modifica di un cluster Aurora, se la capacità minima è impostata su zero ACU. Per specificarlo nell'AWS CLI, utilizzare il parametro `--serverless-v2-scaling-configuration` durante la creazione o la modifica di un cluster Aurora. Per specificarlo nell'API RDS, utilizzare il parametro `ServerlessV2ScalingConfiguration` durante la creazione o la modifica di un cluster Aurora.

L'intervallo minimo che è possibile impostare è 300 secondi (cinque minuti). Questa è l'impostazione predefinita se non si specifica un intervallo. L'intervallo massimo che è possibile impostare è 86.400 secondi (un giorno).

Si supponga di disattivare la funzionalità di pausa automatica per un cluster impostando la capacità minima del cluster su un valore diverso da zero. In tal caso, la proprietà interval viene rimossa dall'attributo `ServerlessV2ScalingConfiguration`. L'assenza di tale proprietà fornisce un'ulteriore conferma che la funzionalità di pausa automatica è disattivata per quel cluster. Se successivamente si riattiva la pausa automatica, si può specificare di nuovo un qualsiasi intervallo personalizzato.

Ripresa di un'istanza Aurora Serverless v2 in pausa automatica

- Quando ci si connette a un'istanza Aurora Serverless v2 in pausa, questa si riattiva e accetta automaticamente la connessione.
- Un tentativo di connessione senza credenziali valide comporta comunque la ripresa dell'istanza database.
- Se ci si connette tramite l'endpoint di scrittura, Aurora riprende l'istanza database di scrittura se è in pausa automatica. Allo stesso tempo, Aurora riattiva tutte le istanze di lettura in pausa automatica

con priorità di failover 0 o 1, il che significa che la loro capacità è legata alla capacità dell'istanza di scrittura.

- Se ci si connette tramite l'endpoint di lettura, Aurora sceglie un'istanza di lettura in modo casuale. Se l'istanza di lettura è in pausa, Aurora la riattiva. Inoltre, Aurora riattiva per prima l'istanza di scrittura, che deve essere sempre attiva se ci sono istanze di lettura attive. Quando Aurora riattiva l'istanza di scrittura, vengono riprese anche tutte le istanze di lettura nei livelli di promozione del failover zero e uno.
- Se si invia una richiesta al cluster tramite l'API dati RDS, Aurora riattiva l'istanza di scrittura se è in pausa. A questo punto, Aurora elabora la richiesta API dati.
- Quando si modifica il valore di un parametro di configurazione in un gruppo di parametri del cluster di database, Aurora riattiva automaticamente tutte le istanze Aurora Serverless v2 in pausa in tutti i cluster che utilizzano quel gruppo di parametri del cluster. Analogamente, quando si modifica il valore di un parametro in un gruppo di parametri database, Aurora riattiva automaticamente tutte le istanze Aurora Serverless v2 in pausa che utilizzano quel gruppo di parametri database. Lo stesso comportamento di ripresa automatica si applica quando si modifica un cluster per assegnare un gruppo di parametri del cluster diverso o quando si modifica un'istanza per assegnare un gruppo di parametri database diverso.
- L'esecuzione di una richiesta di backtrack riattiva automaticamente l'istanza di scrittura Aurora Serverless v2 se è in pausa. Aurora elabora la richiesta di backtrack dopo la ripresa dell'istanza di scrittura. È possibile tornare indietro al momento in cui un'istanza Aurora Serverless v2 è entrata in pausa.
- L'acquisizione di uno snapshot del cluster o l'eliminazione di uno snapshot non comporta la ripresa di alcuna istanza Aurora Serverless v2.
- La creazione di un clone di Aurora comporta la ripresa dell'istanza di scrittura del cluster che viene clonato.
- Se un'istanza in pausa riceve un alto numero di richieste di connessione prima che finisca di riattivarsi, alcune sessioni potrebbero non riuscire a connettersi. Consigliamo di implementare la logica di ripetizione per le connessioni ai cluster Aurora che hanno istanze Aurora Serverless v2 con la pausa automatica abilitata. Ad esempio, in caso di mancata connessione, hai a disposizione tre tentativi per riprovare a eseguirla.
- Aurora può eseguire alcuni tipi di manutenzione spicciola interna senza riattivare un'istanza. Tuttavia, alcuni tipi di manutenzione che avvengono durante la finestra di manutenzione del cluster richiedono che Aurora riattivi l'istanza. Al termine della manutenzione, l'istanza viene nuovamente messa in pausa automaticamente se non viene rilevata alcuna attività dopo l'intervallo specificato.

> **Note**
>
> Aurora non riprende automaticamente un'istanza in pausa per processi pianificati specifici del motore, come quelli dell'estensione `pg_cron` PostgreSQL o del pianificatore di eventi MySQL. Per garantire l'esecuzione di tali processi, avviare manualmente una connessione all'istanza prima dell'orario pianificato. Aurora non mette in coda alcun processo la cui l'ora pianificata corrisponde al periodo di pausa dell'istanza database. Tali processi vengono saltati quando l'istanza viene successivamente riavviata.

- Se si verifica un failover del cluster Aurora mentre un'istanza Aurora Serverless v2 è in pausa automatica, Aurora potrebbe riprendere un'istanza e promuoverla a istanza di scrittura. Lo stesso potrebbe accadere se una o più istanze database vengono rimosse dal cluster mentre un'istanza è in pausa. In questo caso, l'istanza diventa immediatamente un'istanza di scrittura quando viene riattivata.
- Le operazioni che modificano le proprietà del cluster causano anche la ripresa di tutte le istanze Aurora Serverless v2 in pausa automatica. Ad esempio, un'istanza in pausa automatica si riattiva per operazioni come le seguenti:
  - Modifica dell'intervallo di dimensionamento del cluster.
  - Aggiornamento della versione del motore del cluster.
  - Descrizione o download dei file di log da un'istanza in pausa. È possibile esaminare i dati di log storici delle istanze in pausa abilitando il caricamento dei log su CloudWatch e analizzandoli tramite CloudWatch.

Disattivazione della pausa automatica per le istanze Aurora Serverless v2 in un cluster

Segui la procedura riportata in Impostazione dell'intervallo di capacità di Aurora Serverless v2 per un cluster. Per la capacità minima, scegliere un valore pari o superiore a 0,5. Quando si disattiva la funzionalità di pausa automatica, l'intervallo di inattività dell'istanza viene reimpostato. Se si riattiva la pausa automatica, specificare un nuovo intervallo di timeout.

L'esempio dell'interfaccia della riga di comando seguente mostra come disattivare la funzionalità di pausa automatica per un cluster Aurora esistente. L'output `describe-db-clusters` mostra che l'attributo `SecondsUntilAutoPause` viene rimosso quando la capacità minima è impostata su un valore diverso da zero.

```
aws rds modify-db-cluster --db-cluster-identifier serverless-v2-cluster \
```

```
  --serverless-v2-scaling-configuration MinCapacity=2,MaxCapacity=80

aws rds describe-db-clusters --db-cluster-identifier serverless-v2-cluster \
  --query 'DBClusters[*].ServerlessV2ScalingConfiguration|[0]'
{
    "MinCapacity": 2,
    "MaxCapacity": 80.0
}
```

Come funziona la funzionalità di pausa automatica Aurora Serverless v2

È possibile utilizzare le seguenti informazioni per pianificare l'utilizzo della funzionalità di pausa automatica. Comprendere in quali casi le istanze entrano in pausa, vengono riprese o rimangono attive può aiutare a trovare un equilibrio tra disponibilità, reattività e riduzione dei costi.

Argomenti

- Che cosa succede quando le istanze Aurora Serverless v2 entrano in pausa
- Cosa succede quando le istanze Aurora Serverless v2 in pausa automatica riprendono
- Come funziona la fatturazione delle istanze per i cluster Aurora Serverless v2 con pausa automatica
- Situazioni in cui Aurora Serverless v2 non viene messo in pausa automaticamente
- Come funziona la pausa automatica con la funzionalità di arresto/avvio del cluster
- Come funzionano la manutenzione e gli aggiornamenti per i cluster Aurora Serverless v2 con pausa automatica
- Differenze nel comportamento della pausa automatica tra Aurora Serverless v2 e Aurora Serverless v1

Che cosa succede quando le istanze Aurora Serverless v2 entrano in pausa

Quando un'istanza database Aurora Serverless v2 entra in pausa allo scadere di un intervallo di tempo durante il quale non si registrano connessioni:

- Aurora inizia a mettere in pausa l'istanza allo scadere dell'intervallo di tempo durante il quale non si registrano connessioni all'istanza, indipendentemente dal numero di ACU dell'istanza in quel momento.

- Il meccanismo di pausa non è immediato. Un'istanza Aurora Serverless v2 in procinto di entrare in pausa automaticamente potrebbe attendere brevemente per recuperare il ritardo nell'applicazione di tutte le modifiche all'archiviazione Aurora.
- I costi relativi all'istanza vengono sospesi. La metrica `ServerlessV2Usage` ha un valore pari a 0 quando l'istanza è in pausa.
- Il valore dello stato dell'istanza non cambia. Lo stato viene comunque visualizzato come "disponibile".
- L'istanza interrompe la scrittura nei file di log del database. Smette di inviare metriche a CloudWatch e registra zero percento per `CPUUtilization` e `ACUUtilization` e zero per `ServerlessDatabaseCapacity`.
- Aurora emette eventi quando un'istanza database Aurora Serverless v2 inizia la pausa, termina la pausa e se il meccanismo di pausa viene interrotto o non funziona. Per dettagli su questi eventi, consultare Eventi di istanza database.

Cosa succede quando le istanze Aurora Serverless v2 in pausa automatica riprendono

Quando un'istanza database Aurora Serverless v2 riprende dopo essere entrata in pausa automaticamente, si applicano le seguenti condizioni:

- Tutte le modifiche ai parametri incluse nelle modifiche di `pending-reboot` vengono applicate quando l'istanza riprende.
- Aurora emette eventi a livello di istanza quando ogni istanza database Aurora Serverless v2 inizia e termina la ripresa e se l'istanza non riesce a riattivarsi per qualche motivo. Per dettagli su questi eventi, consultare Eventi di istanza database.
- Tutte le connessioni richieste vengono stabilite al termine della ripresa dell'istanza database. Poiché il tempo di ripresa tipico in genere è di circa 15 secondi, si consiglia di regolare le impostazioni del timeout del client su un valore superiore ai 15 secondi. Ad esempio, nelle impostazioni del driver JDBC, è possibile regolare i valori per le impostazioni `connectTimeout` e `sslResponseTimeout` in modo che superino i 15 secondi.

> **Note**
>
> Se un'istanza Aurora Serverless v2 rimane in pausa per più di 24 ore, Aurora può mettere l'istanza in uno stato di sospensione più profondo, che richiede più tempo per riattivarsi. In questo caso, il tempo di ripresa può essere di 30 o più secondi, all'incirca equivalente al

tempo di riavvio dell'istanza. Se il database presenta periodi di inattività che durano più di un giorno, si consiglia di impostare i timeout di connessione su un valore di almeno 30 secondi.

Come funziona la fatturazione delle istanze per i cluster Aurora Serverless v2 con pausa automatica

Durante la pausa automatica, i costi di un'istanza Aurora Serverless v2 sono pari a zero. La metrica `ServerlessV2Usage` è zero durante questo periodo. AWS addebita comunque i costi per l'archiviazione Aurora e altri aspetti del cluster non legati alla specifica istanza database.

Situazioni in cui Aurora Serverless v2 non viene messo in pausa automaticamente

- Se il valore di capacità minima per il cluster di database è superiore a zero ACU, le istanze Aurora Serverless v2 nel cluster non vengono messe automaticamente in pausa. Se sono presenti cluster con istanze Aurora Serverless v2 precedenti alla disponibilità della funzionalità di pausa automatica, l'impostazione della capacità minima più bassa era pari a 0,5 ACU. Per utilizzare la funzionalità di pausa automatica con tali cluster, modificare l'impostazione della capacità minima su zero ACU.
- Se sono aperte connessioni avviate dall'utente in un'istanza Aurora Serverless v2, questa non entrerà in pausa. Inoltre, l'istanza non entrerà in pausa mentre sono in corso attività come l'applicazione di patch e gli aggiornamenti. Le connessioni amministrative utilizzate da Aurora per i controlli dell'integrità non vengono considerate attività e non impediscono la messa in pausa dell'istanza.
- In un cluster Aurora PostgreSQL con replica logica abilitata o in un cluster Aurora MySQL con replica binlog abilitata, l'istanza di scrittura e tutte le istanze di lettura nei livelli di promozione del failover zero e uno non entrano in pausa automaticamente. Aurora esegue una quantità minima costante di controlli sull'integrità della connessione di replica.

  Per i cluster con replica abilitata, è possibile comunque avere istanze di lettura Aurora nel cluster di origine con pausa automatica. Per attivare la funzionalità, impostare la priorità di failover delle istanze su un valore diverso da zero o uno. In questo modo, possono entrare in pausa indipendentemente dall'istanza di scrittura.
- Se al cluster Aurora è associato Server proxy per RDS, il proxy mantiene una connessione aperta a ciascuna istanza database del cluster. Pertanto, tutte le istanze Aurora Serverless v2 in un cluster di questo tipo non entreranno in pausa automaticamente.

- Se il cluster è il cluster primario di un Database globale Aurora, Aurora non mette automaticamente in pausa l'istanza di scrittura Aurora Serverless v2. Questo perché è necessario un livello costante di attività sull'istanza di scrittura per gestire gli altri cluster nel database globale. Poiché l'istanza di scrittura rimane attiva, anche le altre istanze di lettura Aurora Serverless v2 con priorità di failover zero o uno non entrano in pausa automaticamente.
- Le istanze Aurora Serverless v2 nei cluster secondari di un Database globale Aurora non entrano in pausa automaticamente. Se un cluster di database viene promosso a cluster autonomo, la funzionalità di pausa automatica diventa effettiva se vengono soddisfatte tutte le altre condizioni.
- In un cluster con un'integrazione Zero-ETL associata a Redshift, l'istanza di scrittura e tutte le istanze di lettura nei livelli di promozione del failover zero e uno non entrano in pausa automaticamente.
- Oltre all'attività sulla porta del database principale dell'istanza, se un'istanza Aurora PostgreSQL ha la funzionalità Babelfish abilitata, qualsiasi connessione e attività sulla porta T-SQL impedisce la messa in pausa automatica dell'istanza.

Come funziona la pausa automatica con la funzionalità di arresto/avvio del cluster

È possibile arrestare e avviare un cluster Aurora quando la funzionalità di pausa automatica è abilitata. Non importa se alcune istanze sono in pausa. Quando si riavvia il cluster, tutte le istanze Aurora Serverless v2 in pausa vengono riattivate automaticamente.

Durante l'arresto di un cluster Aurora, le istanze Aurora Serverless v2 in pausa non vengono riattivate automaticamente in base ai tentativi di connessione. Una volta riavviato il cluster, si applicano i consueti meccanismi di messa in pausa e ripresa delle istanze Aurora Serverless v2.

Come funzionano la manutenzione e gli aggiornamenti per i cluster Aurora Serverless v2 con pausa automatica

- Durante la pausa automatica di un'istanza Aurora Serverless v2, se si tenta di aggiornare il cluster Aurora, Aurora riprende l'istanza e la aggiorna.
- Aurora riprende periodicamente eventuali istanze Aurora Serverless v2 in pausa automatica per eseguire operazioni di manutenzione come, ad esempio, aggiornamenti a versioni secondarie e modifiche a proprietà come i gruppi di parametri.
- Dopo che un'istanza Aurora Serverless v2 viene riattivata per un'operazione amministrativa come un aggiornamento o interventi di manutenzione, Aurora attende almeno 20 minuti prima di mettere nuovamente in pausa l'istanza. Questo consente di completare tutte le eventuali operazioni in

background. Il periodo di 20 minuti evita inoltre di sospendere e riprendere l'istanza più volte, nel caso in cui venga sottoposta a più operazioni amministrative consecutive.

Differenze nel comportamento della pausa automatica tra Aurora Serverless v2 e Aurora Serverless v1

- Rispetto a Aurora Serverless v1, il tempo di ripresa di Aurora Serverless v2 è migliore. Il tempo di ripresa è in genere di circa 15 secondi se l'istanza è in pausa da meno di 24 ore. Se la pausa supera le 24 ore, invece, il tempo di ripresa potrebbe essere più lungo.
- Il modo in cui Aurora Serverless v2 applica questa funzionalità ai cluster Multi-AZ fa sì che nel cluster siano presenti istanze database sia in pausa sia attive. L'istanza di scrittura riprende ogni volta che è in esecuzione un'istanza di lettura, poiché un'istanza di scrittura è necessario per coordinare determinate attività all'interno del cluster. Poiché Aurora Serverless v1 non utilizza istanze di lettura, l'intero cluster sarebbe sempre in pausa o attivo.
- Quando l'endpoint di lettura sceglie casualmente un'istanza di lettura a cui connettersi, quell'istanza di lettura potrebbe essere già attiva o in pausa automatica. Pertanto, il tempo necessario per accedere all'istanza di lettura potrebbe variare ed essere più difficile da prevedere. I cluster Multi-AZ che utilizzano Aurora Serverless v2 e la pausa automatica potrebbero quindi trarre vantaggio dalla configurazione di endpoint personalizzati per casi d'uso specifici di sola lettura, invece di indirizzare tutte le sessioni di sola lettura all'endpoint di lettura.
- Le istanze Aurora Serverless v2 vengono sottoposte a operazioni di manutenzione con la stessa frequenza delle istanze con provisioning. Poiché Aurora riattiva automaticamente le istanze quando necessitano di manutenzione, è possibile che le istanze Aurora Serverless v2 vengano riattivate più spesso rispetto ai cluster Aurora Serverless v1.

Come funziona la pausa automatica Aurora Serverless v2 per i diversi tipi di cluster Aurora

Le considerazioni relative alla funzionalità di pausa automatica dipendono dal numero di istanze presenti nel cluster Aurora, dai livelli di promozione del failover delle istanze di lettura e dal fatto che tutte le istanze siano Aurora Serverless v2 o siano una combinazione di istanze Aurora Serverless v2 e con provisioning.

Argomenti

- Layout consigliati dei cluster Aurora quando si utilizza la pausa automatica

- Come funziona la pausa automatica Aurora Serverless v2 per l'istanza di scrittura in un cluster di database
- Come funziona la pausa automatica Aurora Serverless v2 per i cluster Multi-AZ
- Come funziona la pausa automatica Aurora Serverless v2 per i cluster con istanze con provisioning

Layout consigliati dei cluster Aurora quando si utilizza la pausa automatica

Quando la funzionalità di pausa automatica è abilitata, è possibile organizzare il cluster Aurora per ottenere il giusto equilibrio tra disponibilità elevata, risposta rapida e scalabilità in base al caso d'uso specifico. Per farlo, scegliere la combinazione di istanze Aurora Serverless v2, istanze con provisioning e livelli di promozione del failover per le istanze database nel cluster.

I seguenti tipi di configurazioni mostrano le varie combinazioni possibili di disponibilità elevata e ottimizzazione dei costi per il cluster:

- Per un sistema di sviluppo e test, è possibile configurare un cluster di database Single-AZ con un'istanza database Aurora Serverless v2. L'istanza singola serve tutte le richieste di lettura e scrittura. Quando il cluster non viene utilizzato per intervalli di tempo significativi, l'istanza database entra in pausa. A quel punto, anche i costi di elaborazione database per il cluster vengono sospesi.
- Se un sistema esegue un'applicazione per cui la disponibilità elevata è una priorità, ma il cluster presenta ancora periodi in cui è completamente inattivo, è possibile configurare un cluster Multi-AZ che includa istanze database di scrittura e di lettura Aurora Serverless v2. Impostare l'istanza di lettura sulla priorità di failover zero o uno, in modo che l'istanza di scrittura e l'istanza di lettura vengano messe in pausa e riprendano contemporaneamente. In questo modo si ottiene il vantaggio di un failover rapido mentre il cluster è attivo. Quando il cluster rimane inattivo per un periodo superiore alla soglia di pausa automatica, i costi dell'istanza database per entrambe le istanze vengono sospesi. Quando il cluster riprende l'elaborazione, la prima sessione del database impiega un po' di tempo per connettersi.
- Supponiamo che il cluster sia costantemente attivo con una quantità minima di attività e richieda una risposta rapida per qualsiasi connessione. In tal caso, è possibile creare un cluster con più di un'istanza di lettura Aurora Serverless v2, quindi disaccoppiare le capacità di alcune istanze di lettura dall'istanza di scrittura. Specificare una priorità di failover zero o uno per l'istanza di scrittura e uno per l'istanza di lettura. Specificare una priorità maggiore di uno per le altre istanze di lettura. In questo modo, le istanze di lettura con livelli di priorità più elevati possono entrare in pausa automaticamente, anche quando l'istanza di scrittura e una delle istanze di lettura rimangono attive.

In questo caso, è possibile utilizzare altre tecniche per garantire che il cluster rimanga sempre disponibile, pur garantendo una riduzione verticale della capacità durante i periodi di inattività:

- È possibile utilizzare le istanze con provisioning per l'istanza di scrittura e per l'istanza di lettura con priorità 0 o 1. In questo modo, due istanze database non entrano mai in pausa automaticamente e sono sempre disponibili per gestire il traffico del database ed eseguire il failover.
- È possibile configurare un endpoint personalizzato che includa le istanze Aurora Serverless v2 nei livelli di priorità più elevati, ma escluda l'istanza di scrittura o le istanze di lettura con livello di promozione 0 o 1. In questo modo, è possibile indirizzare le sessioni di sola lettura non sensibili alla latenza alle istanze di lettura che possono essere messi in pausa automaticamente. Per queste richieste si può evitare l'utilizzo dell'endpoint di lettura, perché Aurora può indirizzare le connessioni dell'endpoint di lettura all'istanza di lettura sempre attiva o a una delle istanze in pausa automatica. L'utilizzo dell'endpoint personalizzato consente di indirizzare le connessioni a diversi gruppi di istanze in base alle proprie preferenze in termini di risposta rapida o di dimensionamento aggiuntivo.

Come funziona la pausa automatica Aurora Serverless v2 per l'istanza di scrittura in un cluster di database

Quando un cluster Aurora database contiene una sola istanza database, il meccanismo per metterla in pausa e riattivarla automaticamente è semplice e dipende solo dall'attività sull'istanza di scrittura. Una configurazione di questo tipo potrebbe essere utile per i cluster utilizzati per attività di sviluppo e test o per l'esecuzione di applicazioni che non richiedono una disponibilità elevata. È importante ricordare che, in un cluster a istanza singola, Aurora indirizza le connessioni all'istanza database di scrittura attraverso l'endpoint di lettura. Pertanto, per un cluster di database a istanza singola, il tentativo di connessione all'endpoint di lettura causa la ripresa dell'istanza database di scrittura messa in pausa automaticamente.

I fattori aggiuntivi riportati di seguito si applicano ai cluster Aurora con più istanze database:

- In genere, all'interno di un cluster di database Aurora si accede spesso all'istanza database di scrittura. Pertanto, è possibile che l'istanza database di scrittura rimanga attiva anche quando una o più istanze database di lettura entrano in pausa automaticamente.
- Alcune attività sulle istanze database di lettura richiedono che l'istanza database di scrittura sia disponibile. Pertanto, le istanze database di scrittura non possono entrare in pausa finché non vengono sospese anche tutte le istanze di lettura. La ripresa di qualsiasi istanza di lettura

riattiva automaticamente l'istanza di scrittura, anche se l'applicazione non accede direttamente a quest'ultima.

- Le istanze di lettura Aurora Serverless v2 nei livelli di promozione del failover zero e uno vengono scalate per sincronizzare la loro capacità con l'istanza di scrittura. Pertanto, quando un'istanza di scrittura Aurora Serverless v2 si riattiva, riprendono anche tutte le istanze di lettura Aurora Serverless v2 incluse nei livelli di promozione del failover zero e uno.

Come funziona la pausa automatica Aurora Serverless v2 per i cluster Multi-AZ

All'interno di un cluster di database Aurora che contiene sia un'istanza di scrittura che una o più istanze database di lettura, potrebbero essere presenti istanze database Aurora Serverless v2 sia in pausa sia attive. L'istanza di scrittura e tutte le istanze di lettura con priorità di failover 0 e 1 vengono sempre messe in pausa e riattivate contemporaneamente. Le istanze di lettura con priorità diversa da 0 o 1 possono entrare in pausa e riattivate indipendentemente dalle altre istanze.

Quando si utilizza questa funzionalità in cluster con più istanze di lettura, è possibile contenere i costi senza sacrificare la disponibilità elevata. L'istanza di scrittura e una o due altre istanze di lettura possono rimanere sempre attive, mentre le istanze di lettura aggiuntive possono entrare in pausa quando non sono necessarie per gestire volumi di traffico di lettura elevati.

Un'istanza di lettura può entrare in pausa automaticamente se la sua capacità può essere scalata in modo indipendente; se invece è legata alla capacità dell'istanza database di scrittura, non è possibile applicare la funzionalità di pausa. Questa proprietà di dimensionamento dipende dalla priorità di failover dell'istanza database di lettura. Quando la priorità dell'istanza di lettura è zero o uno, la capacità dell'istanza di lettura monitora la capacità dell'istanza database di scrittura. Quindi, per consentire la messa in pausa automatica delle istanze database di lettura nella gamma di situazioni più ampia, impostarne la priorità su un valore superiore a uno.

Il tempo necessario per riprendere un'istanza di lettura potrebbe essere leggermente più lungo rispetto a un'istanza di scrittura. Per sapere il più rapidamente possibile se le istanze possono entrare in pausa, connettersi all'endpoint del cluster.

Come funziona la pausa automatica Aurora Serverless v2 per i cluster con istanze con provisioning

Tutte le istanze database con provisioning nel cluster di database Aurora non entrano in pausa automaticamente. Solo le istanze database Aurora Serverless v2, con la classe di istanza `db.serverless`, possono utilizzare la funzionalità di pausa automatica.

Quando il cluster Aurora contiene istanze database con provisioning, nessuna istanza di scrittura Aurora Serverless v2 entra in pausa automaticamente. Questo dipende dal requisito che prevede che l'istanza di scrittura sia sempre disponibile quando sono attive istanze di lettura. Il fatto che l'istanza di scrittura Aurora Serverless v2 resta attiva significa anche che tutte le istanze di lettura Aurora Serverless v2 con priorità di failover 0 e 1 non entreranno in pausa automaticamente in un cluster ibrido che contiene istanze con provisioning.

Monitoraggio dei cluster Aurora che utilizzano la pausa automatica

Per monitorare Aurora, è necessario conoscere già le procedure di monitoraggio in Monitoraggio dei parametri di Amazon Aurora con Amazon CloudWatch e le metriche CloudWatch elencate in Riferimento per i parametri per Amazon Aurora. È importante tenere presente alcune considerazioni specifiche quando si monitorano i cluster Aurora che utilizzano la funzionalità di pausa automatica:

- Durante alcuni intervalli di tempo, le istanze Aurora Serverless v2 non registrano i dati di log e gran parte delle metriche perché sono in pausa. Le uniche metriche inviate a CloudWatch mentre un'istanza è in pausa sono zero percento per `CPUUtilization` e `ACUUtilization` e zero per `ServerlessDatabaseCapacity`.
- È possibile verificare se le istanze Aurora Serverless v2 entrano in pausa più o meno spesso del previsto. A tale scopo, controllare la frequenza con cui la metrica `ServerlessDatabaseCapacity` passa da un valore diverso da zero a zero e per quanto tempo rimane zero. Se le istanze non restano in pausa per il tempo previsto, il risparmio sui costi effettivo sarà inferiore rispetto a quello potenziale. Se le istanze entrano in pausa e vengono riattivate più spesso del previsto, il cluster potrebbe presentare una latenza non necessaria nella risposta alle richieste di connessione. Per informazioni sui fattori che influiscono sulla possibilità e sulla frequenza con cui le istanze Aurora Serverless v2 possono entrare in pausa, consulta Prerequisiti e limitazioni per la funzionalità di pausa automatica Aurora Serverless v2, Situazioni in cui Aurora Serverless v2 non viene messo in pausa automaticamente e Risoluzione dei problemi relativi alla pausa automatica Aurora Serverless v2.
- È anche possibile esaminare un file di log che registra le operazioni di pausa e ripresa automatiche per un'istanza Aurora Serverless v2. Se un'istanza non è stata sospesa alla scadenza dell'intervallo di timeout, nel file di log verrà riportato anche il motivo della mancata attivazione della pausa. Per ulteriori informazioni, consulta Monitoraggio delle attività di pausa e ripresa Aurora Serverless v2.

Argomenti

- Verifica dello stato di pausa di un'istanza Aurora Serverless v2
- Eventi per le operazioni di pausa e ripresa automatiche
- Come funziona la pausa automatica con Approfondimenti sulle prestazioni e Monitoraggio avanzato
- In che modo la funzionalità di pausa automatica Aurora Serverless v2 interagisce con le metriche Aurora

Verifica dello stato di pausa di un'istanza Aurora Serverless v2

Per determinare se un'istanza Aurora Serverless v2 è in stato di pausa, è possibile osservare la metrica `ACUUtilization` relativa all'istanza. Quando l'istanza è in pausa, il valore della metrica è zero.

Quando un'istanza Aurora Serverless v2 è in pausa, lo stato visualizzato sarà comunque Disponibile. Lo stesso vale per un'istanza Aurora Serverless v2 in pausa in fase di ripresa. Il motivo è che è comunque possibile connettersi correttamente a tale istanza, anche se la connessione subisce un leggero ritardo.

In qualsiasi metrica relativa alla disponibilità delle istanze Aurora, il periodo di pausa dell'istanza viene considerato un periodo di disponibilità.

Eventi per le operazioni di pausa e ripresa automatiche

Aurora emette eventi per le istanze Aurora Serverless v2 quando le operazioni di pausa e ripresa automatiche iniziano, terminano o vengono annullate. Gli eventi relativi alla funzionalità di pausa automatica da `RDS-EVENT-0370` a `RDS-EVENT-0374`. Per dettagli su questi eventi, consultare Eventi di istanza database.

Come funziona la pausa automatica con Approfondimenti sulle prestazioni e Monitoraggio avanzato

Mentre un'istanza Aurora Serverless v2 è in pausa, Aurora non raccoglie informazioni di monitoraggio per quell'istanza tramite Approfondimenti sulle prestazioni o Monitoraggio avanzato. Quando l'istanza viene riattivata, potrebbe verificarsi un breve ritardo prima che Aurora riprenda a raccogliere le informazioni di monitoraggio.

In che modo la funzionalità di pausa automatica Aurora Serverless v2 interagisce con le metriche Aurora

Quando è in pausa, un'istanza Aurora Serverless v2 non emette gran parte delle metriche CloudWatch né scrive alcuna informazione nei log del database. Le uniche metriche inviate a CloudWatch mentre un'istanza è in pausa sono zero percento per `CPUUtilization` e `ACUUtilization` e zero per `ServerlessDatabaseCapacity`.

Quando CloudWatch elabora statistiche relative alla disponibilità e al tempo di attività dell'istanza o del cluster, le istanze Aurora Serverless v2 in pausa vengono considerate disponibili.

Quando si avvia un'azione dell'AWS CLI o dell'API RDS per descrivere o scaricare i log di un'istanza Aurora Serverless v2 in pausa, l'istanza riprende automaticamente per rendere disponibili le informazioni di log.

Esempio di metriche CloudWatch

Gli esempi seguenti relativi all'AWS CLI mostrano come osservare la modifica della capacità di un'istanza nel tempo. Durante l'intervallo di tempo, l'istanza viene messa in pausa e ripresa automaticamente. Mentre è in pausa, la metrica `ServerlessDatabaseCapacity` riporta un valore pari a zero. Per determinare se l'istanza è entrata in pausa in qualsiasi momento durante l'intervallo specificato, si può controllare se la capacità minima durante quel periodo di tempo era pari a zero.

Il seguente esempio di Linux rappresenta un'istanza Aurora Serverless v2 messa automaticamente in pausa per un certo periodo di tempo. Raccogliamo i valori `ServerlessDatabaseCapacity` ogni minuto, in un arco di tre minuti. Il valore ACU minimo di 0,0 conferma che l'istanza è entrata in pausa in un dato momento durante ogni minuto.

```
$ aws cloudwatch get-metric-statistics \
  --metric-name "ServerlessDatabaseCapacity" \
  --start-time "$(date -d '3 minutes ago')" --end-time "$(date -d 'now')" --period 60 \
  --statistics Minimum \
  --namespace "AWS/RDS" --dimensions Name=DBInstanceIdentifier,Value=my-autopause-
instance \
  --output text | sort -k 3

ServerlessDatabaseCapacity
DATAPOINTS 0.0 2024-08-02T22:11:00+00:00 None
DATAPOINTS 0.0 2024-08-02T22:12:00+00:00 None
DATAPOINTS 0.0 2024-08-02T22:13:00+00:00 None
```

Successivamente, tentiamo di stabilire una connessione all'istanza Aurora Serverless v2 in pausa. In questo esempio, utilizziamo intenzionalmente una password errata in modo che il tentativo di connessione non abbia successo. Nonostante l'errore, il tentativo di connessione fa sì che Aurora riattivi l'istanza in pausa.

```
$ mysql -h my_cluster_endpoint.rds.amazonaws.com -u admin -pwrong-password

ERROR 1045 (28000): Access denied for user 'admin'@'ip_address' (using password: YES)
```

Nel seguente esempio di Linux, l'istanza in pausa è stata riattivata, è rimasta inattiva per circa cinque minuti e poi è tornata allo stato di pausa. L'istanza si è riattivata con una capacità di 2,0 ACU. Quindi, ha mostrato un leggero aumento verticale, ad esempio per eseguire alcune operazioni di pulizia interna. Poiché non si è verificato alcun tentativo di connessione utente prima della scadenza del periodo di timeout di cinque minuti, l'istanza è passata direttamente da 4,0 ACU allo stato di pausa.

```
$ aws cloudwatch get-metric-statistics \
  --metric-name "ServerlessDatabaseCapacity" \
  --start-time "$(date -d '8 minutes ago')" --end-time "$(date -d 'now')" --period 60 \
  --statistics Minimum \
  --namespace "AWS/RDS" --dimensions Name=DBInstanceIdentifier,Value=my-autopause-
instance \
  --output text | sort -k 3

ServerlessDatabaseCapacity
DATAPOINTS 0.0 2024-08-02T22:13:00+00:00 None
DATAPOINTS 2.0 2024-08-02T22:14:00+00:00 None
DATAPOINTS 3.0 2024-08-02T22:15:00+00:00 None
DATAPOINTS 3.0 2024-08-02T22:16:00+00:00 None
DATAPOINTS 4.0 2024-08-02T22:17:00+00:00 None
DATAPOINTS 4.0 2024-08-02T22:18:00+00:00 None
DATAPOINTS 4.0 2024-08-02T22:19:00+00:00 None
DATAPOINTS 0.0 2024-08-02T22:20:00+00:00 None
```

Se lo scopo del report fosse di mostrare la velocità con cui l'istanza è aumenta verticalmente per gestire il carico di lavoro, potremmo specificare `Maximum` come statistica, invece di `Minimum`. Per la pianificazione della capacità e la stima dei costi, potremmo specificare un periodo di tempo più lungo e utilizzare la statistica `Average`. In questo modo, potremmo determinare i valori di capacità tipici durante i periodi di alta attività, bassa attività e stato di pausa. Per esaminare il comportamento nei

momenti specifici della messa in pausa e della ripresa, potremmo specificare un periodo pari a un secondo ed esaminare un intervallo di tempo più breve. I valori del timestamp nell'output, ad esempio `2024-08-02T22:13:00+00:00`, mostrano il formato con cui specificare parametri precisi per le opzioni `--start-time` e `--end-time`.

Risoluzione dei problemi relativi alla pausa automatica Aurora Serverless v2

Se le istanze Aurora Serverless v2 non entrano in pausa quando previsto, verificare le seguenti possibili cause:

- Verificare che la versione di Aurora in uso supporti una capacità minima di zero ACU. Per gli intervalli di capacità delle diverse versioni di Aurora, consulta Capacità di Aurora Serverless v2.
- Verificare che il valore di capacità minima per il cluster sia impostato su zero ACU.
- Verificare che l'istanza in questione stia effettivamente utilizzando la classe di istanza Aurora Serverless v2 `db.serverless`, non una delle classi di istanza con provisioning.
- Verificare che il cluster non stia utilizzando funzionalità o impostazioni incompatibili di Prerequisiti e limitazioni per la funzionalità di pausa automatica Aurora Serverless v2.
- Esaminare il file di log per controllare quando sono state messe in pausa o riprese le istanze Aurora Serverless v2 o quando Aurora non è stata in grado di mettere in pausa o riprendere un'istanza per qualche motivo. Per ulteriori informazioni, consulta Monitoraggio delle attività di pausa e ripresa Aurora Serverless v2.
- Controllare se alcuni client o applicazioni mantengono le connessioni aperte per lunghi periodi di tempo. Controllare anche se le applicazioni che utilizzano l'API dati RDS o le funzioni Lambda inviano richieste frequenti che non consentono all'istanza di restare inattiva abbastanza a lungo da entrare in pausa. È possibile esaminare le metriche CloudWatch, ad esempio `ConnectionAttempts` e `DatabaseConnections`. Per ulteriori informazioni, consulta Parametri a livello di istanza per Amazon Aurora.
- Se un'istanza di lettura non entra mai in pausa o se accade raramente, verificare la sua priorità di failover. Se l'istanza di lettura viene utilizzata per il dimensionamento della lettura e non come istanza di lettura in standby in caso di failover, impostarla su una priorità compresa tra 2 e 15.
- Se l'istanza di scrittura non entra mai in pausa o se accade raramente, verificare come vengono utilizzate le istanze di lettura. L'istanza di scrittura può essere messa in pausa solo se l'intero cluster è inattivo, perché una connessione a qualsiasi istanza di lettura ne causa la ripresa.
- Se l'applicazione riceve dei timeout durante le richieste di connessione quando le istanze Aurora Serverless v2 sono in fase di ripresa, valutare la possibilità di allungare l'intervallo di timeout utilizzato dal codice dell'applicazione o dal framework di database sottostante. Timeout di

connessione più lunghi riducono la possibilità di errori di connessione durante la ripresa delle istanze Aurora Serverless v2. Tuttavia, i timeout più lunghi possono anche rallentare il rilevamento dei problemi di disponibilità nel cluster da parte dell'applicazione.

In alternativa, valutare la possibilità di allungare l'intervallo della pausa automatica in modo che le applicazioni non incontrino spesso istanze in pausa.

Se il comportamento della pausa automatica e la reattività del cluster per l'applicazione non sembrano avere una coerenza logica, il cluster potrebbe non essere un candidato adatto per l'utilizzo della funzionalità di pausa automatica.

- Quando si prova a stimare per quanto tempo le istanze Aurora Serverless v2 restano in pausa, è importante tenere presente che esistono fattori che rendono difficile formulare previsioni precise.
  - Le istanze potrebbero riattivarsi regolarmente per eseguire operazioni di manutenzione, aggiornamenti a versioni secondarie o modifiche ai gruppi di parametri.
  - Nei cluster Multi-AZ, a volte la ripresa di un'istanza causa anche la riattivazione di altre istanze. La ripresa di un'istanza di lettura riattiva sempre anche l'istanza di scrittura. La ripresa dell'istanza di scrittura comporta sempre la ripresa di tutte le istanze di lettura con priorità di failover 0 e 1.

  Consigliamo di misurare il tempo di pausa effettivo in un periodo di diversi giorni, con un carico di lavoro realistico. Queste misurazioni possono poi essere utilizzate per determinare un valore base per il tempo di pausa previsto di un'istanza.
- Si potrebbe rilevare che le operazioni interne, come l'eliminazione MySQL, la pianificazione di eventi MySQL, l'autovacuum PostgreSQL o i processi PostgreSQL pianificati tramite l'estensione `pg_cron`, non sono in esecuzione o non vengono completate. L'istanza non si riattiva automaticamente per eseguire operazioni che non comportano una connessione utente al database. Se tali operazioni interne sono in corso alla scadenza del timeout della pausa automatica, le attività di manutenzione come l'eliminazione MySQL e l'autovacuum PostgreSQL vengono annullati. Anche i processi programmati dalla pianificazione di eventi MySQL o dall'estensione `pg_cron` PostgreSQL vengono annullati se sono in corso quando Aurora avvia l'operazione di pausa.

  Per garantire che l'istanza sia periodicamente attiva per eseguire le operazioni pianificate, è possibile avviare una connessione per riprendere l'istanza prima dell'inizio del processo. L'intervallo di timeout della pausa automatica può anche essere aumentato per prolungare le operazioni come l'autovacuum una volta che l'attività dell'utente è terminata. È inoltre possibile

utilizzare meccanismi come le funzioni Lambda per eseguire le operazioni del database in base a una pianificazione, in modo da riattivare automaticamente l'istanza quando necessario.

Considerazioni sulla progettazione delle applicazioni per la funzionalità di pausa automatica Aurora Serverless v2

Quando un'istanza database Aurora Serverless v2 riprende dopo essere entrata in pausa automaticamente, inizia con una capacità relativamente bassa, quindi aumenta verticalmente. Questa capacità iniziale si applica anche se, subito prima della pausa automatica, l'istanza database aveva una capacità maggiore.

Utilizzare questa funzionalità con applicazioni che possono tollerare un intervallo di circa 15 secondi quando si stabilisce una connessione. Si tratta di un tipico caso in cui un'istanza Aurora Serverless v2 viene riattivata a causa di una o di poche connessioni in ingresso. Se l'istanza entra in pausa per più di 24 ore, il tempo di ripresa potrebbe essere più lungo.

Se l'applicazione utilizzava già Aurora Serverless v1 e la relativa funzionalità di pausa automatica, gli intervalli di timeout potrebbero essere già impostati per i tentativi di connessione. Se Aurora Serverless v2 è già utilizzato in combinazione con la funzionalità di arresto/avvio del cluster Aurora, il tempo di ripresa per le istanze Aurora Serverless v2 con pausa automatica è in genere molto più breve rispetto a quello necessario per avviare un cluster arrestato.

Quando si codifica la logica di connessione nell'applicazione, riprovare la connessione se il primo tentativo restituisce un errore che ha una causa temporanea. (Se l'errore riguarda l'autenticazione, correggere le credenziali prima di riprovare). Un errore che si verifica immediatamente dopo la ripresa potrebbe riguardare un timeout o i limiti del database. Un nuovo tentativo può risolvere i casi, più rari, in cui la ripresa di un'istanza Aurora Serverless v2 è dovuta a un numero elevato di richieste di connessione simultanee. In questo caso, l'elaborazione di alcune connessioni potrebbe richiedere più tempo del previsto o superare il limite di connessioni simultanee durante il primo tentativo.

Durante lo sviluppo e il debug delle applicazioni, non lasciare le sessioni client o gli strumenti di programmazione con connessioni aperte al database. Aurora non metterà in pausa un'istanza se sono aperte connessioni avviate dall'utente, anche se le connessioni non eseguono istruzioni o transazioni SQL. Quando un'istanza Aurora Serverless v2 in un cluster Aurora non può entrare in pausa, è possibile che anche altre istanze del cluster non vengano sospese. Per ulteriori informazioni, consulta Situazioni in cui Aurora Serverless v2 non viene messo in pausa automaticamente.

Aurora emette eventi quando un'istanza database Aurora Serverless v2 inizia e termina la ripresa e se l'istanza non riesce a riattivarsi per qualche motivo. Per dettagli su questi eventi, consultare Eventi di istanza database.

Migrazione a Aurora Serverless v2

Per convertire un cluster database esistente in modo che utilizzi Aurora Serverless v2, eseguire le operazioni seguenti:

- Eseguire l'aggiornamento da un cluster Aurora con provisioning.
- Aggiornamento da un cluster Aurora Serverless v1.
- Migrare da un database on-premise a un cluster Aurora Serverless v2.

Quando il cluster aggiornato esegue la versione del motore appropriata come elencato in Requisiti e limitazioni per Aurora Serverless v2, puoi iniziare ad aggiungere istanze database Aurora Serverless v2. La prima istanza database aggiunta al cluster aggiornato deve essere un'istanza database con provisioning. Sarà quindi passare all'elaborazione del carico di lavoro di scrittura, del carico di lavoro di lettura o di entrambi nell'istanze database Aurora Serverless v2.

Indice

- Aggiornamento o conversione di cluster esistenti per l'uso di Aurora Serverless v2
  - Percorsi di aggiornamento per cluster compatibili con MySQL per l'uso di Aurora Serverless v2
  - Percorsi di aggiornamento per cluster compatibili con PostgreSQL per l'uso di Aurora Serverless v2
- Passaggio di un cluster con provisioning ad Aurora Serverless v2
- Confronto tra Aurora Serverless v2 e Aurora Serverless v1
  - Confronto dei requisiti di Aurora Serverless v2 e Aurora Serverless v1
  - Confronto di dimensionamento e disponibilità tra Aurora Serverless v2 e Aurora Serverless v1
  - Confronto del supporto delle caratteristiche di Aurora Serverless v2 e Aurora Serverless v1
  - Adattamento dei casi d'uso di Aurora Serverless v1 ad Aurora Serverless v2
- Aggiornamento da un cluster Aurora Serverless v1 ad Aurora Serverless v2
  - Cluster database compatibili con Aurora MySQL
  - Cluster database compatibili con Aurora PostgreSQL
- Migrazione di un database on-premise a Aurora Serverless v2

> **Note**
>
> Questi argomenti descrivono come convertire un cluster database esistente. Per ulteriori informazioni sulla creazione di un nuovo cluster database Aurora Serverless v2, consulta Creazione di un cluster di database che utilizza Aurora Serverless v2.

Aggiornamento o conversione di cluster esistenti per l'uso di Aurora Serverless v2

Se il cluster con provisioning dispone di una versione del motore che supportaAurora Serverless v2, il passaggio ad Aurora Serverless v2 non richiede un aggiornamento. In questo caso, puoi aggiungere istanze database Aurora Serverless v2 al cluster originale. È possibile cambiare il cluster in modo che utilizzi tutte le istanze database Aurora Serverless v2. È inoltre possibile utilizzare una combinazione di istanze database Aurora Serverless v2 e con provisioning nello stesso cluster di database. Per le versioni del motore Aurora che supportano Aurora Serverless v2, consulta Regioni e motori di database Aurora supportati per Aurora Serverless v2.

Se utilizzi una precedente versione del motore che non supporta Aurora Serverless v2, esegui le seguenti operazioni di carattere generale:

1. Aggiornare il cluster.
2. Creare un'istanza database di scrittura con provisioning per il cluster aggiornato.
3. Modificare il cluster in modo che utilizzi istanze database Aurora Serverless v2.

> **Important**
>
> Quando esegui un aggiornamento della versione principale su una versione compatibile con Aurora Serverless v2 utilizzando il ripristino o la clonazione di snapshot, la prima istanza database aggiunta al nuovo cluster deve essere un'istanza database con provisioning. Questa aggiunta avvia la fase finale del processo di aggiornamento.
> Fino a quando non si verifica la fase finale, il cluster non dispone dell'infrastruttura necessaria per il supporto di Aurora Serverless v2. Pertanto, i cluster aggiornati iniziano sempre con un'istanza database di scrittura con provisioning. Sarà quindi possibile convertire o eseguire il failover dell'istanza database con provisioning in un'istanza Aurora Serverless v2.

L'aggiornamento da Aurora Serverless v1 in Aurora Serverless v2 comporta la creazione di un cluster con provisioning come fase intermedia. Vengono quindi eseguiti gli stessi passaggi di aggiornamento utilizzati quando si inizia con un cluster con provisioning.

Percorsi di aggiornamento per cluster compatibili con MySQL per l'uso di Aurora Serverless v2

Se il cluster originale esegue Aurora MySQL, è necessario scegliere la procedura appropriata in base alla versione e alla modalità del motore del cluster.

Se il cluster Aurora MySQL originale è	Procedura per il passaggio ad Aurora Serverless v2
Cluster con provisioning che esegue Aurora MySQL versione 3, compatibile con MySQL 8.0	Questa è la fase finale per tutte le conversioni da cluster Aurora MySQL esistenti. Se necessario, eseguire un aggiornamento della versione secondaria alla versione 3.02.0 o successiva. Utilizzare un'istanza database con provisioning per l'istanza database di scrittura . Aggiungere un'istanza database Aurora Serverless v2 di lettura. Eseguire un failover per convertire l'istanza in istanza database di scrittura. (Facoltativo) Convertire altre istanze database di cui è stato effettuato il provisioning nel cluster in Aurora Serverless v2. Oppure aggiungere nuove istanze database Aurora Serverless v2 e rimuovere le istanze database con provisioning. Per la procedura completa e alcuni esempi, consultare Passaggio di un cluster con provisioning ad Aurora Serverless v2.
Cluster con provisioning che esegue Aurora MySQL versione 2, compatibile con MySQL 5.7	Eseguire un aggiornamento della versione principale ad Aurora MySQL versione 3.02.0 o successive. Eseguire quindi la procedura

Se il cluster Aurora MySQL originale è	Procedura per il passaggio ad Aurora Serverless v2
	per Aurora MySQL versione 3 per convertire il cluster per l'uso di istanze database Aurora Serverless v2.
Cluster Aurora Serverless v1 che esegue Aurora MySQL versione 2, compatibile con MySQL 5.7	Per informazioni sulla pianificazione della conversione da Aurora Serverless v1, prima di iniziare consultare Confronto tra Aurora Serverless v2 e Aurora Serverless v1. Quindi segui la procedura in Aggiornamento da un cluster Aurora Serverless v1 ad Aurora Serverless v2.

Percorsi di aggiornamento per cluster compatibili con PostgreSQL per l'uso di Aurora Serverless v2

Se il cluster originale esegue Aurora PostgreSQL, scegliere la procedura appropriata in base alla versione e alla modalità del motore del cluster.

Se il cluster Aurora PostgreSQL originale è	Procedura per il passaggio ad Aurora Serverless v2
Cluster con provisioning che esegue Aurora PostgreSQL versione 13	Questa è la fase finale per tutte le conversioni da cluster Aurora PostgreSQL esistenti. Se necessario, eseguire un aggiornamento della versione secondaria alla versione 13.6 o successiva. Aggiungere un'istanza database con provisioning per l'istanza database di scrittura. Aggiungere un'istanza database Aurora Serverless v2 di lettura. Eseguire un failover per convertire tale istanza Aurora Serverless v2 in istanza database di scrittura.

Se il cluster Aurora PostgreSQL originale è	Procedura per il passaggio ad Aurora Serverless v2
	(Facoltativo) Convertire altre istanze database di cui è stato effettuato il provisioning nel cluster in Aurora Serverless v2. Oppure aggiungere nuove istanze database Aurora Serverless v2 e rimuovere le istanze database con provisioning. Per la procedura completa e alcuni esempi, consultare Passaggio di un cluster con provisioning ad Aurora Serverless v2.
Cluster con provisioning che esegue Aurora PostgreSQL versione 11 o 12	Eseguire un aggiornamento della versione principale ad Aurora PostgreSQL versione 13.6 o successiva. Eseguire quindi la procedura per Aurora PostgreSQL versione 1.3 per convertire il cluster per l'uso delle istanze database Aurora Serverless v2.
Cluster Aurora Serverless v1 che esegue Aurora PostgreSQL versione 11 o 13	Per informazioni sulla pianificazione della conversione da Aurora Serverless v1, prima di iniziare consultare Confronto tra Aurora Serverless v2 e Aurora Serverless v1. Quindi segui la procedura in Aggiornamento da un cluster Aurora Serverless v1 ad Aurora Serverless v2.

Passaggio di un cluster con provisioning ad Aurora Serverless v2

Per convertire un cluster con provisioning per l'uso di Aurora Serverless v2, procedi nel seguente modo:

1. Verificare se è necessario aggiornare il cluster con provisioning per poter essere utilizzato con istanze database Aurora Serverless v2. Per le versioni Aurora compatibili con Aurora Serverless v2, consultare Requisiti e limitazioni per Aurora Serverless v2.

Se il cluster con provisioning esegue una versione del motore che non è disponibile per Aurora Serverless v2, aggiornare la versione del motore del cluster:

- Se disponi di un cluster con provisioning compatibile con MySQL 5.7, segui le istruzioni di aggiornamento per Aurora MySQL versione 3. Utilizzare la procedura descritta in Come eseguire un aggiornamento in loco.
- Se disponi di un cluster con provisioning compatibile con PostgreSQL che esegue PostgreSQL versione 11 o 12, segui le istruzioni di aggiornamento per Aurora PostgreSQL versione 13. Utilizzare la procedura descritta in Esecuzione di un aggiornamento alla versione principale.

2. Configurare qualsiasi altra proprietà del cluster in modo da soddisfare i requisiti Aurora Serverless v2 riportati in Requisiti e limitazioni per Aurora Serverless v2.
3. Definire la configurazione di dimensionamento per il cluster. Segui la procedura riportata in Impostazione dell'intervallo di capacità di Aurora Serverless v2 per un cluster.
4. Aggiungere una o più istanze database Aurora Serverless v2 al cluster. Eseguire la procedura generale descritta in Aggiunta di repliche di Aurora a un cluster di database. Per ogni nuova istanza DB, specifica il nome speciale della classe di istanza DB Serverless nella Console di gestione AWS AWS CLI o `db.serverless` nell'API Amazon RDS.

   In alcuni casi, è possibile che nel cluster siano già presenti una o più istanze database di lettura con provisioning. In questo caso, è possibile convertire una delle istanze di lettura in un'Istanza database Aurora Serverless v2 invece di creare una nuova istanza database. A tale scopo, segui la procedura in Conversione di un'istanza di lettura o scrittura con provisioning per Aurora Serverless v2.
5. Eseguire un'operazione di failover per convertire una delle istanze database Aurora Serverless v2l'istanza database di scrittura per il cluster.
6. (Facoltativo) Convertire le istanze database con provisioning in Aurora Serverless v2 o rimuoverle dal cluster. Eseguire la procedura generale descritta in Conversione di un'istanza di lettura o scrittura con provisioning per Aurora Serverless v2 o Eliminazione di un'istanza database da un cluster database Aurora.

> **Tip**
>
> La rimozione delle istanze database con provisioning non è obbligatoria. È possibile configurare un cluster contenente sia istanze database Aurora Serverless v2 che istanze database con provisioning. Tuttavia, fino a quando non si acquisisce una certa familiarità

con le prestazioni e le caratteristiche di dimensionamento delle istanze database Aurora Serverless v2, si consiglia di configurare i cluster con istanze database dello stesso tipo.

L' AWS CLI esempio seguente mostra il processo di switchover utilizzando un cluster con provisioning che esegue Aurora MySQL versione 3.02.0. Il cluster è denominato `mysql-80`. Inizialmente il cluster include due istanze database con provisioning denominate `provisioned-instance-1` e `provisioned-instance-2`, ovvero un'istanza di scrittura e una di lettura. Entrambe usano la classe di istanza database `db.r6g.large`.

```
$ aws rds describe-db-clusters --db-cluster-identifier mysql-80 \
  --query '*[].[DBClusterIdentifier,DBClusterMembers[*].
[DBInstanceIdentifier,IsClusterWriter]]' --output text
mysql-80
provisioned-instance-2     False
provisioned-instance-1     True

$ aws rds describe-db-instances --db-instance-identifier provisioned-instance-1 \
  --output text --query '*[].[DBInstanceIdentifier,DBInstanceClass]'
provisioned-instance-1     db.r6g.large

$ aws rds describe-db-instances --db-instance-identifier provisioned-instance-2 \
  --output text --query '*[].[DBInstanceIdentifier,DBInstanceClass]'
provisioned-instance-2     db.r6g.large
```

Viene creata una tabella contenente alcuni dati. In questo modo, è possibile confermare che i dati e il funzionamento del cluster sono gli stessi prima e dopo il passaggio.

```
mysql> create database serverless_v2_demo;
mysql> create table serverless_v2_demo.demo (s varchar(128));
mysql> insert into serverless_v2_demo.demo values ('This cluster started with a
 provisioned writer.');
Query OK, 1 row affected (0.02 sec)
```

Come prima cosa, al cluster viene aggiunto un intervallo di capacità. Se non si esegue questa operazione, viene visualizzato un errore quando al cluster vengono aggiunte istanze database Aurora Serverless v2. Se utilizziamo il Console di gestione AWS per questa procedura, tale passaggio è automatico quando aggiungiamo la prima istanza DB. Aurora Serverless v2

```
$ aws rds create-db-instance --db-instance-identifier serverless-v2-instance-1 \
```

```
  --db-cluster-identifier mysql-80 --db-instance-class db.serverless --engine aurora-
mysql

An error occurred (InvalidDBClusterStateFault) when calling the CreateDBInstance
 operation:
Set the Serverless v2 scaling configuration on the parent DB cluster before creating a
 Serverless v2 DB instance.

$ # The blank ServerlessV2ScalingConfiguration attribute confirms that the cluster
 doesn't have a capacity range set yet.
$ aws rds describe-db-clusters --db-cluster-identifier mysql-80 --query
 'DBClusters[*].ServerlessV2ScalingConfiguration'
[]

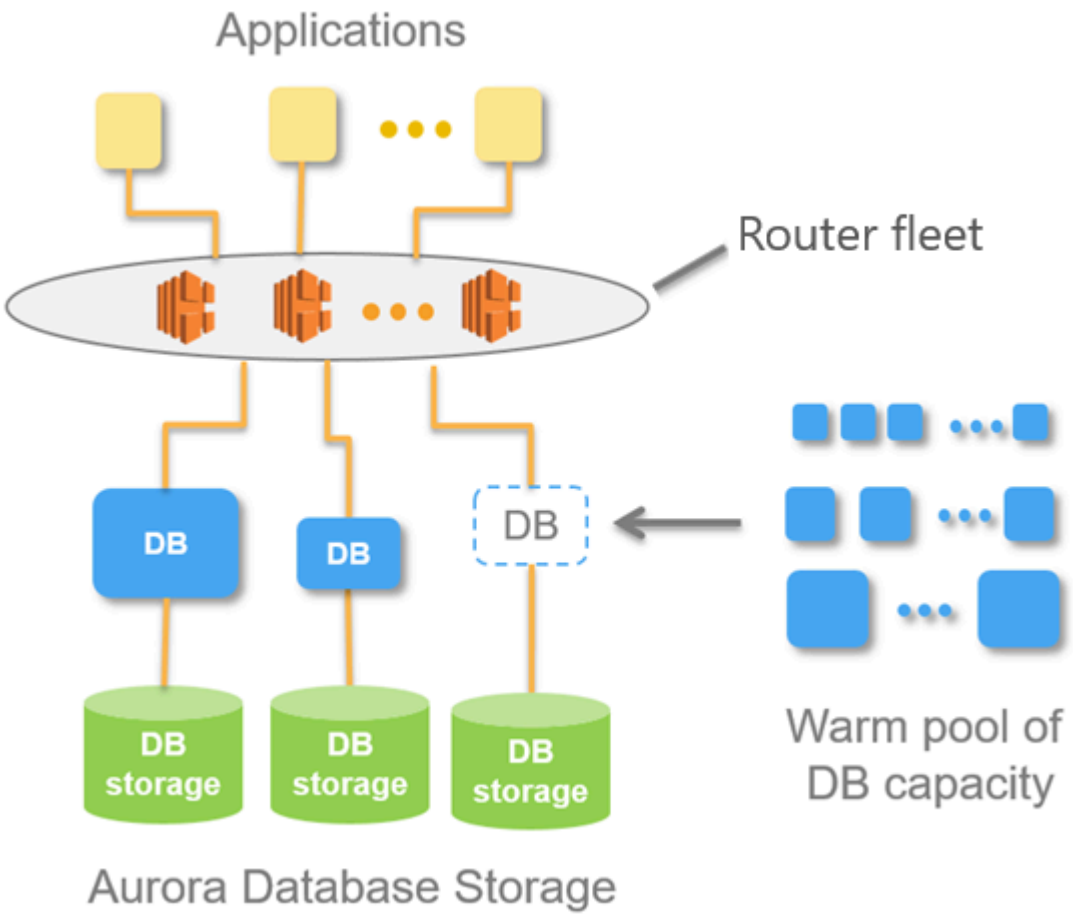
$ aws rds modify-db-cluster --db-cluster-identifier mysql-80 \
  --serverless-v2-scaling-configuration MinCapacity=0.5,MaxCapacity=16
{
  "DBClusterIdentifier": "mysql-80",
  "ServerlessV2ScalingConfiguration": {
    "MinCapacity": 0.5,
    "MaxCapacity": 16
  }
}
```

Vengono creati due lettori Aurora Serverless v2 per sostituire le istanze database originali. Questa operazione viene eseguita specificando la classe di istanza database `db.serverless` per le nuove istanze database.

```
$ aws rds create-db-instance --db-instance-identifier serverless-v2-instance-1 --db-
cluster-identifier mysql-80 --db-instance-class db.serverless --engine aurora-mysql
{
  "DBInstanceIdentifier": "serverless-v2-instance-1",
  "DBClusterIdentifier": "mysql-80",
  "DBInstanceClass": "db.serverless",
  "DBInstanceStatus": "creating"
}

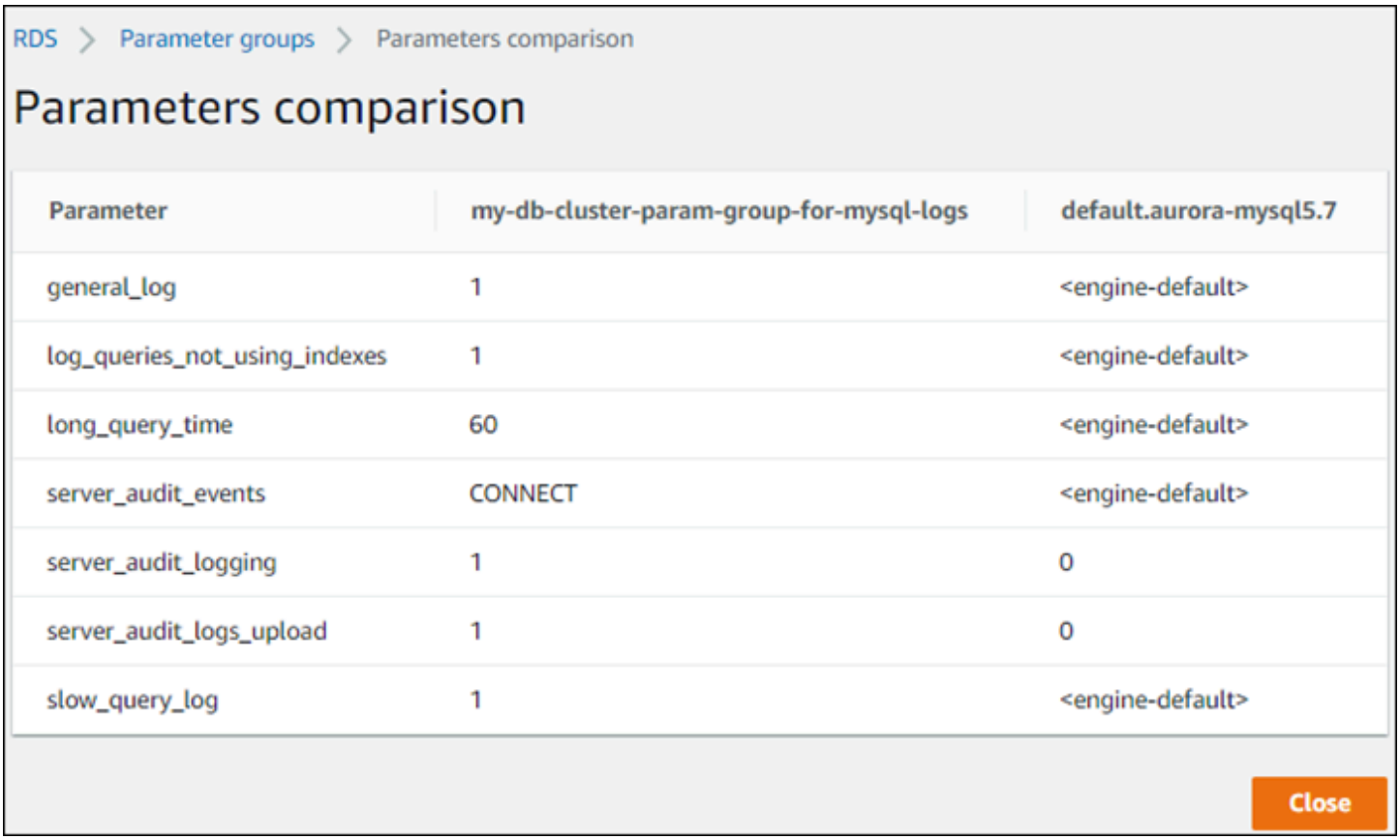
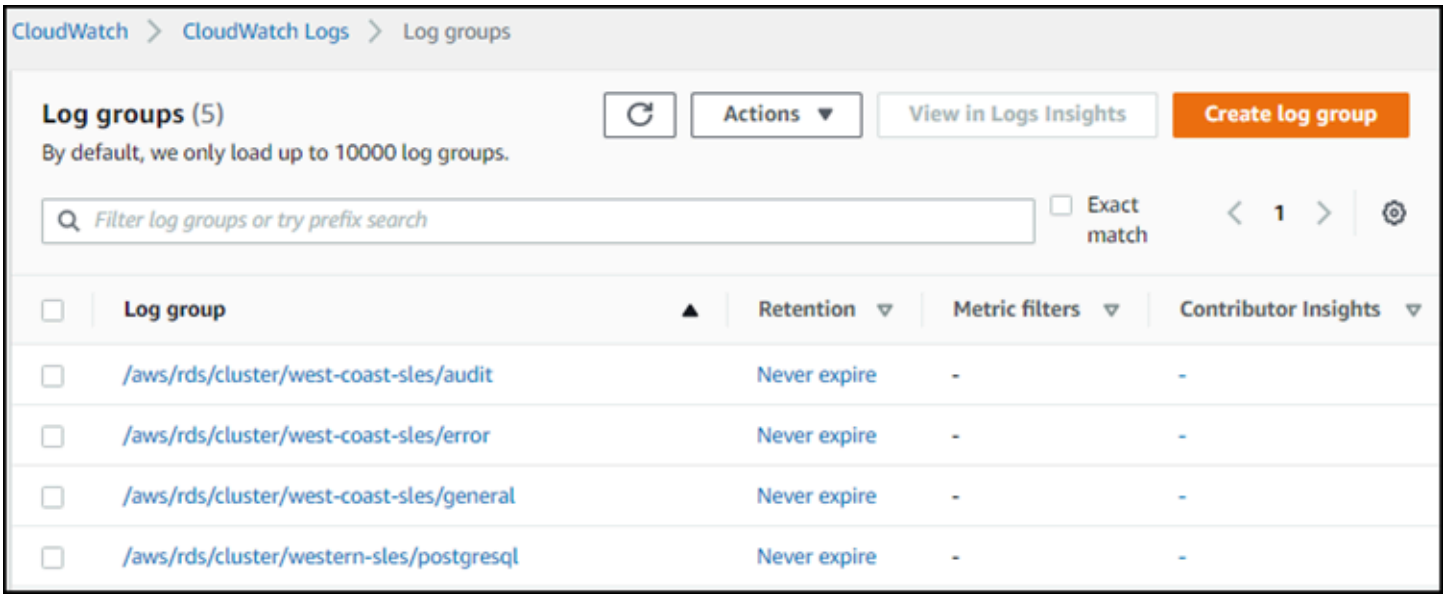
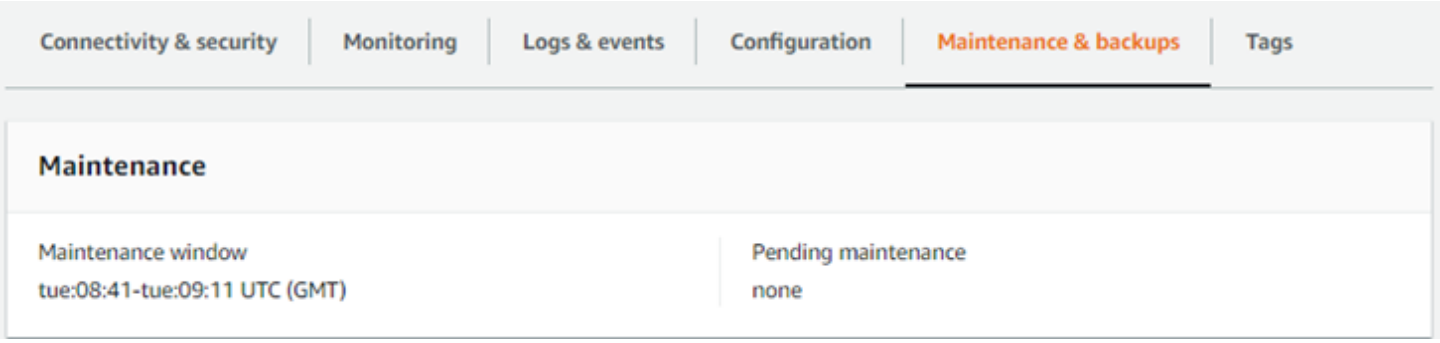
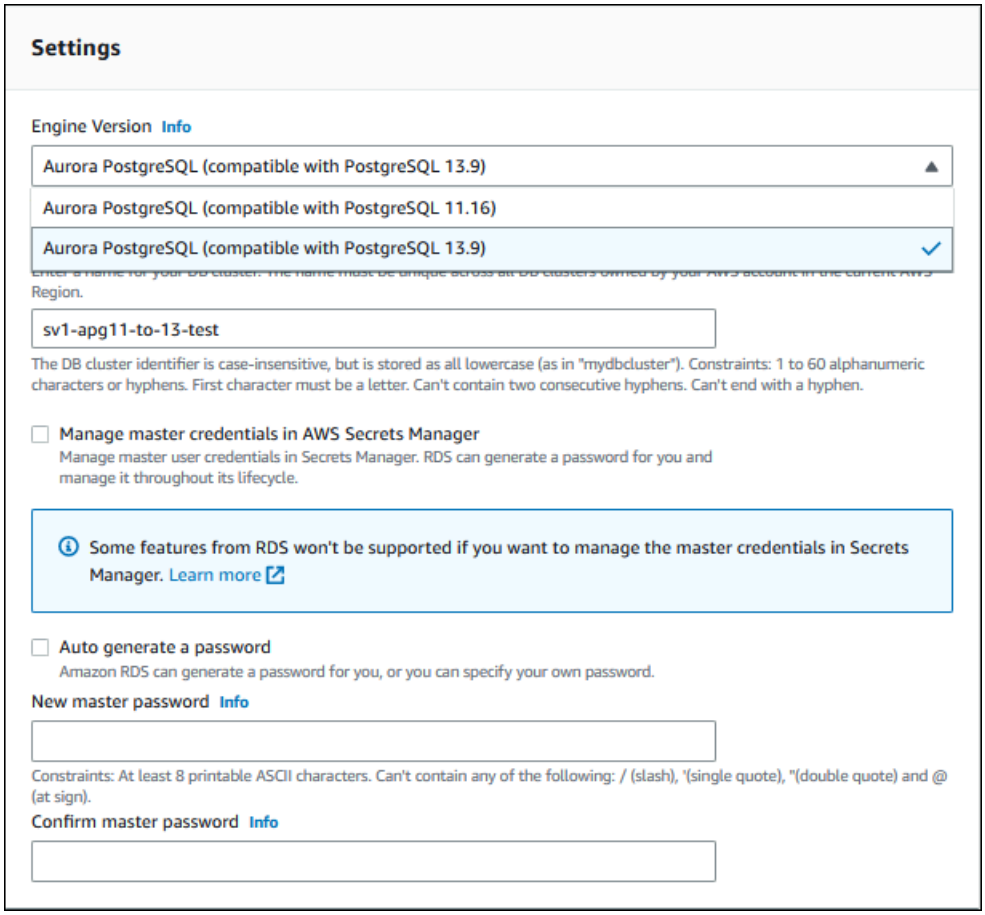
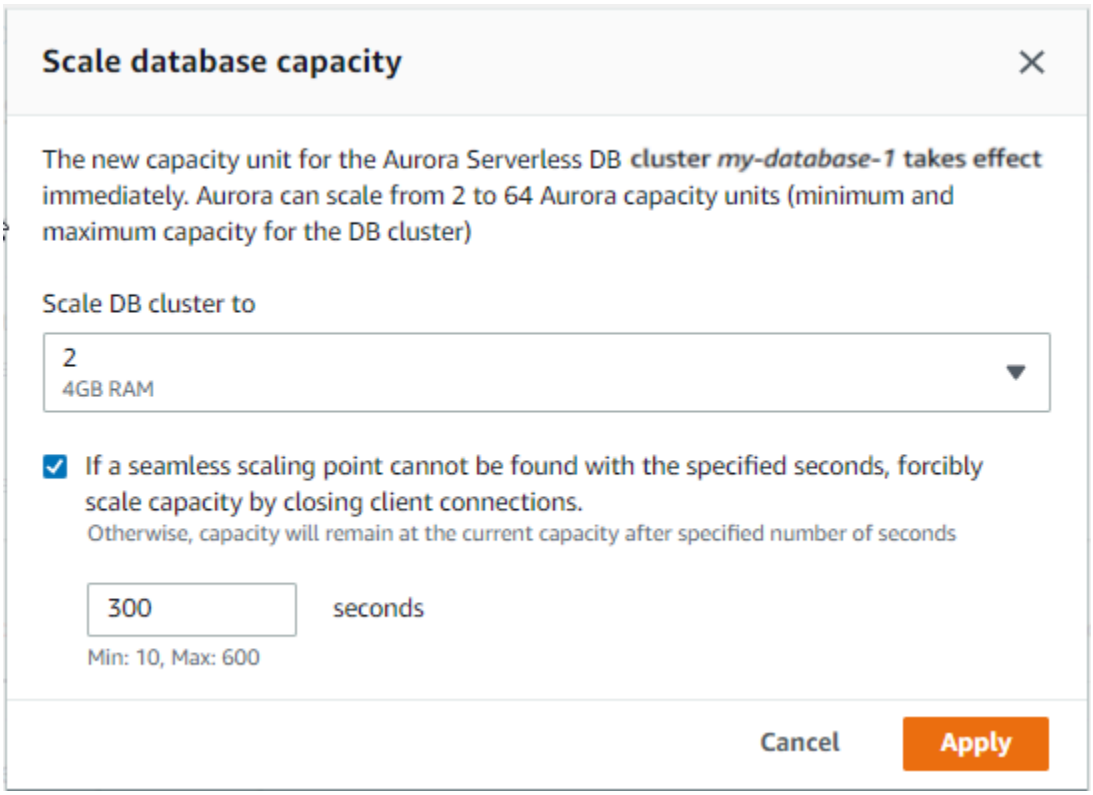
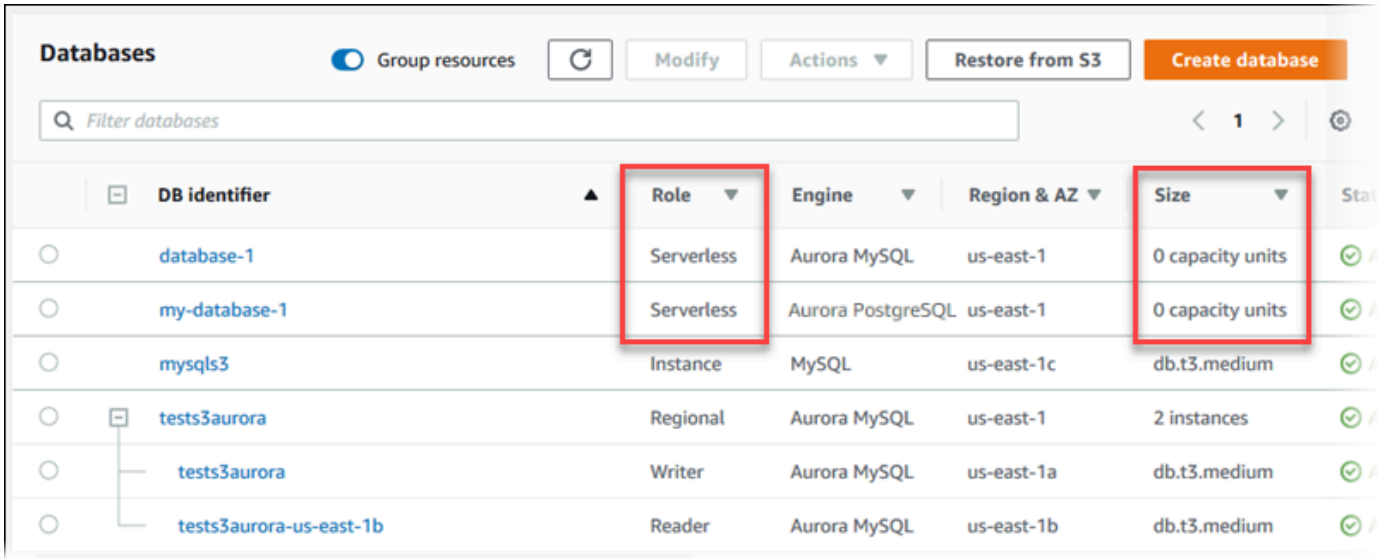
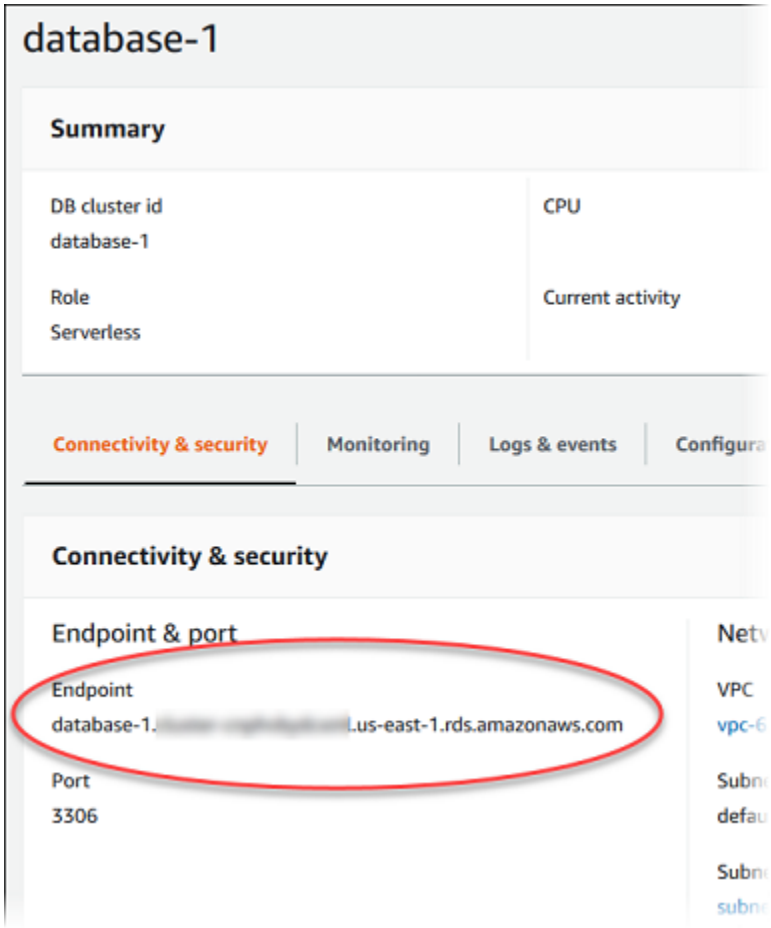
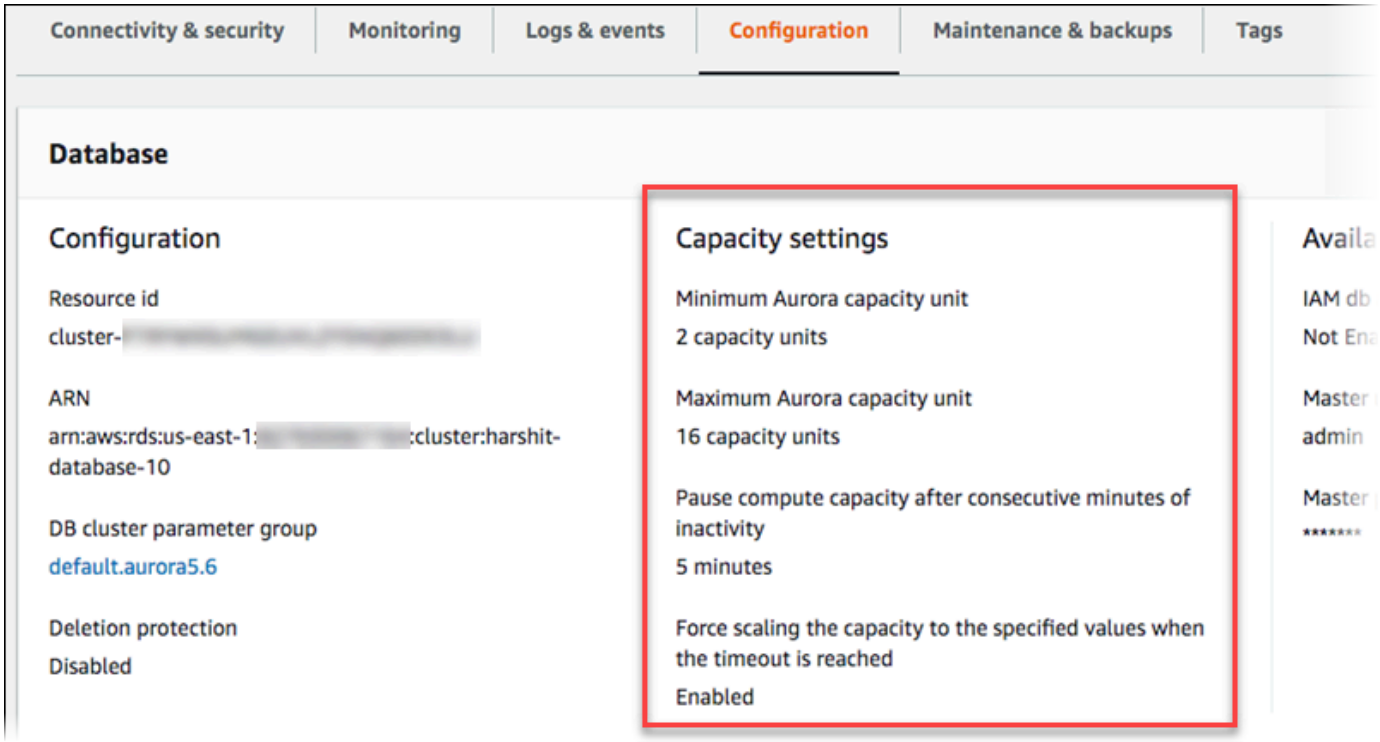
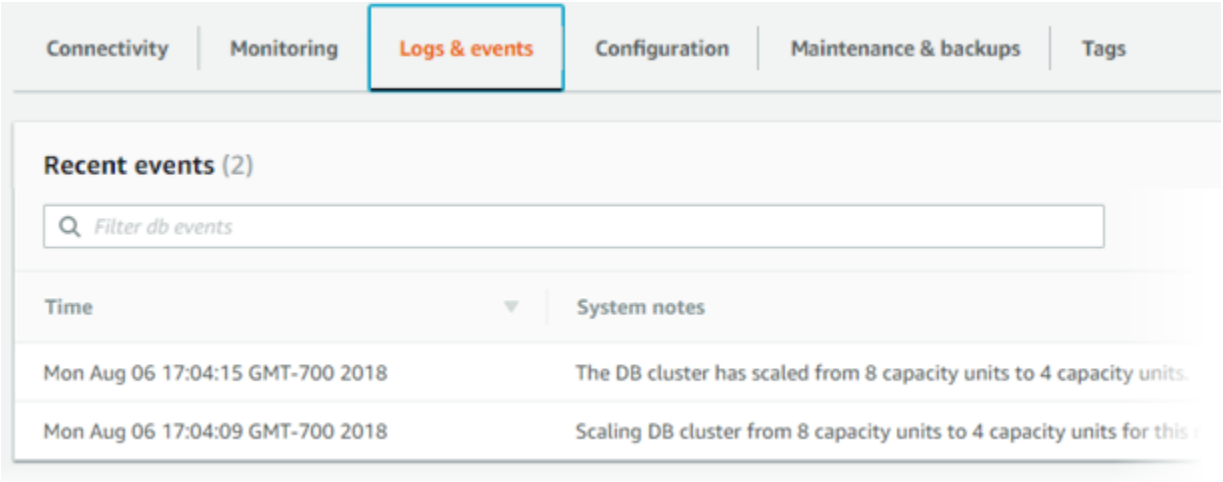
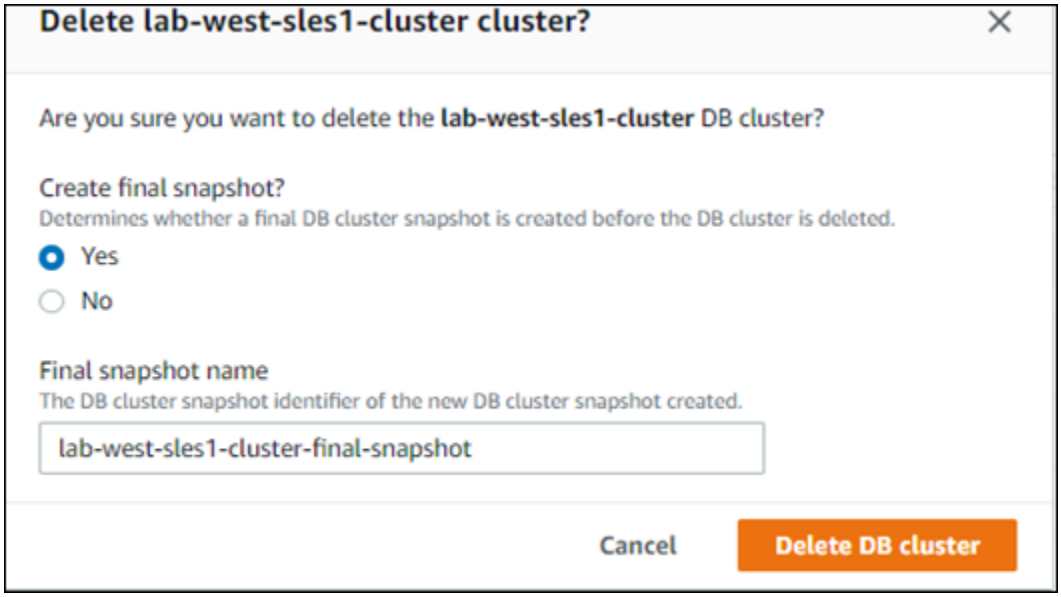
$ aws rds create-db-instance --db-instance-identifier serverless-v2-instance-2 \
  --db-cluster-identifier mysql-80 --db-instance-class db.serverless --engine aurora-
mysql
{
  "DBInstanceIdentifier": "serverless-v2-instance-2",
  "DBClusterIdentifier": "mysql-80",
```

```
  "DBInstanceClass": "db.serverless",
  "DBInstanceStatus": "creating"
}

$ # Wait for both DB instances to finish being created before proceeding.
$ aws rds wait db-instance-available --db-instance-identifier serverless-v2-instance-1
 && \
  aws rds wait db-instance-available --db-instance-identifier serverless-v2-instance-2
```

Viene eseguito un failover per impostare una delle istanze database Aurora Serverless v2 come nuova istanza di scrittura per il cluster.

```
$ aws rds failover-db-cluster --db-cluster-identifier mysql-80 \
  --target-db-instance-identifier serverless-v2-instance-1
{
  "DBClusterIdentifier": "mysql-80",
  "DBClusterMembers": [
    {
      "DBInstanceIdentifier": "serverless-v2-instance-1",
      "IsClusterWriter": false,
      "DBClusterParameterGroupStatus": "in-sync",
      "PromotionTier": 1
    },
    {
      "DBInstanceIdentifier": "serverless-v2-instance-2",
      "IsClusterWriter": false,
      "DBClusterParameterGroupStatus": "in-sync",
      "PromotionTier": 1
    },
    {
      "DBInstanceIdentifier": "provisioned-instance-2",
      "IsClusterWriter": false,
      "DBClusterParameterGroupStatus": "in-sync",
      "PromotionTier": 1
    },
    {
      "DBInstanceIdentifier": "provisioned-instance-1",
      "IsClusterWriter": true,
      "DBClusterParameterGroupStatus": "in-sync",
      "PromotionTier": 1
    }
  ],
  "Status": "available"
```

```
}
```

Sono necessari alcuni secondi per rendere effettiva la modifica. A questo punto, sono disponibili un'istanza Aurora Serverless v2 di scrittura e un'istanza Aurora Serverless v2 di lettura. Pertanto, le istanze database con provisioning originali non sono più necessarie.

```
$ aws rds describe-db-clusters --db-cluster-identifier mysql-80 \
  --query '*[].[DBClusterIdentifier,DBClusterMembers[*].
[DBInstanceIdentifier,IsClusterWriter]]' \
  --output text
mysql-80
serverless-v2-instance-1        True
serverless-v2-instance-2        False
provisioned-instance-2          False
provisioned-instance-1          False
```

L'ultimo passaggio della procedura di conversione prevede l'eliminazione delle istanze database con provisioning.

```
$ aws rds delete-db-instance --db-instance-identifier provisioned-instance-2 --skip-
final-snapshot
{
  "DBInstanceIdentifier": "provisioned-instance-2",
  "DBInstanceStatus": "deleting",
  "Engine": "aurora-mysql",
  "EngineVersion": "8.0.mysql_aurora.3.02.0",
  "DBInstanceClass": "db.r6g.large"
}

$ aws rds delete-db-instance --db-instance-identifier provisioned-instance-1 --skip-
final-snapshot
{
  "DBInstanceIdentifier": "provisioned-instance-1",
  "DBInstanceStatus": "deleting",
  "Engine": "aurora-mysql",
  "EngineVersion": "8.0.mysql_aurora.3.02.0",
  "DBInstanceClass": "db.r6g.large"
}
```

Come controllo finale, si verifica che la tabella originale sia accessibile e scrivibile dall'istanza database Aurora Serverless v2 di scrittura.

```
mysql> select * from serverless_v2_demo.demo;
+--------------------------------------------------+
| s                                                |
+--------------------------------------------------+
| This cluster started with a provisioned writer.  |
+--------------------------------------------------+
1 row in set (0.00 sec)

mysql> insert into serverless_v2_demo.demo values ('And it finished with a Serverless
 v2 writer.');
Query OK, 1 row affected (0.01 sec)

mysql> select * from serverless_v2_demo.demo;
+--------------------------------------------------+
| s                                                |
+--------------------------------------------------+
| This cluster started with a provisioned writer.  |
| And it finished with a Serverless v2 writer.     |
+--------------------------------------------------+
2 rows in set (0.01 sec)
```

Viene inoltre stabilita una connessione all'istanza database Aurora Serverless v2 di lettura e si verifica che in essa siano disponibili in nuovi dati scritti.

```
mysql> select * from serverless_v2_demo.demo;
+--------------------------------------------------+
| s                                                |
+--------------------------------------------------+
| This cluster started with a provisioned writer.  |
| And it finished with a Serverless v2 writer.     |
+--------------------------------------------------+
2 rows in set (0.01 sec)
```

Confronto tra Aurora Serverless v2 e Aurora Serverless v1

Se usi già Aurora Serverless v1, puoi scoprire ulteriori informazioni sulle principali differenze traAurora Serverless v1 e Aurora Serverless v2. Le differenze a livello di architettura, ad esempio il supporto delle istanze database di lettura, offrono nuovi tipi di casi d'uso.

Puoi fare riferimento alle seguenti tabelle per informazioni dettagliate sulle principali differenze tra Aurora Serverless v2 e Aurora Serverless v1.

Argomenti

- Confronto dei requisiti di Aurora Serverless v2 e Aurora Serverless v1
- Confronto di dimensionamento e disponibilità tra Aurora Serverless v2 e Aurora Serverless v1
- Confronto del supporto delle caratteristiche di Aurora Serverless v2 e Aurora Serverless v1
- Adattamento dei casi d'uso di Aurora Serverless v1 ad Aurora Serverless v2

Confronto dei requisiti di Aurora Serverless v2 e Aurora Serverless v1

La tabella seguente fornisce un riepilogo dei diversi requisiti per l'esecuzione di database con Aurora Serverless v2 o Aurora Serverless v1. Rispetto ad Aurora Serverless v1, Aurora Serverless v2 offre versioni superiori dei motori Aurora PostgreSQL e Aurora MySQL.

Funzionalità	Requisiti di Aurora Serverless v2	Requisiti di Aurora Serverless v1
Motori database	Aurora MySQL, Aurora PostgreSQL	Aurora MySQL, Aurora PostgreSQL
Versioni supportate di Aurora MySQL	Per informazioni, consulta Aurora Serverless v2 con Aurora MySQL.	Per informazioni, consulta Aurora Serverless v1 con Aurora MySQL.
Versioni supportate di Aurora PostgreSQL	Per informazioni, consulta Aurora Serverless v2 con Aurora PostgreSQL.	Per informazioni, consulta Aurora Serverless v1 con Aurora PostgreSQL.
Aggiornamento di un cluster di database	Come per i cluster di database sottoposti a provisioning, puoi eseguire gli aggiornam enti manualmente senza dover attendere che Aurora aggiorni il cluster di database automaticamente. Per ulteriori informazioni, consulta Modifica di un cluster database Amazon Aurora.	Gli aggiornamenti della versione secondaria vengono applicati automaticamente non appena diventano disponibi li. Per ulteriori informazi oni, consulta Versioni del motore del database di Aurora Serverless v1 e Aurora. È possibile eseguire manualmente gli aggiornam

Funzionalità	Requisiti di Aurora Serverless v2	Requisiti di Aurora Serverless v1
	Note Per eseguire un aggiornamento della versione principal e da 13.x a 14.x o 15.x per un cluster DB compatibile con Aurora PostgreSQL, la capacità massima del cluster deve essere almeno 2. ACUs	enti della versione principal e. Per ulteriori informazioni, consulta Modifica di un cluster di database Aurora Serverless v1.

Funzionalità	Requisiti di Aurora Serverless v2	Requisiti di Aurora Serverless v1
Conversione da cluster di database con provisioning	È possibile utilizzare i seguenti metodi: • Aggiungere una o più istanze database Aurora Serverless v2 di lettura a un cluster con provisioning esistente. Per utilizzare Aurora Serverless v2 per l'istanza di scrittura, eseguire un failover su una delle istanze database Aurora Serverless v2. Affinché l'intero cluster utilizzi le istanze database Aurora Serverless v2, rimuovere le istanze database di scrittura con provisioning dopo aver promosso l'istanza database Aurora Serverless v2 a istanza di scrittura. • Creare un nuovo cluster con il motore di database e la versione del motore appropriati. Utilizzare un qualsiasi metodo standard. Ad esempio, ripristinare uno snapshot del cluster o creare un clone di un cluster esistente. Scegliere Aurora Serverless v2 per alcune o tutte le istanze database nel nuovo cluster.	Ripristinare lo snapshot del cluster con provisioning per creare il nuovo cluster Aurora Serverless v1.

Funzionalità	Requisiti di Aurora Serverless v2	Requisiti di Aurora Serverless v1
	Se si crea il nuovo cluster tramite la clonazione, non è possibile aggiornar e contemporaneamente la versione del motore. Assicurarsi che il cluster originale esegua già una versione del motore compatibile con Aurora Serverless v2.	
Conversione da cluster Aurora Serverless v1	Segui la procedura riportata in Aggiornamento da un cluster Aurora Serverless v1 ad Aurora Serverless v2.	Non applicabile
Classi di istanze database disponibili	Classe di istanza database speciale `db.serverless` . Nel, è etichettato come Serverless. Console di gestione AWS	Non applicabile. Aurora Serverless v1 utilizza la modalità di motore `serverless` .
Porta	Qualsiasi porta compatibile con MySQL o PostgreSQL	Solo porta MySQL o PostgreSQL di default
Indirizzo IP pubblico consentit o?	Sì	No
Cloud privato virtuale (VPC, Virtual private cloud) obbligato rio?	Sì	Sì. Ogni cluster Aurora Serverless v1 utilizza 2 endpoint di bilanciamento del carico interfaccia e gateway allocati al VPC.

Confronto di dimensionamento e disponibilità tra Aurora Serverless v2 e Aurora Serverless v1

La tabella seguente riepiloga le differenze tra Aurora Serverless v2 e Aurora Serverless v1 relativamente a dimensionamento e disponibilità.

Il dimensionamento in Aurora Serverless v2 è più reattivo, più granulare e meno 'sconvolgente' rispetto al dimensionamento in Aurora Serverless v1. Aurora Serverless v2 può implementare il dimensionamento sia modificando le dimensioni dell'istanza database che aggiungendo altre istanze database al cluster di database. Può anche scalare aggiungendo cluster in altro Regioni AWS a un database globale Aurora. Per contro, Aurora Serverless v1 implementa il dimensionamento solo aumentando o riducendo la capacità dell'istanza di scrittura. Tutte le operazioni di calcolo per un cluster Aurora Serverless v1 vengono eseguite in un'unica zona di disponibilità singola e in un'unica Regione AWS.

Funzionalità di dimension amento ed elevata disponibi lità	Aurora Serverless v2comport amento	Aurora Serverless v1comport amento
Unità di capacità Aurora minima (ACUs) (Aurora MySQL)	0,5 quando il cluster è in esecuzione, 0 quando il cluster è in pausa.	1 quando il cluster è in esecuzione, 0 quando il cluster è in pausa.
Minimo ACUs (Aurora PostgreSQL)	0,5 quando il cluster è in esecuzione, 0 quando il cluster è in pausa.	2 quando il cluster è in esecuzione, 0 quando il cluster è in pausa.
Massimo ACUs (Aurora MySQL)	256	256
Massimo ACUs (Aurora PostgreSQL)	256	384
Arresto di un cluster	È possibile arrestare e avviare manualmente il cluster utilizzando la stessa funzione di arresto e avvio del	Il cluster si interrompe automaticamente dopo un timeout. Alla ripresa dell'atti vità, è necessario attendere

Funzionalità di dimension amento ed elevata disponibi lità	Aurora Serverless v2comport amento	Aurora Serverless v1comport amento
	cluster valida per i cluster con provisioning.	un po' di tempo prima che il cluster torni disponibile.
Dimensionamento delle istanze database	Scalabilità verso l'alto e verso il basso con un incremento minimo di 0,5. ACUs	Scala verso l'alto e verso il basso raddoppiando o dimezzando il. ACUs
Numero di istanze database	Stesso valore valido per un cluster con provisioning: 1 istanza database di scrittura, fino a 15 istanze database di lettura.	1 istanza database che gestisce sia le operazioni di lettura che le operazioni di scrittura.
Il dimensionamento può avvenire durante l'esecuzione di istruzioni SQL?	Sì. Aurora Serverless v2 non richiede l'attesa di un momento di inattività.	No. Ad esempio, il dimension amento attende il completam ento di transazioni con tempi di esecuzione lunghi, tabelle temporanee e blocchi di tabella.
Le istanze database di lettura vengono dimensionate assieme all'istanza di scrittura	Facoltativo	Non applicabile
Spazio di archiviazione massimo	128 TiB	128 TiB
Conservazione della cache del buffer durante il dimension amento	Sì. La cache del buffer viene dimensionata dinamicamente.	No. La cache del buffer viene riavviata a caldo dopo il dimensionamento.
Failover	Sì, con procedura analoga a quella valida per i cluster con provisioning.	Solo modalità "Best effort", soggetto alla disponibilità della capacità. Più lento che in Aurora Serverless v2.

Funzionalità di dimensionamento ed elevata disponibilità	Aurora Serverless v2comportamento	Aurora Serverless v1comportamento
Funzionalità Multi-AZ	Sì, come per le istanze con provisioning. Un cluster Multi-AZ richiede un'istanza database di lettura in una seconda zona di disponibilità. Per un cluster Multi-AZ, Aurora esegue il failover Multi-AZ in caso di guasto a livello di zona di disponibilità.	I cluster Aurora Serverless v1 eseguono tutte le operazioni di calcolo in un'unica zona di disponibilità. Il ripristino in caso di guasto a livello di zona di disponibilità viene eseguito solo in modalità "Best effort" ed è soggetto alla disponibilità della capacità.
Database globali di Aurora	Sì	No
Dimensionamento basato sul carico della memoria	Sì	No
Dimensionamento basato sul carico della CPU	Sì	Sì
Dimensionamento basato sul traffico di rete	Sì, in base alla memoria e al sovraccarico della CPU del traffico di rete. Il parametro `max_connections` rimane costante per evitare l'interruzione delle connessioni durante il dimensionamento.	Sì, in base al numero di connessioni.
Operazione di timeout per gli eventi di dimensionamento	No	Sì

Funzionalità di dimension amento ed elevata disponibi lità	Aurora Serverless v2comport amento	Aurora Serverless v1comport amento
Aggiungere nuove istanze DB al cluster tramite AWS Auto Scaling	Non applicabile. Puoi creare istanze database Aurora Serverless v2 di lettura nei livelli di promozione 2-15, lasciandole ridotte alla capacità inferiore.	No. Le istanze database di lettura non sono disponibili.

Confronto del supporto delle caratteristiche di Aurora Serverless v2 e Aurora Serverless v1

La tabella seguente si riferisce ai seguenti argomenti:

- Caratteristiche disponibili in Aurora Serverless v2 ma non in Aurora Serverless v1
- Caratteristiche che funzionano in modo diverso in Aurora Serverless v1 e Aurora Serverless v2
- Caratteristiche attualmente non disponibili in Aurora Serverless v2

In Aurora Serverless v2 sono presenti molte caratteristiche dei cluster con provisioning che non sono disponibili in Aurora Serverless v1.

Funzionalità	Supporto di Aurora Serverless v2	Supporto di Aurora Serverless v1
Topologia dei cluster	Aurora Serverless v2 è una proprietà delle singole istanze database. Un cluster può contenere più istanze database Aurora Serverles s v2 o una combinazione di istanze Aurora Serverless v2 e istanze database con provisioning.	I cluster Aurora Serverless v1 non utilizzano la nozione di istanze database. Non sarà possibile modificare la proprietà Aurora Serverless v1 dopo aver creato il cluster.

Funzionalità	Supporto di Aurora Serverless v2	Supporto di Aurora Serverless v1
Parametri di configurazione	È possibile modificare quasi tutti gli stessi parametri come avviene per i cluster con provisioning. Per informazioni dettagliate, vedi Uso di gruppi di parametri per Aurora Serverless v2.	È possibile modificare solo un sottoinsieme di parametri.
Gruppi di parametri	Uso di gruppi di parametri a livello di cluster e gruppi di parametri a livello di database Sono disponibili i parametri con il valore `provisioned` nell'attributo `SupportedEngineModes`. In Aurora Serverless v1 è disponibile un maggior numero di parametri.	Solo gruppo di parametri a livello di cluster. Sono disponibili i parametri con il valore `serverless` nell'attributo `SupportedEngineModes`.
Crittografia per volume di cluster	Facoltativo	Obbligatorio. Le limitazioni in Limiti relativi a istanze database crittografate Amazon Aurora si applicano a tutti i cluster Aurora Serverless v1.
Snapshot tra regioni	Sì	L'istantanea deve essere crittografata con la propria chiave AWS Key Management Service (AWS KMS).
Backup automatici conservati dopo l'eliminazione del cluster di database	Sì	No

Funzionalità	Supporto di Aurora Serverless v2	Supporto di Aurora Serverless v1
TLS/SSL	Sì. Stesso supporto disponibile per i cluster con provisioning. Per informazioni sull'utilizzo, consulta Utilizzo di TLS/SSL con Aurora Serverless v2.	Sì. Esistono alcune differenze rispetto al supporto TLS per i cluster con provisioning. Per informazioni sull'utilizzo, consulta Utilizzo di TLS/SSL con Aurora Serverless v1.
Clonazione	Solo versioni di database di origine e destinazione compatibili con Aurora Serverless v2. Non è possibile utilizzare la clonazione per eseguire l'aggiornamento da Aurora Serverless v1 o da una versione precedente di un cluster con provisioning.	Solo versioni di database di origine e destinazione compatibili con Aurora Serverless v1.
Integrazione con Amazon S3	Sì	Sì
Integrazione con Gestione dei segreti AWS	Sì	No
Esportazione di snapshot cluster DB in S3	Sì	No
Associazione di un ruolo IAM	Sì	No
Caricamento dei log su Amazon CloudWatch	Opzionale. Sei tu a scegliere quali registri attivare e su quali registri caricare. CloudWatch	Tutti i log attivati vengono caricati automaticamente. CloudWatch
API di dati disponibile	Sì	Sì
Editor di query disponibile	Sì	Sì

Funzionalità	Supporto di Aurora Serverless v2	Supporto di Aurora Serverless v1
Approfondimenti sulle prestazioni	Sì	No
Amazon RDS Proxy disponibile	Sì	No
Babelfish per Aurora PostgreSQL disponibile	Sì. Supportato per versioni di Aurora PostgreSQL compatibili con Babelfish e Aurora Serverless v2.	No

Adattamento dei casi d'uso di Aurora Serverless v1 ad Aurora Serverless v2

A seconda del caso d'uso relativo ad Aurora Serverless v1, potresti adattare l'approccio in modo da avvalerti delle caratteristiche di Aurora Serverless v2 nel seguente modo.

Supponi di avere un cluster Aurora Serverless v1 con un leggero carico e che la priorità sia mantenere una disponibilità continua e contemporaneamente ridurre al minimo i costi. Con Aurora Serverless v2, è possibile configurare un'impostazione ACU minima minore di 0,5, rispetto a un'impostazione minima di 1 ACU in Aurora Serverless v1. È possibile aumentare la disponibilità creando una configurazione Multi-AZ. Inoltre, l'istanza Reader DB ha un minimo di 0,5. ACUs

Supponi di disporre di un cluster Aurora Serverless v1 utilizzato in uno scenario di sviluppo e test. In questo caso, anche il costo è una priorità elevata, ma non è necessario che il cluster sia sempre disponibile. Aurora Serverless v2 può mettere automaticamente in pausa ogni istanza quando è completamente inattiva. A tale scopo, è necessario specificare una capacità minima di 0 ACUs per il cluster, come spiegato in. Dimensionamento a zero ACU con pausa e ripresa automatiche per Aurora Serverless v2 È anche possibile arrestare manualmente il cluster quando non serve e avviarlo al successivo ciclo di test o sviluppo.

Si supponga di avere un cluster Aurora Serverless v1 con un carico di lavoro pesante. Un cluster equivalente che utilizza Aurora Serverless v2 può essere dimensionato con maggiore granularità. Ad esempio, Aurora Serverless v1 si ridimensiona raddoppiando la capacità, ad esempio da 64 a 128. ACUs Al contrario, l'istanza database Aurora Serverless v2 può ridurre orizzontalmente le risorse con incrementi di 0,5 ACU.

Si supponga che il carico di lavoro richieda una capacità totale superiore a quella disponibile in Aurora Serverless v1. È possibile utilizzare più istanze database Aurora Serverless v2 di lettura per alleggerire l'istanza database di scrittura dalle aree del carico di lavoro a uso più intensivo di operazioni di lettura. È inoltre possibile suddividere il carico di lavoro a uso più intensivo di operazioni di lettura tra più istanze database di lettura.

Per un carico di lavoro intensivo di scrittura, potrebbe essere necessario configurare il cluster con un'istanza database di grandi dimensioni di cui è stato effettuato il provisioning come l'istanza di scrittura. Questa operazione può essere eseguita insieme a una o più istanze database di lettura Aurora Serverless v2.

Aggiornamento da un cluster Aurora Serverless v1 ad Aurora Serverless v2

> **Important**
>
> AWS ha annunciato la end-of-life data delAurora Serverless v1: 31 marzo 2025. Tutti i cluster Aurora Serverless v1 non migrati entro il 31 marzo 2025 verranno spostati su Aurora Serverless v2 durante la finestra di manutenzione. Se l'aggiornamento non riesce, Amazon Aurora converte il cluster Serverless v1 in un cluster con provisioning con la versione del motore equivalente durante la finestra di manutenzione. Se applicabile, Amazon Aurora registrerà il cluster con provisioning convertito in Supporto esteso di Amazon RDS. Per ulteriori informazioni, consulta Estensione del supporto di RDS.

Il processo di aggiornamento di un cluster database da Aurora Serverless v1 a Aurora Serverless v2 implica diversi passaggi. Questo perché non puoi convertire direttamente da Aurora Serverless v1 a Aurora Serverless v2. C'è sempre un passaggio intermedio che prevede la conversione del cluster database Aurora Serverless v1 in un cluster con provisioning.

Cluster database compatibili con Aurora MySQL

È possibile convertire il cluster Aurora Serverless v1 DB in un cluster DB fornito, quindi utilizzare una blue/green distribuzione per aggiornarlo e convertirlo in un Aurora Serverless v2 cluster DB. Questa procedura è consigliata per gli ambienti di produzione. Per ulteriori informazioni, consulta Utilizzo di Amazon Blue/Green Aurora Deployments per gli aggiornamenti del database.

Per utilizzare una blue/green distribuzione per aggiornare un Aurora Serverless v1 cluster che esegue Aurora MySQL versione 2 (compatibile con MySQL 5.7)

1. Converti il cluster di database Aurora Serverless v1 in un cluster Aurora MySQL versione 2 allocato. Segui la procedura riportata in Conversione da istanza Aurora Serverless v1 in istanza con provisioning.
2. Crea blue/green una distribuzione. Segui la procedura riportata in Creazione di una blue/green distribuzione in .
3. Scegli una versione di Aurora MySQL per il cluster verde compatibile con Aurora Serverless v2, ad esempio 3.04.1.

   Per le versioni compatibili, consulta Aurora Serverless v2 con Aurora MySQL.
4. Modifica l'istanza database di scrittura del cluster verde per utilizzare la classe di istanza database Serverless v2 (db.serverless).

   Per informazioni dettagliate, vedi Conversione di un'istanza di lettura o scrittura con provisioning per Aurora Serverless v2.
5. Quando il cluster di database Aurora Serverless v2 aggiornato è disponibile, passa dal cluster blu al cluster verde.

Cluster database compatibili con Aurora PostgreSQL

È possibile convertire il cluster Aurora Serverless v1 DB in un cluster DB fornito, quindi utilizzare una blue/green distribuzione per aggiornarlo e convertirlo in un cluster Aurora Serverless v2 DB. Questa procedura è consigliata per gli ambienti di produzione. Per ulteriori informazioni, consulta Utilizzo di Amazon Blue/Green Aurora Deployments per gli aggiornamenti del database.

Per utilizzare una blue/green distribuzione per aggiornare un Aurora Serverless v1 cluster che esegue Aurora PostgreSQL versione 11

1. Converti il cluster di database Aurora Serverless v1 in un cluster Aurora PostgreSQL allocato. Segui la procedura riportata in Conversione da istanza Aurora Serverless v1 in istanza con provisioning.
2. Crea una distribuzione. blue/green Segui la procedura riportata in Creazione di una blue/green distribuzione in .
3. Scegli una versione di Aurora PostgreSQL per il cluster verde compatibile con Aurora Serverless v2, ad esempio 15.3.

Per le versioni compatibili, consulta Aurora Serverless v2 con Aurora PostgreSQL.

4. Modifica l'istanza database di scrittura del cluster verde per utilizzare la classe di istanza database Serverless v2 (db.serverless).

   Per informazioni dettagliate, vedi Conversione di un'istanza di lettura o scrittura con provisioning per Aurora Serverless v2.

5. Quando il cluster di database Aurora Serverless v2 aggiornato è disponibile, passa dal cluster blu al cluster verde.

È inoltre possibile aggiornare il cluster di database Aurora Serverless v1 direttamente dalla versione 11 alla versione 13 di Aurora PostgreSQL, quindi convertire il cluster in un cluster di database con provisioning e poi convertire il cluster con provisioning in un cluster di database Aurora Serverless v2.

Come aggiornare e poi convertire un cluster Aurora Serverless v1 che esegue Aurora PostgreSQL versione 11

1. Converti il cluster di database Aurora Serverless v1 in un cluster Aurora PostgreSQL allocato. Segui la procedura riportata in Conversione da istanza Aurora Serverless v1 in istanza con provisioning.
2. Aggiorna il Aurora Serverless v1 cluster a una versione di Aurora PostgreSQL versione 13 compatibile, ad esempio, con la 13.12. Aurora Serverless v2 Segui la procedura riportata in Aggiornamento della versione principale.

   Per le versioni compatibili, consulta Aurora Serverless v2 con Aurora PostgreSQL.

3. Aggiungi un'istanza database di lettura Aurora Serverless v2 al cluster. Per ulteriori informazioni, consulta Aggiunta di un'istanza Aurora Serverless v2 di lettura.
4. Failover sull'istanza database Aurora Serverless v2:

   a. Scegli l'istanza database di scrittura del cluster di database.

   b. Per Actions (Operazioni), scegliere Failover.

   c. Nella pagina di conferma, scegli Failover.

Per i cluster di database Aurora Serverless v1 che eseguono Aurora PostgreSQL versione 13, si converte il cluster Aurora Serverless v1 in un cluster di database con provisioning, quindi si converte il cluster con provisioning in un cluster di database Aurora Serverless v2.

Per aggiornare un cluster Aurora Serverless v1 che esegue Aurora PostgreSQL versione 13

1. Converti il cluster di database Aurora Serverless v1 in un cluster Aurora PostgreSQL allocato. Segui la procedura riportata in Conversione da istanza Aurora Serverless v1 in istanza con provisioning.
2. Aggiungi un'istanza database di lettura Aurora Serverless v2 al cluster. Per ulteriori informazioni, consulta Aggiunta di un'istanza Aurora Serverless v2 di lettura.
3. Failover sull'istanza database Aurora Serverless v2:
    a. Scegli l'istanza database di scrittura del cluster di database.
    b. Per Actions (Operazioni), scegliere Failover.
    c. Nella pagina di conferma, scegli Failover.
4. Rimuovi l'istanza di lettura.

Migrazione di un database on-premise a Aurora Serverless v2

Puoi migrare i database on-premise a Aurora Serverless v2, esattamente come con Aurora MySQL e Aurora PostgreSQL con provisioning.

- Per i database MySQL, puoi usare il comando `mysqldump`. Per ulteriori informazioni, consulta Importazione dei dati in un database Amazon RDS per MySQL con tempo di inattività ridotto nella Guida per l'utente di Amazon Relational Database Service.
- Per i database PostgreSQL, puoi utilizzare i comandi `pg_dump` e `pg_restore`. Per ulteriori informazioni, consulta il blog post Best practices for migrating PostgreSQL databases to Amazon RDS and Amazon Aurora.

Utilizzo di Amazon Aurora Serverless v1

> **Important**
>
> AWS ha annunciato la data di fine del ciclo di vita per Aurora Serverless v1: 31 marzo 2025. Tutti i cluster Aurora Serverless v1 di cui non è stata eseguita la migrazione entro il 31 marzo 2025 verranno spostati su Aurora Serverless v2 durante la finestra di manutenzione. Se l'aggiornamento non riesce, Amazon Aurora converte il cluster Serverless v1 in un cluster con provisioning con la versione del motore equivalente durante la finestra di manutenzione. Se applicabile, Amazon Aurora registrerà il cluster con provisioning convertito in Supporto esteso per Amazon RDS. Per ulteriori informazioni, consulta Estensione del supporto di RDS.

Amazon Aurora Serverless v1 (Amazon Aurora Serverless versione 1) è una configurazione con scalabilità automatica on demand di Amazon Aurora. Un cluster di databaseAurora Serverless v1 è un cluster di database che consente di scalare la capacità di calcolo verso l'alto e verso il basso in base alle esigenze dell'applicazione. Ciò è in contrasto con i cluster database con provisioning di Aurora, per i quali è possibile gestire manualmente la capacità. Aurora Serverless v1 offre un'opzione relativamente semplice ed economica per carichi di lavoro poco frequenti, intermittenti o imprevedibili. È economico perché avvia e dimensiona automaticamente la capacità di calcolo in base all'uso dell'applicazione e si chiude quando non viene utilizzato.

Per ulteriori informazioni sui prezzi, consultare Prezzi di Serverless in Edizione compatibile con MySQL o Edizione compatibile con PostgreSQL nella pagina Amazon Aurora pricing.

I cluster Aurora Serverless v1 hanno lo stesso tipo di volume di archiviazione ad alta capacità, distribuito e ad alta disponibilità utilizzato dai cluster database sottoposti a provisioning.

Il volume cluster di un cluster Aurora Serverless v1 è sempre crittografato. Puoi scegliere la chiave di crittografia, ma non puoi disabilitare la crittografia. Ciò significa che è possibile eseguire le stesse operazioni su uno snapshot Aurora Serverless v1 crittografato. Per ulteriori informazioni, consulta Aurora Serverless v1 e snapshot.

Argomenti

- Disponibilità di Regioni e versioni per Aurora Serverless v1
- Vantaggi di Aurora Serverless v1

- Casi d'uso per Aurora Serverless v1
- Limitazioni di Aurora Serverless v1
- Requisiti di configurazione per Aurora Serverless v1
- Utilizzo di TLS/SSL con Aurora Serverless v1
- Funzionamento di Aurora Serverless v1
- Creazione di un cluster di database Aurora Serverless v1
- Ripristino di un cluster database Aurora Serverless v1
- Modifica di un cluster di database Aurora Serverless v1
- Dimensionamento manuale della capacità del cluster database Aurora Serverless v1
- Visualizzazione dei cluster database Aurora Serverless v1
- Eliminazione di un cluster di database Aurora Serverless v1
- Versioni del motore del database di Aurora Serverless v1 e Aurora

Disponibilità di Regioni e versioni per Aurora Serverless v1

La disponibilità e il supporto della funzionalità varia tra le versioni specifiche di ciascun motore di database Aurora e tra Regioni AWS. Per ulteriori informazioni sulla disponibilità di versioni e regioni con Aurora e Aurora Serverless v1, consultare Aurora Serverless v1.

Vantaggi di Aurora Serverless v1

Aurora Serverless v1 offre i vantaggi riportati di seguito:

- Più semplice del provisioning: Aurora Serverless v1 rimuove gran parte della complessità della gestione delle istanze e della capacità del database.
- Scalabile: Aurora Serverless v1 scala perfettamente la capacità di calcolo e memoria in base alle necessità, senza alcuna interruzione alle connessioni client.
- Conveniente: quando utilizzi Aurora Serverless v1, paghi soltanto le risorse del database utilizzate su base al secondo.
- Archiviazione ad alta disponibilità: Aurora Serverless v1 utilizza lo stesso sistema di archiviazione tollerante ai guasti e distribuito con replica in sei direzioni utilizzato da Aurora per la protezione dalla perdita dei dati.

Casi d'uso per Aurora Serverless v1

Aurora Serverless v1 è progettato per i seguenti casi d'uso:

- Applicazioni poco utilizzate – Hai un'applicazione che viene usata soltanto per pochi minuti diverse volte al giorno o alla settimana, come il sito di un blog a volume ridotto. Con Aurora Serverless v1 paghi soltanto le risorse del database utilizzate su base al secondo.
- Nuove applicazioni – Stai distribuendo una nuova applicazione e non sei sicuro delle dimensioni dell'istanza necessarie. Con Aurora Serverless v1, puoi creare un endpoint del database e far sì che il database venga dimensionato automaticamente in base ai requisiti di capacità dell'applicazione.
- Carichi di lavoro variabili – Esegui un'applicazione poco utilizzata, con picchi che vanno da 30 minuti a diverse ore poche volte al giorno o diverse volte all'anno. Esempi di questo tipo sono le applicazioni per le risorse umane, la redazione del budget e la creazione di report operativi. Grazie a Aurora Serverless v1, non dovrai più effettuare il provisioning per la capacità media o di picco.
- Carichi di lavoro imprevedibili – Esegui carichi di lavoro giornalieri con aumenti improvvisi e imprevedibili dell'attività. Un esempio può essere quello di un sito sul traffico in cui c'è un aumento dell'attività quando inizia a piovere. Grazie a Aurora Serverless v1, il database aumenta automaticamente la capacità per soddisfare le esigenze del picco di carico dell'applicazione e la riduce quando il picco è terminato.
- Database di sviluppo e test – Gli sviluppatori utilizzano i database durante l'orario di lavoro, ma non ne hanno bisogno nelle notti o nei fine settimana. Con Aurora Serverless v1, il database si spegne automaticamente quando non viene utilizzato.
- Applicazioni multi-tenant: grazie a Aurora Serverless v1, non devi gestire individualmente la capacità del database per ogni applicazione utilizzata nel parco istanze. Aurora Serverless v1 gestisce la capacità del database a livello individuale per tuo conto.

Limitazioni di Aurora Serverless v1

Le seguenti limitazioni si applicano a Aurora Serverless v1:

- **Important**

  AWS ha annunciato la data di fine del ciclo di vita per Aurora Serverless v1: 31 marzo 2025. Tutti i cluster Aurora Serverless v1 di cui non è stata eseguita la migrazione entro il 31 marzo 2025 verranno spostati su Aurora Serverless v2 durante la finestra

di manutenzione. Se l'aggiornamento non riesce, Amazon Aurora converte il cluster Serverless v1 in un cluster con provisioning con la versione del motore equivalente durante la finestra di manutenzione. Se applicabile, Amazon Aurora registrerà il cluster con provisioning convertito in Supporto esteso per Amazon RDS. Per ulteriori informazioni, consulta Estensione del supporto di RDS.

- Aurora Serverless v1 non supporta le seguenti caratteristiche:
  - Database globali di Aurora
  - Repliche di Aurora
  - AWS Identity and Access Management Autenticazione del database (IAM)
  - Backtrack in Aurora
  - Flussi di attività di database
  - Autenticazione Kerberos
  - Approfondimenti sulle prestazioni
  - Server proxy per RDS
  - Visualizzazione dei log nella Console di gestione AWS
- Le connessioni a un cluster di database Aurora Serverless v1 vengono chiuse automaticamente se mantenute aperte per più di un giorno.
- Tutti i cluster database Aurora Serverless v1 presentano le seguenti limitazioni:
  - Non è possibile esportare snapshot Aurora Serverless v1 in bucket Simple Storage Service (Amazon S3).
  - Non è possibile utilizzare AWS Database Migration Service e CDC (Change Data Capture) con cluster database Aurora Serverless v1. Solo i cluster database Aurora con provisioning supportano CDC con AWS DMS come origine.
  - Non è possibile salvare i dati in file di testo in Amazon S3 o caricare i dati di file di testo in Aurora Serverless v1 da S3.
  - Non è possibile collegare un ruolo IAM a un cluster database Aurora Serverless v1. Tuttavia, è possibile caricare i dati su Aurora Serverless v1 da Simple Storage Service (Amazon S3) utilizzando l'estensione `aws_s3` con la funzione `aws_s3.table_import_from_s3` e il parametro `credentials`. Per ulteriori informazioni, consulta Importazione di dati da Amazon S3 in un cluster database Aurora PostgreSQL.
  - Quando si utilizza l'editor di query, viene creato un segreto Secrets Manager con le credenziali per accedere al database. Eliminando le credenziali dall'editor di query, anche il segreto

associato viene eliminato da Secrets Manager. Una volta eliminato, però, questo segreto non può essere recuperato.

- I cluster database basati su Aurora MySQL che eseguono Aurora Serverless v1non supportano quanto segue:
  - Richiamo delle funzioni AWS Lambda dal cluster database Aurora MySQL. Tuttavia, le funzioni AWS Lambda possono effettuare chiamate al cluster di database Aurora Serverless v1.
  - Ripristino di uno snapshot da un'istanza database che non è Aurora MySQL o RDS for MySQL.
  - Replica dei dati mediante la replica basata su registri binari (binlog). Questa limitazione vale indipendentemente dal fatto che Aurora Serverless v1 del cluster database basato su Aurora MySQL sia l'origine o la destinazione della replica. Per replicare i dati in un cluster database Aurora Serverless v1 da un'istanza database MySQL esterna a Aurora, ad esempio da una istanza che viene eseguita su Amazon EC2, si consiglia di prendere in considerazione l'utilizzo di AWS Database Migration Service. Per ulteriori informazioni, consulta la Guida per l'utente AWS Database Migration Service.
  - Creazione di utenti con accesso basato su host (`'username'@'IP_address'`). Questo perché Aurora Serverless v1 utilizza un parco istanze router tra il client e l'host del database per un dimensionamento senza interruzioni. L'indirizzo IP visto dal cluster database Aurora Serverless è quello dell'host del router e non del client. Per ulteriori informazioni, consulta Architettura di Aurora Serverless v1.

    Utilizzare, invece, il carattere jolly (`'username'@'%'`).
- I cluster database basati su Aurora PostgreSQL che eseguono Aurora Serverless v1 hanno le seguenti limitazioni:
  - La gestione del piano di query di Aurora PostgreSQL (`apg_plan_management`) non è supportata.
  - La funzionalità di replica logica disponibile in Amazon RDS PostgreSQL e Aurora PostgreSQL non è supportata.
  - Le comunicazioni in uscita come quelle abilitate dalle estensioni Amazon RDS for PostgreSQL non sono supportate. Ad esempio, non è possibile accedere ai dati esterni con l'estensione `postgres_fdw/dblink`. Per ulteriori informazioni sulle estensioni PostgreSQL di RDS, consulta PostgreSQL su Amazon RDS nella Guida per l'utente di RDS.
  - Al momento, alcune query SQL e comandi non sono consigliati. Questi includono blocchi di consulenza a livello di sessione, relazioni temporanee, notifiche asincrone (`LISTEN`) e cursori con hold (`DECLARE name ... CURSOR WITH HOLD FOR query`). Inoltre, i comandi `NOTIFY` impediscono il dimensionamento e non sono consigliati.

Per ulteriori informazioni, consulta Scalabilità automatica per Aurora Serverless v1.

- Non è possibile impostare la finestra di backup automatico preferita per un cluster di database Aurora Serverless v1.
- Puoi impostare la finestra di manutenzione per un cluster di database Aurora Serverless v1. Per ulteriori informazioni, consulta Impostazione della finestra di manutenzione preferita del cluster database.

Requisiti di configurazione per Aurora Serverless v1

Quando crei un cluster database Aurora Serverless v1, presta attenzione ai seguenti requisiti:

- Utilizza i seguenti numeri di porta specifici per ogni motore database:
  - Aurora MySQL – `3306`
  - Aurora PostgreSQL – `5432`
- Crea il tuo cluster database Aurora Serverless v1 in un Virtual Private Cloud (VPC) basato sul servizio Amazon VPC. Quando si crea un cluster di database Aurora Serverless v1 nel VPC, si utilizzano due (2) dei cinquanta (50) endpoint di interfaccia e Gateway Load Balancer assegnati al VPC. Questi endpoint vengono creati automaticamente per l'utente. Per aumentare la quota, puoi contattare Supporto. Per ulteriori informazioni, consulta la pagina relativa alle quote di Amazon VPC.
- Non puoi assegnare al cluster database Aurora Serverless v1 un indirizzo IP pubblico. Puoi accedere a un cluster database Aurora Serverless v1 solo da un VPC.
- Crea sottoreti in zone di disponibilità diverse per il gruppo di sottoreti database utilizzato per il cluster database Aurora Serverless v1. In altre parole, non è possibile avere più di una sottorete nella stessa zona di disponibilità.
- Le modifiche a un gruppo di sottoreti utilizzato da un cluster database Aurora Serverless v1 non vengono applicate al cluster.
- Puoi accedere a un cluster database Aurora Serverless v1 da AWS Lambda. Per far ciò, dovrai configurare la funzione Lambda per l'esecuzione nello stesso VPC del cluster database Aurora Serverless v1. Per ulteriori informazioni sull'utilizzo di AWS Lambda, consulta Configurazione di una funzione Lambda per accedere alle risorse in un Amazon VPC nella Guida per gli sviluppatori di AWS Lambda.

Utilizzo di TLS/SSL con Aurora Serverless v1

Per impostazione predefinita, Aurora Serverless v1 utilizza il protocollo TLS/SSL (Transport Layer Security/Secure Sockets Layer) per crittografare le comunicazioni tra i client e il cluster database Aurora Serverless v1. Supporta TLS/SSL versione 1.0, 1.1 e 1.2. Non è necessario configurare il cluster database Aurora Serverless v1 per utilizzare TLS/SSL.

Si applicano le seguenti limitazioni:

- Il supporto TLS/SSL per i cluster database Aurora Serverless v1 al momento non è disponibile nella Regione AWS Cina (Pechino).
- Quando crei utenti di database per un cluster database Aurora Serverless v1 basato su Aurora MySQL, non utilizzare la clausola `REQUIRE` per le autorizzazioni SSL. In questo modo si impedisce agli utenti di connettersi all'istanza database Aurora.
- Per i programmi di utilità MySQL Client e Client PostgreSQL, le variabili di sessione che possono essere utilizzare in altri ambienti non hanno alcun effetto quando si utilizza TLS/SSL tra client e Aurora Serverless v1.
- Per MySQL Client, quando ti connetti con la modalità `VERIFY_IDENTITY` di TLS/SSL, devi utilizzare il comando `mysql` compatibile con MySQL 8.0. Per ulteriori informazioni, consulta Connessione a un'istanza database che esegue il motore del database MySQL.

A seconda del client utilizzato per connettersi al cluster database Aurora Serverless v1, potrebbe non essere necessario specificare TLS/SSL per ottenere una connessione crittografata. Ad esempio, per utilizzare PostgreSQL Client per connettersi a un cluster database Aurora Serverless v1 che esegue l'edizione compatibile con PostgreSQL di Aurora, è sufficiente connettersi come si fa normalmente.

```
psql -h endpoint -U user
```

Dopo aver inserito la password, PostgreSQL Client mostra i dettagli della connessione, tra cui la versione TLS/SSL e il codice.

```
psql (12.5 (Ubuntu 12.5-0ubuntu0.20.04.1), server 10.12)
SSL connection (protocol: TLSv1.2, cipher: ECDHE-RSA-AES256-GCM-SHA384, bits: 256,
 compression: off)
Type "help" for help.
```

> **Important**
>
> Aurora Serverless v1 utilizza il protocollo TLS/SSL (Transport Layer Security/Secure Sockets Layer) per crittografare le connessioni di default, a meno che SSL/TLS non sia disabilitato dall'applicazione client. La connessione TLS/SSL termina al parco istanze del router. La comunicazione tra il parco istanze del router e il cluster di database Aurora Serverless v1 avviene entro il limite della rete interna del servizio.
> È possibile controllare lo stato della connessione client per verificare se la connessione a Aurora Serverless v1 è crittografata con TLS/SSL. PostgreSQL `pg_stat_ssl`, le tabelle `pg_stat_activity` e la relativa funzione `ssl_is_used` non mostrano lo stato TLS/SSL per la comunicazione tra l'applicazione client e Aurora Serverless v1. Allo stesso modo, lo stato TLS/SSL non può essere derivato dall'istruzione `status` MySQL.
> I parametri del cluster Aurora `force_ssl` per PostgreSQL e `require_secure_transport` per MySQL non erano precedentemente supportati per Aurora Serverless v1. Questi parametri sono ora disponibili per Aurora Serverless v1. Per un elenco completo dei parametri supportati da Aurora Serverless v1, chiama l'operazione l'API DescribeEngineDefaultClusterParameters. Per ulteriori informazioni sui gruppi di parametri e su Aurora Serverless v1, consulta Gruppi di parametri per Aurora Serverless v1.

Per utilizzare MySQL Client per connettersi a un cluster database Aurora Serverless v1 che esegue l'edizione compatibile con MySQL di Aurora, devi specificare TLS/SSL nella richiesta. L'esempio seguente include il trust store principale di Amazon CA 1 scaricato da Amazon Trust Services, necessario per la riuscita della connessione.

```
mysql -h endpoint -P 3306 -u user -p --ssl-ca=amazon-root-CA-1.pem --ssl-mode=REQUIRED
```

Specifica la password, quando richiesto. Verrà avviato il monitor MySQL. Puoi verificare che la sessione è crittografata utilizzando il comando `status`.

```
mysql> status
--------------
mysql  Ver 14.14 Distrib 5.5.62, for Linux (x86_64) using readline 5.1
Connection id:          19
Current database:
Current user:           ***@*******
SSL:                    Cipher in use is ECDHE-RSA-AES256-SHA
...
```

Per maggiori informazioni sulla connessione al database Aurora MySQL con MySQL Client, consulta Connessione a un'istanza database che esegue il motore del database MySQL.

Aurora Serverless v1 supporta tutte le modalità TLS/SSL disponibili per MySQL Client (`mysql`) e PostgreSQL Client (`psql`), incluse quelle elencate nella tabella seguente.

Descrizione della modalità TLS/SSL	mysql	psql
Connettiti senza utilizzare TLS/SSL.	DISABLED	disattiva
Prova prima la connessione utilizzando TLS/SSL, ma se necessario, torna all'uso senza SSL.	PREFERRED	prefer (impostazione predefinita)
Applica l'uso di TLS/SSL.	REQUIRED	require
Applica TLS/SSL e verifica la CA.	VERIFY_CA	verify-ca
Applica TLS/SSL, verifica la CA e verifica il nome host della CA.	VERIFY_IDENTITY	verify-full

Aurora Serverless v1 utilizza certificati con caratteri jolly. Se specifichi l'opzione "verifica CA" o "verifica CA e nome host CA" quando utilizzi TLS/SSL, scarica innanzitutto il trust store CA 1 radice di Amazon da Amazon Trust Services. Dopo aver fatto ciò, puoi identificare questo file in formato PEMA nel comando client. Per farlo utilizzando PostgreSQL Client:

Per Linux, macOS o Unix:

```
psql 'host=endpoint user=user sslmode=require sslrootcert=amazon-root-CA-1.pem
 dbname=db-name'
```

Per ulteriori informazioni sull'utilizzo del database Aurora PostgreSQL utilizzando Postgres Client, consulta Connessione a un'istanza database che esegue il modulo di motore del database PostgreSQL.

Per informazioni generali sulla connessione ai cluster database Aurora, consulta Connessione a un cluster database Amazon Aurora.

Suite di crittografia supportate per connessioni a cluster di database Aurora Serverless v1

Utilizzando suite di cifratura configurabili, è possibile avere maggiore controllo sulla sicurezza delle connessioni al database. È possibile specificare un elenco di suite di crittografia che si desidera abilitare per proteggere le connessioni TLS/SSL client al database. Con le suite di cifratura, è possibile controllare la crittografia di connessione accettata dal server di database. In questo modo si impedisce l'uso di sistemi di crittografia non sicuri o obsoleti.

I cluster di database Aurora Serverless v1 basati su Aurora MySQL supportano le stesse suite di crittografia dei cluster di database con provisioning di Aurora MySQL. Per informazioni su queste suite di crittografia, consulta Configurazione di suite di cifratura per connessioni ai cluster di database Aurora MySQL.

I cluster di database Aurora Serverless v1 basati su Aurora PostgreSQL non supportano le suite di crittografia.

Funzionamento di Aurora Serverless v1

> **Important**
>
> AWS ha annunciato la data di fine del ciclo di vita per Aurora Serverless v1: 31 marzo 2025. Tutti i cluster Aurora Serverless v1 non migrati entro il 31 marzo 2025 verranno spostati su Aurora Serverless v2 durante la finestra di manutenzione. Se l'aggiornamento non riesce, Amazon Aurora converte il cluster Serverless v1 in un cluster con provisioning con la versione del motore equivalente durante la finestra di manutenzione. Se applicabile, Amazon Aurora registrerà il cluster con provisioning convertito in Supporto esteso di Amazon RDS. Per ulteriori informazioni, consulta Estensione del supporto di RDS.

Di seguito è descritto il funzionamento di Aurora Serverless v1.

Argomenti

- Architettura di Aurora Serverless v1
- Scalabilità automatica per Aurora Serverless v1

- Operazione di timeout per le modifiche di capacità
- Sospendi e riprendi per Aurora Serverless v1
- Determinazione del numero massimo di connessioni di database per Aurora Serverless v1
- Gruppi di parametri per Aurora Serverless v1
- Registrazione per Aurora Serverless v1
- Aurora Serverless v1 e manutenzione
- Aurora Serverless v1 e failover
- Aurora Serverless v1 e snapshot

Architettura di Aurora Serverless v1

L'immagine seguente mostra una panoramica dell'architettura di Aurora Serverless v1.



Anziché effettuare il provisioning e gestire i server di database, puoi specificare le unità di capacità di Aurora (ACU). Ogni ACU è una combinazione di circa 2 gigabyte (GB) di memoria, CPU

corrispondente e rete. Lo storage del database esegue automaticamente il dimensionamento da 10 gibibyte (GiB) a 128 tebibytes (TiB), lo stesso storage di un cluster di database Aurora standard.

Puoi specificare l'ACU minima e massima. L'unità di capacità Aurora minima è l'ACU più bassa a cui un cluster di database può essere ridotto. L'unità di capacità Aurora massima è l'ACU più alta a cui un cluster di database può essere aumentato. In base alle impostazioni, Aurora Serverless v1 crea automaticamente regole di dimensionamento basate sulle soglie per utilizzo della CPU, delle connessioni e di memoria disponibile.

Aurora Serverless v1 gestisce il pool di risorse nella Regione AWS per ridurre al minimo il tempo di dimensionamento. Quando Aurora Serverless v1 aggiunge nuove risorse al cluster di database Aurora, utilizza il parco istanze del router per passare le connessioni client attive alle nuove risorse. Paghi soltanto i costi delle ACU che vengono attivamente usate nel tuo cluster di database Aurora in un momento specifico.

Scalabilità automatica per Aurora Serverless v1

La capacità allocata al cluster di database Aurora Serverless v1 si ridimensiona senza problemi in base al carico generato dall'applicazione client. Qui, il carico è l'utilizzo della CPU e il numero di connessioni. Quando la capacità è vincolata da una di queste opzioni, Aurora Serverless v1 scala verso l'alto. Anche Aurora Serverless v1 scala verso l'alto quando rileva problemi di prestazioni che possono essere risolti in questo modo.

Puoi visualizzare gli eventi di ridimensionamento per il cluster Aurora Serverless v1 nel Console di gestione AWS . Durante il dimensionamento automatico, Aurora Serverless v1 reimposta il parametro `EngineUptime`. Il valore del parametro di reset non è indice di un problema con il dimensionamento né di un'interruzione della connessione da parte di Aurora Serverless v1. È semplicemente il punto di partenza per i tempi di attività della nuova capacità. Per maggiori informazioni sui parametri, consulta Monitoraggio dei parametri in un cluster di database Amazon Aurora.

Se il tuo cluster DB Aurora Serverless v1 non ha connessioni attive, può ridurre fino a zero capacità (0 ACU). Per ulteriori informazioni, vedi Sospendi e riprendi per Aurora Serverless v1.

Quando è necessario eseguire un'operazione di ridimensionamento, Aurora Serverless v1 prova prima a identificare un punto di dimensionamento, ovvero un momento in cui non vengono elaborate query. Aurora Serverless v1 potrebbe non riuscire a trovare un punto di dimensionamento per i seguenti motivi:

- Query di lunga durata

- Transazioni in corso
- Tabelle temporanee o blocchi di tabelle

Per aumentare il tasso di successo del cluster di database Aurora Serverless v1 quando si trova un punto di dimensionamento, si consiglia di evitare query e transazioni di lunga durata. Per ulteriori informazioni sulle operazioni che provocano il blocco del dimensionamento e su come evitarle, consulta Best practice per l'utilizzo di Aurora Serverless v1 .

Per impostazione predefinita, Aurora Serverless v1 prova a trovare un punto di dimensionamento per 5 minuti (300 secondi). Puoi specificare un periodo di timeout diverso quando crei o modifichi il cluster. Il periodo di timeout può essere compreso tra 60 secondi e 10 minuti (600 secondi). Se Aurora Serverless v1 non riesce a trovare un punto di dimensionamento entro il periodo specificato, l'operazione di scalabilità automatica raggiunge il timeout.

Per impostazione predefinita, se l'autoscaling non trova un punto di dimensionamento prima del timeout, Aurora Serverless v1 mantiene il cluster alla capacità corrente. Questo comportamento di default può essere modificato quando si crea o si modifica il cluster di database Aurora Serverless v1 selezionando l'opzione Force the capacity change (Forza la modifica della capacità). Per ulteriori informazioni, consulta Operazione di timeout per le modifiche di capacità.

Operazione di timeout per le modifiche di capacità

Se la scalabilità automatica raggiunge il timeout senza trovare un punto di dimensionamento, per impostazione predefinita Aurora mantiene la capacità corrente. Se desideri che Aurora forzi la modifica, abilita l'opzione Force the capacity change (Forza la modifica della capacità). Questa opzione è disponibile nella sezione Autoscaling timeout and action (Timeout e operazione di scalabilità automatica) della pagina Create database (Crea database) quando crei il cluster.

L'opzione Force the capacity change (Forza la modifica della capacità) è disabilitata di default. Lascia questa opzione deselezionata per evitare modifiche alla capacità del tuo cluster di database Aurora Serverless v1 nel caso in cui l'operazione di dimensionamento scada senza che sia stato trovato un punto di dimensionamento.

Se selezioni questa opzione, il cluster di database Aurora Serverless v1 applicherà la modifica della capacità anche senza un punto di dimensionamento. Prima di selezionare questa opzione, devi essere consapevole delle conseguenze che essa può avere:

- Tutte le transazioni in corso vengono interrotte e viene visualizzato il seguente messaggio di errore.

Aurora MySQL versione 2 - ERRORE 1105 (HY000): L'ultima transazione è stata interrotta a causa del dimensionamento continuo. Riprova.

Puoi inviare nuovamente la transazione non appena il cluster di database Aurora Serverless v1 diventa disponibile.

- Le connessioni a tabelle temporanee e ai blocchi vengono eliminate.

  Ti consigliamo di selezionare Force the capacity change (Forza la modifica della capacità) solo se è possibile recuperare l'applicazione in seguito a connessioni interrotte o transazioni incomplete.

Le selezioni che effettui nella Console di gestione AWS quando crei un cluster di database Aurora Serverless v1 sono archiviate nell'oggetto `ScalingConfigurationInfo`, in `SecondsBeforeTimeout` e nelle proprietà `TimeoutAction`. Quando si crea il cluster, il valore della proprietà `TimeoutAction` viene impostato su uno dei seguenti valori:

- `RollbackCapacityChange`: questo valore viene impostato quando selezioni l'opzione Roll back the capacity change (Eseguire il rollback della modifica della capacità). Questo è il comportamento che segue di default.
- `ForceApplyCapacityChange`: questo valore viene impostato quando selezioni l'opzione Force the capacity change (Forza la modifica della capacità).

Puoi ottenere il valore di questa proprietà su un cluster di database Aurora Serverless v1 esistente utilizzando il comando di AWS CLI describe-db-cluster, come illustrato di seguito.

Per Linux, macOS o Unix:

```
aws rds describe-db-clusters --region region \
  --db-cluster-identifier your-cluster-name \
  --query '*[].{ScalingConfigurationInfo:ScalingConfigurationInfo}'
```

Per Windows:

```
aws rds describe-db-clusters --region region ^
  --db-cluster-identifier your-cluster-name ^
  --query "*[].{ScalingConfigurationInfo:ScalingConfigurationInfo}"
```

Ad esempio, di seguito è riportata la query e la risposta per un cluster di database Aurora Serverless v1 denominato `west-coast-sles` nella Regione Stati Uniti occidentali (California settentrionale).

```
$ aws rds describe-db-clusters --region us-west-1 --db-cluster-identifier west-coast-
sles
--query '*[].{ScalingConfigurationInfo:ScalingConfigurationInfo}'

[
    {
        "ScalingConfigurationInfo": {
            "MinCapacity": 1,
            "MaxCapacity": 64,
            "AutoPause": false,
            "SecondsBeforeTimeout": 300,
            "SecondsUntilAutoPause": 300,
            "TimeoutAction": "RollbackCapacityChange"
        }
    }
]
```

Come mostra la risposta, questo cluster di database Aurora Serverless v1 utilizza l'impostazione predefinita.

Per ulteriori informazioni, consulta Creazione di un cluster di database Aurora Serverless v1. Dopo aver creato il Aurora Serverless v1, potrai inoltre modificare l'azione di timeout e altre impostazioni della capacità in qualsiasi momento. Per scoprire come, consulta Modifica di un cluster di database Aurora Serverless v1.

Sospendi e riprendi per Aurora Serverless v1

Puoi scegliere di mettere in pausa il cluster di database Aurora Serverless v1 dopo un determinato periodo di tempo in cui non è stata registrata alcuna attività. Specifica il periodo di tempo in cui non viene registrata alcuna attività prima che il cluster di database venga messo in pausa. Quando si seleziona questa opzione, il tempo di inattività predefinito è di cinque minuti, ma è possibile modificare questo valore. Questa è un'impostazione facoltativa.

Quando il cluster di database viene messo in pausa, non avvengono attività di elaborazione o memoria e vengono addebitati soltanto i costi per lo storage. Se vengono richieste connessioni al database quando un cluster di database Aurora Serverless v1 è messo in pausa, il cluster di database si riavvia automaticamente e adempie alle richieste di connessione.

Quando il cluster di database riprende l'attività, ha la stessa capacità che aveva quando Aurora ha messo in pausa il cluster. Il numero di ACU dipende dalla scalabilità del cluster Aurora verso l'alto o verso il basso prima di essere messo in pausa.

> **Note**
>
> Se un cluster di database viene messo in pausa per oltre sette giorni potrebbe essere sottoposto a backup con uno snapshot. In questo caso, Aurora ripristina il cluster di database dalla snapshot quando viene avanzata una richiesta di connessione.

Determinazione del numero massimo di connessioni di database per Aurora Serverless v1

Gli esempi seguenti sono per un cluster di database Aurora Serverless v1 compatibile con MySQL 5.7. Puoi utilizzare un client MySQL o l'editor di query, se è stato configurato l'accesso. Per ulteriori informazioni, consulta Esecuzione di query nell'editor della query.

Trovare il numero massimo di connessioni al database

1. Trova l'intervallo di capacità per il cluster di database Aurora Serverless v1 tramite la AWS CLI.

   ```
   aws rds describe-db-clusters \
       --db-cluster-identifier my-serverless-57-cluster \
       --query 'DBClusters[*].ScalingConfigurationInfo|[0]'
   ```

   Il risultato mostra che il suo intervallo di capacità è di 1-4 ACU.

   ```
   {
       "MinCapacity": 1,
       "AutoPause": true,
       "MaxCapacity": 4,
       "TimeoutAction": "RollbackCapacityChange",
       "SecondsUntilAutoPause": 3600
   }
   ```

2. Esegui la seguente query SQL per trovare il numero massimo di connessioni.

   ```
   select @@max_connections;
   ```

Il risultato mostrato è per la capacità minima del cluster, 1 ACU.

```
@@max_connections
90
```

3. Dimensiona il cluster su 8-32 ACU.

   Per ulteriori informazioni sul dimensionamento, consultare Modifica di un cluster di database Aurora Serverless v1.

4. Conferma l'intervallo di capacità.

```
{
    "MinCapacity": 8,
    "AutoPause": true,
    "MaxCapacity": 32,
    "TimeoutAction": "RollbackCapacityChange",
    "SecondsUntilAutoPause": 3600
}
```

5. Trova il numero massimo di connessioni.

```
select @@max_connections;
```

   Il risultato mostrato è per la capacità minima del cluster, 8 ACU.

```
@@max_connections
1000
```

6. Ridimensiona il cluster al massimo possibile, 256–256 ACU.
7. Conferma l'intervallo di capacità.

```
{
    "MinCapacity": 256,
    "AutoPause": true,
    "MaxCapacity": 256,
    "TimeoutAction": "RollbackCapacityChange",
    "SecondsUntilAutoPause": 3600
}
```

8. Trova il numero massimo di connessioni.

```
select @@max_connections;
```

Il risultato mostrato è per 256 ACU.

```
@@max_connections
6000
```

> **Note**
>
> Il valore `max_connections` non viene dimensionato linearmente con il numero di ACU.

9. Ripristina le dimensioni del cluster a 1-4 ACU.

```
{
    "MinCapacity": 1,
    "AutoPause": true,
    "MaxCapacity": 4,
    "TimeoutAction": "RollbackCapacityChange",
    "SecondsUntilAutoPause": 3600
}
```

Questa volta, il valore `max_connections` è per 4 ACU.

```
@@max_connections
270
```

10. Consenti una riduzione del cluster a 2 ACU.

```
@@max_connections
180
```

Se il cluster è stato configurato per essere sospeso dopo un certo periodo di tempo di inattività, viene ridotto fino a 0 ACU. Tuttavia, `max_connections` non scende sotto il valore per 1 ACU.

```
@@max_connections
90
```

Gruppi di parametri per Aurora Serverless v1

Quando crei il tuo cluster di database Aurora Serverless v1, è possibile scegliere un motore del database Aurora specifico e un gruppo di parametri del cluster di database associato. A differenza di cluster DB Aurora con provisioning, un cluster DB Aurora Serverless v1 ha una singola istanza DB di lettura/scrittura configurata solo con un gruppo di parametri del cluster DB, non ha un gruppo di parametri database separato. Durante la scalabilità automatica, Aurora Serverless v1 deve essere in grado di modificare i parametri affinché il cluster funzioni al meglio per aumentare o diminuire la capacità. Così, con un cluster di database Aurora Serverless v1, alcune delle modifiche apportate ai parametri per un particolare tipo di motore del database potrebbero non essere applicabili.

Ad esempio, un cluster di database basato su Aurora PostgreSQL – Aurora Serverless v1 non può utilizzare `apg_plan_mgmt.capture_plan_baselines` e altri parametri che potrebbero essere utilizzati in cluster di database Aurora PostgreSQL con provisioning per la gestione del piano di query.

È possibile ottenere un elenco di valori predefiniti per i gruppi di parametri predefiniti per i vari motori database Aurora utilizzando il comando della CLI describe-engine-default-cluster-parameters e interrogando la Regione AWS. Di seguito sono riportati i valori che è possibile utilizzare per l'opzione `--db-parameter-group-family`.

Aurora MySQL versione 2	`aurora-mysql5.7`
Aurora PostgreSQL versione 11	`aurora-postgresql11`
Aurora PostgreSQL versione 13	`aurora-postgresql13`

Ti consigliamo di configurare la AWS CLI con il tuo ID chiave di accesso AWS e la chiave di accesso segreto AWS e di impostare la Regione AWS prima di utilizzare i comandi AWS CLI. Fornire la regione alla configurazione CLI consente di evitare di immettere il parametro `--region` l'esecuzione dei comandi. Per ulteriori informazioni sulla configurazione di AWS CLI, consulta Nozioni di base sulla configurazione nella Guida per l'utente di AWS Command Line Interface.

Nell'esempio seguente si recupera un elenco di parametri dal gruppo di cluster di database predefinito per Aurora MySQL versione 2.

Per Linux, macOS o Unix:

```
aws rds describe-engine-default-cluster-parameters \
  --db-parameter-group-family aurora-mysql5.7 --query \
  'EngineDefaults.Parameters[*].
{ParameterName:ParameterName,SupportedEngineModes:SupportedEngineModes} | [?
contains(SupportedEngineModes, `serverless`) == `true`] | [*].{param:ParameterName}' \
  --output text
```

Per Windows:

```
aws rds describe-engine-default-cluster-parameters ^
   --db-parameter-group-family aurora-mysql5.7 --query ^
   "EngineDefaults.Parameters[*].
{ParameterName:ParameterName,SupportedEngineModes:SupportedEngineModes} | [?
contains(SupportedEngineModes, 'serverless') == `true`] | [*].{param:ParameterName}" ^
   --output text
```

Modifica dei valori dei parametri per Aurora Serverless v1

Come spiegato in Gruppi di parametri per Amazon Aurora, non è possibile modificare direttamente i valori in un gruppo di parametri predefinito, indipendentemente dal tipo (gruppo di parametri del cluster di database, gruppo di parametri DB). Al contrario, si crea un gruppo di parametri personalizzato basato sul gruppo di parametri cluster di database predefinito per il motore database Aurora e modificare le impostazioni in base alle esigenze su quel gruppo di parametri. Ad esempio, potresti voler modificare alcune impostazioni per il tuo cluster di database Aurora Serverless v1 per registrare query o caricare log specifici del motore database in Amazon CloudWatch .

Per creare un gruppo di parametri del cluster di database personalizzato

1. Accedi alla Console di gestione AWS e apri la console Amazon RDS all'indirizzo https://console.aws.amazon.com/rds/.
2. Scegliere Gruppi di parametri.
3. Seleziona Crea gruppo di parametri per aprire il riquadro dei dettagli del gruppo di parametri.
4. Seleziona il gruppo cluster di database predefinito appropriato per il motore database che desideri utilizzare per il cluster di databaseAurora Serverless v1. Assicurati di scegliere le seguenti opzioni:

    a. Per Famiglia gruppo parametri, seleziona la famiglia appropriata per il motore del database scelto. Assicurati che la selezione abbia un nome contenente il prefisso `aurora-`.

b. Per Type (Tipo), scegli DB Cluster Parameter Group (Gruppo di parametri del cluster di database).

c. Per Nome gruppo e Descrizione, specifica nomi significativi per l'utente o per altri utenti che potrebbero dover utilizzare il cluster di database Aurora Serverless v1 e i relativi parametri.

d. Scegli Create (Crea).

Il gruppo di parametri del cluster di database personalizzato viene aggiunto all'elenco dei gruppi di parametri disponibili per l'account nella Regione AWS. Ora puoi utilizzare il gruppo di parametri del cluster DB personalizzato quando crei nuovi cluster DB Aurora Serverless v1. Puoi anche modificare un cluster di database Aurora Serverless v1 esistente per utilizzare il gruppo di parametri del cluster di database personalizzato. Una volta che il cluster di database Aurora Serverless v1 inizia a utilizzare il gruppo di parametri del cluster di database personalizzato, sarà possibile modificare i valori per i parametri dinamici utilizzando la Console di gestione AWS o il AWS CLI.

Potrai inoltre utilizzare la console per visualizzare un confronto affiancato dei valori nel gruppo di parametri del cluster di database personalizzati rispetto a quello dei parametri di default, come illustrato nello screenshot seguente.



Quando modifichi i valori dei parametri in un cluster di database attivo, Aurora Serverless v1 avvia il dimensionamento senza interruzioni per applicare le modifiche ai parametri. Se il tuo cluster di database Aurora Serverless v1 è in pausa, riprende l'attività e inizia il dimensionamento in modo che possa apportare la modifica. L'operazione di dimensionamento per la modifica di un gruppo di parametri forza sempre la modifica della capacità, quindi tieni presente che la modifica dei parametri potrebbe comportare la perdita di connessioni se non è possibile trovare un punto di dimensionamento durante il periodo di dimensionamento.

Registrazione per Aurora Serverless v1

Per impostazione predefinita, i log degli errori per Aurora Serverless v1 sono abilitati e caricati automaticamente su Amazon CloudWatch. Puoi anche fare in modo che il cluster di database Aurora Serverless v1 carichi log specifici del motore database Aurora su CloudWatch. Per fare ciò, abilita i parametri di configurazione nel gruppo di parametri del cluster di database personalizzati. Il tuo cluster di database Aurora Serverless v1 carica quindi tutti i log disponibili su Amazon CloudWatch. A questo punto, procedi all'analisi dei dati di log con CloudWatch Logs e utilizza CloudWatch per creare allarmi e visualizzare parametri.

Per Aurora MySQL, i log che è possibile abilitare sono mostrati nella tabella seguente. Quando vengono abilitati, vengono caricati automaticamente dal cluster di database Aurora Serverless v1 ad Amazon CloudWatch.

Log Aurora MySQL	Descrizione
`general_log`	Crea il log generale. Imposta su 1 per attivare questa opzione. Il valore predefinito è disattiva to (0).
`log_queries_not_using_indexes`	Registra tutte le query nel log delle query lente che non utilizzano un indice. Il valore predefini to è disattivato (0). Imposta su 1 per attivare questo log.
`long_query_time`	Impedisce che le query in esecuzione rapida vengano registrate nel log delle query lente. Può essere impostato su un valore variabile compreso tra 0 e 31.536.000. Il valore predefini to è 0 (non attivo).

Log Aurora MySQL	Descrizione
`server_audit_events`	L'elenco degli eventi da catturare nei log. I valori supportati sono `CONNECT`, `QUERY`, `QUERY_DCL` , `QUERY_DDL` , `QUERY_DML` e `TABLE`.
`server_audit_logging`	Imposta su 1 per attivare la registrazione di controllo del server. Se attivi questa opzione, puoi specificare gli eventi di controllo da inviare a CloudWatch riportandoli nel parametro `server_audit_events` .
`slow_query_log`	Crea un log di query lente. Imposta su 1 per attivare il log di query lente. Il valore predefinito è disattivato (0).

Per ulteriori informazioni, consulta Utilizzo dell'audit avanzato con un cluster di database Amazon Aurora MySQL.

Per Aurora PostgreSQL, i log che è possibile abilitare sono mostrati nella tabella seguente. Quando vengono abilitati, vengono caricati automaticamente dal cluster di database Aurora Serverless v1 ad Amazon CloudWatch insieme ai normali log degli errori.

Log Aurora PostgreSQL	Descrizione
`log_connections`	Attivato di default e non può essere modificato. Registra i dettagli per tutte le nuove connessio ni client.
`log_disconnections`	Attivato di default e non può essere modificato. Registra tutte le disconnessioni client.
`log_hostname`	Disattivato per impostazione predefinita e non modificabile. I nomi degli host non vengono registrati.

Log Aurora PostgreSQL	Descrizione
`log_lock_waits`	Il valore predefinito è 0 (disattivato). Imposta su 1 per registrare le attese di blocco.
`log_min_duration_statement`	La durata minima (in millisecondi) per l'esecuzione di un'istruzione prima della registrazione.
`log_min_messages`	Imposta i livelli dei messaggi che vengono registrati. I valori supportati sono `debug5`, `debug4`, `debug3`, `debug2`, `debug1`, `info`, `notice`, `warning`, `error`, `log`, `fatal`, `panic`. Per registrare i dati delle prestazioni nel log `postgres`, imposta il valore su `debug1`.
`log_temp_files`	Registra l'utilizzo di file temporanei che si trovano al di sopra dei kilobyte (kB) specificati.
`log_statement`	Controlla le istruzioni SQL specifiche che vengono registrate. I valori supportati sono `none`, `ddl`, `mod` e `all`. Il valore predefinito è `none`.

Dopo aver attivato i log per Aurora MySQL o Aurora PostgreSQL per il tuo cluster di database Aurora Serverless v1, puoi visualizzare i log in CloudWatch.

Visualizzazione dei log Aurora Serverless v1 con Amazon CloudWatch

Aurora Serverless v1 carica automaticamente ("pubblica") su Amazon CloudWatch tutti i log abilitati nel gruppo di parametri del cluster di database personalizzato. Non è necessario scegliere o specificare i tipi di log. Il caricamento dei log inizia non appena si attiva il parametro di configurazione log. Se in seguito questo parametro viene disattivato, gli altri caricamenti saranno interrotti. Tuttavia, tutti i log che sono già stati pubblicati in CloudWatch saranno conservati fino a quando non saranno eliminati.

Per ulteriori informazioni sull'uso di CloudWatch con i log di Aurora MySQL, consulta Monitoraggio degli eventi di log in Amazon CloudWatch.

Per ulteriori informazioni su CloudWatch e Aurora PostgreSQL, consulta Pubblicazione di log Aurora PostgreSQL in Amazon CloudWatch Logs.

Per visualizzare i log per il cluster di database Aurora Serverless v1

1. Aprire la console CloudWatch all'indirizzo https://console.aws.amazon.com/cloudwatch/.
2. Scegli il tuo Regione AWS.
3. Scegli Log groups (Gruppi di log).
4. Seleziona il log del cluster di database Aurora Serverless v1 dall'elenco. Per i log di errori, il modello di denominazione è il seguente.

```
/aws/rds/cluster/cluster-name/error
```

Ad esempio, nello screenshot seguente sono riportati gli elenchi dei log pubblicati per un cluster di database Aurora Serverless v1 Aurora PostgreSQL denominato `western-sles`. Sono riportati anche diversi elenchi per il cluster di database Aurora Serverless v1 Aurora PostgreSQL denominato `west-coast-sles`. Seleziona il log desiderato per iniziare ad esplorarne il contenuto.



Aurora Serverless v1 e manutenzione

La manutenzione per un cluster di database Aurora Serverless v1, ad esempio l'applicazione delle funzionalità, correzioni e aggiornamenti di sicurezza più recenti, viene eseguita automaticamente. Aurora Serverless v1 ha una finestra di manutenzione che è possibile visualizzare nella Console di gestione AWS in Manutenzione e backup per il tuo cluster di database Aurora Serverless v1. È possibile trovare la data e l'ora in cui può essere eseguita la manutenzione e verificare se è presente una manutenzione in sospeso per il cluster di database Aurora Serverless v1 come mostrato nella seguente figura.



Puoi impostare la finestra di manutenzione quando crei il cluster di database Aurora Serverless v1 e modificarla in un secondo momento. Per ulteriori informazioni, consulta Impostazione della finestra di manutenzione preferita del cluster database.

Le finestre di manutenzione vengono utilizzate per gli aggiornamenti programmati delle versioni principali. Le patch e gli aggiornamenti a versioni secondarie vengono applicati immediatamente durante il dimensionamento. Il dimensionamento avviene in base alle impostazioni per `TimeoutAction`:

- `ForceApplyCapacityChange`: la modifica viene applicata immediatamente.
- `RollbackCapacityChange`: Aurora aggiorna forzatamente il cluster al massimo dopo 3 giorni dal primo tentativo di patch.

Come per qualsiasi modifica forzata senza un punto di ridimensionamento appropriato, ciò potrebbe interrompere il carico di lavoro.

Quando possibile, Aurora Serverless v1 esegue la manutenzione in modalità non invasiva. Quando è necessaria la manutenzione, il cluster di database Aurora Serverless v1 dimensionerà automaticamente la propria capacità per gestire le operazioni necessarie. Prima di scalare, Aurora

Serverless v1 cerca un punto di dimensionamento. Esegue l'operazione per un massimo di sette giorni, se necessario.

Alla fine di ogni giorno che Aurora Serverless v1 non riesce a trovare un punto di dimensionamento, viene creato un evento cluster. Questo evento notifica la manutenzione in sospeso e la necessità di eseguire il dimensionamento per eseguire la manutenzione. La notifica include la data in cui Aurora Serverless v1 può forzare il dimensionamento del cluster di database.

Per ulteriori informazioni, consulta Operazione di timeout per le modifiche di capacità.

Aurora Serverless v1 e failover

Se l'istanza DB per un cluster di database Aurora Serverless v1 diventa non disponibile o la zona di disponibilità (AZ) restituisce un errore, Aurora crea nuovamente l'istanza DB in una AZ diversa. Tuttavia, il cluster Aurora Serverless v1 non è un cluster Multi-AZ. Questo perché è costituito da una singola istanza DB in un'unica zona di disponibiità. Perciò, questo meccanismo di failover richiede più tempo rispetto a un cluster Aurora con provisioning o alle istanze Aurora Serverless v2. Il tempo di failover per Aurora Serverless v1 è indefinito perché dipende dalla richiesta e dalla disponibilità di capacità in altre AZ nella Regione AWS specificata.

Poiché Aurora separa la capacità di calcolo e lo storage, il volume di storage per il cluster è suddiviso in più zone di disponibilità. I dati rimangono disponibili anche se un'interruzione riguarda l'istanza DB o la zona di disponibilità associata.

Aurora Serverless v1 e snapshot

Il volume del cluster per un cluster Aurora Serverless v1 è sempre crittografato. Puoi scegliere la chiave di crittografia, ma non è possibile disabilitare la crittografia. Per copiare o condividere uno snapshot di un cluster Aurora Serverless v1, esegui la crittografia dello snapshot utilizzando la tua AWS KMS key. Per ulteriori informazioni, consulta Copia di uno snapshot del cluster di database. Per ulteriori informazioni sulla crittografia e Amazon Aurora, consulta Creazione di un cluster di database Amazon Aurora

Creazione di un cluster di database Aurora Serverless v1

> **Important**
>
> AWS ha annunciato la data di fine del ciclo di vita per Aurora Serverless v1: 31 marzo 2025. Tutti i cluster Aurora Serverless v1 di cui non è stata effettuata la migrazione entro il 31 marzo

2025 verranno spostati su Aurora Serverless v2 durante la finestra di manutenzione. Se l'aggiornamento non riesce, Amazon Aurora converte il cluster Serverless v1 in un cluster con provisioning con la versione del motore equivalente durante la finestra di manutenzione. Se applicabile, Amazon Aurora registrerà il cluster con provisioning convertito in supporto esteso per Amazon RDS. Per ulteriori informazioni, consulta Estensione del supporto di RDS.

Attraverso la seguente procedura crei un cluster Aurora Serverless v1 senza oggetti o dati dello schema. Se invece desideri creare un cluster Aurora Serverless v1 che sia il duplicato di uno con provisioning esistente, oppure un cluster Aurora Serverless v1, esegui l'operazione di ripristino o clonazione degli snapshot. Per ottenere queste informazioni, consulta Ripristino da uno snapshot cluster database e Clonazione di un volume per un cluster di database Amazon Aurora. Non è possibile convertire un cluster con provisioning esistente in uno Aurora Serverless v1. Inoltre, non è possibile riconvertire un cluster Aurora Serverless v1 esistente in uno con provisioning.

Quando crei un cluster di database Aurora Serverless v1, puoi impostare una capacità minima e massima. Un'unità di capacità equivale a una specifica configurazione di memoria e calcolo. Aurora Serverless v1 crea regole di dimensionamento basate sulle soglie di utilizzo della CPU, delle connessioni e della memoria disponibile, e scala perfettamente a una delle tante unità di capacità disponibili in base alle esigenze delle tue applicazioni. Per ulteriori informazioni, consulta Architettura di Aurora Serverless v1.

È possibile impostare i seguenti valori specifici per il cluster di database Aurora Serverless v1:

- Unità di capacità minima di Aurora: Aurora Serverless v1 può ridurre la capacità fino a questa unità di capacità.
- Unità di capacità massima di Aurora: Aurora Serverless v1 può aumentare la capacità fino a questa unità di capacità.

È inoltre possibile scegliere le seguenti opzioni di configurazione del dimensionamento facoltative:

- Forza il dimensionamento della capacità ai valori specificati quando viene raggiunto il timeout: è possibile scegliere questa impostazione se si desidera che Aurora Serverless v1 forzi Aurora Serverless v1 per il dimensionamento anche se non riesce a trovare un punto di dimensionamento prima del timeout. Se desideri che Aurora Serverless v1 annulli le modifiche di capacità se non riesce a trovare un punto di ridimensionamento, non scegliere questa impostazione. Per ulteriori informazioni, consulta Operazione di timeout per le modifiche di capacità.

- Sospendi la capacità di calcolo dopo minuti consecutivi di inattività: puoi scegliere questa impostazione se desideri che Aurora Serverless v1 dimensioni a zero quando non è presente alcuna attività nel cluster di database per un periodo di tempo specificato. Con questa impostazione abilitata, il cluster di database Aurora Serverless v1 riprende automaticamente l'elaborazione e si adatta alla capacità necessaria per gestire il carico di lavoro quando il traffico del database riprende. Per ulteriori informazioni, vedi Sospendi e riprendi per Aurora Serverless v1.

Prima di poter creare un cluster di database Aurora Serverless v1, è necessario un account AWS. È inoltre necessario aver completato le attività di configurazione per lavorare con Amazon Aurora. Per ulteriori informazioni, consulta Configurazione dell'ambiente per Amazon Aurora. È inoltre necessario completare altre fasi preliminari per la creazione di un cluster di database Aurora. Per ulteriori informazioni, consulta Creazione di un cluster database Amazon Aurora.

Aurora Serverless v1 è disponibile in determinate Regioni AWS solo per versioni specifiche di Aurora MySQL e Aurora PostgreSQL. Per ulteriori informazioni, consulta Aurora Serverless v1.

> **Note**
>
> Il volume del cluster per un cluster Aurora Serverless v1 è sempre crittografato. Quando crei il cluster di database Aurora Serverless v1, non puoi disattivare la crittografia, ma puoi scegliere di utilizzare la tua chiave di crittografia. Con Aurora Serverless v2, è possibile scegliere se crittografare o meno il volume cluster.

Puoi creare un cluster di database Aurora Serverless v1 utilizzando l'interfaccia AWS CLI o l'API RDS.

> **Note**
>
> Se viene visualizzato il seguente messaggio di errore quando si tenta di creare il cluster, l'account necessita di autorizzazioni aggiuntive.
> `Unable to create the resource. Verify that you have permission to create service linked role. Otherwise wait and try again later.`
> Per ulteriori informazioni, consulta Utilizzo di ruoli collegati ai servizi per Amazon Aurora.

Non puoi connetterti direttamente all'istanza database nel cluster di database Aurora Serverless v1. Per connetterti al cluster di database Aurora Serverless v1, utilizza l'endpoint del database.

Puoi trovare l'endpoint per il cluster di database Aurora Serverless v1 nella scheda Connettività e sicurezza per il cluster in Console di gestione AWS. Per ulteriori informazioni, consulta Connessione a un cluster database Amazon Aurora.

AWS CLI

Per creare un nuovo cluster database Aurora Serverless v1 tramite AWS CLI, esegui il comando create-db-cluster e specifica `serverless` per l'opzione `--engine-mode`.

Puoi eventualmente specificare l'opzione `--scaling-configuration` in modo che configuri la capacità minima, quella massima e la pausa automatica quando non sono presenti connessioni.

I seguenti esempi di comando creano un nuovo cluster di database Serverless impostando l'opzione `--engine-mode` su `serverless`. Gli esempi specificano inoltre i valori per l'opzione `--scaling-configuration`.

Esempio per Aurora MySQL

Il seguente comando creano un nuovo cluster database serverless compatibile con Aurora MySQL. I valori di capacità validi per Aurora MySQL sono 1, 2, 4, 8, 16, 32, 64, 128 e 256.

Per Linux, macOS o Unix:

```
aws rds create-db-cluster --db-cluster-identifier sample-cluster \
    --engine aurora-mysql --engine-version 5.7.mysql_aurora.2.11.4 \
    --engine-mode serverless \
    --scaling-configuration
 MinCapacity=4,MaxCapacity=32,SecondsUntilAutoPause=1000,AutoPause=true \
    --master-username username --master-user-password password
```

Per Windows:

```
aws rds create-db-cluster --db-cluster-identifier sample-cluster ^
    --engine aurora-mysql --engine-version 5.7.mysql_aurora.2.11.4 ^
    --engine-mode serverless ^
    --scaling-configuration
 MinCapacity=4,MaxCapacity=32,SecondsUntilAutoPause=1000,AutoPause=true ^
    --master-username username --master-user-password password
```

Esempio per Aurora PostgreSQL

Il comando seguente crea un nuovo cluster di database serverless compatibile con PostgreSQL 13.9. I valori di capacità validi per Aurora PostgreSQL sono 2, 4, 8, 16, 32, 64, 192 e 384.

Per Linux, macOS o Unix:

```
aws rds create-db-cluster --db-cluster-identifier sample-cluster \
    --engine aurora-postgresql --engine-version 13.9 \
    --engine-mode serverless \
    --scaling-configuration
 MinCapacity=8,MaxCapacity=64,SecondsUntilAutoPause=1000,AutoPause=true \
    --master-username username --master-user-password password
```

Per Windows:

```
aws rds create-db-cluster --db-cluster-identifier sample-cluster ^
    --engine aurora-postgresql --engine-version 13.9 ^
    --engine-mode serverless ^
    --scaling-configuration
 MinCapacity=8,MaxCapacity=64,SecondsUntilAutoPause=1000,AutoPause=true ^
    --master-username username --master-user-password password
```

API RDS

Per creare un nuovo cluster di database Aurora Serverless v1 tramite l'API RDS, esegui l'operazione CreateDBCluster e specifica `serverless` per il parametro `EngineMode`.

Puoi eventualmente specificare il parametro `ScalingConfiguration` in modo che configuri la capacità minima, quella massima e la pausa automatica quando non sono presenti connessioni. I valori di capacità validi includono quanto segue:

- Aurora MySQL: 1, 2, 4, 8, 16, 32, 64, 128 e 256.
- Aurora PostgreSQL: 2, 4, 8, 16, 32, 64, 192 e 384.

Ripristino di un cluster database Aurora Serverless v1

> **Important**
>
> AWS ha annunciato la data di fine del ciclo di vita per Aurora Serverless v1: 31 marzo 2025. Tutti i cluster Aurora Serverless v1 di cui non è stata effettuata la migrazione entro il 31 marzo 2025 verranno spostati su Aurora Serverless v2 durante la finestra di manutenzione. Se l'aggiornamento non riesce, Amazon Aurora converte il cluster Serverless v1 in un cluster con provisioning con la versione del motore equivalente durante la finestra di manutenzione. Se applicabile, Amazon Aurora registrerà il cluster con provisioning convertito in supporto esteso per Amazon RDS. Per ulteriori informazioni, consulta Estensione del supporto di RDS.

Puoi configurare un cluster di database Aurora Serverless v1 durante il ripristino dello snapshot di un cluster di database con provisioning utilizzando l'interfaccia AWS CLI o l'API RDS.

Quando ripristini uno snapshot in un cluster database Aurora Serverless v1, puoi impostare i seguenti valori specifici:

- Unità di capacità minima di Aurora: Aurora Serverless v1 può ridurre la capacità fino a questa unità di capacità.
- Unità di capacità massima di Aurora: Aurora Serverless v1 può aumentare la capacità fino a questa unità di capacità.
- Azione di timeout: l'azione da eseguire quando una modifica della capacità scade perché non riesce a trovare un punto di dimensionamento. Aurora Serverless v1 Il cluster database può forzare il cluster database alle nuove impostazioni di capacità impostando l'opzione Forza la capacità di dimensionamento ai valori specificati.... In alternativa, è possibile ripristinare la modifica della capacità per annullarla se non si sceglie l'opzione. Per ulteriori informazioni, consulta Operazione di timeout per le modifiche di capacità.
- Pause after inactivity (Pausa dopo inattività): periodo di tempo senza traffico di database trascorso il quale la capacità di calcolo viene ridotta a zero. Quando il traffico di database riprende, Aurora aumenta automaticamente la capacità di calcolo e si dimensiona per gestire il traffico.

Per informazioni generali sul ripristino di un cluster database da uno snapshot, consulta Ripristino da uno snapshot cluster database.

AWS CLI

Puoi configurare un cluster database Aurora Serverless durante il ripristino della snapshot di un cluster database con provisioning tramite Console di gestione AWS, AWS CLI o l'API RDS.

Quando ripristini uno snapshot in un cluster database Aurora Serverless, puoi impostare i seguenti valori specifici:

- Unità di capacità minima di Aurora: Aurora Serverless può ridurre la capacità fino a questa unità di capacità.
- Unità di capacità massima di Aurora: Aurora Serverless può aumentare la capacità fino a questa unità di capacità.
- Azione di timeout: l'azione da eseguire quando una modifica della capacità scade perché non riesce a trovare un punto di dimensionamento. Aurora Serverless v1 Il cluster database può forzare il cluster database alle nuove impostazioni di capacità impostando l'opzione Forza la capacità di dimensionamento ai valori specificati.... In alternativa, è possibile ripristinare la modifica della capacità per annullarla se non si sceglie l'opzione. Per ulteriori informazioni, consulta Operazione di timeout per le modifiche di capacità.
- Pause after inactivity (Pausa dopo inattività): periodo di tempo senza traffico di database trascorso il quale la capacità di calcolo viene ridotta a zero. Quando il traffico di database riprende, Aurora aumenta automaticamente la capacità di calcolo e si dimensiona per gestire il traffico.

> **Note**
>
> La versione dello snapshot del cluster database deve essere compatibile conAurora Serverless v1. Per l'elenco delle versioni supportate, consulta Aurora Serverless v1.

Per ripristinare uno snapshot in un cluster Aurora Serverless v1 con compatibilità MySQL 5.7, includi i seguenti parametri aggiuntivi:

- `--engine` **`aurora-mysql`**
- `--engine-version` **`5.7`**

I parametri `--engine` e `--engine-version` consentono di creare un cluster Aurora Serverless v1 compatibile con MySQL 5.7 da uno snapshot Aurora o Aurora Serverless v1compatibile con MySQL

5.6. Nell'esempio seguente viene ripristinata uno snapshot da un cluster compatibile con MySQL 5.6 denominato *mydbclustersnapshot* in un cluster Aurora Serverless v1 compatibile con MySQL 5.7 denominato *mynewdbcluster*.

Per Linux, macOS o Unix:

```
aws rds restore-db-cluster-from-snapshot \
    --db-cluster-identifier mynewdbcluster \
    --snapshot-identifier mydbclustersnapshot \
    --engine-mode serverless \
    --engine aurora-mysql \
    --engine-version 5.7
```

Per Windows:

```
aws rds restore-db-cluster-from-snapshot ^
    --db-instance-identifier mynewdbcluster ^
    --db-snapshot-identifier mydbclustersnapshot ^
    --engine aurora-mysql ^
    --engine-version 5.7
```

Puoi eventualmente specificare l'opzione `--scaling-configuration` in modo che configuri la capacità minima, quella massima e la pausa automatica quando non sono presenti connessioni. I valori di capacità validi includono quanto segue:

- Aurora MySQL: 1, 2, 4, 8, 16, 32, 64, 128 e 256.
- Aurora PostgreSQL: 2, 4, 8, 16, 32, 64, 192 e 384.

Nell'esempio seguente viene eseguito il ripristino da uno snapshot del cluster di database creato in precedenza denominato *mydbclustersnapshot* in un nuovo cluster di database denominato *mynewdbcluster*. È possibile impostare `--scaling-configuration` in modo che il nuovo cluster Aurora Serverless v1 DB può dimensionare da 8 ACU a 64 ACU (unità di capacità Aurora) in base alle esigenze per elaborare il carico di lavoro. Al termine dell'elaborazione e dopo 1000 secondi senza connessioni da supportare, il cluster si arresta fino a quando le richieste di connessione non richiedono il riavvio.

Per Linux, macOS o Unix:

```
aws rds restore-db-cluster-from-snapshot \
```

```
    --db-cluster-identifier mynewdbcluster \
    --snapshot-identifier mydbclustersnapshot \
    --engine-mode serverless --scaling-configuration
 MinCapacity=8,MaxCapacity=64,TimeoutAction='ForceApplyCapacityChange',SecondsUntilAutoPause=10
```

Per Windows:

```
aws rds restore-db-cluster-from-snapshot ^
    --db-instance-identifier mynewdbcluster ^
    --db-snapshot-identifier mydbclustersnapshot ^
    --engine-mode serverless --scaling-configuration
 MinCapacity=8,MaxCapacity=64,TimeoutAction='ForceApplyCapacityChange',SecondsUntilAutoPause=10
```

API RDS

Per configurare un cluster database Aurora Serverless v1 durante un ripristino da un cluster database tramite l'API RDS, esegui l'operazione RestoreDBClusterFromSnapshot e specifica `serverless` per il parametro `EngineMode`.

Puoi eventualmente specificare il parametro `ScalingConfiguration` in modo che configuri la capacità minima, quella massima e la pausa automatica quando non sono presenti connessioni. I valori di capacità validi includono quanto segue:

- Aurora MySQL: 1, 2, 4, 8, 16, 32, 64, 128 e 256.
- Aurora PostgreSQL: 2, 4, 8, 16, 32, 64, 192 e 384.

Modifica di un cluster di database Aurora Serverless v1

> **Important**
>
> AWS ha annunciato la data di fine del ciclo di vita per Aurora Serverless v1: 31 marzo 2025. Tutti i cluster Aurora Serverless v1 di cui non è stata effettuata la migrazione entro il 31 marzo 2025 verranno spostati su Aurora Serverless v2 durante la finestra di manutenzione. Se l'aggiornamento non riesce, Amazon Aurora converte il cluster Serverless v1 in un cluster con provisioning con la versione del motore equivalente durante la finestra di manutenzione. Se applicabile, Amazon Aurora registrerà il cluster con provisioning convertito in supporto esteso per Amazon RDS. Per ulteriori informazioni, consulta Estensione del supporto di RDS.

Dopo aver configurato un cluster di database Aurora Serverless v1, puoi modificare determinate proprietà con la Console di gestione AWS, la AWS CLI o l'API RDS. La maggior parte delle proprietà che puoi modificare sono le stesse di altri tipi di cluster Aurora.

Di seguito sono riportate le modifiche più importanti di Aurora Serverless v1:

- Modifica della configurazione di dimensionamento
- Aggiornamento della versione principale
- Conversione da istanza Aurora Serverless v1 in istanza con provisioning

Modifica della configurazione di dimensionamento di un cluster di database Aurora Serverless v1

Puoi impostare le capacità minima e massima per il cluster di database. Ogni unità di capacità equivale a una specifica configurazione di memoria e calcolo. Aurora Serverless crea automaticamente regole di dimensionamento in base alle soglie per utilizzo della CPU, connessioni e memoria disponibile. Puoi anche specificare se Aurora Serverless sospende il database in assenza di attività e lo ripristina quando l'attività ricomincia.

Puoi impostare i seguenti valori specifici per la configurazione di dimensionamento:

- Unità di capacità minima di Aurora: Aurora Serverless può ridurre la capacità fino a questa unità di capacità.
- Unità di capacità massima di Aurora: Aurora Serverless può aumentare la capacità fino a questa unità di capacità.
- Il timeout e l'operazione di scalabilità automatica: questa sezione specifica per quanto tempo Aurora Serverlessattende di trovare un punto di dimensionamento prima del timeout. Specifica anche l'operazione da eseguire quando una modifica della capacità raggiunge il timeout perché non riesce a trovare un punto di dimensionamento. Aurora può forzare la modifica della capacità per impostare la capacità sul valore specificato non appena possibile. Oppure, può eseguire il rollback della modifica della capacità per annullarla. Per ulteriori informazioni, consulta Operazione di timeout per le modifiche di capacità.
- Pausa dopo inattività: utilizzare l'impostazione facoltativa Ridimensiona la capacità a 0 ACU quando il cluster è inattivo per ridimensionare il database a una capacità di elaborazione pari a zero quando il cluster è inattivo. Quando il traffico di database riprende, Aurora aumenta automaticamente la capacità di calcolo e si dimensiona per gestire il traffico.

> **Note**
>
> Quando si modifica l'intervallo di capacità per un cluster database Aurora Serverless, la modifica viene implementata subito, indipendentemente dal fatto che si scelga di applicarla immediatamente o durante la successiva finestra di manutenzione pianificata.

Console

Puoi modificare la configurazione di dimensionamento di un cluster di database Aurora dalla Console di gestione AWS.

Per modificare un cluster di database Aurora Serverless v1

1. Apri la console Amazon RDS all'indirizzo https://console.aws.amazon.com/rds/.
2. Nel riquadro di navigazione, scegliere Databases (Database).
3. Seleziona il cluster di database Aurora Serverless v1 che desideri modificare.
4. Per Actions (Operazioni), scegliere Modify cluster (Modifica cluster).
5. Nella sezione Capacity settings (Impostazioni di capacità), modificare la configurazione di dimensionamento.
6. Scegli Continua.
7. Nella pagina Modifica il cluster DB rivedi le modifiche apportate, quindi scegli quando applicarle.
8. Scegliere Modify cluster (Modifica cluster).

AWS CLI

Per modificare la configurazione di dimensionamento di un cluster di database Aurora Serverless v1 tramite AWS CLI, emetti il comando modify-db-cluster AWS CLI. Specifica l'opzione `--scaling-configuration` in modo che configuri la capacità minima, quella massima e la pausa automatica quando non sono presenti connessioni. I valori di capacità validi includono quanto segue:

- Aurora MySQL: `1`, `2`, `4`, `8`, `16`, `32`, `64`, `128` e `256`.
- Aurora PostgreSQL: `2`, `4`, `8`, `16`, `32`, `64`, `192` e `384`.

In questo esempio, modifica la configurazione di dimensionamento di un cluster di database Aurora Serverless v1 denominato *`sample-cluster`*.

Per Linux, macOS o Unix:

```
aws rds modify-db-cluster \
    --db-cluster-identifier sample-cluster \
    --scaling-configuration
 MinCapacity=8,MaxCapacity=64,SecondsUntilAutoPause=500,TimeoutAction='ForceApplyCapacityChange
```

Per Windows:

```
aws rds modify-db-cluster ^
    --db-cluster-identifier sample-cluster ^
    --scaling-configuration
 MinCapacity=8,MaxCapacity=64,SecondsUntilAutoPause=500,TimeoutAction='ForceApplyCapacityChange
```

API RDS

Puoi modificare la configurazione di dimensionamento di un cluster di database Aurora con l'operazione API ModifyDBCluster. Specifica il parametro `ScalingConfiguration` in modo che configuri la capacità minima, quella massima e la pausa automatica quando non sono presenti connessioni. I valori di capacità validi includono quanto segue:

- Aurora MySQL: 1, 2, 4, 8, 16, 32, 64, 128 e 256.
- Aurora PostgreSQL: 2, 4, 8, 16, 32, 64, 192 e 384.

Aggiornamento della versione principale di un cluster di database Aurora Serverless v1

> **Important**
>
> AWS ha annunciato la data di fine del ciclo di vita per Aurora Serverless v1: 31 marzo 2025. Tutti i cluster Aurora Serverless v1 di cui non è stata effettuata la migrazione entro il 31 marzo 2025 verranno spostati su Aurora Serverless v2 durante la finestra di manutenzione. Se l'aggiornamento non riesce, Amazon Aurora converte il cluster Serverless v1 in un cluster con provisioning con la versione del motore equivalente durante la finestra di manutenzione. Se applicabile, Amazon Aurora registrerà il cluster con provisioning convertito in supporto esteso per Amazon RDS. Per ulteriori informazioni, consulta Estensione del supporto di RDS.

Puoi aggiornare la versione principale per un cluster database Aurora Serverless v1 compatibile con PostgreSQL 11 a una versione corrispondente compatibile con PostgreSQL 13.

Console

Puoi eseguire un aggiornamento locale di un cluster di database Aurora Serverless v1 usando la Console di gestione AWS.

Per aggiornare un cluster di database Aurora Serverless v1

1. Apri la console di Amazon RDS all'indirizzo https://console.aws.amazon.com/rds/.
2. Nel riquadro di navigazione, scegliere Databases (Database).
3. Seleziona il cluster di database Aurora Serverless v1 che desideri aggiornare.
4. Per Actions (Operazioni), scegliere Modify cluster (Modifica cluster).
5. In Versione, scegli 13 come numero di versione di Aurora PostgreSQL.

   Il seguente esempio mostra un aggiornamento locale da Aurora PostgreSQL 11.16 a 13.9.



Se esegui un aggiornamento della versione principale, non modificare tutte le altre proprietà. Per modificare una qualsiasi delle altre proprietà, esegui un'operazione Modifica al termine dell'aggiornamento.

6. Scegli Continua.
7. Nella pagina Modifica il cluster DB rivedi le modifiche apportate, quindi scegli quando applicarle.
8. Scegliere Modify cluster (Modifica cluster).

AWS CLI

Per eseguire un aggiornamento locale da un cluster di database Aurora Serverless v1 compatibile con PostgreSQL 11 a uno compatibile con PostgreSQL 13, specifica il parametro `--engine-version` con un numero di versione Aurora PostgreSQL versione 13 compatibile con Aurora Serverless v1. Includi anche il parametro `--allow-major-version-upgrade`.

In questo esempio, viene modificata la versione principale di un cluster database Aurora Serverless v1 compatibile con PostgreSQL 11 denominata `sample-cluster`. Questo consente di eseguire un aggiornamento locale a un cluster di database Aurora Serverless v1 compatibile con PostgreSQL 13.

```
aws rds modify-db-cluster \
    --db-cluster-identifier sample-cluster \
    --engine-version 13.serverless_12 \
    --allow-major-version-upgrade
```

Per Windows:

```
aws rds modify-db-cluster ^
    --db-cluster-identifier sample-cluster ^
    --engine-version 13.serverless_12 ^
    --allow-major-version-upgrade
```

API RDS

Per eseguire un aggiornamento locale da un cluster di database Aurora Serverless v1 compatibile con PostgreSQL 11 a uno compatibile con PostgreSQL 13, specifica il parametro `EngineVersion` con un numero di versione Aurora PostgreSQL versione 13 compatibile con Aurora Serverless v1. Includi anche il parametro `AllowMajorVersionUpgrade`.

Conversione di cluster di database Aurora Serverless v1 in allocato

Puoi convertire un cluster di database Aurora Serverless v1 in un cluster di database allocato. Per eseguire la conversione, utilizza l'interfaccia AWS CLI o l'API Amazon RDS per modificare la classe dell'istanza database impostandola su Con provisioning. Utilizza i passaggi seguenti per modificare la classe dell'istanza database.

L'esempio seguente dimostra come utilizzare l'interfaccia AWS CLI per convertire un cluster di database Aurora Serverless v1 in un cluster con provisioning.

AWS CLI

Per convertire un cluster di database Aurora Serverless v1 in un cluster allocato, esegui il comando AWS CLI modify-db-cluster.

I parametri seguenti sono obbligatori:

- `--db-cluster-identifier`: il cluster di database Aurora Serverless v1 che stai convertendo in allocato.
- `--engine-mode`: usa il valore `provisioned`.
- `--allow-engine-mode-change`
- `--db-cluster-instance-class`: scegli la classe di istanza database per il cluster di database allocato in base alla capacità del cluster di database Aurora Serverless v1.

In questo esempio, si converte un cluster di database Aurora Serverless v1 denominato `sample-cluster` e si utilizza la classe di istanza database `db.r5.xlarge`.

Per Linux, macOS o Unix:

```
aws rds modify-db-cluster \
--db-cluster-identifier sample-cluster \
--engine-mode provisioned \
--allow-engine-mode-change \
--db-cluster-instance-class db.r5.xlarge
```

Per Windows:

```
aws rds modify-db-cluster ^
```

```
--db-cluster-identifier sample-cluster ^
--engine-mode provisioned ^
--allow-engine-mode-change ^
--db-cluster-instance-class db.r5.xlarge
```

L'esempio seguente dimostra come utilizzare l'API Amazon RDS per convertire un cluster di database Aurora Serverless v1 in un cluster con provisioning.

API RDS

Per convertire un cluster di database Aurora Serverless v1 in un cluster allocato, utilizza l'operazione API ModifyDBCluster.

I parametri seguenti sono obbligatori:

- `DBClusterIdentifier`: il cluster di database Aurora Serverless v1 che stai convertendo in allocato.
- `EngineMode`: usa il valore `provisioned`.
- `AllowEngineModeChange`
- `DBClusterInstanceClass`: scegli la classe di istanza database per il cluster di database allocato in base alla capacità del cluster di database Aurora Serverless v1.

Considerazioni sulla conversione da un cluster di database Aurora Serverless v1 a un cluster con provisioning

Le seguenti considerazioni si applicano alla conversione da un cluster di database Aurora Serverless v1 a un cluster con provisioning.

- Puoi utilizzare questa conversione come parte dell'aggiornamento del tuo cluster di database da Aurora Serverless v1 a Aurora Serverless v2. Per ulteriori informazioni, consulta Aggiornamento da un cluster Aurora Serverless v1 ad Aurora Serverless v2.
- Il processo di conversione crea un'istanza database di lettura nel cluster di database, promuove l'istanza di lettura in un'istanza di scrittura ed elimina l'istanza Aurora Serverless v1 originale. Quando si converte il cluster di database, non è possibile eseguire altre modifiche contemporaneamente, ad esempio la modifica della versione del motore di database o del gruppo di parametri del cluster di database. L'operazione di conversione viene applicata immediatamente e non può essere annullata.

- Durante la conversione, viene acquisito uno snapshot del cluster di database di backup in caso di errore. L'identificatore per lo snapshot del cluster di database ha il formato `pre-modify-engine-mode-`*`DB_cluster_identifier`*`-`*`timestamp`*.
- Aurora utilizza la versione secondaria predefinita corrente del motore di database per il cluster di database allocato.
- Se non fornisci una classe di istanza database per il cluster di database convertito, Aurora ne consiglia una in base alla capacità massima del cluster di database Aurora Serverless v1 originale. La capacità consigliata per le mappature delle classi di istanza è illustrata nella tabella seguente.

Capacità Serverless massima (ACU)	Classe di istanza database allocata
1	db.t3.small
2	db.t3.medium
4	db.t3.large
8	db.r5.large
16	db.r5.xlarge
32	db.r5.2xlarge
64	db.r5.4xlarge
128	db.r5.8xlarge
192	db.r5.12xlarge
256	db.r5.16xlarge
384	db.r5.24xlarge

> **Note**
>
> A seconda della classe di istanza database scelta e dell'utilizzo del database, è possibile che vengano calcolati costi diversi per un cluster di database allocato rispetto a Aurora Serverless v1.

Se converti il tuo cluster di database Aurora Serverless v1 in una classe di istanza database (db.t*) espandibile, potresti incorrere in costi aggiuntivi per l'utilizzo del cluster di database. Per ulteriori informazioni, consulta Tipi di classi di istanza database.

Dimensionamento manuale della capacità del cluster database Aurora Serverless v1

Important

AWS ha annunciato la data di fine del ciclo di vita per Aurora Serverless v1: 31 marzo 2025. Tutti i cluster Aurora Serverless v1 di cui non è stata effettuata la migrazione entro il 31 marzo 2025 verranno spostati su Aurora Serverless v2 durante la finestra di manutenzione. Se l'aggiornamento non riesce, Amazon Aurora converte il cluster Serverless v1 in un cluster con provisioning con la versione del motore equivalente durante la finestra di manutenzione. Se applicabile, Amazon Aurora registrerà il cluster con provisioning convertito in supporto esteso per Amazon RDS. Per ulteriori informazioni, consulta Estensione del supporto di RDS.

In genere, i cluster database Aurora Serverless v1 vengono dimensionati senza problemi in base al carico di lavoro. Tuttavia, la capacità potrebbe non essere sempre dimensionata in maniera rapida da soddisfare estremi improvvisi, come ad esempio un aumento esponenziale delle transazioni. In questi casi è possibile avviare manualmente l'operazione di dimensionamento impostando un nuovo valore di capacità. Dopo aver impostato la capacità esplicitamente, Aurora Serverless v1 può dimensionare automaticamente il cluster database. Esegue questa operazione in base al periodo di attesa per la riduzione.

Puoi impostare la capacità di un cluster database Aurora Serverless v1 su un valore specifico con Console di gestione AWS, AWS CLI o l'API RDS.

Console

Puoi impostare le capacità di un cluster database Aurora dalla Console di gestione AWS.

Per modificare un cluster database Aurora Serverless v1

1. Apri la console Amazon RDS all'indirizzo https://console.aws.amazon.com/rds/.

2. Nel riquadro di navigazione, scegliere Databases (Database).
3. Seleziona il cluster database Aurora Serverless v1 che desideri modificare.
4. Per Actions (Operazioni), selezionare Set capacity (Imposta capacità).
5. Nella finestra Dimensionamento della capacità del database seleziona quanto segue:

    a. Per il selettore a discesa Scale DB cluster to (Dimensionamento del cluster database a), seleziona la nuova capacità desiderata per il cluster database.

    b. Nella casella di controllo If a seamless scaling point cannot be found (Se non è possibile trovare un punto di ridimensionamento senza soluzione di continuità...), scegli il comportamento desiderato per il tuo cluster database Aurora Serverless v1 in base all'impostazione di `TimeoutAction`, come riportato di seguito:

        - Deseleziona questa opzione se desideri che la capacità rimanga invariata se Aurora Serverless v1 non riesce a trovare un punto di ridimensionamento prima del timeout.
        - Seleziona questa opzione se desideri forzare la modifica della capacità del cluster database Aurora Serverless v1 anche se non riesce a trovare un punto di ridimensionamento prima del timeout. Questa opzione può comportare l'eliminazione delle connessioni Aurora Serverless v1 che impediscono di trovare un punto di ridimensionamento.

    c. Nel campo seconds (secondi) specifica per quanto tempo il cluster database Aurora Serverless v1 può cercare un punto di dimensionamento prima del timeout. Puoi specificare un qualsiasi valore compreso tra 10 e 600 secondi (10 minuti). L'impostazione predefinita è 5 minuti (300 secondi). In questo esempio si impone al cluster database Aurora Serverless v1 di eseguire il dimensionamento verso il basso fino a 2 ACU anche se non riesce a trovare un punto di ridimensionamento entro cinque minuti.



6. Scegliere Apply (Applica).

Per ulteriori informazioni sui punti di ridimensionamento, `TimeoutAction`, e sui periodi di tempo di recupero, consulta Scalabilità automatica per Aurora Serverless v1.

AWS CLI

Per impostare la capacità di un cluster database Aurora Serverless v1 tramite AWS CLI, emetti il comando AWS CLI modify-current-db-cluster-capacity e specifica l'opzione `--capacity`. I valori di capacità validi includono quanto segue:

- Aurora MySQL: 1, 2, 4, 8, 16, 32, 64, 128 e 256.
- Aurora PostgreSQL: 2, 4, 8, 16, 32, 64, 192 e 384.

In questo esempio, viene impostata la capacità di un cluster database Aurora Serverless v1 denominato *`sample-cluster`* su *`64`*.

```
aws rds modify-current-db-cluster-capacity --db-cluster-identifier sample-cluster --
capacity 64
```

API RDS

Puoi impostare le capacità di un cluster database Aurora con l'operazione API ModifyCurrentDBClusterCapacity. Specifica il parametro `Capacity`. I valori di capacità validi includono quanto segue:

- Aurora MySQL: 1, 2, 4, 8, 16, 32, 64, 128 e 256.
- Aurora PostgreSQL: 2, 4, 8, 16, 32, 64, 192 e 384.

Visualizzazione dei cluster database Aurora Serverless v1

> **Important**
>
> AWS ha annunciato la data di fine del ciclo di vita per Aurora Serverless v1: 31 marzo 2025. Tutti i cluster Aurora Serverless v1 di cui non è stata effettuata la migrazione entro il 31 marzo 2025 verranno spostati su Aurora Serverless v2 durante la finestra di manutenzione. Se l'aggiornamento non riesce, Amazon Aurora converte il cluster Serverless v1 in un cluster con provisioning con la versione del motore equivalente durante la finestra di manutenzione. Se applicabile, Amazon Aurora registrerà il cluster con provisioning convertito in supporto esteso per Amazon RDS. Per ulteriori informazioni, consulta Estensione del supporto di RDS.

Dopo aver creato uno o più cluster database Aurora Serverless v1, puoi visualizzare quali cluster database sono di tipo Serverless e quali di tipo Istanza. Puoi anche visualizzare il numero corrente di unità di capacità di Aurora (ACU) utilizzate da ogni cluster database Aurora Serverless v1. Ogni ACU è la combinazione di capacità di calcolo (CPU) e memoria (RAM).

Per visualizzare i cluster database Aurora Serverless v1

1. Accedi alla Console di gestione AWS e apri la console Amazon RDS all'indirizzo https://console.aws.amazon.com/rds/.
2. Nell'angolo in alto a destra della Console di gestione AWS, seleziona la Regione AWS in cui hai creato i cluster database Aurora Serverless v1.
3. Nel riquadro di navigazione, scegli Databases (Database).

Per ogni cluster database, il tipo viene indicato su Role (Ruolo). I cluster database Aurora Serverless v1 mostrano Serverless per il tipo. Puoi visualizzare la capacità corrente di un cluster database Aurora Serverless v1 in Dimensione.



4. Seleziona il nome di un cluster database Aurora Serverless v1 per visualizzarne i dettagli.

   Nella scheda Connettività e sicurezza, prendi nota dell'endpoint del database. Utilizza questo endpoint per connetterti al cluster database Aurora Serverless v1.



Seleziona la scheda Configurazione per visualizzare le impostazioni di capacità.



Viene generato un evento di dimensionamento quando il cluster database aumenta o riduce il dimensionamento, viene messo in pausa o ripristinato. Selezionare la scheda Logs & events (Log ed eventi) per visualizzare gli eventi recenti. La seguente immagine mostra esempi di tali eventi.



Monitoraggio della capacità e dimensionamento degli eventi per il cluster database Aurora Serverless v1

Puoi visualizzare il cluster database Aurora Serverless v1 in CloudWatch per monitorare la capacità allocata al cluster database con il parametro `ServerlessDatabaseCapacity`. Inoltre, puoi monitorare tutti i parametri standard Aurora CloudWatch come `CPUUtilization`, `DatabaseConnections`, `Queries` e così via.

Puoi fare in modo che Aurora pubblichi alcuni o tutti i log di database su CloudWatch. Puoi selezionare i log da pubblicare abilitando parametri di configurazione come `general_log` e `slow_query_log` nel gruppo di parametri del cluster di database associato al cluster Aurora Serverless v1. A differenza dei cluster con provisioning, i cluster Aurora Serverless v1 non richiedono la specifica nelle impostazioni del cluster database dei tipi di log da caricare in CloudWatch. I cluster Aurora Serverless v1 caricano automaticamente tutti i log disponibili. Quando disabiliti un parametro di configurazione del log, la pubblicazione del log su CloudWatch si arresta. Puoi anche eliminare i log in CloudWatch se non sono più necessari.

Per iniziare a utilizzare Amazon CloudWatch per il tuo cluster database Aurora Serverless v1, consulta Visualizzazione dei log Aurora Serverless v1 con Amazon CloudWatch. Per ulteriori informazioni su come monitorare i cluster database Aurora attraverso CloudWatch, consulta Monitoraggio degli eventi di log in Amazon CloudWatch.

Per la connessione a un cluster database Aurora Serverless v1, utilizza l'endpoint di database. Per ulteriori informazioni, consulta Connessione a un cluster database Amazon Aurora.

> **Note**
>
> Non puoi connetterti direttamente a istanze database specifiche nei cluster database Aurora Serverless v1.

Eliminazione di un cluster di database Aurora Serverless v1

> **Important**
>
> AWS ha annunciato la data di fine del ciclo di vita per Aurora Serverless v1: 31 marzo 2025. Tutti i cluster Aurora Serverless v1 di cui non è stata effettuata la migrazione entro il 31 marzo

2025 verranno spostati su Aurora Serverless v2 durante la finestra di manutenzione. Se l'aggiornamento non riesce, Amazon Aurora converte il cluster Serverless v1 in un cluster con provisioning con la versione del motore equivalente durante la finestra di manutenzione. Se applicabile, Amazon Aurora registrerà il cluster con provisioning convertito in supporto esteso per Amazon RDS. Per ulteriori informazioni, consulta Estensione del supporto di RDS.

A seconda di come viene creato un cluster di database Aurora Serverless v1, la protezione dall'eliminazione potrebbe essere attivata per impostazione predefinita. Non è possibile eliminare immediatamente un cluster di database Aurora Serverless v1 in cui è abilitata la protezione dall'eliminazione. Per eliminare cluster database Aurora Serverless v1 con protezione dall'eliminazione utilizzando Console di gestione AWS, modifica innanzitutto il cluster per rimuovere questa protezione. Per informazioni sull'utilizzo di AWS CLI per questa attività, consulta AWS CLI.

Per disabilitare la protezione dall'eliminazione tramite Console di gestione AWS

1. Accedi alla Console di gestione AWS e apri la console Amazon RDS all'indirizzo https://console.aws.amazon.com/rds/.
2. Nel pannello di navigazione, seleziona DB clusters (Cluster database).
3. Seleziona il cluster database Aurora Serverless v1 dall'elenco.
4. Seleziona Modifica per aprire la configurazione del cluster database. La pagina Modifica cluster database consente di visualizzare le impostazioni, le impostazioni della capacità e altri dettagli di configurazione per il cluster database Aurora Serverless v1. La protezione dall'eliminazione è disponibile nella sezione Additional configuration (Configurazione aggiuntiva).
5. Deseleziona l'opzione Enable deletion protection (Abilita protezione eliminazione) nella scheda delle proprietà Additional configuration (Configurazione aggiuntive).
6. Scegli Continue (Continua). Viene visualizzato il Riepilogo delle modifiche .
7. Seleziona Modifica cluster per accettare il riepilogo delle modifiche. Puoi inoltre scegliere Indietro per modificare le modifiche o Annulla per ignorarle.

Una volta disattivata la protezione dall'eliminazione, potrai eliminare il cluster database Aurora Serverless v1 utilizzando Console di gestione AWS.

Console

Per eliminare un cluster database Aurora Serverless v1

1. Accedi alla Console di gestione AWS e apri la console Amazon RDS all'indirizzo https://console.aws.amazon.com/rds/.
2. Nella sezione Risorse seleziona Cluster database.
3. Seleziona il cluster database Aurora Serverless v1 che desideri eliminare.
4. In Actions (Azioni), selezionare Delete (Elimina). Verrà richiesto di confermare che si desidera eliminare il cluster database Aurora Serverless v1.
5. Si consiglia di mantenere le opzioni preselezionate:
   - Sì per Crea snapshot finale
   - Il nome del cluster database Aurora Serverless v1 più `-final-snapshot` per Nome snapshot finale. Tuttavia, puoi modificare il nome della snapshot finale in questo campo.



   Se scegli No per Crea snapshot finale non potrai ripristinare il cluster database utilizzando le snapshot o il ripristino point-in-time.
6. Seleziona Elimina cluster database.

Aurora Serverless v1 elimina il cluster di database. Se decidi di avere uno snapshot finale, lo stato del cluster database Aurora Serverless v1 diventa "Backup in corso" prima che venga eliminato e non viene più visualizzato nell'elenco.

AWS CLI

Prima di iniziare, configura AWS CLI con l'ID chiave di accesso di AWS, la chiave di accesso segreta di AWS e la Regione AWS in cui si trova il cluster database Aurora Serverless v1. Per ulteriori informazioni, consulta i passaggi di base della configurazione nella Guida per l'utente di AWS Command Line Interface.

Non puoi eliminare un cluster database Aurora Serverless v1 fino a quando disattivi la protezione dall'eliminazione per i cluster configurati con questa opzione. Se provi a eliminare un cluster con questa opzione di protezione attivata, viene visualizzato il seguente messaggio di errore.

```
An error occurred (InvalidParameterCombination) when calling the DeleteDBCluster
  operation: Cannot delete protected Cluster, please disable deletion protection and
 try again.
```

Puoi modificare l'impostazione di protezione da eliminazione del cluster database Aurora Serverless v1 utilizzando il comando AWS CLI comando modify-db-cluster come riportato di seguito:

```
aws rds modify-db-cluster --db-cluster-identifier your-cluster-name --no-deletion-
protection
```

Con questo comando vengono restituite le proprietà revisionate per il cluster database specificato. Adesso puoi eliminare il cluster database Aurora Serverless v1.

Consigliamo di creare sempre uno snapshot finale ogni volta che si elimina un cluster database Aurora Serverless v1. Il seguente esempio di utilizzo del comando AWS CLI delete-db-cluster mostra come completare questa operazione. Specifica il nome del cluster database e un nome per la snapshot.

Per Linux, macOS o Unix:

```
aws rds delete-db-cluster --db-cluster-identifier \
  your-cluster-name --no-skip-final-snapshot \
  --final-db-snapshot-identifier name-your-snapshot
```

Per Windows:

```
aws rds delete-db-cluster --db-cluster-identifier ^
  your-cluster-name --no-skip-final-snapshot ^
  --final-db-snapshot-identifier name-your-snapshot
```

Versioni del motore del database di Aurora Serverless v1 e Aurora

> **Important**
>
> AWS ha annunciato la data di fine del ciclo di vita per Aurora Serverless v1: 31 marzo 2025. Tutti i cluster Aurora Serverless v1 di cui non è stata effettuata la migrazione entro il 31 marzo 2025 verranno spostati su Aurora Serverless v2 durante la finestra di manutenzione. Se l'aggiornamento non riesce, Amazon Aurora converte il cluster Serverless v1 in un cluster con provisioning con la versione del motore equivalente durante la finestra di manutenzione. Se applicabile, Amazon Aurora registrerà il cluster con provisioning convertito in supporto esteso per Amazon RDS. Per ulteriori informazioni, consulta Estensione del supporto di RDS.

Aurora Serverless v1 è disponibile in determinate Regioni AWS solo per versioni specifiche di Aurora MySQL e Aurora PostgreSQL. Per l'elenco aggiornato delle Regioni AWS che supportano Aurora Serverless v1 e le versioni specifiche di Aurora MySQL e Aurora PostgreSQL disponibili in ogni regione, consulta Aurora Serverless v1.

Aurora Serverless v1 utilizza il motore del database Aurora associato per identificare le release supportate specifiche per ogni motore del database supportato, come segue:

- Aurora MySQL Serverless
- Aurora PostgreSQL Serverless

Quando una release secondaria dei motori di database diventa disponibile per Aurora Serverless v1, viene applicata automaticamente nelle varie Regioni AWS in cui è disponibile Aurora Serverless v1. In altre parole, non è necessario aggiornare il cluster database Aurora Serverless v1 per ottenere una nuova release secondaria per il motore del database del cluster se è disponibile per Aurora Serverless v1.

Aurora MySQL Serverless

Aurora Serverless v1 è disponibile in determinate Regioni AWS solo per versioni specifiche di Aurora MySQL. Per l'elenco aggiornato delle Regioni AWS che supportano Aurora Serverless v1 e le versioni specifiche di Aurora MySQL disponibili in ogni Regione, consulta Aurora Serverless v1 con Aurora MySQL.

Per informazioni sui miglioramenti e sulle correzioni di bug per la versione 2 di Aurora MySQL, consulta Aggiornamenti del motore del database per Amazon Aurora MySQL versione 2 nelle Note di rilascio di Aurora MySQL.

Per utilizzare una versione più recente di Aurora MySQL, puoi utilizzare Aurora Serverless v2. Per informazioni sulle Regioni AWS e sulle versioni di Aurora MySQL che puoi utilizzare con Aurora Serverless v2, consulta Aurora Serverless v2 con Aurora MySQL. Per informazioni sull'utilizzo di Aurora Serverless v2, consulta Uso di Aurora Serverless v2.

Aurora PostgreSQL Serverless

Aurora Serverless v1 è disponibile in determinate Regioni AWS solo per versioni specifiche di Aurora PostgreSQL. Per l'elenco aggiornato delle Regioni AWS che supportano Aurora Serverless v1 e le versioni specifiche di Aurora PostgreSQL disponibili in ogni Regione, consulta Aurora Serverless v1 con Aurora PostgreSQL.

Se desideri utilizzare Aurora PostgreSQL per il cluster di database Aurora Serverless v1, puoi scegliere tra le versioni compatibili con Aurora PostgreSQL 13. Le release secondarie per Aurora edizione compatibile con PostgreSQL includono solo le modifiche compatibili con le versioni precedenti. Il cluster database Aurora Serverless v1 viene aggiornato in modo trasparente quando una release secondaria di Aurora PostgreSQL diventa disponibile per Aurora Serverless v1 nella tua Regione AWS.

Per utilizzare una versione più recente di Aurora PostgreSQL, puoi utilizzare Aurora Serverless v2. Per informazioni sulle Regioni AWS e sulle versioni di Aurora PostgreSQL che puoi utilizzare con Aurora Serverless v2, consulta Aurora Serverless v2 con Aurora PostgreSQL. Per informazioni sull'utilizzo di Aurora Serverless v2, consulta Uso di Aurora Serverless v2.

Aggiornamenti automatici a versioni secondarie per Aurora Serverless v1

Quando una release secondaria dei motori di database diventa disponibile per Aurora Serverless v1, viene applicata automaticamente nelle varie Regioni AWS in cui è disponibile Aurora Serverless v1.

In altre parole, non è necessario aggiornare il cluster database Aurora Serverless v1 per ottenere una nuova release secondaria per il motore del database del cluster se è disponibile per Aurora Serverless v1.

Utilizzo dell’API dati di Amazon RDS

Utilizzando l’API dati di RDS (API dati), puoi lavorare con un’interfaccia di servizi web per il cluster di database Aurora. L’API dati non richiede una connessione persistente al cluster di database. Fornisce invece un endpoint HTTP sicuro e l'integrazione con AWS SDKs. Puoi usare l'endpoint per eseguire istruzioni SQL senza gestire connessioni.

Gli utenti non devono passare le credenziali con le chiamate a Data API, poiché Data API utilizza le credenziali del database archiviate in. Gestione dei segreti AWS Per archiviare le credenziali in Secrets Manager, agli utenti devono essere concesse le autorizzazioni appropriate per l’utilizzo di Secrets Manager e anche l’API dati. Per ulteriori informazioni sull'autorizzazione degli utenti, consulta Autorizzazione di accesso all’API dati di Amazon RDS.

Puoi anche utilizzare Data API per integrare Amazon Aurora con altre AWS applicazioni come AWS Lambda AWS AppSync, e. AWS Cloud9 L’API dati fornisce un modo più sicuro per utilizzare AWS Lambda. Consente di accedere al cluster database senza dover configurare una funzione Lambda per accedere alle risorse in un cloud privato virtuale (VPC). Per ulteriori informazioni, consulta AWS Lambda, AWS AppSync e AWS Cloud9.

Puoi abilitare l’API dati al momento della creazione del cluster di database Aurora. È inoltre possibile modificare la configurazione in un secondo momento. Per ulteriori informazioni, consulta Abilitazione dell’API dati di Amazon RDS.

Dopo aver abilitato l’API dati, puoi anche utilizzare l’editor di query per eseguire query ad hoc senza configurare uno strumento di query per accedere ad Aurora in un VPC. Per ulteriori informazioni, consulta Utilizzo dell’editor di query Aurora.

Argomenti

- Disponibilità per Regione e versione dell’API dati di Amazon RDS
- Utilizzo IPv6 con Amazon RDS Data API
- Limitazioni dell’API dati di Amazon RDS
- Confronto dei comportamenti delle API dati di Amazon RDS per i cluster Aurora Serverless v2 e con provisioning con i cluster Aurora Serverless v1
- Autorizzazione di accesso all’API dati di Amazon RDS
- Abilitazione dell’API dati di Amazon RDS
- Creazione di un endpoint Amazon VPC per l’API dati di Amazon RDS (AWS PrivateLink)

- Chiamata dell'API dati di Amazon RDS
- Utilizzo della libreria client Java per l'API dati
- Elaborazione dei risultati delle query dell'API dati di Amazon RDS in formato JSON
- Risoluzione dei problemi relativi all'API dati di Amazon RDS
- Registrazione di log delle chiamate API dati di Amazon RDS con AWS CloudTrail
- Monitoraggio delle query dell'API dati RDS con Approfondimenti sulle prestazioni

Disponibilità per Regione e versione dell'API dati di Amazon RDS

Per informazioni sulle Regioni e sulle versioni del motore disponibili per l'API dati, consulta le sezioni seguenti.

Tipo di cluster	Disponibilità di regioni e versioni
Aurora PostgreSQL con provisioning e Serverless v2	API dati con Aurora PostgreSQL Serverless v2 e cluster allocati
Aurora MySQL con provisioning e Serverless v2	API dati con Aurora MySQL Serverless v2 e cluster allocati
Aurora PostgreSQL Serverless v1	API dati con Aurora PostgreSQL Serverless v1
Aurora MySQL Serverless v1	API dati con Aurora MySQL Serverless v1

Se richiedi moduli crittografici convalidati da FIPS 140-2 quando accedi all'API dati tramite interfaccia a riga di comando o API, utilizza un endpoint FIPS. Per ulteriori informazioni sugli endpoint FIPS disponibili, consulta il Federal Information Processing Standard (FIPS) 140-2.

Utilizzo IPv6 con Amazon RDS Data API

Amazon RDS Data API supporta la IPv6 connettività tramite endpoint dual-stack. Ciò consente di connettersi a Data API utilizzando IPv6 gli indirizzi mantenendo la compatibilità con le versioni precedenti con. IPv4

IPv6 supporto per endpoint

Data API fornisce endpoint dual-stack che supportano sia le connessioni che le connessioni. IPv4 IPv6 Questi endpoint utilizzano il dominio `.aws` anziché il tradizionale dominio `.amazonaws.com`.

I tipi di endpoint disponibili sono:

Endpoint pubblici dual stack

- Formato: `rds-data.`*`region`*`.api.aws`
- Ad esempio: `rds-data.us-east-1.api.aws`

Endpoint dual stack FIPS

- Formato: `rds-data-fips.`*`region`*`.api.aws`
- Ad esempio: `rds-data-fips.us-east-1.api.aws`

PrivateLink IPv6 endpoint

- Disponibile tramite endpoint VPC con supporto IPv6
- Consente la IPv6 connettività privata all'interno del tuo VPC

Endpoint solo per versioni precedenti IPv4

Gli `.amazonaws.com` endpoint esistenti continuano a supportare IPv4 solo connessioni:

- `rds-data.`*`region`*`.amazonaws.com`
- `rds-data-fips.`*`region`*`.amazonaws.com`

> **Note**
>
> Gli endpoint legacy restano invariati per garantire la compatibilità con le applicazioni esistenti.

Utilizzo degli endpoint IPv6

Per utilizzarla IPv6 con Data API, aggiorna l'applicazione per utilizzare i nuovi endpoint dual-stack. L'applicazione verrà utilizzata automaticamente, IPv6 se disponibile, oppure ricorrerà a. IPv4

Per indicazioni generali sulla configurazione del tuo VPC, consulta la sezione Migrazione verso IPv6 IPv6 nella Amazon VPC User Guide.

Puoi configurare gli IPv6 endpoint in due modi:

- Utilizzo della variabile di ambiente: impostata `AWS_USE_DUALSTACK_ENDPOINT=true` nel proprio IPv6 ambiente. AWS CLI and AWS SDKs costruirà automaticamente gli `api.aws` endpoint appropriati senza la necessità di specificare l'endpoint URLs manualmente.
- Utilizzo di un endpoint esplicito URLs: specifica l'URL dell'endpoint dual-stack direttamente nei AWS CLI comandi o nella configurazione dell'SDK, come mostrato negli esempi seguenti.

AWS CLI configurazione

Configura AWS CLI gli IPv6 endpoint da utilizzare specificando l'URL dell'endpoint:

Per Linux, macOS o Unix:

```
aws rds-data execute-statement \
    --endpoint-url https://rds-data.us-east-1.api.aws \
    --resource-arn "arn:aws:rds:us-east-1:123456789012:cluster:my-cluster" \
    --secret-arn "arn:aws:secretsmanager:us-east-1:123456789012:secret:my-secret" \
    --database "mydb" \
    --sql "SELECT * FROM users LIMIT 10"
```

Per Windows:

```
aws rds-data execute-statement ^
    --endpoint-url https://rds-data.us-east-1.api.aws ^
    --resource-arn "arn:aws:rds:us-east-1:123456789012:cluster:my-cluster" ^
    --secret-arn "arn:aws:secretsmanager:us-east-1:123456789012:secret:my-secret" ^
    --database "mydb" ^
    --sql "SELECT * FROM users LIMIT 10"
```

AWS Configurazione SDK

Configura AWS SDKs per utilizzare gli endpoint dual-stack:

Python

```
import boto3
```

```
# Create RDS Data API client with IPv6 dual-stack endpoint
client = boto3.client(
    'rds-data',
    endpoint_url='https://rds-data.us-east-1.api.aws'
)

# Execute a SQL statement
response = client.execute_statement(
    resourceArn='arn:aws:rds:us-east-1:123456789012:cluster:my-cluster',
    secretArn='arn:aws:secretsmanager:us-east-1:123456789012:secret:my-secret',
    database='mydb',
    sql='SELECT * FROM users LIMIT 10'
)

print(response['records'])
```

Java

```
import software.amazon.awssdk.services.rdsdata.RdsDataClient;
import software.amazon.awssdk.services.rdsdata.model.ExecuteStatementRequest;
import software.amazon.awssdk.services.rdsdata.model.ExecuteStatementResponse;
import java.net.URI;

// Create RDS Data API client with IPv6 dual-stack endpoint
RdsDataClient client = RdsDataClient.builder()
    .endpointOverride(URI.create("https://rds-data.us-east-1.api.aws"))
    .build();

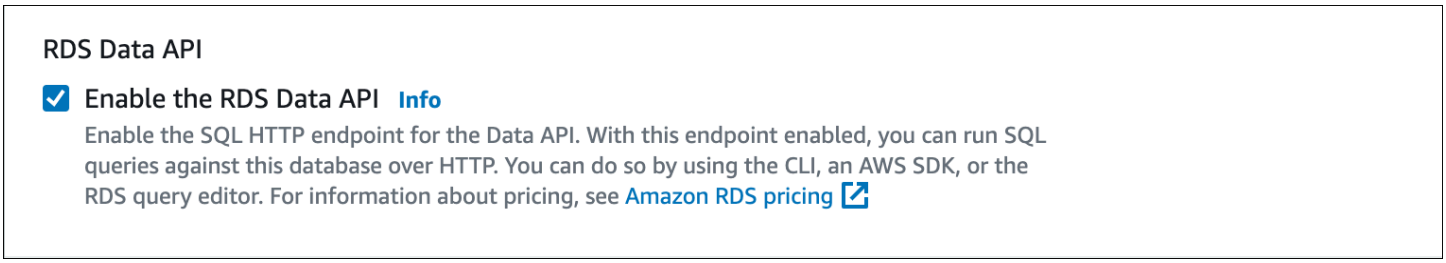
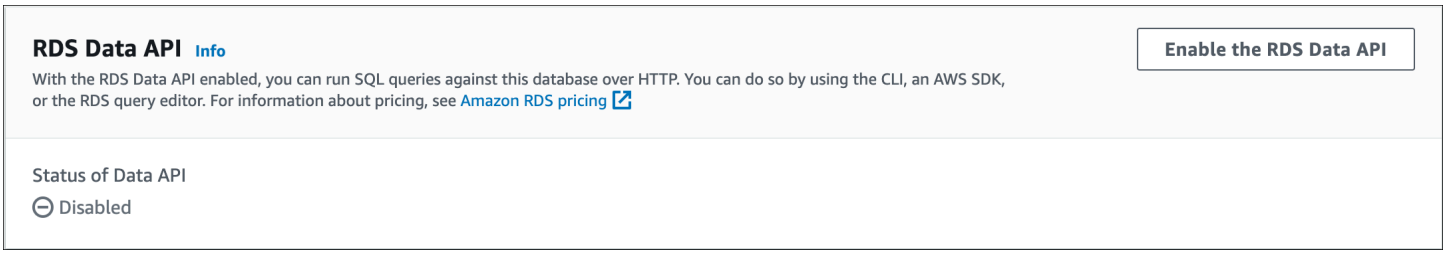
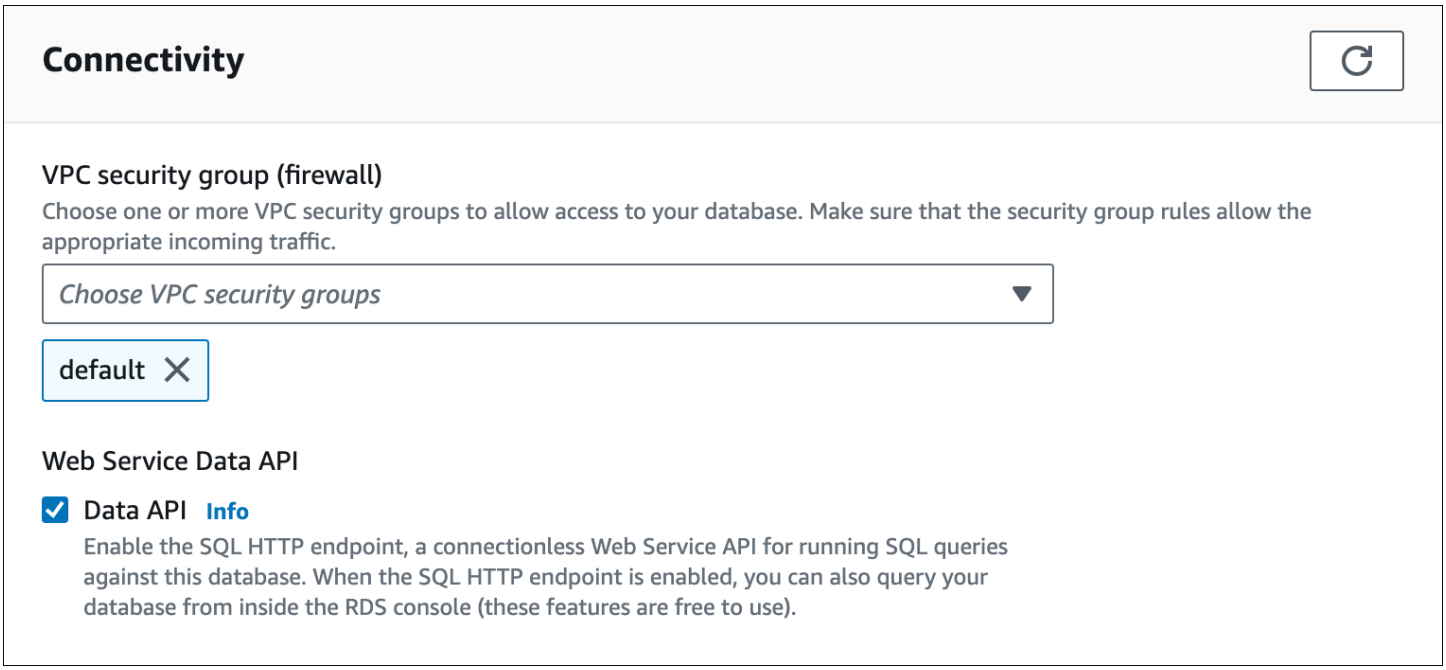
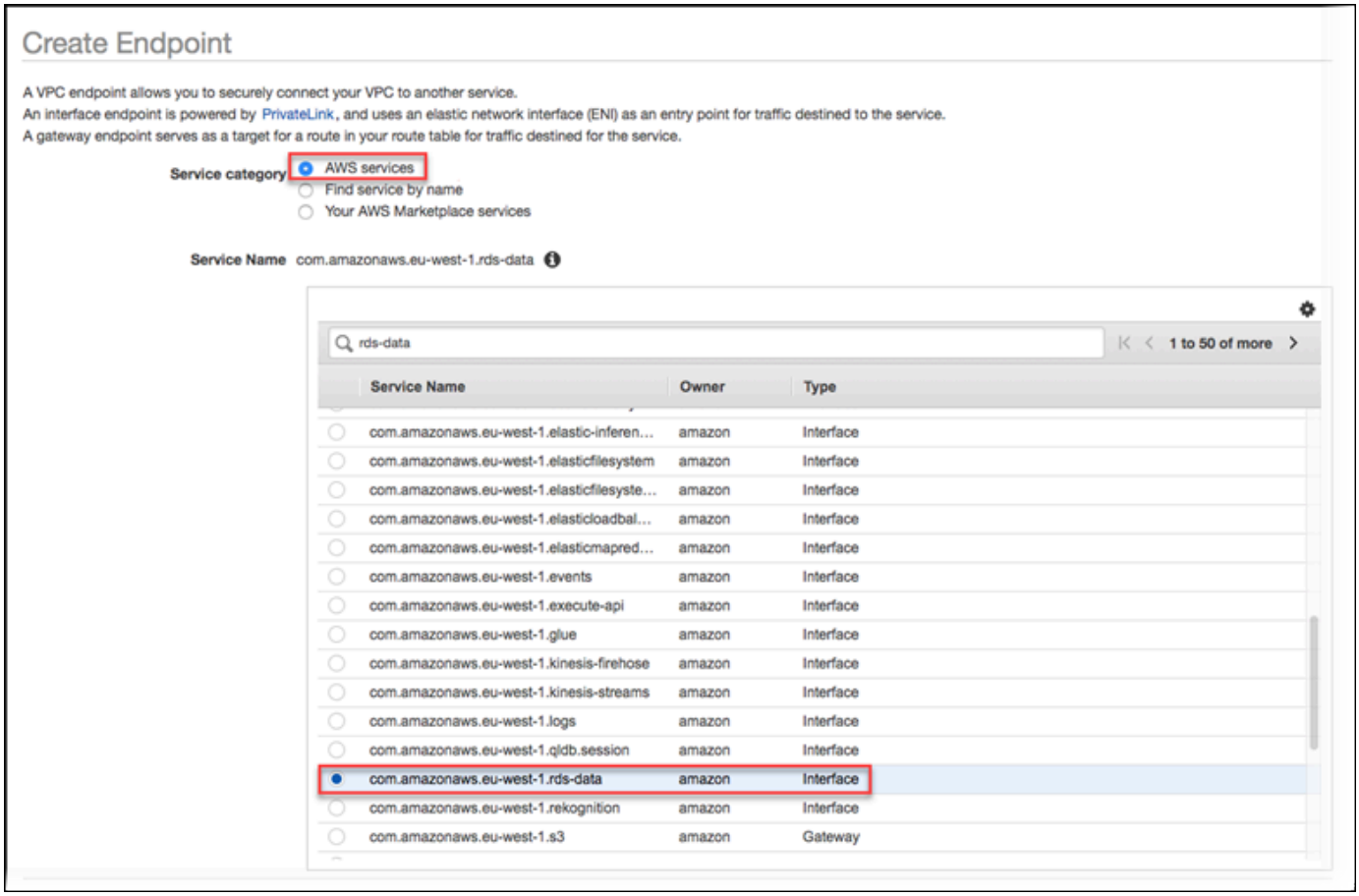
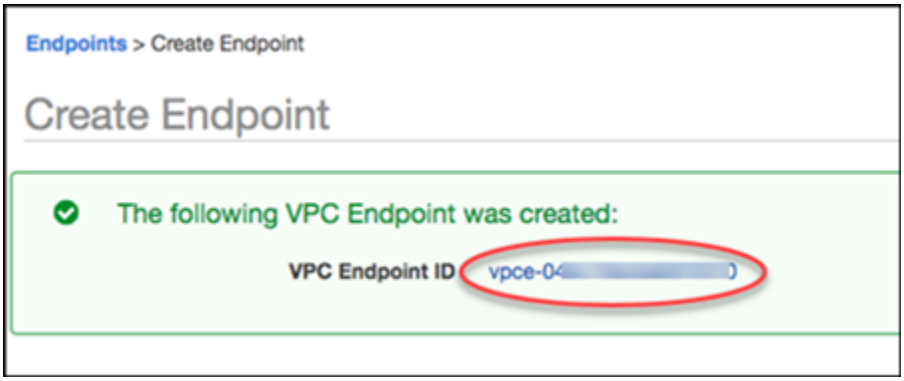
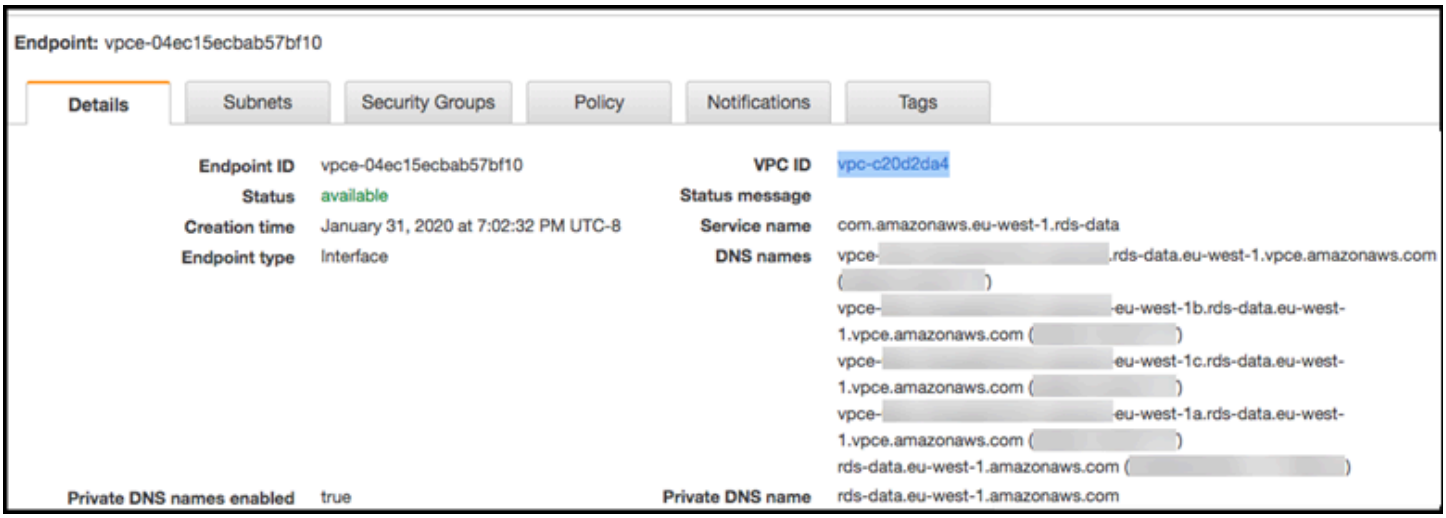
// Execute a SQL statement
ExecuteStatementRequest request = ExecuteStatementRequest.builder()
    .resourceArn("arn:aws:rds:us-east-1:123456789012:cluster:my-cluster")
    .secretArn("arn:aws:secretsmanager:us-east-1:123456789012:secret:my-secret")
    .database("mydb")
    .sql("SELECT * FROM users LIMIT 10")
    .build();

ExecuteStatementResponse response = client.executeStatement(request);
System.out.println(response.records());
```

JavaScript

```
const { RDSDataClient, ExecuteStatementCommand } = require("@aws-sdk/client-rds-
data");
```

```
// Create RDS Data API client with IPv6 dual-stack endpoint
const client = new RDSDataClient({
    endpoint: "https://rds-data.us-east-1.api.aws"
});

// Execute a SQL statement
const command = new ExecuteStatementCommand({
    resourceArn: "arn:aws:rds:us-east-1:123456789012:cluster:my-cluster",
    secretArn: "arn:aws:secretsmanager:us-east-1:123456789012:secret:my-secret",
    database: "mydb",
    sql: "SELECT * FROM users LIMIT 10"
});

const response = await client.send(command);
console.log(response.records);
```

Utilizzo con AWS PrivateLink IPv6

Puoi creare endpoint VPC per Data API che supportano la IPv6 connettività all'interno del tuo VPC. Per istruzioni dettagliate sulla creazione di endpoint VPC per l'API dati, consulta Creazione di un endpoint Amazon VPC per l'API dati di Amazon RDS (AWS PrivateLink).

Quando crei un endpoint VPC per il IPv6 supporto, assicurati che:

- Il VPC e le sottoreti sono configurati per supportare IPv6
- I gruppi di sicurezza consentono il IPv6 traffico sulle porte richieste (in genere 443 per HTTPS)
- ACLs Le reti sono configurate per consentire il traffico IPv6

Considerazioni sulla migrazione

Durante la migrazione agli IPv6 endpoint, considera quanto segue:

- Migrazione graduale: è possibile migrare le applicazioni gradualmente aggiornando l'endpoint URLs un'applicazione alla volta.
- Compatibilità di rete: assicuratevi che l'infrastruttura di rete supporti IPv6 prima della migrazione.
- Politiche di sicurezza: aggiorna le regole del gruppo di sicurezza e la rete ACLs per consentire IPv6 il traffico, se necessario.

- Monitoraggio: aggiorna le configurazioni di monitoraggio e registrazione per gestire gli indirizzi. IPv6

> **Note**
>
> Indirizzi di connessione al database: quando si utilizzano gli IPv6 endpoint for Data API, le connessioni al database e i registri del database sottostanti continueranno a mostrare gli indirizzi. IPv4 Si tratta di un comportamento previsto e non influisce sulla funzionalità delle applicazioni IPv6 abilitate.

Risoluzione dei problemi IPv6 di connessione

Se riscontri problemi di IPv6 connettività, controlla quanto segue:

Configurazione della rete

Verifica che la rete supporti IPv6 e che il IPv6 routing sia configurato correttamente.

Risoluzione DNS

Verifica che il risolutore DNS sia in grado di risolvere i record AAAA per gli endpoint dual stack.

Gruppi di sicurezza

Aggiorna le regole del gruppo di sicurezza per consentire il IPv6 traffico sulle porte richieste (in genere 443 per HTTPS).

Librerie client

Verifica che le librerie dei client HTTP IPv6 supportino la connettività dual-stack.

Limitazioni dell'API dati di Amazon RDS

L'API dati di RDS presenta le seguenti limitazioni:

- È possibile eseguire query dell'API dati solo sulle istanze di scrittura di un cluster di database. Tuttavia, le istanze di scrittura possono accettare sia query di scrittura che di lettura.
- Con i database globali Aurora, è possibile abilitare l'API dati sia sul cluster di database primario che su quello secondario. Tuttavia, un cluster secondario non dispone di un'istanza di scrittura

finché non viene promosso a primario. L'API dati richiede l'accesso all'istanza di scrittura per l'elaborazione delle query, anche per le query di sola lettura. Di conseguenza, le query di lettura e scrittura inviate al cluster secondario non riescono finché questo non dispone di un'istanza di scrittura. Una volta che il cluster secondario viene promosso e dispone di un'istanza di scrittura, le query dell'API dati su tale istanza database vengono eseguite correttamente.

- L'API dati non è supportata sulle classi di istanza database T.
- Per i cluster di database Aurora Serverless v2 e con provisioning, l'API dati di RDS non supporta alcuni tipi di dati. Per l'elenco dei tipi supportati, consulta the section called "API dati con Aurora Serverless v2 rispetto ad Aurora Serverless v1".
- Per i database Aurora PostgreSQL versione 14 e successive, l'API dati supporta solo `scram-sha-256` per la crittografia delle password.
- Il limite di dimensioni della risposta è 1 MiB. Se la chiamata restituisce più di 1 MiB di dati di risposta, verrà terminata.
- Per Aurora Serverless v1, il numero massimo di richieste al secondo è 1.000. Per tutti gli altri database supportati, non esiste alcun limite.
- Il limite delle dimensioni dell'API dati è 64 KB per riga nel set di risultati restituito dal database. Assicurati che ogni riga di un set di risultati non superi i 64 KB.

Confronto dei comportamenti delle API dati di Amazon RDS per i cluster Aurora Serverless v2 e con provisioning con i cluster Aurora Serverless v1

I miglioramenti più recenti apportati alle API dati di Amazon RDS rendono le API dati disponibili per i cluster che utilizzano versioni recenti dei motori PostgreSQL o MySQL. Questi cluster possono essere configurati per utilizzare Aurora Serverless v2 oppure classi di istanza con provisioning, come `db.r6g` o `db.r6i`.

Le sezioni seguenti descrivono le differenze delle API dati di Amazon RDS tra cluster Aurora Serverless v2 e cluster di database con provisioning e cluster di database Aurora Serverless v1. Aurora Serverless v1 I cluster di database utilizzano la modalità motore `serverless`. I cluster di database con provisioning utilizzano la modalità motore `provisioned`. Un cluster di database Aurora Serverless v2 utilizza anch'esso la modalità motore `provisioned` e contiene una o più istanze database Aurora Serverless v2 con la classe di istanza `db.serverless`.

Numero massimo di richieste al secondo

Aurora Serverless v1

Le API dati possono eseguire fino a 1.000 richieste al secondo.

Aurora Serverless v2

Le API dati possono eseguire un numero illimitato di richieste al secondo.

Abilitazione o disabilitazione dell'API dati di Amazon RDS su un database esistente

Aurora Serverless v1

- Con l'API Amazon RDS: utilizza l'operazione `ModifyCluster` e specifica `True` o `False`, a seconda dei casi, per il parametro `EnableHttpEndpoint`.
- Con la AWS CLI: utilizza l'operazione `modify-db-cluster` con l'opzione `--enable-http-endpoint` o `--no-enable-http-endpoint`, a seconda dei casi.

Aurora Serverless v2

- Con l'API Amazon RDS: utilizza le operazioni `EnableHttpEndpoint` e `DisableHttpEndpoint`.
- Con la AWS CLI: utilizza le operazioni `enable-http-endpoint` e `disable-http-endpoint`.

Eventi CloudTrail

Aurora Serverless v1

Gli eventi derivanti dalle chiamate API dati sono eventi di gestione. Questi eventi vengono automaticamente inclusi in un trail per impostazione predefinita. Per ulteriori informazioni, consulta the section called "Esclusione di eventi API dati da un trail CloudTrail (solo Aurora Serverless v1)".

Aurora Serverless v2

Gli eventi derivanti dalle chiamate API dati sono eventi di dati. Questi eventi vengono automaticamente esclusi da un trail per impostazione predefinita. Per ulteriori informazioni, consulta the section called "Inclusione di eventi API dati in un trail CloudTrail".

Supporto per istruzioni multiple

Aurora Serverless v1

- Per Aurora MySQL, le istruzioni multiple non sono supportate.
- Per Aurora PostgreSQL, le istruzioni multiple restituiscono solo la risposta della prima query.

Aurora Serverless v2

Le istruzioni multiple non sono supportate. Il tentativo di eseguire più istruzioni in una singola chiamata API restituisce "`An error occurred (ValidationException) when calling the ExecuteStatement operation: Multistatements aren't supported.`". Per eseguire più istruzioni, effettua chiamate API `ExecuteStatement` separate oppure utilizza `BatchExecuteStatement` per l'elaborazione in batch.

L'esempio seguente mostra il messaggio di errore risultante da una chiamata API che tenta di eseguire più istruzioni.

```
 aws rds-data execute-statement \
    --resource-arn "arn:aws:rds:region:account:cluster:cluster-name" \
    --secret-arn "arn:aws:secretsmanager:region:account:secret:secret-name" \
    --database "your_database" \
    --sql "SELECT * FROM your_table; Select * FROM next_table;

                                "An error occurred (ValidationException) when calling
 the ExecuteStatement operation: Multistatements aren't supported.
```

L'esempio seguente mostra l'esecuzione di più istruzioni tramite chiamate API `ExecuteStatement` separate.

```
aws rds-data execute-statement \
    --resource-arn "arn:aws:rds:region:account:cluster:cluster-name" \
    --secret-arn "arn:aws:secretsmanager:region:account:secret:secret-name" \
    --database "your_database" \
    --sql "SELECT * FROM your_table;"

aws rds-data execute-statement \
    --resource-arn "arn:aws:rds:region:account:cluster:cluster-name" \
    --secret-arn "arn:aws:secretsmanager:region:account:secret:secret-name" \
    --database "your_database" \
```

```
    --sql "SELECT * FROM next_table;"
```

Richieste simultanee per lo stesso ID transazione

Aurora Serverless v1

Le richieste successive attendono fino al completamento della richiesta corrente. L'applicazione deve gestire gli errori di timeout se il periodo di attesa è troppo lungo.

Aurora Serverless v2

Quando l'API dati riceve più richieste con lo stesso ID transazione, restituisce immediatamente questo errore:

`DatabaseErrorException: Transaction is still running a query`

Questo errore si verifica in due situazioni:

- L'applicazione effettua richieste asincrone (ad esempio con promesse JavaScript) utilizzando lo stesso ID transazione.
- Una richiesta precedente con tale ID transazione è ancora in elaborazione.

L'esempio seguente mostra tutte le richieste eseguite in parallelo con `promise.all()`.

```
const api_calls = [];
for (let i = 0; i < 10; i++) {
api_calls.push(
    client.send(
    new ExecuteStatementCommand({
        ...params,
        sql: `insert into table_name values (i);`,
        transactionId
    })
    )
);
}
await Promise.all(api_calls);
```

Per risolvere questo errore, attendi il completamento della richiesta corrente prima di inviare un'altra richiesta con lo stesso ID transazione o rimuovi l'ID transazione per consentire le richieste parallele.

L'esempio seguente mostra una chiamata API che utilizza l'esecuzione sequenziale con lo stesso ID transazione.

```
 for (let i = 0; i < 10; i++) {
    await client.send(
    new ExecuteStatementCommand({
        ...params,
        sql: `insert into table_name values (i);`,
        transactionId
    })
    ).promise()
);
}
```

Comportamento di BatchExecuteStatement

Per ulteriori informazioni su `BatchExecuteStatement`, consulta BatchExecuteStatement.

Aurora Serverless v1

L'oggetto dei campi generati nel risultato dell'aggiornamento include i valori inseriti.

Aurora Serverless v2

- Per Aurora MySQL, l'oggetto campi generati nel risultato dell'aggiornamento include i valori inseriti.
- Per Aurora PostgreSQL, l'oggetto campi generati è vuoto.

Comportamento di ExecuteSQL

Per ulteriori informazioni su `ExecuteSQL`, consulta ExecuteSQL.

Aurora Serverless v1

L'operazione `ExecuteSQL` è obsoleta.

Aurora Serverless v2

L'operazione `ExecuteSQL` non è supportata.

Comportamento di ExecuteStatement

Per ulteriori informazioni su `ExecuteStatement`, consulta ExecuteStatement.

Aurora Serverless v1

Il parametro `ExecuteStatement` supporta lo strumento di recupero di colonne di array multidimensionali e di tutti i tipi di dati avanzati.

Aurora Serverless v2

Il parametro `ExecuteStatement` non supporta di colonne di array multidimensionali. Non supporta inoltre alcuni tipi di dati PostgreSQL, inclusi i tipi geometrici e monetari. Quando l'API dati rileva un tipo di dato non supportato, restituisce l'errore: `UnsupportedResultException: The result contains the unsupported data type data_type`.

Per risolvere il problema, esegui il casting del tipo di dati non supportato su `TEXT`. L'esempio seguente mostra come eseguire il casting di un tipo di dati non supportato su `TEXT`.

```
SELECT custom_type::TEXT FROM my_table;--
ORSELECT CAST(custom_type AS TEXT) FROM my_table;
```

Per un elenco dei tipi di dati supportati per ciascun motore di database Aurora, consulta il riferimento alle operazioni dell'API dati.

Comportamento del parametro Schema

Aurora Serverless v1

Il parametro `Schema` non è supportato. Quando includi il parametro `Schema` in una chiamata API, l'API dati ignora il parametro.

Aurora Serverless v2

Il parametro `Schema` è obsoleto. Quando includi il parametro `Schema` in una chiamata API, l'API dati restituisce l'errore: `ValidationException: The schema parameter isn't supported`. L'esempio seguente mostra una chiamata API dati che restituisce l'errore `ValidationException`.

```
aws rds-data execute-statement \
--resource-arn "arn:aws:rds:region:account:cluster:cluster-name" \
--secret-arn "arn:aws:secretsmanager:region:account:secret:secret-name" \
--database "your_database" \
--schema "your_schema" \
--sql "SELECT * FROM your_table LIMIT 10"
```

Per risolvere questo problema, rimuovi il parametro `Schema` dalla chiamata API.

L'esempio seguente mostra una chiamata API dati con il parametro `Schema` rimosso.

```
aws rds-data execute-statement \
--resource-arn "arn:aws:rds:region:account:cluster:cluster-name" \
--secret-arn "arn:aws:secretsmanager:region:account:secret:secret-name" \
--database "your_database" \
--sql "SELECT * FROM your_table LIMIT 10"
```

Autorizzazione di accesso all'API dati di Amazon RDS

Gli utenti possono invocare le operazioni dell'API dati di Amazon RDS (API dati) solo se sono autorizzati a farlo. Puoi concedere a un utente l'autorizzazione a utilizzare Data API allegando una policy AWS Identity and Access Management (IAM) che ne definisce i privilegi. È inoltre possibile associare la policy a un ruolo se si utilizzano ruoli IAM. Una policy AWS gestita include `AmazonRDSDataFullAccess` le autorizzazioni per Data API.

La `AmazonRDSDataFullAccess` politica include anche le autorizzazioni da cui l'utente può ottenere il valore di un segreto. Gestione dei segreti AWS Gli utenti devono utilizzare Secrets Manager per archiviare i segreti che possono utilizzare nelle loro chiamate all'API dati. Utilizzare segreti significa che gli utenti non devono includere le credenziali del database per le risorse target nelle chiamate all'API dati. L'API dati chiama in modo trasparente Secrets Manager, che consente (o nega) la richiesta dell'utente per il segreto. Per informazioni sulla configurazione di segreti da utilizzare con l'API dati, consulta Archiviazione delle credenziali del database in Gestione dei segreti AWS.

La policy `AmazonRDSDataFullAccess` fornisce l'accesso completo (tramite l'API dati) alle risorse. Puoi restringere l'ambito definendo le tue policy che specificano l'Amazon Resource Name (ARN) di una risorsa.

Ad esempio, la policy seguente mostra un esempio delle autorizzazioni minime richieste per un utente per accedere all'API dati per il cluster di database identificato dal relativo ARN. La policy include le autorizzazioni necessarie per accedere Secrets Manager e ottenere l'autorizzazione all'istanza DB per l'utente.

JSON

```
{
```

```
    "Version":"2012-10-17",
    "Statement": [
        {
            "Sid": "SecretsManagerDbCredentialsAccess",
            "Effect": "Allow",
            "Action": [
                "secretsmanager:GetSecretValue"
            ],
            "Resource": "arn:aws:secretsmanager:*:*:secret:rds-db-credentials/*"
        },
        {
            "Sid": "RDSDataServiceAccess",
            "Effect": "Allow",
            "Action": [
                "rds-data:BatchExecuteStatement",
                "rds-data:BeginTransaction",
                "rds-data:CommitTransaction",
                "rds-data:ExecuteStatement",
                "rds-data:RollbackTransaction"
            ],
            "Resource": "arn:aws:rds:us-east-2:111122223333:cluster:prod"
        }
    ]
}
```

Si consiglia di utilizzare un ARN specifico per l'elemento "Risorse" nelle istruzioni delle policy (come illustrato nell'esempio) anziché un carattere jolly (*).

Utilizzo dell'autorizzazione basata su tag

L'API dati di RDS (API dati) e Secrets Manager supportano entrambi l'autorizzazione basata su tag. Itag sono coppie chiave-valore che etichettano una risorsa, ad esempio un cluster RDS, con un valore stringa aggiuntivo, ad esempio:

- `environment:production`
- `environment:development`

È possibile applicare tag alle risorse per l'allocazione dei costi, il supporto delle operazioni, il controllo degli accessi e molti altri motivi. Se non disponi già di tag sulle tue risorse e desideri applicarli, puoi saperne di più alla pagina Applicazione di tag alle risorse Amazon RDS. È possibile utilizzare i tag

nelle istruzioni delle policy per limitare l'accesso ai cluster RDS etichettati con questi tag. Ad esempio, un cluster DB Aurora potrebbe avere tag che identificano l'ambiente come produzione o sviluppo.

Nell'esempio seguente viene illustrato come utilizzare i tag nelle istruzioni delle policy. Questa istruzione richiede che sia il cluster e il segreto passato nella richiesta API dati abbiano un tag `environment:production`.

Ecco come viene applicata la policy: quando un utente effettua una chiamata utilizzando l'API dati, la richiesta viene inviata al servizio. L'API dati verifica innanzitutto che l'ARN del cluster passato nella richiesta sia contrassegnato con `environment:production`. Quindi chiama Secrets Manager per recuperare il valore del segreto dell'utente nella richiesta. Secrets Manager verifica inoltre che il segreto dell'utente sia contrassegnato con `environment:production`. In tal caso, l'API dati utilizza quindi il valore recuperato per la password DB dell'utente. Infine, se anche questo è corretto, la richiesta dell'API dati viene richiamata correttamente per l'utente.

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Sid": "SecretsManagerDbCredentialsAccess",
            "Effect": "Allow",
            "Action": [
                 "secretsmanager:GetSecretValue"
               ],
            "Resource": "arn:aws:secretsmanager:*:*:secret:rds-db-credentials/*",
            "Condition": {
                    "StringEquals": {
                        "aws:ResourceTag/environment": [
                                        "production"
                                       ]
                     }
             }
        },
        {
            "Sid": "RDSDataServiceAccess",
            "Effect": "Allow",
            "Action": [
                  "rds-data:*"
               ],
```

```
            "Resource": "arn:aws:rds:us-east-2:111122223333:cluster:*",
            "Condition": {
                    "StringEquals": {
                        "aws:ResourceTag/environment": [
                                        "production"
                                       ]
                     }
                }
            }
        ]
}
```

L'esempio mostra operazioni separate per `rds-data` e `secretsmanager` per l'API dati e Secrets Manager. Tuttavia, è possibile combinare azioni e definire le condizioni dei tag in molti modi diversi per supportare i casi d'uso specifici. Per ulteriori informazioni, consulta Utilizzo delle policy basate su identità (policy IAM) per Secrets Manager.

Nell'elemento "Condizione" delle policy, è possibile scegliere le chiavi tag tra le seguenti:

- `aws:TagKeys`
- `aws:ResourceTag/${TagKey}`

Per ulteriori informazioni sui tag delle risorse e su come utilizzarli`aws:TagKeys`, consulta Controllo dell'accesso alle AWS risorse tramite i tag delle risorse.

> **Note**
>
> Sia Data API che Gestione dei segreti AWS autorizzano gli utenti. Se non si dispone delle autorizzazioni per tutte le azioni definite in una policy, viene visualizzato un errore `AccessDeniedException`.

Archiviazione delle credenziali del database in Gestione dei segreti AWS

Quando chiami l'API dati di Amazon RDS, passi le credenziali per il cluster di database Aurora usando un segreto in Secrets Manager. Per utilizzare questo metodo per passare le credenziali, specifica il nome del segreto o l'Amazon Resource Name (ARN) del segreto.

Per archiviare le credenziali del cluster database in un segreto

1. Utilizzare Secrets Manager per creare un segreto contenente le credenziali per il cluster DB Aurora.

   Per le istruzioni, consulta Creazione di un segreto del database nella Guida per l'utente di Gestione dei segreti AWS.
2. Usa la console Secrets Manager per visualizzare i dettagli del segreto che hai creato o esegui il `aws secretsmanager describe-secret` AWS CLI comando.

   Prendere nota del nome e dell'ARN del segreto, poiché possono essere utilizzati nelle chiamate all'API dati.

Per ulteriori informazioni relative all'utilizzo di Secrets Manager, consulta la Guida per l'utente di AWS Secrets Manager.

Per informazioni su come Amazon Aurora gestisce la gestione delle identità e degli accessi, consulta Come funziona Amazon Aurora con IAM.

Per informazioni sulla creazione di una policy IAM, consulta Creazione di policy IAM nella Guida per l'utente di IAM. Per informazioni sull'aggiunta di una policy IAM a un utente, consulta Aggiunta e rimozione di autorizzazioni per identità IAM nella Guida per l'utente di IAM.

Abilitazione dell'API dati di Amazon RDS

Per utilizzare l'API dati di Amazon RDS (API dati), è necessario abilitarla per il cluster di database Aurora. Puoi abilitare l'API dati durante la creazione o la modifica del cluster di database.

> **Note**
>
> La disponibilità dell'API dati per il cluster dipende dalla versione di Aurora, dal motore di database e dalla Regione AWS. Per le versioni meno recenti di Aurora, l'API dati funziona solo con i cluster Aurora Serverless v1. Per le versioni più recenti di Aurora, l'API dati è compatibile con i cluster che utilizzano sia istanze con provisioning sia istanze Aurora Serverless v2. Per verificare se il cluster può utilizzare l'API dati, consulta Regioni e motori di database Aurora supportati per l'API dati RDS.

Argomenti

- Abilitazione dell'API dati di RDS durante la creazione di un database
- Abilitazione dell'API dati RDS su un database esistente

Abilitazione dell'API dati di RDS durante la creazione di un database

Durante la creazione di un database che supporta l'API dati RDS (API dati), puoi abilitare questa funzionalità. Le seguenti procedure descrivono come eseguire questa operazione utilizzando la Console di gestione AWS, la AWS CLI o l'API RDS.

Console

Per abilitare l'API dati durante la creazione di un cluster di database, seleziona la casella di controllo Abilita l'API dati in RDS nella sezione Connettività della pagina Crea database, come mostrato nello screenshot seguente.



Per istruzioni su come creare un cluster di database Aurora in grado di utilizzare l'API dati RDS, consulta:

- Per i cluster Aurora Serverless v2 e con provisioning: Creazione di un cluster database Amazon Aurora
- Per Aurora Serverless v1: Creazione di un cluster di database Aurora Serverless v1

AWS CLI

Per abilitare l'API dati durante la creazione di un cluster di database Aurora, esegui il comando create-db-cluster AWS CLI con l'opzione `--enable-http-endpoint`.

Nell'esempio seguente viene creato un cluster di database Aurora PostgreSQL con l'API dati abilitata.

Per Linux, macOS o Unix:

```
aws rds create-db-cluster \
```

```
    --db-cluster-identifier my_pg_cluster \
    --engine aurora-postgresql \
    --enable-http-endpoint
```

Per Windows:

```
aws rds create-db-cluster ^
    --db-cluster-identifier my_pg_cluster ^
    --engine aurora-postgresql ^
    --enable-http-endpoint
```

API RDS

Per abilitare l'API dati durante la creazione di un cluster di database Aurora, utilizza l'operazione CreateDBCluster e imposta il valore del parametro `EnableHttpEndpoint` su `true`.

Abilitazione dell'API dati RDS su un database esistente

Puoi modificare un cluster di database che supporta l'API dati RDS (API dati) per abilitare o disabilitare questa funzionalità.

Argomenti

- Abilitazione o disabilitazione dell'API dati (Aurora Serverless v2 e con provisioning)
- Abilitazione o disabilitazione dell'API dati (solo Aurora Serverless v1)

Abilitazione o disabilitazione dell'API dati (Aurora Serverless v2 e con provisioning)

Utilizza le seguenti procedure per abilitare o disabilitare l'API dati su database Aurora Serverless v2 e con provisioning. Per abilitare o disabilitare l'API dati su database Aurora Serverless v1, utilizza le procedure descritte in the section called "Abilitazione o disabilitazione dell'API dati (solo Aurora Serverless v1)".

Console

Puoi abilitare o disabilitare l'API dati utilizzando la console RDS per un cluster di database che supporta questa funzionalità. A tale scopo, apri la pagina dei dettagli del cluster del database su cui vuoi agire e, nella scheda Connettività e sicurezza, vai alla sezione API dati RDS. Questa sezione mostra lo stato dell'API dati e consente di abilitarla o disabilitarla.

Lo screenshot seguente mostra che l'opzione API dati RDS non è abilitata.



AWS CLI

Per abilitare o disabilitare l'API dati su un database esistente, esegui il comando enable-http-endpoint o disable-http-endpoint AWS CLI, specificando l'ARN del cluster di database.

L'esempio seguente mostra come abilitare l'API dati.

Per Linux, macOS o Unix:

```
aws rds enable-http-endpoint \
    --resource-arn cluster_arn
```

Per Windows:

```
aws rds enable-http-endpoint ^
    --resource-arn cluster_arn
```

API RDS

Per abilitare o disabilitare l'API dati su un database esistente, utilizza le operazioni EnableHttpEndpoint e DisableHttpEndpoint.

Abilitazione o disabilitazione dell'API dati (solo Aurora Serverless v1)

Utilizza le seguenti procedure per abilitare o disabilitare l'API dati su database Aurora Serverless v1 esistenti. Per abilitare o disabilitare l'API dati su database Aurora Serverless v2 e con provisioning, utilizza le procedure descritte in the section called "Abilitazione o disabilitazione dell'API dati".

Console

Quando modifichi un cluster di database Aurora Serverless v1, puoi abilitare l'API dati nella sezione Connettività della console RDS.

Lo screenshot seguente mostra la funzione API di dati abilitata durante la modifica di un cluster di database Aurora.



Per istruzioni su come modificare un cluster di database Aurora Serverless v1, consulta Modifica di un cluster di database Aurora Serverless v1.

AWS CLI

Per abilitare o disabilitare l'API dati, esegui il comando modify-db-cluster AWS CLI con `--enable-http-endpoint` o `--no-enable-http-endpoint`, a seconda dei casi.

L'esempio seguente mostra come abilitare l'API dati su `sample-cluster`.

Per Linux, macOS o Unix:

```
aws rds modify-db-cluster \
    --db-cluster-identifier sample-cluster \
    --enable-http-endpoint
```

Per Windows:

```
aws rds modify-db-cluster ^
    --db-cluster-identifier sample-cluster ^
    --enable-http-endpoint
```

API RDS

Per abilitare l’API dati, utilizza l’operazione ModifyDBCluster e imposta il valore di `EnableHttpEndpoint` su `true` o `false`, a seconda dei casi.

Creazione di un endpoint Amazon VPC per l’API dati di Amazon RDS (AWS PrivateLink)

Amazon VPC consente di avviare risorse AWS, ad esempio cluster DB Aurora e applicazioni, in un virtual private cloud (VPC). AWS PrivateLink fornisce connettività privata tra VPC e servizi AWS in modo sicuro sulla rete Amazon. Utilizzando AWS PrivateLink, puoi creare endpoint Amazon VPC che consentono di connettersi a servizi su account e VPC diversi basati su Amazon VPC. Per ulteriori informazioni su AWS PrivateLink, consulta Servizi endpoint VPC (AWS PrivateLink) nella Guida per l'utente di Amazon Virtual Private Cloud.

Puoi chiamare l’API dati di RDS (API dati) utilizzando gli endpoint Amazon VPC. L’utilizzo di un endpoint Amazon VPC mantiene il traffico tra le applicazioni nell’Amazon VPC e l’API dati nella rete AWS, senza utilizzare indirizzi IP pubblici. Gli endpoint Amazon VPC consentono di soddisfare i requisiti di conformità e normativi relativi alla limitazione della connettività Internet. Ad esempio, se utilizzi un endpoint Amazon VPC, puoi mantenere il traffico tra un’applicazione in esecuzione su un’istanza Amazon EC2 e l’API dati nei VPC che li contengono.

Dopo aver creato l'endpoint Amazon VPC, puoi iniziare a utilizzarlo senza apportare modifiche al codice o alla configurazione nell'applicazione.

Per creare un endpoint Amazon VPC per l’API dati

1. Accedi alla Console di gestione AWS e apri la console Amazon VPC all'indirizzo https://console.aws.amazon.com/vpc/.
2. Scegliere Endpoint, quindi Create Endpoint (Crea endpoint).
3. Nella pagina Crea endpoint, per Categoria di servizio, seleziona Servizi AWS. Per Service Name (Nome servizio), scegliere rds-data.



4. Per VPC, scegliere il VPC in cui creare l'endpoint.

   Scegliere il VPC che contiene l'applicazione che effettua chiamate API dati.

5. In Sottoreti, scegli la sottorete per ogni zona di disponibilità (AZ) utilizzata dal servizio AWS che esegue l'applicazione.

   Per creare un endpoint Amazon VPC, specificare l'intervallo di indirizzi IP privati in cui l'endpoint sarà accessibile. A tale scopo, scegliere la sottorete per ogni zona di disponibilità. Questo ha l'effetto di limitare l'endpoint VPC all'intervallo di indirizzi IP privati specifico per ciascuna zona di disponibilità e crea inoltre un endpoint Amazon VPC in ogni zona di disponibilità.

6. Per Enable DNS Name (Abilita nome DNS), seleziona Enable for this endpoint (Abilita per questo endpoint).

   Il DNS privato risolve il nome host DNS dell'API dati standard (`https://rds-data.`*`region`*`.amazonaws.com`) negli indirizzi IP privati associati al nome host DNS specifico dell'endpoint Amazon VPC. Di conseguenza, puoi accedere all'endpoint VPC dell'API dati utilizzando gli SDK AWS CLI o AWS senza apportare modifiche al codice o alla configurazione per aggiornare l'URL dell'endpoint dell'API dati.

7. Per Security group (Gruppo di sicurezza), scegli un gruppo di sicurezza da associare all'endpoint Amazon VPC.

   Scegli il gruppo di sicurezza che consente l'accesso al servizio AWS che esegue l'applicazione. Ad esempio, se un'istanza Amazon EC2 esegue l'applicazione, scegli il gruppo di sicurezza che consente l'accesso all'istanza Amazon EC2. Il gruppo di sicurezza consente di controllare il traffico verso l'endpoint Amazon VPC dalle risorse del VPC.

8. Per Policy, scegli Full Access (Accesso completo) per consentire a chiunque all'interno del Amazon VPC di accedere all'API dati tramite questo endpoint. Oppure scegli Custom (Personalizzato) per specificare una policy che limita l'accesso.

   Se scegli Personalizzato, immetti la policy nello strumento di creazione delle policy.

9. Selezionare Create endpoint (Crea endpoint).

Dopo aver creato l'endpoint, scegli il collegamento nella Console di gestione AWS per visualizzare i dettagli dell'endpoint.



La scheda Details (Dettagli) dell'endpoint mostra i nomi host DNS generati durante la creazione dell'endpoint Amazon VPC.



È possibile utilizzare l'endpoint standard (`rds-data.`*`region`*`.amazonaws.com`) o uno degli endpoint specifici di VPC per chiamare l'API dati all'interno di Amazon VPC. L'endpoint API dati standard esegue automaticamente l'instradamento all'endpoint Amazon VPC. Questo routing si verifica perché il nome host DNS privato è stato abilitato al momento della creazione dell'endpoint Amazon VPC.

Quando utilizzi un endpoint Amazon VPC in una chiamata all'API dati, tutto il traffico tra l'applicazione e l'API dati rimane negli Amazon VPC che li contengono. Puoi utilizzare un endpoint Amazon VPC per qualsiasi tipo di chiamata API dati. Per informazioni sulla chiamata dell'API dati, consulta Chiamata dell'API dati di Amazon RDS.

Chiamata dell'API dati di Amazon RDS

Con l'API dati di Amazon RDS (API dati) abilitata sul cluster di database Aurora, puoi eseguire istruzioni SQL sul cluster di database Aurora utilizzando l'API dati o la AWS CLI. L'API dati supporta i linguaggi di programmazione supportati dagli SDK AWS. Per ulteriori informazioni, consulta Strumenti per creare in AWS.

Argomenti

- Riferimento alle operazioni dell'API dati di Amazon RDS
- Chiamata dell'API dati di Amazon RDS con la AWS CLI
- Chiamata dell'API dati di Amazon RDS da un'applicazione Python
- Chiamata dell'API dati di Amazon RDS da un'applicazione Java

- Controllo del comportamento di timeout dell'API dati

Riferimento alle operazioni dell'API dati di Amazon RDS

L'API dati di Amazon RDS fornisce le seguenti operazioni per eseguire istruzioni SQL.

Operazione API dati	AWS CLI command	Descrizione
ExecuteStatement	aws rds-data execute-statement	Esegue un'istruzione SQL in un database.
BatchExecuteStatement	aws rds-data batch-execute-statement	Esegue un'istruzione SQL batch su un array di dati per operazioni di aggiornamento in blocco e di inserimento. Puoi eseguire un'istruzione DML (Data Manipolation Language) con una matrice di set di parametri. Un'istruzione SQL batch può fornire un miglioramento significativo delle prestazioni su singole operazioni di inserimento e aggiornamento.

Puoi utilizzare entrambe le operazioni per eseguire singole istruzioni SQL o per eseguire transazioni. L'API dati fornisce le seguenti operazioni per le transazioni.

Operazione API dati	AWS CLI command	Descrizione
BeginTransaction	aws rds-data begin-transaction	Inizia una transazione SQL.
CommitTransaction	aws rds-data commit-transaction	Termina una transazione SQL ed esegue il commit delle modifiche.
RollbackTransaction	aws rds-data rollback-transaction	Esegue un rollback di una transazione.

Le operazioni per l'esecuzione di istruzioni SQL e il supporto delle transazioni dispongono dei seguenti parametri API dati e opzioni AWS CLI comuni. Alcune operazioni supportano altri parametri o opzioni.

Parametro operazione API dati	AWS CLIOpzione comando	Campo obbligatorio	Descrizione
`resourceArn`	`--resource-arn`	Sì	Il nome della risorsa Amazon (ARN) del cluster di database Aurora. Il cluster deve trovarsi nello stesso Account AWS dell'utente IAM o del ruolo IAM che invoca l'API dati. Per accedere a un cluster in un account diverso, assumi un ruolo in tale account.
`secretArn`	`--secret-arn`	Sì	Il nome o l'ARN del secreto che abilita l'accesso al cluster database.

L'API dati di RDS supporta i seguenti tipi di dati per Aurora MySQL:

- `TINYINT(1)`, `BOOLEAN`, `BOOL`
- `TINYINT`
- `SMALLINT` [`SIGNED` | `UNSIGNED`]
- `MEDIUMINT` [`SIGNED` | `UNSIGNED`]
- `INT` [`SIGNED` | `UNSIGNED`]
- `BIGINT` [`SIGNED` | `UNSIGNED`]
- `FLOAT`
- `DOUBLE`
- `VARCHAR`, `CHAR`, `TEXT`, `ENUM`
- `VARBINARY`, `BINARY`, `BLOB`
- `DATE`, `TIME`, `DATETIME`, `TIMESTAMP`
- `DECIMAL`
- `JSON`

- `BIT`, `BIT(N)`

L'API dati di RDS supporta i seguenti tipi scalari di Aurora PostgreSQL:

- `BOOL`
- `BYTEA`
- `DATE`
- `CIDR`
- `DECIMAL`, `NUMERIC`
- `ENUM`
- `FLOAT8`, `DOUBLE PRECISION`
- `INET`
- `INT`, `INT4`, `SERIAL`
- `INT2`, `SMALLINT`, `SMALLSERIAL`
- `INT8`, `BIGINT`, `BIGSERIAL`
- `JSONB`, `JSON`
- `REAL`, `FLOAT`
- `TEXT`, `CHAR(N)`, `VARCHAR`, `NAME`
- `TIME`
- `TIMESTAMP`
- `UUID`
- `VECTOR`

L'API dati di RDS supporta i seguenti tipi di array di Aurora PostgreSQL:

- `BOOL[]`, `BIT[]`
- `DATE[]`
- `DECIMAL[]`, `NUMERIC[]`
- `FLOAT8[]`, `DOUBLE PRECISION[]`
- `INT[]`, `INT4[]`
- `INT2[]`
- `INT8[]`, `BIGINT[]`

- JSON[]
- REAL[], FLOAT[]
- TEXT[], CHAR(N)[], VARCHAR[], NAME[]
- TIME[]
- TIMESTAMP[]
- UUID[]

Puoi utilizzare parametri nelle chiamate API dati per ExecuteStatement e BatchExecuteStatement e quando esegui i comandi AWS CLI execute-statement e batch-execute-statement. Per utilizzare un parametro, specifica una coppia nome-valore nel tipo di dati SqlParameter. Specifica il valore con il tipo di dati Field. La tabella seguente associa i tipi di dati Java Database Connectivity (JDBC) ai tipi di dati specificati nelle chiamate API dati.

Tipo di dati JDBC	Tipo di dati API dati
INTEGER, TINYINT, SMALLINT, BIGINT	LONG (o STRING)
FLOAT, REAL, DOUBLE	DOUBLE
DECIMAL	STRING
BOOLEAN, BIT	BOOLEAN
BLOB, BINARY, LONGVARBINARY, VARBINARY	BLOB
CLOB	STRING
Altri tipi (inclusi i tipi correlati a data e ora)	STRING

> **Note**
>
> Puoi specificare il tipo di dati LONG o STRING nella chiamata all'API data per i valori LONG restituiti dal database. Si consiglia di farlo per evitare di perdere precisione per numeri estremamente grandi, cosa che può accadere quando si lavora con JavaScript.

Alcuni tipi, ad esempio `DECIMAL` e `TIME`, richiedono un suggerimento in modo che l'API dati passi i valori `String` al database come tipo corretto. Per utilizzare un suggerimento, includere i valori per `typeHint` nel tipo di dati `SqlParameter`. i seguito sono riportati i valori possibili per `typeHint`:

- `DATE` – Il valore del parametro `String` corrispondente viene inviato come oggetto di tipo `DATE` al database. Il formato accettato è `YYYY-MM-DD`.
- `DECIMAL` – Il valore del parametro `String` corrispondente viene inviato come oggetto di tipo `DECIMAL` al database.
- `JSON` – Il valore del parametro `String` corrispondente viene inviato come oggetto di tipo `JSON` al database.
- `TIME` – Il valore del parametro `String` corrispondente viene inviato come oggetto di tipo `TIME` al database. Il formato accettato è `HH:MM:SS[.FFF]`.
- `TIMESTAMP` – Il valore del parametro `String` corrispondente viene inviato come oggetto di tipo `TIMESTAMP` al database. Il formato accettato è `YYYY-MM-DD HH:MM:SS[.FFF]`.
- `UUID` – Il valore del parametro `String` corrispondente viene inviato come oggetto di tipo `UUID` al database.

> **Note**
>
> Attualmente, l'API dati non supporta array di identificatori univoci universali (UUID).

> **Note**
>
> Per Amazon Aurora PostgreSQL, l'API dati restituisce sempre il tipo di dati Aurora PostgreSQL `TIMESTAMPTZ` nel fuso orario UTC.

Chiamata dell'API dati di Amazon RDS con la AWS CLI

Puoi chiamare l'API dati di Amazon RDS (API dati) utilizzando la AWS CLI.

Gli esempi seguenti utilizzano la AWS CLI per l'API dati. Per ulteriori informazioni, consulta la Documentazione di riferimento AWS CLI per l'API dati.

In ogni esempio, sostituisci il nome della risorsa Amazon (ARN) del cluster di database con l'ARN per il cluster di database Aurora. Inoltre, sostituisci l'ARN segreto con l'ARN del segreto in Secrets Manager che consente l'accesso al cluster database.

> **Note**
>
> AWS CLI può formattare le risposte in JSON.

Argomenti

- Avvio di una transazione SQL
- Esecuzione di un'istruzione SQL
- Esecuzione di un'istruzione SQL batch su un array di dati
- Esecuzione del commit di una transazione SQL
- Rollback di una transazione SQL

Avvio di una transazione SQL

Puoi avviare una transazione SQL utilizzando il comando CLI `aws rds-data begin-transaction`. La chiamata restituisce un identificatore di transazione.

> **Important**
>
> All'interno dell'API dati, una transazione scade se non ci sono chiamate che utilizzano il suo ID transazione in un periodo di tre minuti. Se una transazione scade prima del commit, viene automaticamente sottoposta a rollback dall'API dati.
> Le istruzioni DDL (Data Definition Language) MySQL all'interno di una transazione causano un commit implicito. Consigliamo di eseguire ogni istruzioni DDL MySQL in un comando `execute-statement` separato con l'opzione `--continue-after-timeout`.

Oltre alle opzioni comuni, specifica l'opzione `--database`, che fornisce il nome del database.

Ad esempio, il comando CLI seguente avvia una transazione SQL.

Per Linux, macOS o Unix:

```
aws rds-data begin-transaction --resource-arn "arn:aws:rds:us-
east-1:123456789012:cluster:mydbcluster" \
--database "mydb" --secret-arn "arn:aws:secretsmanager:us-
east-1:123456789012:secret:mysecret"
```

Per Windows:

```
aws rds-data begin-transaction --resource-arn "arn:aws:rds:us-
east-1:123456789012:cluster:mydbcluster" ^
--database "mydb" --secret-arn "arn:aws:secretsmanager:us-
east-1:123456789012:secret:mysecret"
```

Di seguito è riportato un esempio della risposta.

```
{
    "transactionId": "ABC1234567890xyz"
}
```

Esecuzione di un'istruzione SQL

Puoi eseguire un'istruzione SQL utilizzando il comando CLI `aws rds-data execute-statement`.

Puoi eseguire un'istruzione SQL in una transazione specificando l'identificatore di transazione con l'opzione `--transaction-id`. Puoi avviare una transazione utilizzando il comando CLI `aws rds-data begin-transaction`. Puoi terminare ed eseguire il commit di una transazione utilizzando il comando CLI `aws rds-data commit-transaction`.

> **Important**
>
> Se non specifichi l'opzione `--transaction-id`, viene eseguito automaticamente il commit delle modifiche risultanti dalla chiamata.

Oltre alle opzioni comuni, specifica le seguenti opzioni:

- `--sql` (obbligatoria) – Un'istruzione SQL da eseguire sul cluster database.
- `--transaction-id` (facoltativa) – L'identificatore di una transazione che è stata avviata utilizzando il comando CLI `begin-transaction`. Specifica l'ID transazione della transazione in cui desideri includere l'istruzione SQL.

- `--parameters` (facoltativa) – I parametri per l'istruzione SQL.
- `--include-result-metadata | --no-include-result-metadata` (opzionale) – Un valore che indica se includere metadati nei risultati. Il valore di default è `--no-include-result-metadata`.
- `--database` (opzionale) – Il nome del database.

  L'opzione `--database` potrebbe non funzionare quando si esegue un'istruzione SQL dopo l'esecuzione di `--sql "use database_name;"` nella richiesta precedente. Si consiglia di utilizzare l'opzione `--database` invece di eseguire le istruzioni `--sql "use database_name;"`.
- `--continue-after-timeout | --no-continue-after-timeout` (facoltativo): un valore che indica se continuare l'esecuzione dell'istruzione dopo che la chiamata supera l'intervallo di timeout di 45 secondi dell'API dati. Il valore di default è `--no-continue-after-timeout`.

  Per istruzioni DDL (Data Definition Language), è consigliabile continuare a eseguire l'istruzione dopo il timeout della chiamata per evitare errori e la possibilità di strutture dati danneggiate.
- `--format-records-as "JSON" | "NONE"` - Un valore facoltativo che specifica se formattare il set di risultati come stringa JSON. Il valore di default è `"NONE"`. Per informazioni sull'utilizzo dell'elaborazione di set di risultati JSON, consulta Elaborazione dei risultati delle query dell'API dati di Amazon RDS in formato JSON.

Il cluster database restituisce una risposta per la chiamata.

> **Note**
>
> Il limite di dimensioni della risposta è 1 MiB. Se la chiamata restituisce più di 1 MiB di dati di risposta, verrà terminata.
> Per Aurora Serverless v1, il numero massimo di richieste al secondo è 1.000. Per tutti gli altri database supportati, non esiste alcun limite.

Ad esempio, il comando CLI seguente esegue una singola istruzione SQL e omette i metadati nei risultati (il valore predefinito).

Per Linux, macOS o Unix:

```
aws rds-data execute-statement --resource-arn "arn:aws:rds:us-
east-1:123456789012:cluster:mydbcluster" \
--database "mydb" --secret-arn "arn:aws:secretsmanager:us-
east-1:123456789012:secret:mysecret" \
--sql "select * from mytable"
```

Per Windows:

```
aws rds-data execute-statement --resource-arn "arn:aws:rds:us-
east-1:123456789012:cluster:mydbcluster" ^
--database "mydb" --secret-arn "arn:aws:secretsmanager:us-
east-1:123456789012:secret:mysecret" ^
--sql "select * from mytable"
```

Di seguito è riportato un esempio della risposta.

```
{
    "numberOfRecordsUpdated": 0,
    "records": [
        [
            {
                "longValue": 1
            },
            {
                "stringValue": "ValueOne"
            }
        ],
        [
            {
                "longValue": 2
            },
            {
                "stringValue": "ValueTwo"
            }
        ],
        [
            {
                "longValue": 3
            },
            {
                "stringValue": "ValueThree"
            }
```

```
        ]
    ]
}
```

Il comando CLI seguente esegue una singola istruzione SQL in una transazione specificando l'opzione `--transaction-id`.

Per Linux, macOS o Unix:

```
aws rds-data execute-statement --resource-arn "arn:aws:rds:us-
east-1:123456789012:cluster:mydbcluster" \
--database "mydb" --secret-arn "arn:aws:secretsmanager:us-
east-1:123456789012:secret:mysecret" \
--sql "update mytable set quantity=5 where id=201" --transaction-id "ABC1234567890xyz"
```

Per Windows:

```
aws rds-data execute-statement --resource-arn "arn:aws:rds:us-
east-1:123456789012:cluster:mydbcluster" ^
--database "mydb" --secret-arn "arn:aws:secretsmanager:us-
east-1:123456789012:secret:mysecret" ^
--sql "update mytable set quantity=5 where id=201" --transaction-id "ABC1234567890xyz"
```

Di seguito è riportato un esempio della risposta.

```
{
    "numberOfRecordsUpdated": 1
}
```

Il comando CLI seguente esegue una singola istruzione SQL con parametri.

Per Linux, macOS o Unix:

```
aws rds-data execute-statement --resource-arn "arn:aws:rds:us-
east-1:123456789012:cluster:mydbcluster" \
--database "mydb" --secret-arn "arn:aws:secretsmanager:us-
east-1:123456789012:secret:mysecret" \
--sql "insert into mytable values (:id, :val)" --parameters "[{\"name\": \"id\",
 \"value\": {\"longValue\": 1}},{\"name\": \"val\", \"value\": {\"stringValue\":
 \"value1\"}}]"
```

Per Windows:

```
aws rds-data execute-statement --resource-arn "arn:aws:rds:us-
east-1:123456789012:cluster:mydbcluster" ^
--database "mydb" --secret-arn "arn:aws:secretsmanager:us-
east-1:123456789012:secret:mysecret" ^
--sql "insert into mytable values (:id, :val)" --parameters "[{\"name\": \"id\",
 \"value\": {\"longValue\": 1}},{\"name\": \"val\", \"value\": {\"stringValue\":
 \"value1\"}}]"
```

Di seguito è riportato un esempio della risposta.

```
{
    "numberOfRecordsUpdated": 1
}
```

Il comando CLI seguente esegue un'istruzione SQL DDL (Data Definition Language). L'istruzione DDL rinomina la colonna `job` nella colonna `role`.

> **Important**
>
> Per istruzioni DDL, è consigliabile continuare a eseguire l'istruzione dopo il timeout della chiamata. Quando un'istruzione DDL termina prima che l'esecuzione sia terminata, può causare errori e verosimilmente strutture dati danneggiate. Per continuare l'esecuzione di un'istruzione dopo che una chiamata supera il timeout di 45 secondi dell'API dati di RDS, specifica l'opzione `--continue-after-timeout`.

Per Linux, macOS o Unix:

```
aws rds-data execute-statement --resource-arn "arn:aws:rds:us-
east-1:123456789012:cluster:mydbcluster" \
--database "mydb" --secret-arn "arn:aws:secretsmanager:us-
east-1:123456789012:secret:mysecret" \
--sql "alter table mytable change column job role varchar(100)" --continue-after-
timeout
```

Per Windows:

```
aws rds-data execute-statement --resource-arn "arn:aws:rds:us-
east-1:123456789012:cluster:mydbcluster" ^
```

```
--database "mydb" --secret-arn "arn:aws:secretsmanager:us-
east-1:123456789012:secret:mysecret" ^
--sql "alter table mytable change column job role varchar(100)" --continue-after-
timeout
```

Di seguito è riportato un esempio della risposta.

```
{
    "generatedFields": [],
    "numberOfRecordsUpdated": 0
}
```

> **Note**
>
> I dati `generatedFields` non sono supportati da Aurora PostgreSQL. Per ottenere i valori dei campi generati, utilizza la clausola `RETURNING`. Per ulteriori informazioni, consulta Returning Data From Modified Rows nella documentazione PostgreSQL.

Esecuzione di un'istruzione SQL batch su un array di dati

Puoi eseguire un'istruzione SQL batch su un'array di dati utilizzando il comando CLI `aws rds-data batch-execute-statement`. Puoi utilizzare questo comando per eseguire un'operazione di importazione in blocco o di aggiornamento.

Puoi eseguire un'istruzione SQL in una transazione specificando l'identificatore di transazione con l'opzione `--transaction-id`. Puoi avviare una transazione utilizzando il comando CLI `aws rds-data begin-transaction`. Puoi terminare ed eseguire il commit di una transazione utilizzando il comando CLI `aws rds-data commit-transaction`.

> **Important**
>
> Se non specifichi l'opzione `--transaction-id`, viene eseguito automaticamente il commit delle modifiche risultanti dalla chiamata.

Oltre alle opzioni comuni, specifica le seguenti opzioni:

- `--sql` (obbligatoria) – Un'istruzione SQL da eseguire sul cluster database.

> **Tip**
>
> Perché le istruzioni siano compatibili con MySQL, non includere un punto e virgola alla fine del parametro `--sql`. Un punto e virgola finale potrebbe causare un errore di sintassi.

- `--transaction-id` (facoltativa) – L'identificatore di una transazione che è stata avviata utilizzando il comando CLI `begin-transaction`. Specifica l'ID transazione della transazione in cui desideri includere l'istruzione SQL.
- `--parameter-set` (facoltativa) – Il set di parametri per l'operazione batch.
- `--database` (opzionale) – Il nome del database.

Il cluster database restituisce una risposta alla chiamata.

> **Note**
>
> Non è previsto alcun limite massimo al numero di set di parametri. Tuttavia, la dimensione massima della richiesta HTTP inviata tramite l'API dati è 4 MiB. Se la richiesta supera questo limite, l'API dati restituisce un errore e non elabora la richiesta. Questo limite di 4 MiB include la dimensione delle intestazioni HTTP e la notazione JSON nella richiesta. Pertanto, il numero di set di parametri che è possibile includere dipende da una combinazione di fattori, ad esempio la dimensione dell'istruzione SQL e la dimensione di ogni set di parametri.
> Il limite di dimensioni della risposta è 1 MiB. Se la chiamata restituisce più di 1 MiB di dati di risposta, verrà terminata.
> Per Aurora Serverless v1, il numero massimo di richieste al secondo è 1.000. Per tutti gli altri database supportati, non esiste alcun limite.

Ad esempio, il seguente comando CLI esegue un'istruzione SQL batch su un'array di dati con un set di parametri.

Per Linux, macOS o Unix:

```
aws rds-data batch-execute-statement --resource-arn "arn:aws:rds:us-
east-1:123456789012:cluster:mydbcluster" \
--database "mydb" --secret-arn "arn:aws:secretsmanager:us-
east-1:123456789012:secret:mysecret" \
--sql "insert into mytable values (:id, :val)" \
```

```
--parameter-sets "[[{\"name\": \"id\", \"value\": {\"longValue\": 1}},{\"name\":
 \"val\", \"value\": {\"stringValue\": \"ValueOne\"}}],
[{\"name\": \"id\", \"value\": {\"longValue\": 2}},{\"name\": \"val\", \"value\":
 {\"stringValue\": \"ValueTwo\"}}],
[{\"name\": \"id\", \"value\": {\"longValue\": 3}},{\"name\": \"val\", \"value\":
 {\"stringValue\": \"ValueThree\"}}]]"
```

Per Windows:

```
aws rds-data batch-execute-statement --resource-arn "arn:aws:rds:us-
east-1:123456789012:cluster:mydbcluster" ^
--database "mydb" --secret-arn "arn:aws:secretsmanager:us-
east-1:123456789012:secret:mysecret" ^
--sql "insert into mytable values (:id, :val)" ^
--parameter-sets "[[{\"name\": \"id\", \"value\": {\"longValue\": 1}},{\"name\":
 \"val\", \"value\": {\"stringValue\": \"ValueOne\"}}],
[{\"name\": \"id\", \"value\": {\"longValue\": 2}},{\"name\": \"val\", \"value\":
 {\"stringValue\": \"ValueTwo\"}}],
[{\"name\": \"id\", \"value\": {\"longValue\": 3}},{\"name\": \"val\", \"value\":
 {\"stringValue\": \"ValueThree\"}}]]"
```

> **Note**
>
> Non includere interruzioni di riga nell'opzione `--parameter-sets`.

Esecuzione del commit di una transazione SQL

Utilizzando il comando CLI `aws rds-data commit-transaction`, puoi terminare una transazione SQL avviata con `aws rds-data begin-transaction` ed eseguire il commit delle modifiche.

Oltre alle opzioni di comando, specifica l'opzione seguente:

- `--transaction-id` (obbligatorio) – L'identificatore di una transazione che è stata avviata utilizzando il comando CLI `begin-transaction`. Specifica l'ID transazione della transazione che desideri terminare e di cui eseguire il commit.

Ad esempio, il comando CLI seguente termina una transazione SQL ed esegue il commit delle modifiche.

Per Linux, macOS o Unix:

```
aws rds-data commit-transaction --resource-arn "arn:aws:rds:us-
east-1:123456789012:cluster:mydbcluster" \
--secret-arn "arn:aws:secretsmanager:us-east-1:123456789012:secret:mysecret" \
--transaction-id "ABC1234567890xyz"
```

Per Windows:

```
aws rds-data commit-transaction --resource-arn "arn:aws:rds:us-
east-1:123456789012:cluster:mydbcluster" ^
--secret-arn "arn:aws:secretsmanager:us-east-1:123456789012:secret:mysecret" ^
--transaction-id "ABC1234567890xyz"
```

Di seguito è riportato un esempio della risposta.

```
{
    "transactionStatus": "Transaction Committed"
}
```

Rollback di una transazione SQL

Utilizzando il comando CLI `aws rds-data rollback-transaction`, puoi eseguire il rollback di una transazione SQL avviata con `aws rds-data begin-transaction`. Il rollback di una transazione annulla le relative modifiche.

> **Important**
>
> Se l'ID transazione è scaduto, il rollback della transazione è stato eseguito automaticamente. In questo caso, un comando `aws rds-data rollback-transaction` che specifica l'ID transazione scaduto restituisce un errore.

Oltre alle opzioni di comando, specifica l'opzione seguente:

- `--transaction-id` (obbligatorio) – L'identificatore di una transazione che è stata avviata utilizzando il comando CLI `begin-transaction`. Specifica l'ID transazione della transazione di cui si desidera eseguire il rollback.

Ad esempio, il comando AWS CLI seguente esegue il rollback di una transazione SQL.

Per Linux, macOS o Unix:

```
aws rds-data rollback-transaction --resource-arn "arn:aws:rds:us-
east-1:123456789012:cluster:mydbcluster" \
--secret-arn "arn:aws:secretsmanager:us-east-1:123456789012:secret:mysecret" \
--transaction-id "ABC1234567890xyz"
```

Per Windows:

```
aws rds-data rollback-transaction --resource-arn "arn:aws:rds:us-
east-1:123456789012:cluster:mydbcluster" ^
--secret-arn "arn:aws:secretsmanager:us-east-1:123456789012:secret:mysecret" ^
--transaction-id "ABC1234567890xyz"
```

Di seguito è riportato un esempio della risposta.

```
{
    "transactionStatus": "Rollback Complete"
    }
```

Chiamata dell’API dati di Amazon RDS da un’applicazione Python

Puoi chiamare l’API dati di Amazon RDS (API dati) da un’applicazione Python.

Gli esempi seguenti utilizzano il kit SDK AWS for Python (Boto). Per ulteriori informazioni su Boto, consulta la documentazione di SDK AWS for Python (Boto 3).

In ogni esempio, sostituisci il nome della risorsa Amazon (ARN) del cluster di database con l’ARN del cluster di database Aurora. Inoltre, sostituisci l'ARN segreto con l'ARN del segreto in Secrets Manager che consente l'accesso al cluster database.

Argomenti

- Esecuzione di una query SQL
- Esecuzione di un'istruzione SQL DML
- Esecuzione di una transazione SQL

Esecuzione di una query SQL

Puoi eseguire un'istruzione SELECT e recuperare i risultati da un'applicazione Python.

L'esempio seguente esegue una query SQL.

```
import boto3

rdsData = boto3.client('rds-data')

cluster_arn = 'arn:aws:rds:us-east-1:123456789012:cluster:mydbcluster'
secret_arn = 'arn:aws:secretsmanager:us-east-1:123456789012:secret:mysecret'

response1 = rdsData.execute_statement(
            resourceArn = cluster_arn,
            secretArn = secret_arn,
            database = 'mydb',
            sql = 'select * from employees limit 3')

print (response1['records'])
[
    [
        {
            'longValue': 1
        },
        {
            'stringValue': 'ROSALEZ'
        },
        {
            'stringValue': 'ALEJANDRO'
        },
        {
            'stringValue': '2016-02-15 04:34:33.0'
        }
    ],
    [
        {
            'longValue': 1
        },
        {
            'stringValue': 'DOE'
        },
        {
            'stringValue': 'JANE'
```

```
        },
        {
            'stringValue': '2014-05-09 04:34:33.0'
        }
    ],
    [
        {
            'longValue': 1
        },
        {
            'stringValue': 'STILES'
        },
        {
            'stringValue': 'JOHN'
        },
        {
            'stringValue': '2017-09-20 04:34:33.0'
        }
    ]
]
```

Esecuzione di un'istruzione SQL DML

Puoi eseguire un'istruzione DML (Data Manipulation Language) per inserire, aggiornare o eliminare dati nel database. Puoi anche utilizzare parametri in istruzioni DML.

> **Important**
>
> Se una chiamata non fa parte di una transazione perché non include il parametro `transactionID`, viene eseguito automaticamente il commit delle modifiche risultanti dalla chiamata.

L'esempio seguente esegue un'istruzione SQL insert e utilizza i parametri.

```
import boto3

cluster_arn = 'arn:aws:rds:us-east-1:123456789012:cluster:mydbcluster'
secret_arn = 'arn:aws:secretsmanager:us-east-1:123456789012:secret:mysecret'

rdsData = boto3.client('rds-data')
```

```
param1 = {'name':'firstname', 'value':{'stringValue': 'JACKSON'}}
param2 = {'name':'lastname', 'value':{'stringValue': 'MATEO'}}
paramSet = [param1, param2]

response2 = rdsData.execute_statement(resourceArn=cluster_arn,
                                      secretArn=secret_arn,
                                      database='mydb',
                                      sql='insert into employees(first_name, last_name)
 VALUES(:firstname, :lastname)',
                                      parameters = paramSet)

print (response2["numberOfRecordsUpdated"])
```

Esecuzione di una transazione SQL

Puoi avviare una transazione SQL, eseguire una o più istruzioni SQL, quindi eseguire il commit delle modifiche con un'applicazione Python.

> **Important**
>
> Una transazione scade se non ci sono chiamate che utilizzano il suo ID transazione in un periodo di tre minuti. Se una transazione scade prima che venga eseguito il commit della stessa, viene automaticamente sottoposta a rollback.
> Se non specifichi un ID transazione, viene eseguito automaticamente il commit delle modifiche risultanti dalla chiamata.

L'esempio seguente esegue una transazione SQL che inserisce una riga in una tabella.

```
import boto3

rdsData = boto3.client('rds-data')

cluster_arn = 'arn:aws:rds:us-east-1:123456789012:cluster:mydbcluster'
secret_arn = 'arn:aws:secretsmanager:us-east-1:123456789012:secret:mysecret'

tr = rdsData.begin_transaction(
     resourceArn = cluster_arn,
     secretArn = secret_arn,
     database = 'mydb')
```

```
response3 = rdsData.execute_statement(
     resourceArn = cluster_arn,
     secretArn = secret_arn,
     database = 'mydb',
     sql = 'insert into employees(first_name, last_name) values('XIULAN', 'WANG')',
     transactionId = tr['transactionId'])

cr = rdsData.commit_transaction(
     resourceArn = cluster_arn,
     secretArn = secret_arn,
     transactionId = tr['transactionId'])

cr['transactionStatus']
'Transaction Committed'

response3['numberOfRecordsUpdated']
1
```

> **Note**
>
> Se esegui un'istruzione DDL (Data Definition Language), è consigliabile continuare a eseguire l'istruzione dopo il timeout della chiamata. Quando un'istruzione DDL termina prima che l'esecuzione sia terminata, può causare errori e verosimilmente strutture dati danneggiate. Per continuare l'esecuzione di un'istruzione dopo che una chiamata supera l'intervallo di timeout di 45 secondi dell'API dati di RDS, imposta il parametro `continueAfterTimeout` su `true`.

Chiamata dell'API dati di Amazon RDS da un'applicazione Java

Puoi chiamare l'API dati di Amazon RDS (API dati) da un'applicazione Java.

Gli esempi seguenti utilizzano il kit SDK AWS per Java. Per ulteriori informazioni, consulta la Guida per sviluppatori di AWS SDK per Java.

In ogni esempio, sostituisci il nome della risorsa Amazon (ARN) del cluster di database con l'ARN del cluster di database Aurora. Inoltre, sostituisci l'ARN segreto con l'ARN del segreto in Secrets Manager che consente l'accesso al cluster database.

Argomenti

- Esecuzione di una query SQL

- Esecuzione di una transazione SQL
- Esecuzione di un'operazione SQL batch

Esecuzione di una query SQL

Puoi eseguire un'istruzione SELECT e recuperare i risultati con un'applicazione Java.

L'esempio seguente esegue una query SQL.

```
package com.amazonaws.rdsdata.examples;

import com.amazonaws.services.rdsdata.AWSRDSData;
import com.amazonaws.services.rdsdata.AWSRDSDataClient;
import com.amazonaws.services.rdsdata.model.ExecuteStatementRequest;
import com.amazonaws.services.rdsdata.model.ExecuteStatementResult;
import com.amazonaws.services.rdsdata.model.Field;

import java.util.List;

public class FetchResultsExample {
  public static final String RESOURCE_ARN = "arn:aws:rds:us-
east-1:123456789012:cluster:mydbcluster";
  public static final String SECRET_ARN = "arn:aws:secretsmanager:us-
east-1:123456789012:secret:mysecret";

  public static void main(String[] args) {
    AWSRDSData rdsData = AWSRDSDataClient.builder().build();

    ExecuteStatementRequest request = new ExecuteStatementRequest()
            .withResourceArn(RESOURCE_ARN)
            .withSecretArn(SECRET_ARN)
            .withDatabase("mydb")
            .withSql("select * from mytable");

    ExecuteStatementResult result = rdsData.executeStatement(request);

    for (List<Field> fields: result.getRecords()) {
      String stringValue = fields.get(0).getStringValue();
      long numberValue = fields.get(1).getLongValue();

      System.out.println(String.format("Fetched row: string = %s, number = %d",
 stringValue, numberValue));
```

```
    }
  }
}
```

Esecuzione di una transazione SQL

Puoi avviare una transazione SQL, eseguire una o più istruzioni SQL, quindi eseguire il commit delle modifiche con un'applicazione Java.

> **Important**
>
> Una transazione scade se non ci sono chiamate che utilizzano il suo ID transazione in un periodo di tre minuti. Se una transazione scade prima che venga eseguito il commit della stessa, viene automaticamente sottoposta a rollback.
> Se non specifichi un ID transazione, viene eseguito automaticamente il commit delle modifiche risultanti dalla chiamata.

L'esempio seguente esegue una transazione SQL.

```
package com.amazonaws.rdsdata.examples;

import com.amazonaws.services.rdsdata.AWSRDSData;
import com.amazonaws.services.rdsdata.AWSRDSDataClient;
import com.amazonaws.services.rdsdata.model.BeginTransactionRequest;
import com.amazonaws.services.rdsdata.model.BeginTransactionResult;
import com.amazonaws.services.rdsdata.model.CommitTransactionRequest;
import com.amazonaws.services.rdsdata.model.ExecuteStatementRequest;

public class TransactionExample {
  public static final String RESOURCE_ARN = "arn:aws:rds:us-
east-1:123456789012:cluster:mydbcluster";
  public static final String SECRET_ARN = "arn:aws:secretsmanager:us-
east-1:123456789012:secret:mysecret";

  public static void main(String[] args) {
    AWSRDSData rdsData = AWSRDSDataClient.builder().build();

    BeginTransactionRequest beginTransactionRequest = new BeginTransactionRequest()
            .withResourceArn(RESOURCE_ARN)
            .withSecretArn(SECRET_ARN)
            .withDatabase("mydb");
```

```
    BeginTransactionResult beginTransactionResult =
 rdsData.beginTransaction(beginTransactionRequest);
    String transactionId = beginTransactionResult.getTransactionId();

    ExecuteStatementRequest executeStatementRequest = new ExecuteStatementRequest()
            .withTransactionId(transactionId)
            .withResourceArn(RESOURCE_ARN)
            .withSecretArn(SECRET_ARN)
            .withSql("INSERT INTO test_table VALUES ('hello world!')");
    rdsData.executeStatement(executeStatementRequest);

    CommitTransactionRequest commitTransactionRequest = new CommitTransactionRequest()
            .withTransactionId(transactionId)
            .withResourceArn(RESOURCE_ARN)
            .withSecretArn(SECRET_ARN);
    rdsData.commitTransaction(commitTransactionRequest);
  }
}
```

> **Note**
>
> Se esegui un'istruzione DDL (Data Definition Language), è consigliabile continuare a eseguire l'istruzione dopo il timeout della chiamata. Quando un'istruzione DDL termina prima che l'esecuzione sia terminata, può causare errori e verosimilmente strutture dati danneggiate. Per continuare l'esecuzione di un'istruzione dopo che una chiamata supera l'intervallo di timeout di 45 secondi dell'API dati di RDS, imposta il parametro `continueAfterTimeout` su `true`.

Esecuzione di un'operazione SQL batch

Puoi eseguire operazioni di inserimento a blocchi e di aggiornamento su un'array di dati con un'applicazione Java. Puoi eseguire un'istruzione DML con un array di set di parametri.

> **Important**
>
> Se non specifichi un ID transazione, viene eseguito automaticamente il commit delle modifiche risultanti dalla chiamata.

L'esempio seguente illustra un'operazione di inserimento in batch.

```
package com.amazonaws.rdsdata.examples;

import com.amazonaws.services.rdsdata.AWSRDSData;
import com.amazonaws.services.rdsdata.AWSRDSDataClient;
import com.amazonaws.services.rdsdata.model.BatchExecuteStatementRequest;
import com.amazonaws.services.rdsdata.model.Field;
import com.amazonaws.services.rdsdata.model.SqlParameter;

import java.util.Arrays;

public class BatchExecuteExample {
  public static final String RESOURCE_ARN = "arn:aws:rds:us-
east-1:123456789012:cluster:mydbcluster";
  public static final String SECRET_ARN = "arn:aws:secretsmanager:us-
east-1:123456789012:secret:mysecret";

  public static void main(String[] args) {
      AWSRDSData rdsData = AWSRDSDataClient.builder().build();

    BatchExecuteStatementRequest request = new BatchExecuteStatementRequest()
            .withDatabase("test")
            .withResourceArn(RESOURCE_ARN)
            .withSecretArn(SECRET_ARN)
            .withSql("INSERT INTO test_table2 VALUES (:string, :number)")
            .withParameterSets(Arrays.asList(
                    Arrays.asList(
                            new SqlParameter().withName("string").withValue(new
 Field().withStringValue("Hello")),
                            new SqlParameter().withName("number").withValue(new
 Field().withLongValue(1L))
                    ),
                    Arrays.asList(
                            new SqlParameter().withName("string").withValue(new
 Field().withStringValue("World")),
                            new SqlParameter().withName("number").withValue(new
 Field().withLongValue(2L))
                    )
            ));

    rdsData.batchExecuteStatement(request);
  }
}
```

Controllo del comportamento di timeout dell'API dati

Tutte le chiamate all'API dati sono sincrone. Supponiamo di eseguire un'operazione dell'API dati che esegue un'istruzione SQL come `INSERT` o `CREATE TABLE`. Se la chiamata API dati viene completata correttamente, l'elaborazione SQL è terminata al momento del ritorno della chiamata.

Per impostazione predefinita, l'API dati annulla un'operazione e restituisce un errore di timeout se l'elaborazione non termina entro 45 secondi. In tal caso, i dati non vengono inseriti, la tabella non viene creata e così via.

Puoi utilizzare l'API dati per eseguire operazioni di lunga durata che non possono essere completate entro 45 secondi. Se prevedi che un'operazione, come `INSERT` in blocco o un'operazione DDL su una tabella di grandi dimensioni, richieda più di 45 secondi, puoi specificare il parametro `continueAfterTimeout` per l'operazione `ExecuteStatement`. L'applicazione riceve comunque l'errore di timeout. Tuttavia, l'operazione continua a essere eseguita e non viene annullata. Per vedere un esempio, consulta Esecuzione di una transazione SQL.

Se l'SDK AWS per il linguaggio di programmazione ha un proprio periodo di timeout per le chiamate API o per le connessioni socket HTTP, assicurati che tutti questi periodi di timeout siano superiori a 45 secondi. Per alcuni SDK, il periodo di timeout è inferiore a 45 secondi per impostazione predefinita. Consigliamo di impostare qualsiasi periodo di timeout specifico dell'SDK o del client su almeno un minuto. Così eviti che l'applicazione riceva un errore di timeout mentre l'operazione dell'API dati viene completata correttamente. In tal modo, puoi sapere con certezza se ripetere o meno l'operazione.

Ad esempio, supponiamo che l'SDK restituisca un errore di timeout all'applicazione, ma che l'operazione dell'API dati venga comunque completata entro l'intervallo di timeout dell'API dati. In tal caso, ripetere l'operazione potrebbe inserire dati duplicati o produrre risultati non corretti. L'SDK potrebbe ripetere automaticamente l'operazione, causando dati errati senza alcun intervento dell'applicazione.

L'intervallo di timeout è particolarmente importante per l'SDK Java 2. In tale SDK, il timeout della chiamata API e il timeout del socket HTTP sono entrambi pari a 30 secondi per impostazione predefinita. Ecco un esempio di impostazione di tali timeout su un valore più alto:

```
public RdsDataClient createRdsDataClient() {
    return RdsDataClient.builder()
        .region(Region.US_EAST_1) // Change this to your desired Region
        .overrideConfiguration(createOverrideConfiguration())
```

```
        .httpClientBuilder(createHttpClientBuilder())
        .credentialsProvider(defaultCredentialsProvider()) // Change this to your
 desired credentials provider
        .build();
}

private static ClientOverrideConfiguration createOverrideConfiguration() {
    return ClientOverrideConfiguration.builder()
        .apiCallTimeout(Duration.ofSeconds(60))
        .build();
}

private HttpClientBuilder createHttpClientBuilder() {
    return ApacheHttpClient.builder() // Change this to your desired HttpClient
        .socketTimeout(Duration.ofSeconds(60));
}
```

Ecco un esempio equivalente che utilizza il client di dati asincrono:

```
public static RdsDataAsyncClient createRdsDataAsyncClient() {
    return RdsDataAsyncClient.builder()
        .region(Region.US_EAST_1) // Change this to your desired Region
        .overrideConfiguration(createOverrideConfiguration())
        .credentialsProvider(defaultCredentialsProvider())  // Change this to your
 desired credentials provider
        .build();
}

private static ClientOverrideConfiguration createOverrideConfiguration() {
    return ClientOverrideConfiguration.builder()
        .apiCallAttemptTimeout(Duration.ofSeconds(60))
        .build();
}

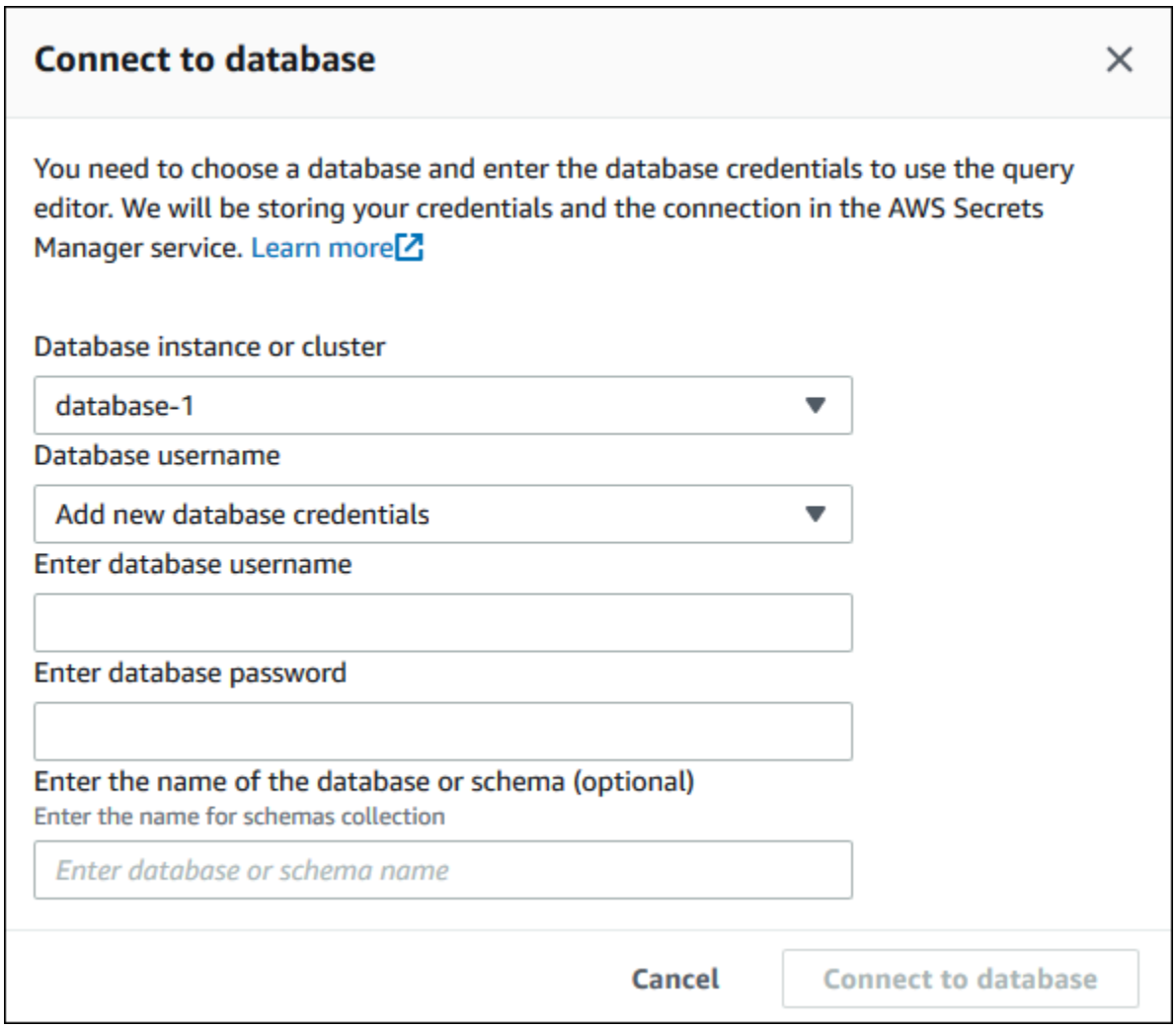
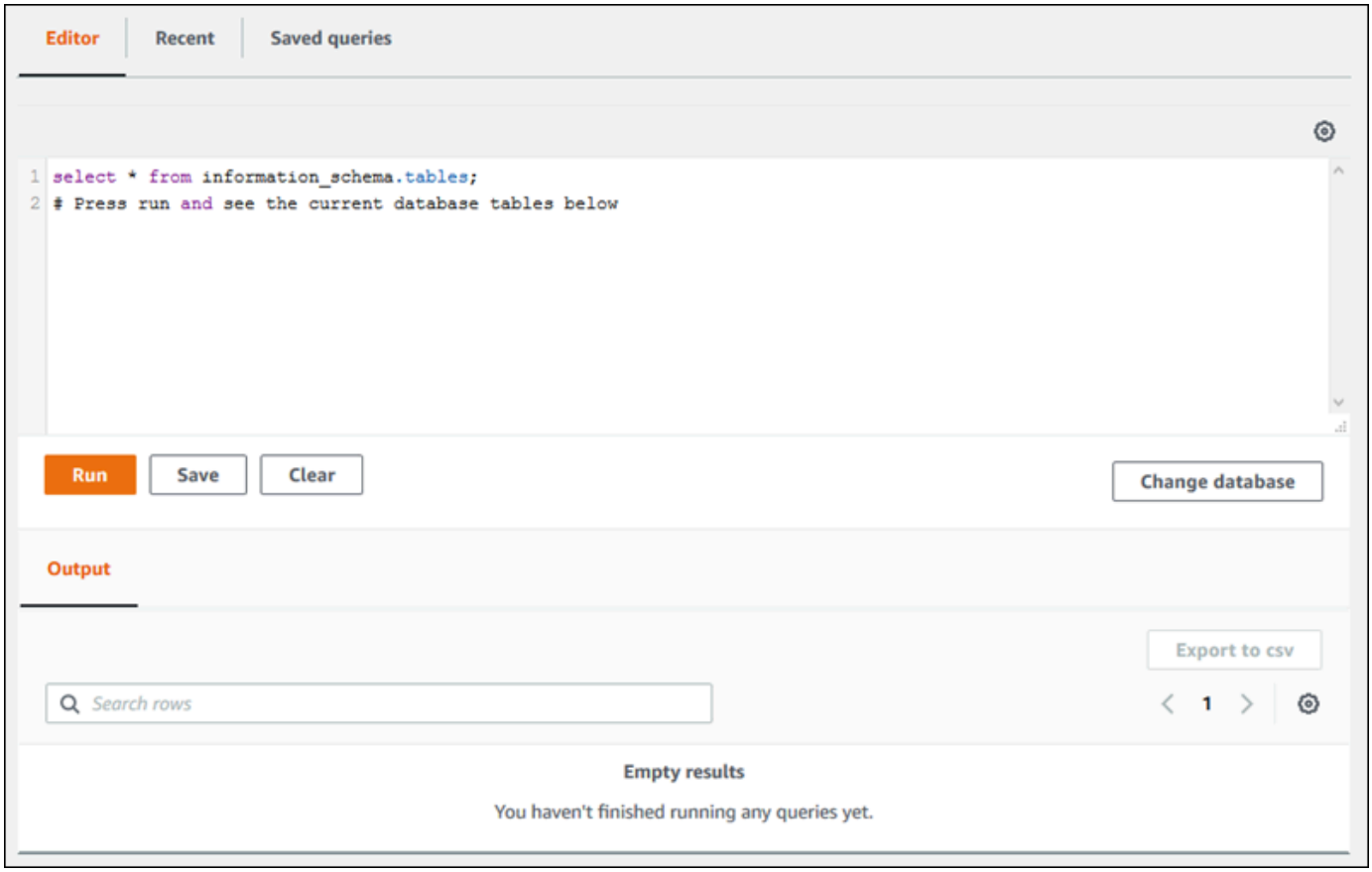
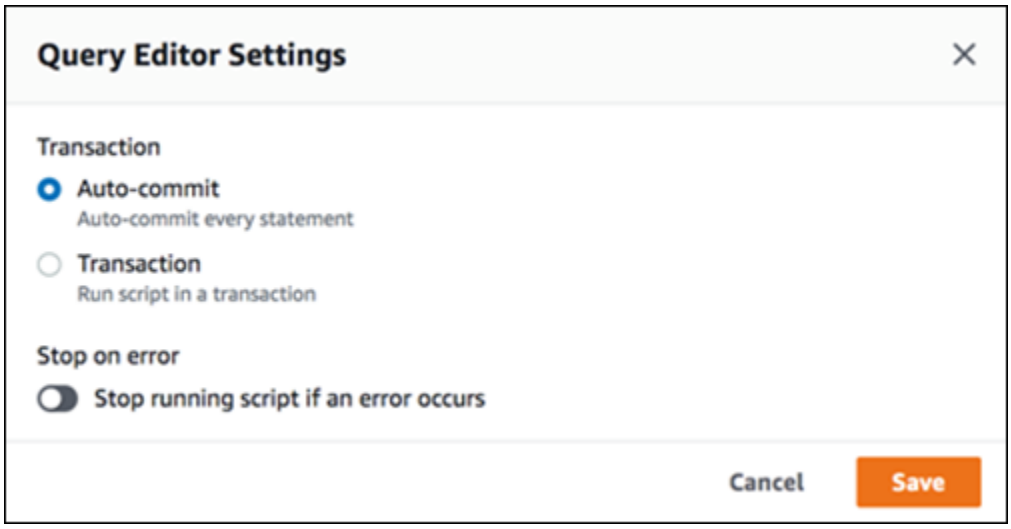
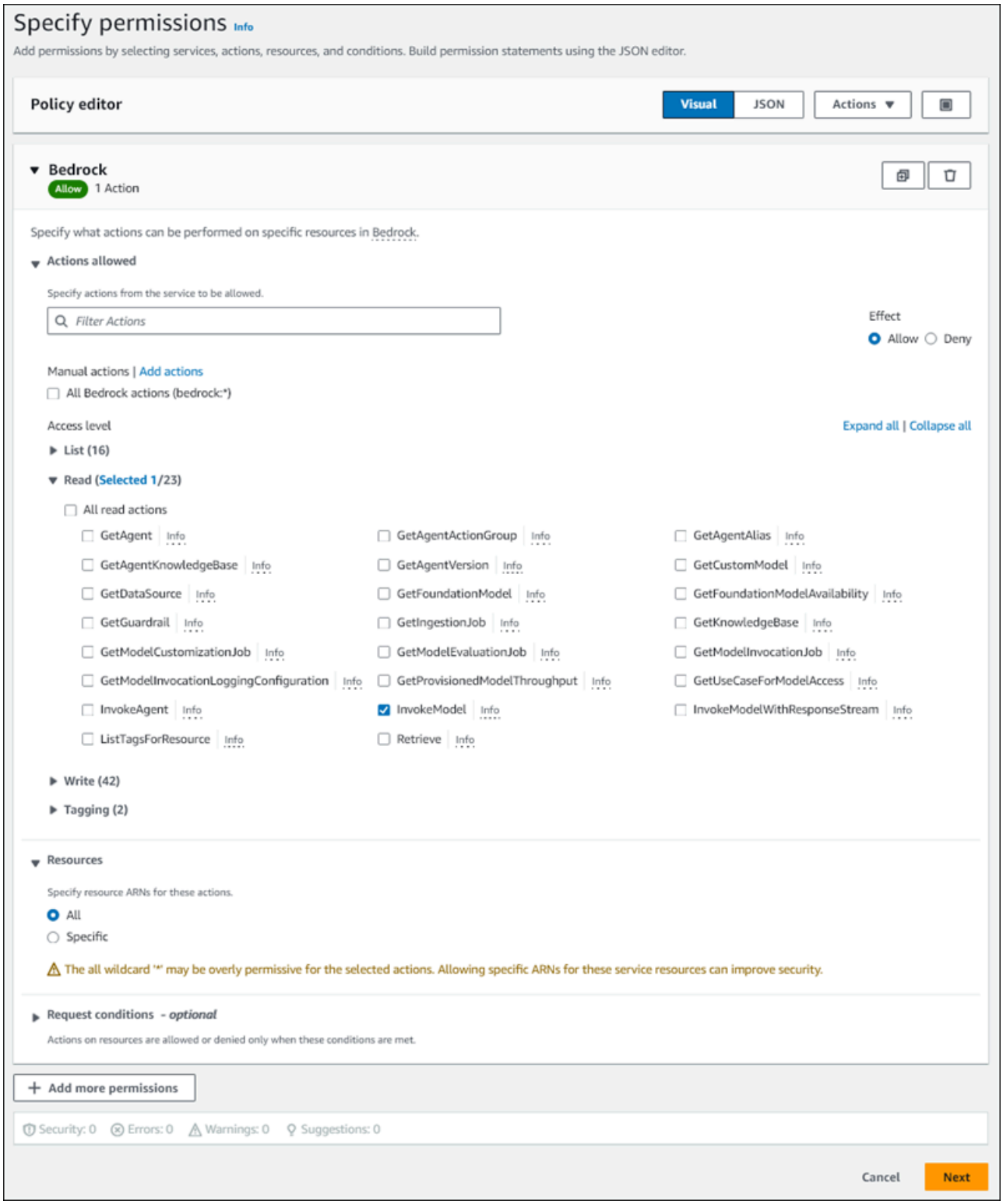
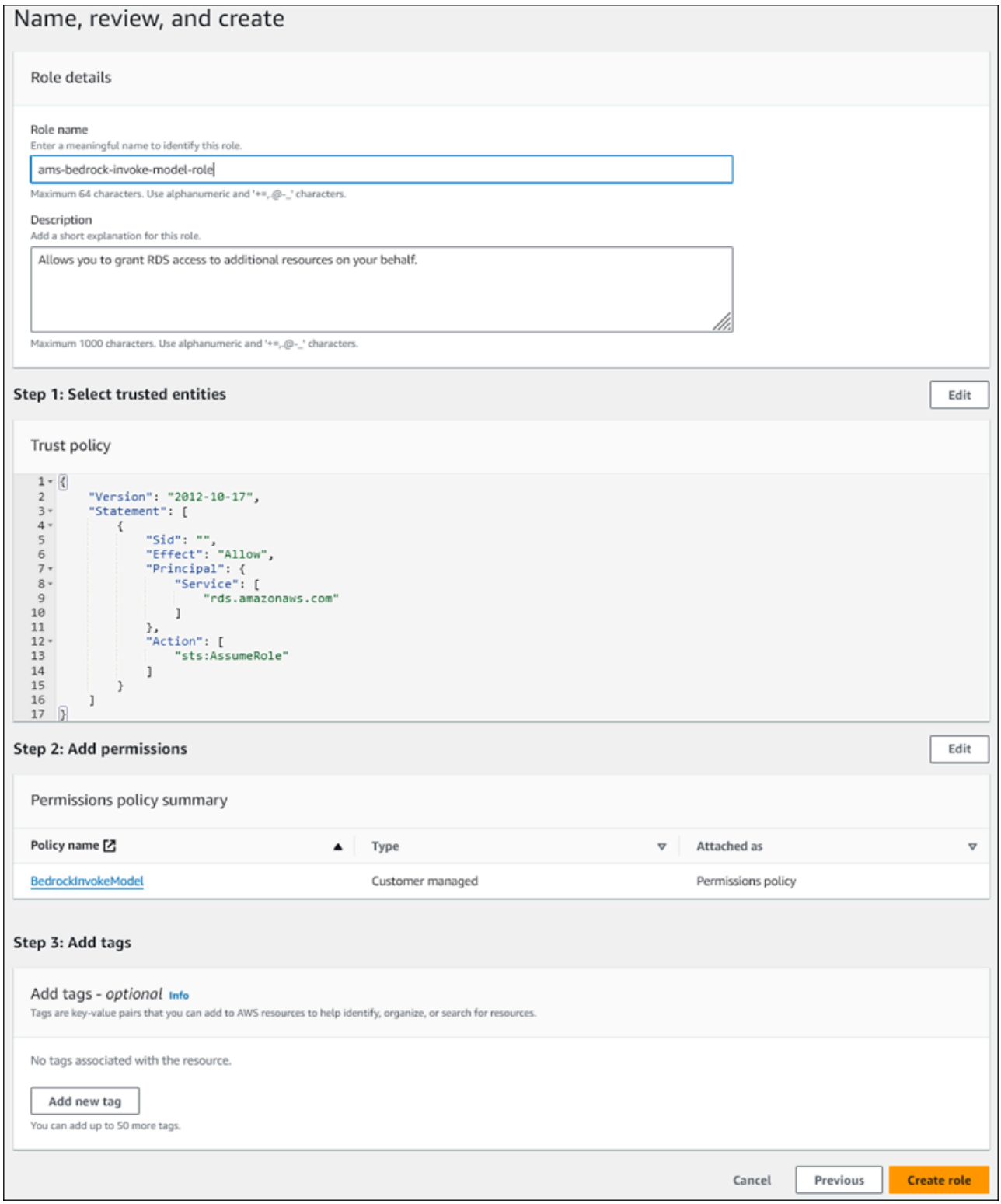
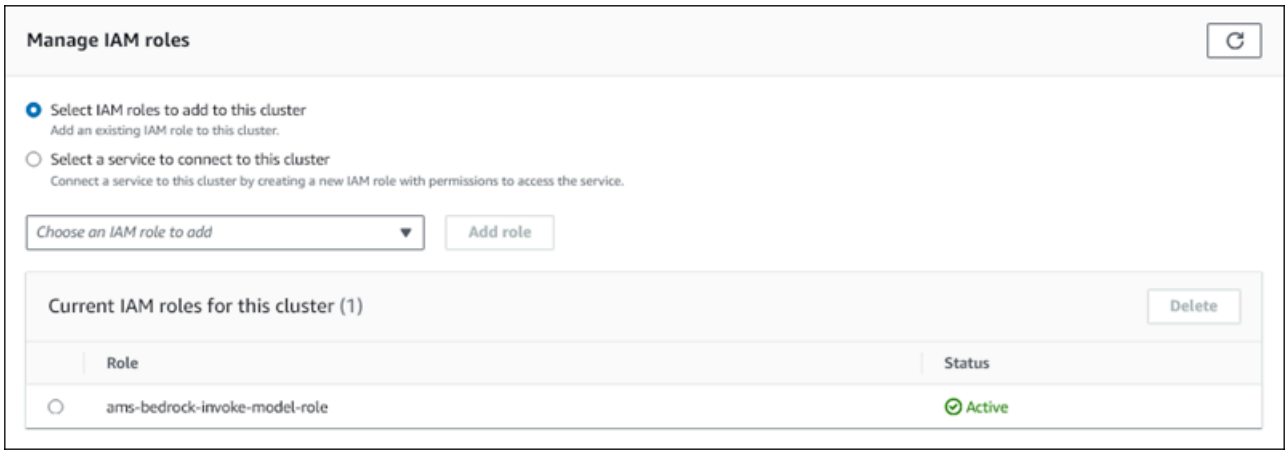
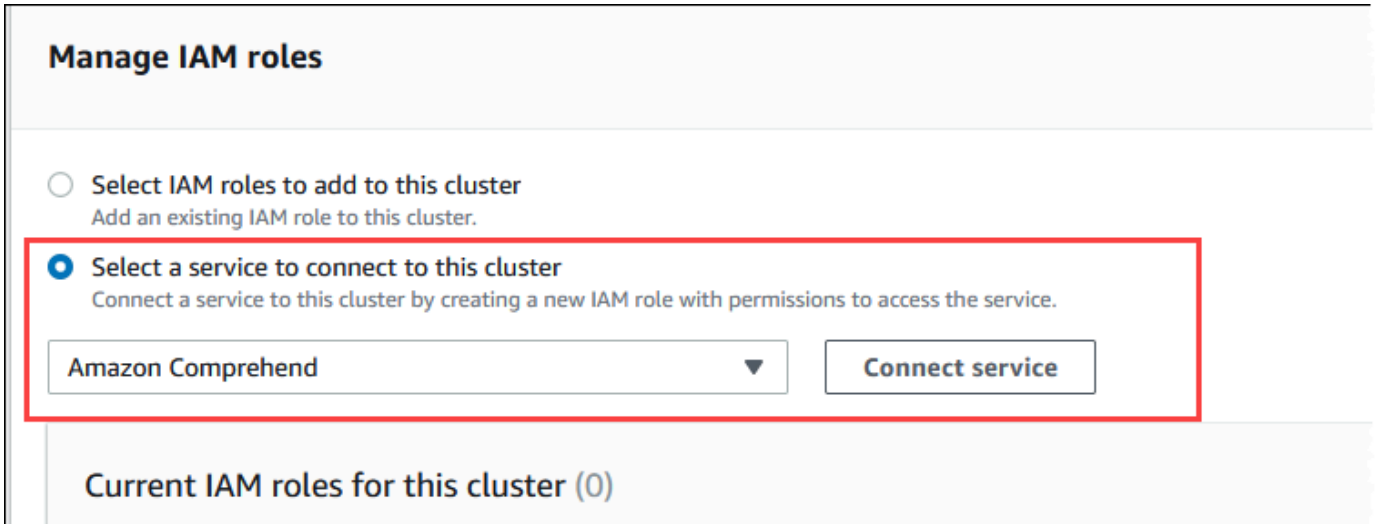
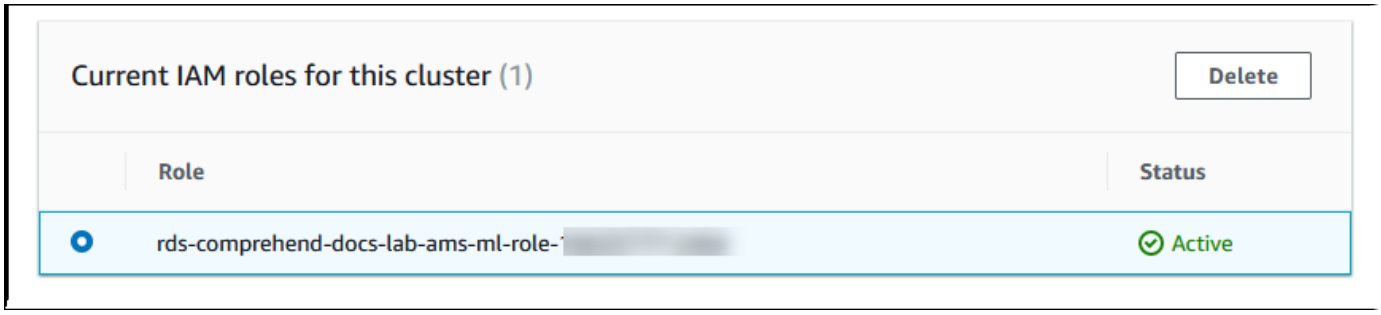
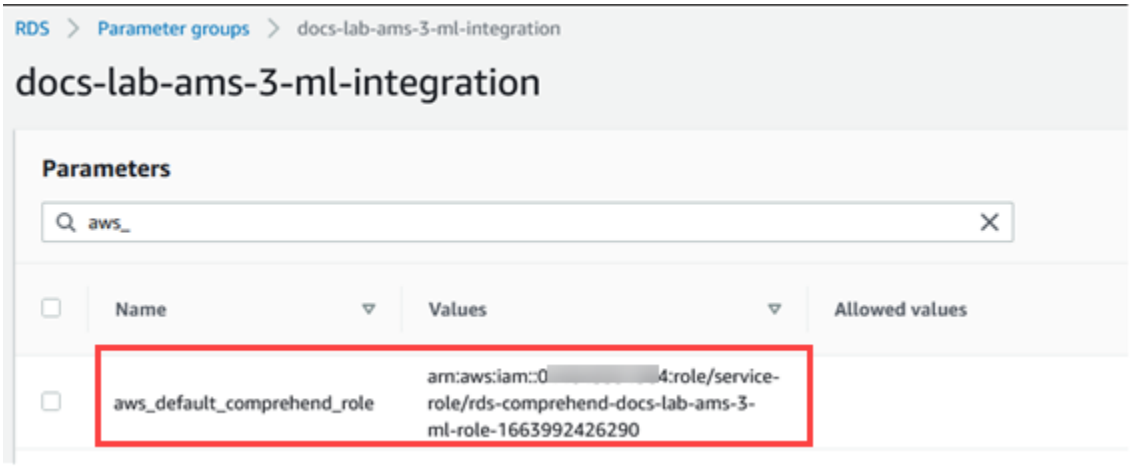
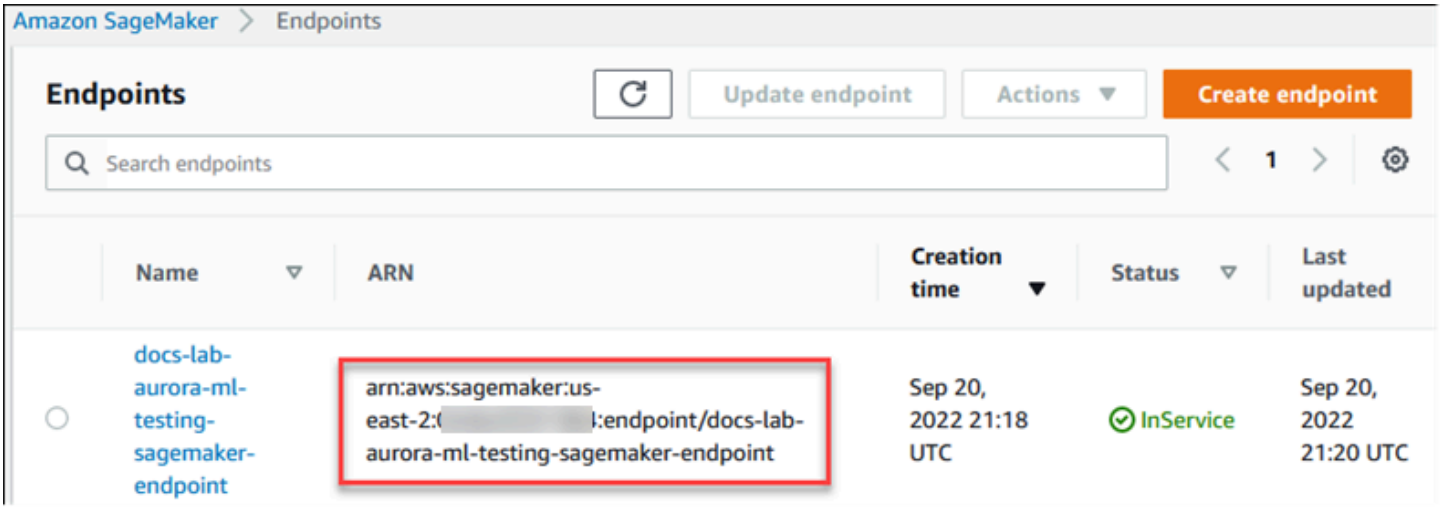
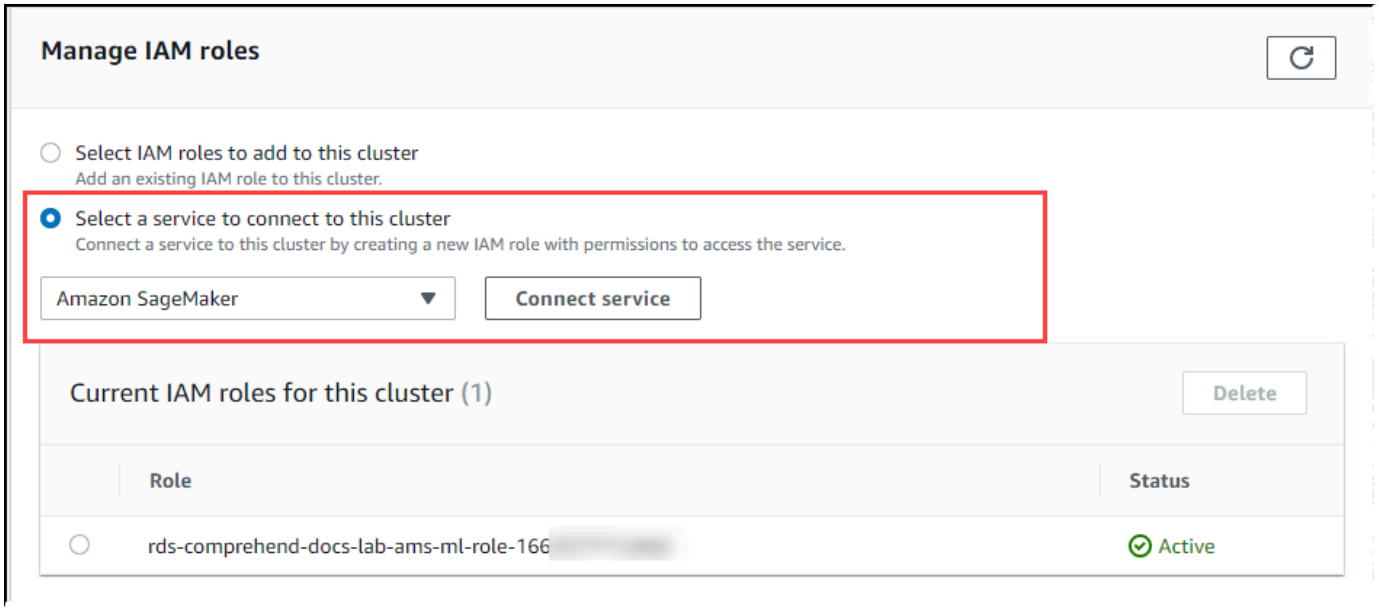
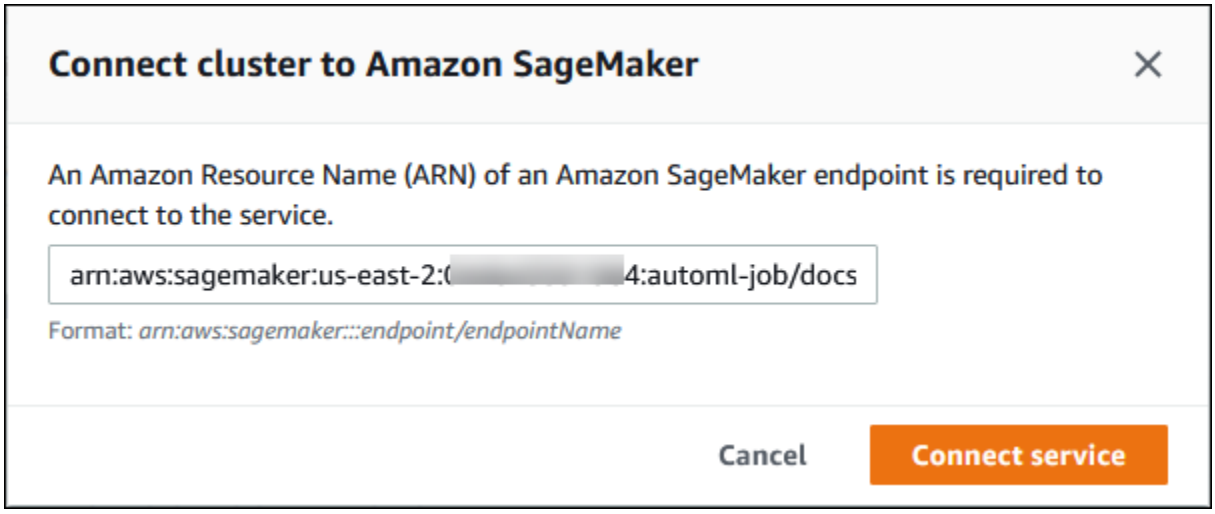
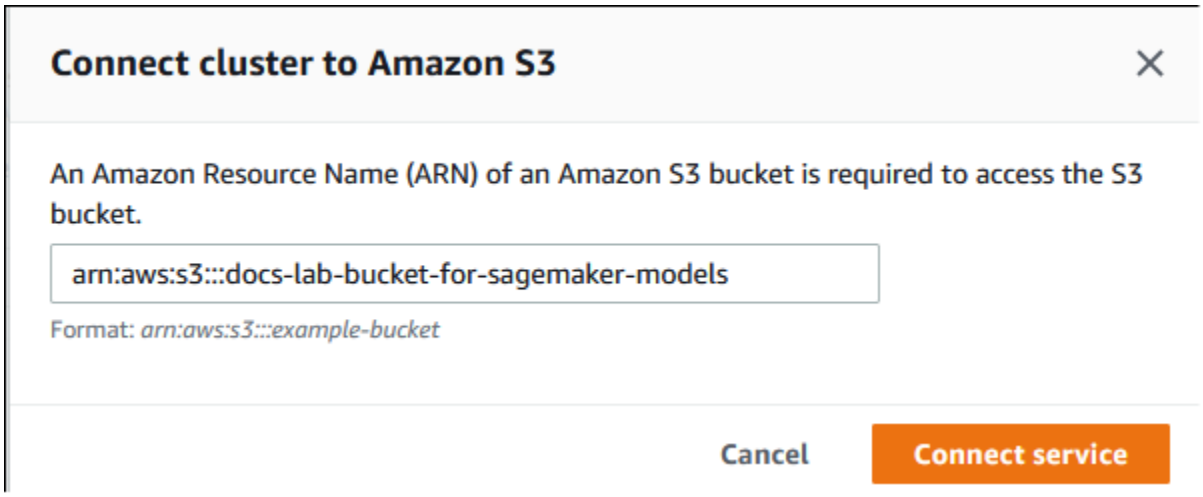
private HttpClientBuilder createHttpClientBuilder() {
    return NettyNioAsyncHttpClient.builder() // Change this to your desired
 AsyncHttpClient
        .readTimeout(Duration.ofSeconds(60));
}
```

Utilizzo della libreria client Java per l’API dati

Puoi scaricare e utilizzare una libreria client Java per l’API dati di RDS (API dati). Questa è la libreria client Java fornisce un modo alternativo per utilizzare l’API dati. Utilizzando questa libreria, puoi mappare le tue classi lato client alle richieste e alle risposte dell’API dati. Questo supporto per la mappatura può facilitare l'integrazione con alcuni tipi Java specifici, ad esempio `Date`, `Time` e `BigDecimal`.

Download della libreria client Java per l'API dati

La libreria client Java dell'API dati è open source in GitHub nella seguente posizione:

https://github.com/awslabs/rds-data-api-client-library-java

Puoi creare manualmente la libreria dai file di origine, ma la best practice consiste nell'utilizzare la libreria utilizzando la gestione delle dipendenze di Apache Maven. Aggiungi la seguente dipendenza al file Maven POM.

Per la versione 2.x, compatibile con SDK 2.x AWS, utilizza quanto segue:

```
<dependency>
   <groupId>software.amazon.rdsdata</groupId>
   <artifactId>rds-data-api-client-library-java</artifactId>
   <version>2.0.0</version>
</dependency>
```

Per la versione 1.x, compatibile con SDK 1.x AWS, utilizza quanto segue:

```
<dependency>
    <groupId>software.amazon.rdsdata</groupId>
    <artifactId>rds-data-api-client-library-java</artifactId>
    <version>1.0.8</version>
</dependency>
```

Esempi di librerie client Java

Di seguito puoi trovare alcuni esempi comuni di utilizzo della libreria client Java dell'API dati. Questi esempi presuppongono che sia disponibile una tabella `accounts` con due colonne: `accountId` e `name`. Hai anche il seguente oggetto di trasferimento dati (DTO).

```
public class Account {
    int accountId;
    String name;
    // getters and setters omitted
}
```

La libreria client consente di passare i DTO come parametri di input. L'esempio seguente mostra come i DTO dei clienti sono mappati ai set di parametri di input.

```
var account1 = new Account(1, "John");
var account2 = new Account(2, "Mary");
client.forSql("INSERT INTO accounts(accountId, name) VALUES(:accountId, :name)")
         .withParamSets(account1, account2)
         .execute();
```

In alcuni casi, è più facile lavorare con valori semplici come parametri di input. Puoi farlo con la seguente sintassi.

```
client.forSql("INSERT INTO accounts(accountId, name) VALUES(:accountId, :name)")
         .withParameter("accountId", 3)
         .withParameter("name", "Zhang")
         .execute();
```

Di seguito è riportato un altro esempio che funziona con valori semplici come parametri di input.

```

client.forSql("INSERT INTO accounts(accountId, name) VALUES(?, ?)", 4, "Carlos")
         .execute();

```

La libreria client fornisce il mapping automatico ai DTO quando viene restituito un risultato. Gli esempi seguenti mostrano come il risultato è mappato ai DTO.

```
List<Account> result = client.forSql("SELECT * FROM accounts")
          .execute()
          .mapToList(Account.class);

Account result = client.forSql("SELECT * FROM accounts WHERE account_id = 1")
          .execute()
          .mapToSingle(Account.class);
```

In molti casi, il set di risultati del database contiene solo un singolo valore. Al fine di semplificare il recupero di tali risultati, la libreria client offre le seguenti API:

```
int numberOfAccounts = client.forSql("SELECT COUNT(*) FROM accounts")
          .execute()
          .singleValue(Integer.class);
```

> **Note**
>
> La funzione `mapToList` converte un set di risultati SQL in un elenco di oggetti definito dall'utente. Non è supportato l'utilizzo dell'istruzione `.withFormatRecordsAs(RecordsFormatType.JSON)` in una chiamata `ExecuteStatement` per la libreria client Java, perché ha lo stesso scopo. Per ulteriori informazioni, consulta Elaborazione dei risultati delle query dell'API dati di Amazon RDS in formato JSON.

Elaborazione dei risultati delle query dell'API dati di Amazon RDS in formato JSON

Quando invochi l'operazione `ExecuteStatement`, puoi scegliere di restituire i risultati della query come stringa in formato JSON. In questo modo puoi utilizzare le funzionalità di parsificazione JSON del linguaggio di programmazione per interpretare e riformattare il set di risultati. Ciò può aiutare a evitare di scrivere codice aggiuntivo per scorrere il set di risultati e interpretare ogni valore di colonna.

Per richiedere il set di risultati in formato JSON, passi il parametro opzionale`formatRecordsAs` con il valore di `JSON`. Il set di risultati in formato JSON viene restituito nel campo `formattedRecords` della struttura `ExecuteStatementResponse`.

L'operazione `BatchExecuteStatement` non restituisce un set di risultati. Pertanto, l'opzione JSON non si applica a tale operazione.

Per personalizzare le chiavi nella struttura hash JSON, definisci gli alias di colonna nel set di risultati. Puoi farlo utilizzando la clausola `AS` nell'elenco delle colonne della query SQL.

Puoi utilizzare la funzionalità JSON per semplificare la lettura del set di risultati e mappare il suo contenuto su framework specifici del linguaggio. Poiché il volume del set di risultati con codifica ASCII è maggiore della rappresentazione di default, puoi scegliere la rappresentazione di default per le

query che restituiscono un numero elevato di righe o contenuti di colonne di grandi dimensioni che consumano più memoria di quella disponibile per l'applicazione.

Argomenti

- Recupero dei risultati delle query in formato JSON
- Mappatura dei tipi di dati
- Risoluzione dei problemi
- Esempi

Recupero dei risultati delle query in formato JSON

Per ricevere il set di risultati come stringa JSON, includi `.withFormatRecordsAs(RecordsFormatType.JSON)` nella chiamata di `ExecuteStatement`. Il valore restituito è ritornato come stringa JSON nel campo `formattedRecords`. In questo caso, il valore di `columnMetadata` è `null`. Le etichette di colonna sono le chiavi dell'oggetto che rappresenta ogni riga. Questi nomi di colonna vengono ripetuti per ogni riga del set dei risultati. I valori delle colonne sono stringhe con virgolette, valori numerici o valori speciali che rappresentano `true`, `false` o `null`. I metadati delle colonne come i vincoli di lunghezza e il tipo specifico di numeri e stringhe non vengono conservati nella risposta JSON.

Se ometti la chiamata `.withFormatRecordsAs()` o specifichi un parametro di `NONE`, il set di risultati viene restituito in formato binario utilizzando i campi `Records` e `columnMetadata`.

Mappatura dei tipi di dati

I valori SQL nel set di risultati sono mappati su un set più piccolo di tipi JSON. I valori sono rappresentati in JSON come stringhe, numeri e alcune costanti speciali come `true`, `false` e `null`. Puoi convertire questi valori in variabili nell'applicazione, utilizzando una tipizzazione forte o debole secondo quanto previsto per il linguaggio di programmazione.

Tipo di dati JDBC	Tipo di dati JSON
`INTEGER`, `TINYINT`, `SMALLINT`, `BIGINT`	Numero di default. Stringa se l'opzione `LongReturnType` è impostata su `STRING`.
`FLOAT`, `REAL`, `DOUBLE`	Numero

Tipo di dati JDBC	Tipo di dati JSON
`DECIMAL`	Stringa di default. Numero se l'opzione `DecimalReturnType` è impostata su `DOUBLE_OR_LONG` .
`STRING`	Stringa
`BOOLEAN`, `BIT`	Booleano
`BLOB`, `BINARY`, `VARBINARY` , `LONGVARBINARY`	Stringa nella codifica base64.
`CLOB`	Stringa
`ARRAY`	Array
`NULL`	`null`
Altri tipi (inclusi i tipi correlati a data e ora)	Stringa

Risoluzione dei problemi

La risposta JSON è limitata a 10 megabyte. Se la risposta è superiore a questo limite, il programma riceve un errore `BadRequestException`. In questo caso, è possibile risolvere l'errore utilizzando una delle seguenti tecniche:

- Ridurre il numero di righe nel set dei risultati. A tale scopo, aggiungi una clausola `LIMIT`. Puoi dividere un set di risultati di grandi dimensioni in blocchi più piccoli inviando diverse query con clausole `LIMIT` e `OFFSET`.

  Se il set di risultati include righe filtrate in base alla logica dell'applicazione, puoi rimuovere tali righe dal set di risultati aggiungendo ulteriori condizioni nella clausola `WHERE`.
- Ridurre il numero di colonne nel set dei risultati. A tale scopo, rimuovi gli elementi dall'elenco di selezione della query.
- Abbrevia le etichette delle colonne utilizzando gli alias di colonna nella query. Il nome di ogni colonna viene ripetuto nella stringa JSON per ogni riga del set di risultati. Pertanto, il risultato di una query con nomi di colonna lunghi e molte righe potrebbe superare il limite di dimensione. In

particolare, utilizza alias di colonna per espressioni complicate al fine di evitare di ripetere l'intera espressione nella stringa JSON.

- Sebbene con SQL sia possibile utilizzare alias di colonna per generare un set di risultati con più di una colonna con lo stesso nome, i nomi di chiave duplicati non sono consentiti in JSON. La Data API di RDS restituisce un errore se si richiede il set di risultati in formato JSON e più di una colonna ha lo stesso nome. Assicurati quindi che tutte le etichette delle colonne abbiano nomi univoci.

Esempi

I seguenti esempi Java mostrano come invocare `ExecuteStatement` richiedendo la risposta sotto forma di stringa in formato JSON e quindi come interpretare il set di risultati. Sostituisci i valori appropriati per i parametri *`databaseName`*, *`secretStoreArn`* e *`clusterArn`*.

Nell'esempio Java seguente viene illustrata una query che restituisce un valore numerico decimale nel set di risultati. Le invocazioni di `assertThat` verificano che i campi della risposta presentino le proprietà attese in base alle regole per i set di risultati in formato JSON.

Questo esempio funziona con lo schema e i dati di esempio seguenti:

```
create table test_simplified_json (a float);
insert into test_simplified_json values(10.0);
```

```
public void JSON_result_set_demo() {
    var sql = "select * from test_simplified_json";
    var request = new ExecuteStatementRequest()
      .withDatabase(databaseName)
      .withSecretArn(secretStoreArn)
      .withResourceArn(clusterArn)
      .withSql(sql)
      .withFormatRecordsAs(RecordsFormatType.JSON);
    var result = rdsdataClient.executeStatement(request);
}
```

Il valore del campo `formattedRecords` del programma precedente è:

```
[{"a":10.0}]
```

I campi `Records` e `ColumnMetadata` nella risposta sono entrambi nulli, a causa della presenza del set di risultati JSON.

Nell'esempio Java seguente viene illustrata una query che restituisce un valore numerico decimale nel set di risultati. L'esempio invoca `getFormattedRecords` per restituire solo la stringa in formato JSON e ignorare gli altri campi di risposta che sono vuoti o nulli. L'esempio deserializza il risultato in una struttura che rappresenta un elenco di record. Ogni record presenta campi i cui nomi corrispondono agli alias di colonna del set di risultati. Questa tecnica semplifica il codice che analizza il set di risultati. L'applicazione non deve eseguire il ciclo tra le righe e le colonne del set di risultati e convertire ogni valore nel tipo appropriato.

Questo esempio funziona con lo schema e i dati di esempio seguenti:

```
create table test_simplified_json (a int);
insert into test_simplified_json values(17);
```

```
public void JSON_deserialization_demo() {
    var sql = "select * from test_simplified_json";
    var request = new ExecuteStatementRequest()
      .withDatabase(databaseName)
      .withSecretArn(secretStoreArn)
      .withResourceArn(clusterArn)
      .withSql(sql)
      .withFormatRecordsAs(RecordsFormatType.JSON);
    var result = rdsdataClient.executeStatement(request)
      .getFormattedRecords();

/* Turn the result set into a Java object, a list of records.
   Each record has a field 'a' corresponding to the column
   labelled 'a' in the result set. */
    private static class Record { public int a; }
    var recordsList = new ObjectMapper().readValue(
        response, new TypeReference<List<Record>>() {
        });
}
```

Il valore del campo `formattedRecords` del programma precedente è:

```
[{"a":17}]
```

Per recuperare la colonna a della riga dei risultati 0, l'applicazione fa riferimento a `recordsList.get(0).a`.

Al contrario, l'esempio Java seguente mostra il tipo di codice necessario per costruire una struttura dati che contiene il set di risultati quando non si utilizza il formato JSON. In questo caso, ogni riga del set di risultati contiene campi con informazioni su un singolo utente. La creazione di una struttura dati per rappresentare il set di risultati richiede il ciclo tra le righe. Per ogni riga, il codice recupera il valore di ciascun campo, esegue una conversione di tipo appropriata e assegna il risultato al campo corrispondente dell'oggetto che rappresenta la riga. Quindi il codice aggiunge l'oggetto che rappresenta ogni utente alla struttura dati che rappresenta l'intero set di risultati. Se la query viene modificata per riordinare, aggiungere o rimuovere campi nel set di risultati, anche il codice dell'applicazione dovrebbe essere modificato.

```
/* Verbose result-parsing code that doesn't use the JSON result set format */
for (var row: response.getRecords()) {
    var user = User.builder()
      .userId(row.get(0).getLongValue())
      .firstName(row.get(1).getStringValue())
      .lastName(row.get(2).getStringValue())
      .dob(Instant.parse(row.get(3).getStringValue()))
      .build();
    result.add(user);
  }
```

I seguenti valori di esempio mostrano i valori del campo `formattedRecords` per set di risultati con diversi numeri di colonne, alias di colonna e tipi di dati di colonna.

Se il set di risultati include più righe, ogni riga viene rappresentata come un oggetto che è un elemento di una matrice. Ogni colonna del set di risultati diventa una chiave nell'oggetto. Le chiavi sono ripetute per ogni riga del set dei risultati. Pertanto, per i set di risultati costituiti da molte righe e colonne, potrebbe essere necessario definire alias di colonna brevi per evitare di superare il limite di lunghezza dell'intera risposta.

Questo esempio funziona con lo schema e i dati di esempio seguenti:

```
create table sample_names (id int, name varchar(128));
insert into sample_names values (0, "Jane"), (1, "Mohan"), (2, "Maria"), (3, "Bruce"),
 (4, "Jasmine");
```

```
[{"id":0,"name":"Jane"},{"id":1,"name":"Mohan"},
{"id":2,"name":"Maria"},{"id":3,"name":"Bruce"},{"id":4,"name":"Jasmine"}]
```

Se una colonna del set di risultati è definita come espressione, il testo dell'espressione diventa la chiave JSON. Pertanto, in genere è conveniente definire un alias di colonna descrittivo per ogni espressione nell'elenco di selezione della query. Ad esempio, la seguente query include espressioni come chiamate di funzione e operazioni aritmetiche nell'elenco di selezione.

```
select count(*), max(id), 4+7 from sample_names;
```

Tali espressioni vengono passate al set di risultati JSON come chiavi.

```
[{"count(*)":5,"max(id)":4,"4+7":11}]
```

L'aggiunta di colonne AS con etichette descrittive rende le chiavi più semplici da interpretare nel set di risultati JSON.

```
select count(*) as rows, max(id) as largest_id, 4+7 as addition_result from
 sample_names;
```

Con la query SQL revisionata, le etichette di colonna definite dalle clausole AS vengono utilizzate come nomi chiave.

```
[{"rows":5,"largest_id":4,"addition_result":11}]
```

Il valore di ogni coppia chiave-valore nella stringa JSON può essere una stringa con virgolette. La stringa potrebbe contenere caratteri unicode. Se la stringa contiene sequenze di escape o i caratteri " o \, tali caratteri sono preceduti da caratteri backslash di escape. I seguenti esempi di stringhe JSON dimostrano queste possibilità. Ad esempio, il risultato `string_with_escape_sequences` contiene i caratteri speciali backspace, nuova riga, ritorno a capo, tabulazione, form feed e \.

```
[{"quoted_string":"hello"}]
[{"unicode_string":"####"}]
[{"string_with_escape_sequences":"\b \n \r \t \f \\ '"}]
```

Il valore di ogni coppia chiave-valore nella stringa JSON può rappresentare anche un numero. Il numero potrebbe essere un numero intero, un valore a virgola mobile, un valore negativo o un valore rappresentato come notazione esponenziale. I seguenti esempi di stringhe JSON dimostrano queste possibilità.

```
[{"integer_value":17}]
[{"float_value":10.0}]
```

```
[{"negative_value":-9223372036854775808,"positive_value":9223372036854775807}]
[{"very_small_floating_point_value":4.9E-324,"very_large_floating_point_value":1.79769313486231
```

I valori booleani e null sono rappresentati con le parole chiave speciali senza virgolette `true`,`false` e `null`. I seguenti esempi di stringhe JSON dimostrano queste possibilità.

```
[{"boolean_value_1":true,"boolean_value_2":false}]
[{"unknown_value":null}]
```

Se selezioni un valore di tipo BLOB, il risultato viene rappresentato nella stringa JSON come valore codificato in base64. Per riconvertire il valore nella sua rappresentazione originale puoi utilizzare la funzione di decodifica appropriata nella lingua dell'applicazione. Ad esempio, in Java invochi la funzione `Base64.getDecoder().decode()`. Il seguente output di esempio mostra il risultato della selezione di un valore BLOB di `hello world` e della restituzione del set di risultati come stringa in formato JSON.

```
[{"blob_column":"aGVsbG8gd29ybGQ="}]
```

L'esempio Python seguente mostra come accedere ai valori dal risultato di una chiamata alla funzione Python `execute_statement`. Il set di risultati è un valore stringa nel campo `response['formattedRecords']`. Il codice trasforma la stringa JSON in una struttura dati invocando la funzione `json.loads`. Quindi ogni riga del set di risultati è un elemento elenco all'interno della struttura dati e all'interno di ogni riga puoi fare riferimento a ciascun campo del set di risultati usandone il nome.

```
import json

result = json.loads(response['formattedRecords'])
print (result[0]["id"])
```

L’esempio JavaScript seguente mostra come accedere ai valori dal risultato di una chiamata alla funzione JavaScript `executeStatement`. Il set di risultati è un valore stringa nel campo `response.formattedRecords`. Il codice trasforma la stringa JSON in una struttura dati invocando la funzione `JSON.parse`. Quindi ogni riga del set di risultati è un elemento vettore all'interno della struttura dati e all'interno di ogni riga puoi fare riferimento a ciascun campo del set di risultati usandone il nome.

```
<script>
```

```
    const result = JSON.parse(response.formattedRecords);
    document.getElementById("display").innerHTML = result[0].id;
</script>
```

Risoluzione dei problemi relativi all'API dati di Amazon RDS

Utilizza le seguenti sezioni, identificate dai messaggi di errore più comuni, per individuare e risolvere i problemi che possono verificarsi con l'API dati di Amazon RDS (API dati).

Argomenti

- La transazione <transaction_ID> non è stata trovata
- Il pacchetto per la query è troppo grande
- La risposta del database è andata oltre il limite delle dimensioni
- HttpEndpointnon è abilitato per il cluster <cluster_ID>
- DatabaseErrorException: Transaction sta ancora eseguendo una query
- Eccezione per risultato non supportato
- Le istruzioni multiple non sono supportate
- Il parametro Schema non è supportato
- IPv6 problemi di connettività

La transazione <transaction_ID> non è stata trovata

In questo caso, l'ID transazione specificato in una chiamata API dati non è stato trovato. Al messaggio di errore viene aggiunta la causa di questo problema, ovvero una delle seguenti:

- La transazione potrebbe essere scaduta.

  Assicurati che ogni chiamata della transazione venga eseguita entro tre minuti dalla precedente.

  È anche possibile che l'ID di transazione specificato non sia stato creato da una BeginTransactionchiamata. Assicurati che la chiamata abbia un ID transazione valido.
- Una chiamata precedente ha comportato l'interruzione della transazione.

  La transazione era già terminata dalla chiamata `CommitTransaction` o `RollbackTransaction`.

- La transazione è stata interrotta a causa di un errore di una chiamata precedente.

  Verifica se le chiamate precedenti hanno generato eccezioni.

Per informazioni sull'esecuzione di transazioni, consulta Chiamata dell'API dati di Amazon RDS.

Il pacchetto per la query è troppo grande

In questo caso, il set di risultati restituito per una riga era troppo grande. Il limite delle dimensioni dell'API dati è 64 KB per riga nel set di risultati restituito dal database.

Per risolvere questo problema, verifica che ogni riga in un set di risultati sia corrispondente o inferiore a 64 KB.

La risposta del database è andata oltre il limite delle dimensioni

In questo caso, le dimensioni del set di risultati restituito dal database erano troppo grandi. Il limite dell'API dati è 1 MiB nel set di risultati restituito dal database.

Per risolvere questo problema, assicurati che le chiamate API dati restituiscano 1 MiB di dati o meno. Se devi restituire più di 1 MiB, è possibile utilizzare chiamate `ExecuteStatement` molteplici con la clausola `LIMIT` nella query.

Per ulteriori informazioni sulla clausola `LIMIT`, consulta SELECT Syntax nella documentazione MySQL.

HttpEndpointnon è abilitato per il cluster <cluster_ID>

Controlla le seguenti potenziali cause di questo problema:

- Il cluster di database Aurora non supporta l'API dati. Per informazioni sui tipi di cluster di database supportati dall'API dati di RDS, consulta the section called "Disponibilità per Regione e versione dell'API dati di Amazon RDS".
- L'API dati non è abilitata per il cluster di database Aurora. Per utilizzare l'API dati con un cluster di database Aurora, è necessario abilitarla per il cluster di database. Per informazioni su come abilitare l'API dati, consulta Abilitazione dell'API dati di Amazon RDS.
- Il cluster di database è stato rinominato dopo che l'API dati era stata abilitata. In tal caso, disattiva l'API dati per quel cluster e poi riabilitala.

- L'ARN specificato non corrisponde esattamente all'ARN del cluster. Verifica che l'ARN restituito da un'altra origine o costruito dalla logica del programma corrisponda esattamente all'ARN del cluster. Ad esempio, assicurati che l'ARN utilizzato presenti il corretto formato maiuscolo/minuscolo per tutti i caratteri alfabetici.

DatabaseErrorException: Transaction sta ancora eseguendo una query

Se l’applicazione invia una richiesta con un ID transazione e tale transazione sta attualmente elaborando un’altra richiesta, l’API dati restituisce immediatamente questo errore all’applicazione. Questa condizione potrebbe verificarsi se l’applicazione effettua richieste asincrone, utilizzando un meccanismo come le “promesse” in JavaScript.

Per risolvere questo problema, attendi il completamento della richiesta precedente, quindi riprova. Puoi continuare a riprovare finché l’errore non si verifica più o finché l’applicazione non riceve un errore di tipo diverso.

Questa condizione può verificarsi con l’API dati per le istanze Aurora Serverless v2 e con provisioning. Nell’API dati per Aurora Serverless v1, le richieste successive con lo stesso ID transazione attendono automaticamente che la richiesta precedente venga completata. Tuttavia, quel comportamento precedente poteva generare timeout se la richiesta precedente impiegava troppo tempo. Se stai trasferendo un’applicazione precedente che utilizza l’API dati e invia richieste concorrenti, modifica la logica di gestione delle eccezioni per gestire questo nuovo tipo di errore.

Eccezione per risultato non supportato

L’API dati non supporta tutti i tipi di dati. Questo errore si verifica quando viene eseguita una query che restituisce un tipo di dato non supportato.

Per risolvere il problema, esegui il casting del tipo di dati non supportato su `TEXT`. Ad esempio:

```
SELECT custom_type::TEXT FROM my_table;
-- OR
SELECT CAST(custom_type AS TEXT) FROM my_table;
```

Le istruzioni multiple non sono supportate

Le istruzioni multiple non sono supportate nell’API dati per Aurora Serverless v2 e per i cluster con provisioning. Il tentativo di eseguire più istruzioni in una singola chiamata API restituisce questo errore.

Per eseguire più istruzioni, utilizza chiamate API `ExecuteStatement` separate oppure utilizza l'API `BatchExecuteStatement` per l'elaborazione in batch.

Il parametro Schema non è supportato

Aurora Serverless v1 ignora silenziosamente il parametro Schema. Tuttavia, Aurora Serverless v2 e i cluster con provisioning rifiutano esplicitamente le chiamate API che includono il parametro Schema.

Per risolvere questo problema, rimuovi il parametro Schema da tutte le chiamate API dati quando utilizzi Aurora Serverless v2 o cluster con provisioning.

IPv6 problemi di connettività

Se riscontri problemi durante la connessione a Data API tramite IPv6 endpoint, verifica le seguenti cause potenziali:

- La rete non supporta IPv6: verifica che l'infrastruttura di rete supporti IPv6 e che il IPv6 routing sia configurato correttamente.
- Problemi di risoluzione DNS: che il risolutore DNS sia in grado di risolvere i record AAAA per gli endpoint dual stack (ad esempio `rds-data.us-east-1.api.aws`).
- Configurazione del gruppo di sicurezza: aggiorna le regole del gruppo di sicurezza per consentire IPv6 il traffico sulla porta 443 (HTTPS). Aggiungi regole per i blocchi IPv6 CIDR (ad esempio, `::/0` per tutti gli IPv6 indirizzi).
- Configurazione ACL di rete: assicurati che la rete ACLs consenta il IPv6 traffico sulle porte richieste.
- Compatibilità con le librerie client: verifica che le librerie e i client HTTP AWS SDKs IPv6 supportino la connettività dual-stack.
- Configurazione degli endpoint VPC: se in uso PrivateLink, assicurati che l'endpoint VPC sia configurato per il supporto IPv6 e che alle sottoreti associate siano assegnati blocchi CIDR. IPv6

Per risolvere i problemi di connettività: IPv6

1. Verifica la connettività utilizzando IPv4 -only endpoints (`.amazonaws.com`) per verificare che il problema sia specifico per. IPv6
2. Utilizza gli strumenti di diagnostica di rete per verificare la IPv6 connettività agli endpoint dual-stack.

3. Controlla CloudTrail i log per eventuali errori di autenticazione o autorizzazione quando usi gli endpoint. IPv6
4. Verifica che l'applicazione sia configurata correttamente per utilizzare il nuovo endpoint dual-stack. URLs

Registrazione di log delle chiamate API dati di Amazon RDS con AWS CloudTrail

API dati RDS (API dati) è integrato con AWS CloudTrail, un servizio che offre un record delle operazioni eseguite da un utente, un ruolo o un servizio AWS nell'API dati. CloudTrail acquisisce tutte le chiamate API per l'API dati, come gli eventi, incluse le chiamate dalla console Amazon RDS e dalle chiamate di codice alle operazioni dell'API dati. Se viene creato un trail, è possibile abilitare la distribuzione continua di eventi CloudTrail in un bucket Amazon S3, inclusi gli eventi per l'API dati. Utilizzando i dati raccolti da CloudTrail, è possibile stabilire molte informazioni. Queste informazioni includono la richiesta che è stata fatta all'API dati, l'indirizzo IP da cui è stata effettuata la richiesta, chi ha effettuato la richiesta, quando è stata fatta e ulteriori dettagli.

Per ulteriori informazioni su CloudTrail, consulta la Guida per l'utente di AWS CloudTrail.

Utilizzo delle informazioni dell'API dati in CloudTrail

CloudTrail è abilitato sull'account AWS al momento della sua creazione. Quando si verifica un'attività supportata (eventi di gestione) nell'API dati, questa viene registrata in un evento CloudTrail insieme ad altri eventi di servizio AWS in Cronologia eventi. È possibile visualizzare, cercare e scaricare gli eventi di gestione recenti nell'account AWS. Per ulteriori informazioni, consulta Working with CloudTrail Event history nella Guida per l'utente di AWS CloudTrail.

Per una registrazione continua degli eventi nell'account AWS, inclusi gli eventi relativi all'API dati, crea un trail. Un trail consente a CloudTrail di distribuire i file di log in un bucket Amazon S3. Per impostazione predefinita, quando si crea un trail nella console, il trail sarà valido in tutte le Regioni AWS. Il percorso registra gli eventi da tutte le regioni AWSnella partizione AWS e distribuisce i file di log nel bucket Simple Storage Service (Amazon S3) specificato. Inoltre, è possibile configurare altri servizi AWS per analizzare con maggiore dettaglio e usare i dati evento raccolti nei log CloudTrail. Per ulteriori informazioni, consulta gli argomenti seguenti nella Guida per l'utente di AWS CloudTrail:

- Panoramica della creazione di un percorso
- Servizi e integrazioni CloudTrail supportati

- Configurazione delle notifiche Amazon SNS per CloudTrail
- Ricezione di file di log CloudTrail da più regioni e Ricezione di file di log CloudTrail da più account

Tutte le operazioni dell'API dati vengono registrate da CloudTrail e documentate nella Documentazione di riferimento API del servizio dati Amazon RDS. Ad esempio, tutte le chiamate alle operazioni `BatchExecuteStatement`, `BeginTransaction`, `CommitTransaction` e `ExecuteStatement` generano voci nei file di log di CloudTrail.

Ogni evento o voce di log contiene informazioni sull'utente che ha generato la richiesta. Le informazioni di identità consentono di determinare quanto segue:

- Se la richiesta è stata effettuata con credenziali utente o root.
- Se la richiesta è stata effettuata con le credenziali di sicurezza temporanee per un ruolo o un utente federato.
- Se la richiesta è stata effettuata da un altro servizio AWS.

Per ulteriori informazioni, consulta Elemento CloudTrail userIdentity.

Esclusione di eventi API dati da un trail AWS CloudTrail

La maggior parte degli utenti di API dati si basa sugli eventi in un percorso AWS CloudTrail per fornire un record delle operazioni di API dati. I dati degli eventi non rivelano il nome del database, il nome dello schema o le istruzioni SQL nelle richieste all'API dati. Tuttavia, sapere quale utente ha effettuato un tipo di chiamata su uno specifico cluster di database in un determinato momento può aiutare a rilevare modelli di accesso anomali.

Inclusione di eventi API dati in un trail AWS CloudTrail

Per Aurora PostgreSQL Serverless v2 e i database con provisioning, le seguenti operazioni dell'API dati vengono registrate in AWS CloudTrail come eventi di dati. Gli eventi di dati sono operazioni API del piano dati con volumi elevati di cui CloudTrail non registra log per impostazione predefinita. Per gli eventi di dati sono previsti costi aggiuntivi. Per ulteriori informazioni sui prezzi di CloudTrail, consulta Prezzi di AWS CloudTrail.

- BatchExecuteStatement
- BeginTransaction
- CommitTransaction

- ExecuteStatement
- RollbackTransaction

È possibile utilizzare la console CloudTrail, AWS CLI o le operazioni dell'API CloudTrail per registrare log di queste operazioni dell'API dati. Nella console CloudTrail, scegli API dati RDS - Cluster database per il tipo di evento dati. Per ulteriori informazioni, consulta Registrazione degli eventi di dati con Console di gestione AWS nella Guida per l'utente di AWS CloudTrail.

Utilizzando AWS CLI, esegui il comando `aws cloudtrail put-event-selectors` per registrare log di queste operazioni dell'API dati per il trail. Per registrare log di tutti gli eventi API dati nei cluster di database, specifica `AWS::RDS::DBCluster` per il tipo di risorsa. L'esempio seguente registra tutti gli eventi API dati nei cluster di database. Per ulteriori informazioni, consulta Registrazione degli eventi di dati con AWS Command Line Interface nella Guida per l'utente di AWS CloudTrail.

```
aws cloudtrail put-event-selectors --trail-name trail_name --advanced-event-selectors \
'{
   "Name": "RDS Data API Selector",
   "FieldSelectors": [
      {
         "Field": "eventCategory",
         "Equals": [
            "Data"
         ]
      },
      {
         "Field": "resources.type",
         "Equals": [
            "AWS::RDS::DBCluster"
         ]
      }
   ]
}'
```

È possibile configurare selettori di eventi avanzati per filtrare ulteriormente i campi `readOnly`, `eventName`, e `resources.ARN`. Per ulteriori informazioni su questi campi, consulta AdvancedFieldSelector.

Esclusione di eventi API dati da un trail AWS CloudTrail (solo Aurora Serverless v1)

Per Aurora Serverless v1, gli eventi API dati sono eventi di gestione. Per impostazione predefinita, tutti gli eventi API dati sono inclusi in un trail AWS CloudTrail. Tuttavia, poiché l'API dati può generare un numero elevato di eventi, si potrebbe voler escludere questi eventi da un trail CloudTrail. L'impostazione Escludi eventi API dei dati Amazon RDS esclude tutti gli eventi API dati dal trail. Non è possibile escludere eventi API dati specifici.

Per escludere gli eventi API dati da un percorso, procedi nel seguente modo:

- Nella console CloudTrail, scegli l'impostazione Escludi eventi API dati di Amazon RDS durante la creazione o l'aggiornamento di un percorso.
- Nell'API CloudTrail, utilizza l'operazione PutEventSelectors. Se si utilizzano selettori di eventi avanzati, è possibile escludere gli eventi API dati impostando il campo `eventSource` su un valore diverso da `rdsdata.amazonaws.com`. Se si utilizzano selettori di eventi di base, è possibile escludere gli eventi API dati impostando il valore dell'attributo `ExcludeManagementEventSources` su `rdsdata.amazonaws.com`. Per ulteriori informazioni, consulta Registrazione degli eventi con AWS Command Line Interface nella Guida per l'utente di AWS CloudTrail.

> **Warning**
>
> L'esclusione di eventi API dati da un log CloudTrail può oscurare le operazioni delle API dati. Presta attenzione quando concedi alle entità principali l'autorizzazione `cloudtrail:PutEventSelectors` necessaria per eseguire questa operazione.

È possibile disattivare questa esclusione in qualsiasi momento modificando l'impostazione della console o i selettori di eventi per un percorso. Il percorso inizierà quindi a registrare gli eventi API dati. Tuttavia, non è possibile ripristinare gli eventi API dati che si sono verificati mentre era in atto l'esclusione.

Quando si escludono eventi API dati utilizzando la console o l'API, anche l'operazione API `PutEventSelectors` di CloudTrail risultante viene registrata nei log CloudTrail. Se gli eventi dell'API dati non vengono visualizzati nei log CloudTrail, ricerca un evento `PutEventSelectors` con l'attributo `ExcludeManagementEventSources` impostato su `rdsdata.amazonaws.com`.

Per ulteriori informazioni, consulta Registrazione di eventi di gestione per i percorsi nella Guida per l'utente di AWS CloudTrail.

Informazioni sulle voci dei file di registro dell'API dei dati

Un percorso è una configurazione che consente la distribuzione di eventi come file di log in un bucket Amazon S3 specificato. I file di log di CloudTrail possono contenere una o più voci di log. Un evento rappresenta una singola richiesta da un'origine e include informazioni sull'operazione richiesta, data e ora dell'operazione, parametri della richiesta e così via. I file di log CloudTrail non sono una traccia dello stack ordinata delle chiamate API pubbliche e di conseguenza non devono apparire in base a un ordine specifico.

Aurora PostgreSQL Serverless v2 e database con provisioning

L'esempio seguente mostra una voce di log di CloudTrail che illustra l'operazione `ExecuteStatement` per Aurora PostgreSQL Serverless v2 e i database con provisioning. Per questi database, tutti gli eventi API dati sono eventi di dati in cui l'origine dell'evento è rdsdataapi.amazonaws.com e il tipo di evento è Rds Data Service.


```
{
    "eventVersion": "1.05",
    "userIdentity": {
        "type": "IAMUser",
        "principalId": "AKIAIOSFODNN7EXAMPLE",
        "arn": "arn:aws:iam::123456789012:user/johndoe",
        "accountId": "123456789012",
        "accessKeyId": "AKIAI44QH8DHBEXAMPLE",
        "userName": "johndoe"
    },
    "eventTime": "2019-12-18T00:49:34Z",
    "eventSource": "rdsdataapi.amazonaws.com",
    "eventName": "ExecuteStatement",
    "awsRegion": "us-east-1",
    "sourceIPAddress": "192.0.2.0",
    "userAgent": "aws-cli/1.16.102 Python/3.7.2 Windows/10 botocore/1.12.92",
    "requestParameters": {
        "continueAfterTimeout": false,
        "database": "**********",
        "includeResultMetadata": false,
        "parameters": [],
```

```
        "resourceArn": "arn:aws:rds:us-east-1:123456789012:cluster:my-database-1",
        "schema": "**********",
        "secretArn": "arn:aws:secretsmanager:us-
east-1:123456789012:secret:dataapisecret-ABC123",
        "sql": "**********"
    },
    "responseElements": null,
    "requestID": "6ba9a36e-b3aa-4ca8-9a2e-15a9eada988e",
    "eventID": "a2c7a357-ee8e-4755-a0d0-aed11ed4253a",
    "eventType": "Rds Data Service",
    "recipientAccountId": "123456789012"
}
```

Aurora Serverless v1

L'esempio seguente mostra come viene visualizzata la voce di log CloudTrail dell'esempio precedente per Aurora Serverless v1. Per Aurora Serverless v1, tutti gli eventi sono eventi di gestione in cui l'origine dell'evento è rdsdata.amazonaws.com e il tipo di evento è AwsApiCall.

```
{
    "eventVersion": "1.05",
    "userIdentity": {
        "type": "IAMUser",
        "principalId": "AKIAIOSFODNN7EXAMPLE",
        "arn": "arn:aws:iam::123456789012:user/johndoe",
        "accountId": "123456789012",
        "accessKeyId": "AKIAI44QH8DHBEXAMPLE",
        "userName": "johndoe"
    },
    "eventTime": "2019-12-18T00:49:34Z",
    "eventSource": "rdsdata.amazonaws.com",
    "eventName": "ExecuteStatement",
    "awsRegion": "us-east-1",
    "sourceIPAddress": "192.0.2.0",
    "userAgent": "aws-cli/1.16.102 Python/3.7.2 Windows/10 botocore/1.12.92",
    "requestParameters": {
        "continueAfterTimeout": false,
        "database": "**********",
        "includeResultMetadata": false,
        "parameters": [],
        "resourceArn": "arn:aws:rds:us-east-1:123456789012:cluster:my-database-1",
        "schema": "**********",
```

```
        "secretArn": "arn:aws:secretsmanager:us-
east-1:123456789012:secret:dataapisecret-ABC123",
        "sql": "**********"
    },
    "responseElements": null,
    "requestID": "6ba9a36e-b3aa-4ca8-9a2e-15a9eada988e",
    "eventID": "a2c7a357-ee8e-4755-a0d0-aed11ed4253a",
    "eventType": "AwsApiCall",
    "recipientAccountId": "123456789012"
}
```

Monitoraggio delle query dell'API dati RDS con Approfondimenti sulle prestazioni

Se il cluster Aurora esegue istanze Aurora Serverless v2 o con provisioning, è possibile utilizzare Approfondimenti sulle prestazioni con l'API dati RDS.

Per ulteriori informazioni su come utilizzare Approfondimenti sulle prestazioni con Aurora, consulta Monitoraggio del carico DB con Performance Insights su Amazon Aurora.

Come vengono rappresentate le query dell'API dati RDS in Approfondimenti sulle prestazioni

Con l'API dati, il cluster Aurora elabora le query in base alle chiamate API dati inviate dall'applicazione. L'API dati esegue anche alcune istruzioni SQL come parte del proprio funzionamento interno, come l'annullamento di query che superano la soglia di timeout. Entrambi i tipi di operazioni SQL sono mostrati nelle statistiche e nei grafici di Approfondimenti sulle prestazioni.

- Per le query dell'API dati inviate a un cluster Aurora, il campo Host nella dashboard delle API è contrassegnato come API dati RDS. Per Aurora PostgreSQL, il campo application_name ha il valore `rds-data-api`. Cercare queste etichette quando si analizza il carico del database utilizzando Host principali o Applicazioni principali come dimensione.
- Tutte le query interne eseguite dall'API dati per gestire aspetti del database, come il pool di connessioni e i timeout delle query, sono annotate con il prefisso API dati RDS. Esempio: `/* RDS Data API */ select * from my_table;` cerca questi prefissi quando si analizza il carico del database tramite SQL principale come dimensione. Le istruzioni sono annotate con un commento SQL di `/* RDS Data API */`.

Utilizzo dell’editor di query Aurora

L’editor di query Aurora consente di eseguire le istruzioni SQL sul cluster di database Aurora tramite la Console di gestione AWS. È possibile eseguire query SQL, istruzioni di manipolazione dei dati (DML) e istruzioni di definizione dei dati (DDL). L’utilizzo dell’interfaccia della console consente di eseguire la manutenzione del database, produrre report e condurre esperimenti SQL. È possibile evitare di configurare la configurazione di rete per la connessione al cluster DB da un sistema client separato, EC2 ad esempio un'istanza o un computer portatile.

L’editor di query richiede un cluster di database Aurora con l’API dati in RDS (API dati) abilitata. Per informazioni sui cluster di database che supportano l’API dati e su come abilitarla, consulta Utilizzo dell’API dati di Amazon RDS. Il codice SQL che è possibile eseguire è soggetto alle limitazioni dell’API dati. Per ulteriori informazioni, consulta the section called “Limitazioni”.

Disponibilità dell'editor di query

L'editor di query è disponibile per i cluster Aurora DB che utilizzano le versioni dei motori Aurora MySQL e Aurora PostgreSQL che supportano Data API e dove Data API è disponibile. Regioni AWS Per ulteriori informazioni, consulta Regioni e motori di database Aurora supportati per l’API dati RDS.

Autorizzazione di accesso all'editor della query

Un utente deve essere autorizzato a eseguire query nell'editor della query. È possibile autorizzare un utente a eseguire query nell'editor di query aggiungendo la policy, una `AmazonRDSDataFullAccess` policy predefinita (IAM), a quell'utente. AWS Identity and Access Management

> **Note**
>
> Assicurarsi di utilizzare lo stesso nome utente e password utilizzati per l’utente del database, ad esempio il nome utente e la password amministrativi, quando si crea l’utente IAM. Per ulteriori informazioni, consulta Creazione di un utente IAM nell'Account AWS nella Guida per l'utente di AWS Identity and Access Management.

Puoi anche creare una policy IAM che concede l'accesso all'editor della query. Dopo aver creato la policy, aggiungila a ciascun utente che richiede l'accesso all'editor della query.

La seguente policy fornisce le autorizzazioni minime richieste per un utente per accedere all'editor della query.

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Sid": "QueryEditor0",
            "Effect": "Allow",
            "Action": [
                "secretsmanager:GetSecretValue",
                "secretsmanager:PutResourcePolicy",
                "secretsmanager:PutSecretValue",
                "secretsmanager:DeleteSecret",
                "secretsmanager:DescribeSecret",
                "secretsmanager:TagResource"
            ],
            "Resource": "arn:aws:secretsmanager:*:*:secret:rds-db-credentials/*"
        },
        {
            "Sid": "QueryEditor1",
            "Effect": "Allow",
            "Action": [
                "secretsmanager:GetRandomPassword",
                "tag:GetResources",
                "secretsmanager:CreateSecret",
                "secretsmanager:ListSecrets",
                "dbqms:CreateFavoriteQuery",
                "dbqms:DescribeFavoriteQueries",
                "dbqms:UpdateFavoriteQuery",
                "dbqms:DeleteFavoriteQueries",
                "dbqms:GetQueryString",
                "dbqms:CreateQueryHistory",
                "dbqms:UpdateQueryHistory",
                "dbqms:DeleteQueryHistory",
                "dbqms:DescribeQueryHistory",
                "rds-data:BatchExecuteStatement",
                "rds-data:BeginTransaction",
                "rds-data:CommitTransaction",
                "rds-data:ExecuteStatement",
```

```
                "rds-data:RollbackTransaction"
            ],
            "Resource": "*"
        }
    ]
}
```

Per informazioni sulla creazione di una policy IAM, consulta Creazione di policy IAM nella Guida per l'utente di AWS Identity and Access Management.

Per informazioni sull'aggiunta di una policy IAM a un utente, consulta Aggiunta e rimozione di autorizzazioni per identità IAM nella Guida per l'utente di AWS Identity and Access Management.

Esecuzione di query nell'editor della query

È possibile eseguire le istruzioni SQL in un cluster di database Aurora nell'editor di query. Il codice SQL che è possibile eseguire è soggetto alle limitazioni dell'API dati. Per ulteriori informazioni, consulta the section called "Limitazioni".

Per eseguire una query nell'editor della query

1. Accedi a Console di gestione AWS e apri la console Amazon RDS all'indirizzo https://console.aws.amazon.com/rds/.
2. Nell'angolo superiore destro diConsole di gestione AWS, scegli il cluster Regione AWS in cui hai creato i cluster Aurora DB su cui desideri interrogare.
3. Nel pannello di navigazione, seleziona Database.
4. Scegliere il cluster di database Aurora su cui eseguire le query SQL.
5. In Actions (Operazioni), scegliere Query. Se è la prima volta che si esegue la connessione al database, viene visualizzata la pagina Connect to database (Connetti al database).



6. Immetti le seguenti informazioni:

   a. Per l'istanza o il cluster di database, scegliere il cluster di database Aurora su cui eseguire le query SQL.

   b. In Database username (Nome utente del database), scegliere il nome utente dell'utente del database a cui connettersi o selezionare Add new database credentials (Aggiungi nuove credenziali del database. Se si sceglie Add new database credentials (Aggiungi nuove credenziali del database), immettere il nome utente per le nuove credenziali del database in Enter database username (Immetti il nome utente database).

   c. In Enter database password (Immetti la password database), immettere la password per l'utente del database scelto.

   d. Nell'ultima casella, immettere il nome del database o dello schema da utilizzare per il cluster database Aurora.

   e. Scegliere Connect to database (Connetti al database).

> **Note**
>
> Se la connessione ha esito positivo, le informazioni relative a connessione e autenticazione vengono archiviate in Gestione dei segreti AWS. Non è necessario inserire nuovamente le informazioni di connessione.

7. Nell'editor della query, immettere la query SQL che si desidera eseguire sul database.



Ogni istruzione SQL è sottoposta automaticamente a commit o è possibile eseguire istruzioni SQL in uno script come parte di una transazione. Per controllare questo comportamento, scegliere l'icona a forma di ingranaggio sopra la finestra di query.

Viene visualizzata la finestra Query Editor Settings (Impostazioni editor della query).



Se si sceglie Auto-commit (Commit automatico), viene eseguito automaticamente il commit di ogni istruzione SQL. Se si sceglie Transazione, è possibile eseguire un gruppo di istruzioni in uno script. Alla fine dello script viene eseguito automaticamente il commit delle istruzioni a meno che prima non sia stato eseguito esplicitamente il commit o il rollback. Inoltre, è possibile scegliere di interrompere uno script in esecuzione se si verifica un errore abilitando Stop on error (Interrompi in caso di errore).

> **Note**
>
> In un gruppo di istruzioni, le istruzioni DDL (Data Definition Language) possono causare il commit di istruzioni DML (Data Manipulation Language) precedenti. È possibile anche includere istruzioni `COMMIT` e `ROLLBACK` in un gruppo di istruzioni in uno script.

Dopo aver effettuato le scelte nella finestra Query Editor Settings (Impostazioni editor della query), scegliere Save (Salva).

8. Scegliere Run (Esegui) o premere Ctrl+Invio. L'editor della query visualizza i risultati della query.

   Dopo aver eseguito la query, salvarla in Saved queries (Query salvate) scegliendo Save As (Salva con nome).

   Esportare i risultati della query in formato foglio di calcolo scegliendo Export to csv (Esporta in formato .csv).

Puoi trovare, modificare ed eseguire nuovamente query precedenti. A tale scopo, seleziona la scheda Recent (Recente) o Saved queries (Query salvate), scegli il testo della query, quindi seleziona Run (Esegui).

Per cambiare il database, scegli Change database (Cambia database).

Riferimento all'API DBQMS (Database Query Metadata Service)

Il Database Query Metadata Service (`dbqms`) è un servizio solo interno. Fornisce le query recenti e salvate per l'editor di query nella Console di gestione AWS per più servizi AWS, tra cui Amazon RDS.

Operazioni DBQMS supportate

- CreateFavoriteQuery
- CreateQueryHistory
- CreateTab
- DeleteFavoriteQueries
- DeleteQueryHistory
- DeleteTab

- DescribeFavoriteQueries
- DescribeQueryHistory
- DescribeTabs
- GetQueryString
- UpdateFavoriteQuery
- UpdateQueryHistory
- UpdateTab

CreateFavoriteQuery

Salva una nuova query preferita. Ogni utente può creare fino a 1000 query salvate. Questo limite è soggetto a modifiche in futuro.

CreateQueryHistory

Salva una nuova voce della cronologia delle query.

CreateTab

Salva una nuova scheda della query. Ogni utente può creare fino a 10 schede di query.

DeleteFavoriteQueries

Elimina una o più query salvate.

DeleteQueryHistory

Elimina le voci della cronologia delle query.

DeleteTab

Elimina le voci della scheda di query.

DescribeFavoriteQueries

Elenca le query salvate create da un utente in un determinato account.

DescribeQueryHistory

Elenca le voci della cronologia delle query.

DescribeTabs

Elenca le schede delle query create da un utente in un determinato account.

GetQueryString

Recupera il testo completo della query da un ID di query.

UpdateFavoriteQuery

Aggiorna la stringa di query, la descrizione, il nome o la data di scadenza.

UpdateQueryHistory

Aggiorna lo stato della cronologia delle query.

UpdateTab

Aggiorna il nome della scheda di query e la stringa di query.

Utilizzo di machine learning di Amazon Aurora

Con il machine learning Amazon Aurora, è possibile integrare il cluster di database Aurora con uno dei seguenti servizi di machine learning AWS, a seconda delle esigenze. Ciascuno supporta casi d'uso di machine learning specifici.

Amazon Bedrock

Amazon Bedrock è un servizio completamente gestito che rende disponibili i principali modelli di fondazione delle aziende di intelligenza artificiale tramite un'API, insieme a strumenti per sviluppatori per creare e scalare applicazioni di IA generativa. Con Amazon Bedrock, paghi per eseguire inferenze su qualsiasi modello di fondazione di terze parti. I prezzi si basano sul volume dei token di input e di output, nonché sull'eventuale acquisto o meno della velocità di trasmissione effettiva assegnata per il modello. Per ulteriori informazioni, consulta Che cos'è Amazon Bedrock? nella Guida per l'utente di Amazon Bedrock

Amazon Comprehend

Amazon Comprehend è un servizio gestito di elaborazione del linguaggio naturale che viene utilizzato per estrarre informazioni dai documenti. Con Amazon Comprehend, puoi ottenere il sentiment in base al contenuto dei documenti, analizzando entità, frasi chiave, lingua e altre funzionalità. Per ulteriori informazioni, consultare Che cos'è Amazon Comprehend? nella Guida per gli sviluppatori di Amazon Comprehend.

SageMaker AI

Amazon SageMaker AI è un servizio di machine learning completamente gestito. I data scientist e gli sviluppatori utilizzano Amazon SageMaker AI per creare, eseguire il training e provare modelli di machine learning per un'ampia gamma di attività di inferenza, come il rilevamento delle frodi e i consigli sui prodotti. Quando un modello di machine learning è pronto per l'utilizzo in produzione, può essere distribuito nell'ambiente ospitato di Amazon SageMaker AI. Per ulteriori informazioni, consulta Che cos'è Amazon SageMaker AI? nella Guida per gli sviluppatori di Amazon SageMaker AI.

L'utilizzo di Amazon Comprehend con il cluster di database Aurora comporta meno operazioni di configurazione preliminari rispetto a SageMaker AI. Se non hai esperienza con il machine learning di AWS, ti consigliamo di iniziare utilizzando Amazon Comprehend.

Argomenti

- Utilizzo di machine learning di Amazon Aurora con Aurora MySQL
- Utilizzo del machine learning di Amazon Aurora con Aurora PostgreSQL

Utilizzo di machine learning di Amazon Aurora con Aurora MySQL

L'utilizzo del machine learning di Amazon Aurora con il cluster di database Aurora MySQL, consente di utilizzare Amazon Bedrock, Amazon Comprehend o Amazon SageMaker AI, a seconda delle esigenze. Ciascuno di essi supporta casi d'uso di machine learning diversi.

Indice

- Requisiti per l'utilizzo di machine learning di Aurora con Aurora MySQL
- Disponibilità di regioni e versioni
- Funzionalità supportate e limitazioni del machine learning di Aurora con Aurora MySQL
- Configurazione del cluster database Aurora per utilizzare machine learning di Aurora
  - Configurazione del cluster di database Aurora MySQL per utilizzare Amazon Bedrock
  - Configurazione del cluster database Aurora MySQL per utilizzare Amazon Comprehend
  - Configurazione del cluster di database Aurora MySQL per utilizzare SageMaker AI
    - Configurazione del cluster di database Aurora MySQL per utilizzare Amazon S3 for SageMaker AI (facoltativo)
  - Concessione agli utenti del database dell'accesso a machine learning di Aurora
    - Concessione dell'accesso alle funzioni di Amazon Bedrock
    - Concessione dell'accesso alle funzioni Amazon Comprehend
    - Concessione dell'accesso alle funzioni di SageMaker AI
- Utilizzo di Amazon Bedrock con il cluster di database Aurora MySQL
- Utilizzo di Amazon Comprehend con il cluster database Aurora MySQL
- Utilizzo di SageMaker AI con il cluster di database Aurora MySQL
  - Requisito del set di caratteri per le funzioni di SageMaker AI che restituiscono stringhe
  - Esportazione dei dati in Amazon S3 per l'addestramento dei modelli SageMaker AI (avanzato)
- Considerazioni sulle prestazioni per l'utilizzo del machine learning di Aurora con Aurora MySQL
  - Modello e prompt
  - Cache delle query
  - Ottimizzazione batch per le chiamate di funzione Aurora Machine Learning

- Monitoraggio del machine learning di Aurora

Requisiti per l'utilizzo di machine learning di Aurora con Aurora MySQL

I servizi di machine learning di AWS sono servizi gestiti che vengono impostati ed eseguiti negli ambienti di produzione. Il machine learning di Aurora supporta l'integrazione con Amazon Bedrock, Amazon Comprehend e SageMaker AI. Prima di provare a configurare il cluster database Aurora MySQL per utilizzare il machine learning di Aurora, occorre accertarsi di aver compreso i requisiti e prerequisiti riportati di seguito.

- I servizi di machine learning devono essere in esecuzione nella stessa Regione AWS del cluster di database Aurora MySQL. Non è possibile utilizzare i servizi di machine learning da un cluster di database Aurora MySQL in una Regione diversa.
- Se il cluster di database Aurora MySQL si trova in un cloud pubblico virtuale (VPC) diverso dai servizi Amazon Bedrock, Amazon Comprehend o SageMaker AI, il gruppo di sicurezza del VPC deve consentire connessioni in uscita al servizio di machine learning di Aurora di destinazione. Per ulteriori informazioni, consulta Controlla il traffico verso le tue risorse AWS utilizzando gruppi di sicurezza nella Amazon VPC User Guide.
- È possibile aggiornare un cluster Aurora che esegue una versione precedente di Aurora MySQL a una versione più recente se si desidera utilizzare Aurora machine learning con tale cluster. Per ulteriori informazioni, consulta Aggiornamenti del motore del database per Amazon Aurora MySQL.
- Il cluster di database Aurora MySQL deve utilizzare un gruppo di parametri cluster di database personalizzato. Al termine del processo di configurazione per ogni servizio di machine learning di Aurora che desideri utilizzare, aggiungi il nome della risorsa Amazon (ARN) del ruolo IAM associato che è stato creato per il servizio. Ti consigliamo di creare in anticipo un gruppo di parametri cluster database personalizzato per Aurora MySQL e di configurare il cluster database Aurora MySQL per utilizzarlo in modo che sia pronto per la modifica alla fine del processo di configurazione.
- Per SageMaker AI:
  - I componenti di machine learning da utilizzare per le inferenze devono essere configurati e pronti all'uso. Durante il processo di configurazione per il cluster di database Aurora MySQL, è necessario disporre dell'ARN dell'endpoint SageMaker AI. I data scientist del tuo team sono probabilmente le persone più preparate per gestire l'utilizzo di SageMaker AI in modo da preparare i modelli ed eseguire le altre attività correlate. Per iniziare a utilizzare Amazon SageMaker AI, consulta Nozioni di base su Amazon SageMaker. Per ulteriori informazioni su inferenze ed endpoint, consulta Real-time inference (Inferenza in tempo reale).

- Per utilizzare SageMaker AI con i tuoi dati di addestramento, occorre configurare un bucket Amazon S3 come parte della configurazione di Aurora MySQL per il machine learning di Aurora. A questo scopo, si segue lo stesso processo generale di configurazione dell'integrazione SageMaker AI. Per un riepilogo di questo processo di configurazione facoltativo, consultare Configurazione del cluster di database Aurora MySQL per utilizzare Amazon S3 for SageMaker AI (facoltativo).

- Per i database globali Aurora, occorre configurare i servizi di machine learning di Aurora che si desidera utilizzare in tutte le Regioni AWS che costituiscono il database globale Aurora. Ad esempio, se desideri utilizzare il machine learning di Aurora con SageMaker AI per il database globale Aurora, procedi come indicato di seguito per ciascun cluster di database Aurora MySQL in ogni Regione AWS:
  - Configura i servizi Amazon SageMaker AI con gli stessi endpoint e modelli di addestramento di SageMaker AI. Anche questi devono utilizzare gli stessi nomi.
  - Crea i ruoli IAM come descritto in Configurazione del cluster database Aurora per utilizzare machine learning di Aurora.
  - Aggiungi l'ARN del ruolo IAM al gruppo di parametri cluster database Aurora per ciascun cluster database Aurora MySQL in ogni Regione AWS.

  Queste attività richiedono che il machine learning di Aurora sia disponibile per la versione di Aurora MySQL in tutte le Regioni AWS che costituiscono il database globale Aurora.

Disponibilità di regioni e versioni

La disponibilità e il supporto della funzionalità varia tra le versioni specifiche di ciascun motore di database Aurora e tra Regioni AWS.

- Per informazioni sulla disponibilità di versioni e Regioni per Amazon Comprehend e Amazon SageMaker AI con Aurora MySQL, consulta Utilizzo di machine learning di Aurora con Aurora MySQL.
- Amazon Bedrock è supportato solo su Aurora MySQL 3.06 e versioni successive.

  Per informazioni sulla disponibilità delle Regioni per Amazon Bedrock, consulta Model support by Regione AWS nella Amazon Bedrock User Guide.

Funzionalità supportate e limitazioni del machine learning di Aurora con Aurora MySQL

Quando utilizzi Aurora MySQL con il machine learning di Aurora, si applicano le seguenti limitazioni:

- L'estensione per il machine learning di Aurora non supporta le interfacce vettoriali.
- L'utilizzo delle integrazioni del machine learning di Aurora all'interno di un trigger non è supportato.
- Le funzioni di machine learning di Aurora non sono compatibili con la replica della registrazione di log binari (binlog).
  - L'impostazione `--binlog-format=STATEMENT` genera un'eccezione per le chiamate alle funzioni Aurora machine learning.
  - Le funzioni di machine Learning di Aurora sono non deterministiche e le funzioni archiviate non deterministiche non sono compatibili con il formato binlog.

  Per ulteriori informazioni, consulta Binary Logging Formats nella documentazione di MySQL.
- Le funzioni archiviate che chiamano tabelle con colonne sempre generate non sono supportate. Questo vale per qualsiasi funzione archiviata di Aurora MySQL. Per ulteriori informazioni su questo tipo di colonna, consultare CREATE TABLE and Generated Columns nella documentazione di MySQL.
- Le funzioni di Amazon Bedrock non supportano `RETURNS JSON`. Se necessario, puoi utilizzare `CONVERT` o `CAST` per eseguire la conversione da `TEXT` a `JSON`.
- Amazon Bedrock non supporta richieste batch.
- Il machine learning di Aurora MYSQL supporta qualsiasi endpoint SageMaker AI che legga e scriva in formato CSV (valori separati da virgole) impostando `ContentType` su `text/csv`. Attualmente, questo formato è accettato dai seguenti algoritmi SageMaker AI integrati:
  - Linear Learner
  - Random Cut Forest
  - XGBoost

  Per ulteriori informazioni su questi algoritmi, consulta Scelta di un algoritmo nella Guida per gli sviluppatori di Amazon SageMaker AI.

Configurazione del cluster database Aurora per utilizzare machine learning di Aurora

Nei seguenti argomenti sono illustrate le procedure di configurazione separate per ciascuno di questi servizi di machine learning di Aurora.

Argomenti

- Configurazione del cluster di database Aurora MySQL per utilizzare Amazon Bedrock
- Configurazione del cluster database Aurora MySQL per utilizzare Amazon Comprehend
- Configurazione del cluster di database Aurora MySQL per utilizzare SageMaker AI
  - Configurazione del cluster di database Aurora MySQL per utilizzare Amazon S3 for SageMaker AI (facoltativo)
- Concessione agli utenti del database dell'accesso a machine learning di Aurora
  - Concessione dell’accesso alle funzioni di Amazon Bedrock
  - Concessione dell'accesso alle funzioni Amazon Comprehend
  - Concessione dell’accesso alle funzioni di SageMaker AI

Configurazione del cluster di database Aurora MySQL per utilizzare Amazon Bedrock

Il machine learning di Aurora si basa su policy e ruoli (IAM) AWS Identity and Access Management per consentire al cluster di database Aurora MySQL di accedere ai servizi Amazon Bedrock e utilizzarli. Le seguenti procedure creano un ruolo e una policy di autorizzazione IAM per consentire al cluster di database di integrarsi con Amazon Bedrock.

Per creare la policy IAM

1. Accedi a Console di gestione AWS e apri la console IAM all'indirizzo https://console.aws.amazon.com/iam/.
2. Nel riquadro di navigazione, scegli Policy.
3. Scegliere Create a policy (Crea una policy).
4. Nella pagina Specifica le autorizzazioni, in Seleziona un servizio scegli Bedrock.

   Vengono visualizzate le autorizzazioni di Amazon Bedrock.
5. Espandi Lettura, quindi seleziona InvokeModel.
6. In Risorse, seleziona Tutte.

La pagina Specifica le autorizzazioni dovrebbe essere simile alla figura seguente.



7. Scegli Next (Successivo).
8. Nella pagina Rivedi e crea, inserisci un nome per la policy, ad esempio **`BedrockInvokeModel`**.

9. Esamina la policy, quindi scegli Crea policy.

A questo punto, crea il ruolo IAM che utilizza la policy di autorizzazione di Amazon Bedrock.

Per creare il ruolo IAM

1. Accedi a Console di gestione AWS e apri la console IAM all'indirizzo https://console.aws.amazon.com/iam/.
2. Nel riquadro di navigazione scegliere Roles (Ruoli).
3. Scegli Crea ruolo.
4. Nella pagina Seleziona un'entità attendibile, in Caso d'uso scegli RDS.
5. Seleziona RDS - Aggiungi ruolo al database, quindi scegli Avanti.
6. Nella pagina Aggiungi autorizzazioni, in Policy di autorizzazione, seleziona la policy IAM che hai creato, quindi scegli Avanti.
7. Nella pagina Nomina, verifica e crea, inserisci un nome per il ruolo, ad esempio **`ams-bedrock-invoke-model-role`**.

   Il ruolo dovrebbe essere simile alla figura seguente.



8. Verifica il ruolo, quindi scegli Crea ruolo.

Associa quindi il ruolo IAM Amazon Bedrock al cluster di database.

Per associare il ruolo IAM al tuo cluster di database

1. Accedi alla Console di gestione AWS e apri la console Amazon RDS all'indirizzo https://console.aws.amazon.com/rds/.
2. Nel riquadro di navigazione seleziona Database.
3. Scegli il cluster di database Aurora MySQL da collegare ai servizi Amazon Bedrock.
4. Scegliere la scheda Connectivity & security (Connettività e sicurezza).
5. Nella sezione Gestisci ruoli IAM, scegli Seleziona IAM da aggiungere a questo cluster.
6. Scegli l'IAM che hai creato, quindi seleziona Aggiungi ruolo.

   Il ruolo IAM viene associato al tuo cluster di database, prima con lo stato In attesa, quindi con lo stato Attivo. Al termine del processo, il ruolo è disponibile nell'elenco Ruoli IAM attuali per questo cluster.



È necessario impostare l'ARN di questo ruolo IAM sul parametro `aws_default_bedrock_role` del parametro del cluster di database personalizzato associato al cluster di database Aurora MySQL. Se il cluster database Aurora MySQL non utilizza un gruppo di parametri cluster database personalizzato, è necessario crearne uno da utilizzare con il cluster database Aurora MySQL per completare l'integrazione. Per ulteriori informazioni, consulta Gruppi di parametri del cluster di database per i cluster di database Amazon Aurora.

Per configurare i parametri del cluster di database

1. Nella console Amazon RDS, apri la scheda Configurazione del cluster database Aurora MySQL.

2. Individua il gruppo di parametri del cluster di database configurato per il cluster. Scegli il collegamento per aprire il gruppo di parametri del cluster di database personalizzato, quindi seleziona Modifica.
3. Individua il parametro `aws_default_bedrock_role` nel gruppo di parametri cluster database personalizzato.
4. Nel campo Valore, immetti l'ARN del ruolo IAM.
5. Scegli Salva modifiche per salvare l'impostazione.
6. Riavvia l'istanza primaria del cluster database Aurora MySQL per rendere effettiva questa impostazione del parametro.

L'integrazione con IAM per Amazon Bedrock è stata completata. Continua a configurare il cluster di database Aurora MySQL in modo che funzioni con Amazon Bedrock in base ai Concessione agli utenti del database dell'accesso a machine learning di Aurora.

Configurazione del cluster database Aurora MySQL per utilizzare Amazon Comprehend

Machine learning di Aurora si basa su ruoli e policy AWS Identity and Access Management per consentire al cluster database Aurora MySQL di accedere e utilizzare i servizi Amazon Comprehend. La seguente procedura crea automaticamente un ruolo e una policy IAM per il cluster in modo che possa utilizzare Amazon Comprehend.

Come configurare il cluster database Aurora MySQL per utilizzare Amazon Comprehend

1. Accedi alla Console di gestione AWS e apri la console Amazon RDS all'indirizzo https://console.aws.amazon.com/rds/.
2. Nel riquadro di navigazione seleziona Database.
3. Scegli il cluster di database Aurora MySQL da collegare ai servizi Amazon Comprehend.
4. Scegliere la scheda Connectivity & security (Connettività e sicurezza).
5. Scorri fino alla sezione Gestisci ruoli IAM, quindi scegli Seleziona un servizio per connetterti a questo cluster.
6. Scegli Amazon Comprehend dal menu, quindi seleziona Connetti un servizio.



7. La finestra di dialogo Connetti il cluster ad Amazon Comprehend non richiede informazioni aggiuntive. Tuttavia, è possibile che venga visualizzato un messaggio che segnala che l'integrazione tra Aurora e Amazon Comprehend è al momento è in fase di anteprima. Assicurati di leggere il messaggio prima di continuare. Se preferisci non continuare, puoi scegliere Annulla.
8. Scegli Connetti un servizio per completare il processo di integrazione.

   Aurora crea il ruolo IAM. Inoltre, crea la policy che consente al cluster di database Aurora MySQL di utilizzare i servizi Amazon Comprehend e collegare la policy al ruolo. Al termine del processo, il ruolo è disponibile nell'elenco Ruoli IAM attuali per questo cluster, come illustrato nell'immagine seguente.



   È necessario impostare l'ARN di questo ruolo IAM sul parametro `aws_default_comprehend_role` del parametro del cluster di database personalizzato associato al cluster di database Aurora MySQL. Se il cluster database Aurora MySQL non utilizza un gruppo di parametri cluster database personalizzato, è necessario crearne uno da utilizzare con il cluster database Aurora MySQL per completare l'integrazione. Per ulteriori informazioni, consulta Gruppi di parametri del cluster di database per i cluster di database Amazon Aurora.

Dopo aver creato il gruppo di parametri cluster database personalizzato e averlo associato al cluster database Aurora MySQL, puoi continuare a seguire questi passaggi.

Se il cluster utilizza un gruppo di parametri cluster database personalizzato, procedi come descritto di seguito.

a. Nella console Amazon RDS, apri la scheda Configurazione del cluster database Aurora MySQL.
b. Individua il gruppo di parametri del cluster di database configurato per il cluster. Scegli il collegamento per aprire il gruppo di parametri del cluster di database personalizzato, quindi seleziona Modifica.
c. Individua il parametro `aws_default_comprehend_role` nel gruppo di parametri cluster database personalizzato.
d. Nel campo Valore, immetti l'ARN del ruolo IAM.
e. Scegli Salva modifiche per salvare l'impostazione. Nell'immagine seguente, è disponibile un esempio.



Riavvia l'istanza primaria del cluster database Aurora MySQL per rendere effettiva questa impostazione del parametro.

L'integrazione con IAM per Amazon Comprehend è stata completata. Continua a configurare il cluster database Aurora MySQL per funzionare con Amazon Comprehend concedendo l'accesso agli utenti del database appropriati.

Configurazione del cluster di database Aurora MySQL per utilizzare SageMaker AI

La seguente procedura crea automaticamente la policy e il ruolo IAM per il cluster di database Aurora MySQL in modo da poter utilizzare SageMaker AI. Prima di provare a seguire questa procedura, occorre assicurarsi che l'endpoint SageMaker AI sia disponibile, in modo da poterlo inserire quando necessario. In genere, i data scientist del team eseguono il lavoro per produrre un endpoint che è possibile utilizzare dal cluster database Aurora MySQL. Tali endpoint sono disponibili nella console SageMaker AI. Nel riquadro di navigazione, apri il menu Inferenza e scegli Endpoint. Nell'immagine seguente, è disponibile un esempio.



Per configurare il cluster di database Aurora MySQL per utilizzare SageMaker AI

1. Accedi alla Console di gestione AWS e apri la console Amazon RDS all'indirizzo https://console.aws.amazon.com/rds/.
2. Scegli Database dal menu di navigazione di Amazon RDS e quindi seleziona il cluster di database Aurora MySQL da collegare ai servizi SageMaker AI.
3. Scegliere la scheda Connectivity & security (Connettività e sicurezza).
4. Scorri fino alla sezione Gestisci ruoli IAM, quindi scegli Seleziona un servizio per connetterti a questo cluster. Scegli SageMaker AI dal selettore.



5. Scegliere Connect service (Connetti servizio).
6. Nella finestra di dialogo Connetti il cluster a SageMaker AI, inserisci l'ARN dell'endpoint SageMaker AI.



7. Aurora crea il ruolo IAM. Inoltre, crea la policy che consente al cluster di database Aurora MySQL di utilizzare i servizi SageMaker AI e collegare la policy al ruolo. Al termine del processo, il ruolo è disponibile nell'elenco Ruoli IAM attuali per questo cluster.
8. Apri la console IAM all'indirizzo https://console.aws.amazon.com/iam/.
9. Scegli Ruoli dalla sezione Gestione degli accessi del menu di navigazione AWS Identity and Access Management.
10. Trova il ruolo tra quelli elencati. Il suo nome utilizza il seguente schema.

```
rds-sagemaker-your-cluster-name-role-auto-generated-digits
```

11. Apri la pagina di riepilogo del ruolo e individua l'ARN. Prendi nota dell'ARN o copialo utilizzando il widget copy.
12. Apri la console di Amazon RDS all'indirizzo https://console.aws.amazon.com/rds/.
13. Scegli il cluster database Aurora MySQL, quindi seleziona la sua scheda Configurazione.
14. Individua il gruppo di parametri cluster database e scegli il collegamento per aprire il gruppo di parametri cluster database personalizzato. Trova il parametro `aws_default_sagemaker_role` e immetti l'ARN del ruolo IAM nel campo Valore, quindi salva l'impostazione.
15. Riavvia l'istanza primaria del cluster database Aurora MySQL per rendere effettiva questa impostazione del parametro.

La configurazione di IAM è ora completata. Continua a configurare il cluster di database Aurora MySQL in modo che funzioni con SageMaker AI concedendo l'accesso agli utenti del database appropriati.

Se per l'addestramento desideri utilizzare i tuoi modelli SageMaker AI anziché i componenti predefiniti di SageMaker AI, devi aggiungere al cluster di database Aurora MySQL anche il bucket Amazon S3, come descritto nella sezione Configurazione del cluster di database Aurora MySQL per utilizzare Amazon S3 for SageMaker AI (facoltativo) di seguito.

Configurazione del cluster di database Aurora MySQL per utilizzare Amazon S3 for SageMaker AI (facoltativo)

Per utilizzare SageMaker AI con i tuoi modelli anziché utilizzare i componenti predefiniti forniti da SageMaker AI, devi configurare un bucket Amazon S3 per il cluster di database Aurora MySQL da utilizzare. Per ulteriori informazioni sulla creazione di un bucket Amazon S3, consultaCreazione di un bucketnellaGuida all'utente di Amazon Simple Storage Service.

Per configurare il cluster di database Aurora MySQL per utilizzare un bucket Amazon S3 per SageMaker AI

1. Accedi alla Console di gestione AWS e apri la console Amazon RDS all'indirizzo https://console.aws.amazon.com/rds/.
2. Scegli Database dal menu di navigazione di Amazon RDS e quindi seleziona il cluster di database Aurora MySQL da collegare ai servizi SageMaker AI.

3. Scegliere la scheda Connectivity & security (Connettività e sicurezza).
4. Scorri fino alla sezione Gestisci ruoli IAM, quindi scegli Seleziona un servizio per connetterti a questo cluster. Scegli Amazon S3 dal selettore.
5. Scegliere Connect service (Connetti servizio).
6. Nella finestra di dialogo Connetti il cluster ad Amazon S3, inserisci l'ARN del bucket Amazon S3, come mostrato nell'immagine seguente.



7. Scegli Connetti un servizio per completare questo processo.

Per ulteriori informazioni sull'utilizzo di bucket Amazon S3 con SageMaker AI, consulta Specify an Amazon S3 Bucket to Upload Training Datasets and Store Output Data nella Guida per gli sviluppatori di Amazon SageMaker AI. Per ulteriori informazioni su come utilizzare SageMaker AI, consulta Get Started with Amazon SageMaker AI Notebook Instances nella Guida per gli sviluppatori di Amazon SageMaker AI.

Concessione agli utenti del database dell'accesso a machine learning di Aurora

Per invocare le funzioni di machine learning di Aurora, gli utenti del database devono disporre dell'autorizzazione appropriata. Il modo in cui l'autorizzazione viene concessa dipende dalla versione di MySQL utilizzata per il cluster database Aurora MySQL, come descritto di seguito. L'operazione utilizzata dipende dalla versione di MySQL utilizzata dal cluster database Aurora MySQL.

- Per Aurora MySQL versione 3 (compatibile con MySQL 8.0), gli utenti del database devono disporre del ruolo del database appropriato. Per ulteriori informazioni, consulta Utilizzo di ruoli nel Manuale di riferimento di MySQL 8.0.

- Per Aurora MySQL versione 2 (compatibile con MySQL 5.7), gli utenti del database devono disporre di privilegi. Per ulteriori informazioni, consulta Access Control and Account Management nel Manuale di riferimento di MySQL 5.7.

Nella tabella seguente sono raffigurati i ruoli e i privilegi necessari agli utenti del database per utilizzare le funzioni di machine learning.

Aurora MySQL versione 3 (ruolo)	Aurora MySQL versione 2 (privilegio)
AWS_BEDROCK_ACCESS	–
AWS_COMPREHEND_ACCESS	INVOKE COMPREHEND
AWS_SAGEMAKER_ACCESS	INVOKE SAGEMAKER

Concessione dell'accesso alle funzioni di Amazon Bedrock

Per fornire agli utenti del database l'accesso alle funzioni di Amazon Bedrock, utilizza la seguente istruzione SQL:

```
GRANT AWS_BEDROCK_ACCESS TO user@domain-or-ip-address;
```

Per utilizzare Amazon Bedrock, gli utenti del database devono inoltre disporre di autorizzazioni `EXECUTE` per le funzioni che hai creato.

```
GRANT EXECUTE ON FUNCTION database_name.function_name TO user@domain-or-ip-address;
```

Infine, i ruoli degli utenti del database devono essere impostati su `AWS_BEDROCK_ACCESS`:

```
SET ROLE AWS_BEDROCK_ACCESS;
```

Le funzioni di Amazon Bedrock sono ora disponibili per l'uso.

Concessione dell'accesso alle funzioni Amazon Comprehend

Per fornire agli utenti del database l'accesso alle funzioni di Amazon Comprehend, utilizza l'istruzione appropriata per la versione di Aurora MySQL.

- Aurora MySQL versione 3 (compatibile con MySQL 8.0)

```
GRANT AWS_COMPREHEND_ACCESS TO user@domain-or-ip-address;
```

- Aurora MySQL versione 2 (compatibile con MySQL 5.7)

```
GRANT INVOKE COMPREHEND ON *.* TO user@domain-or-ip-address;
```

Le funzioni di Amazon Comprehend sono ora disponibili per l'uso. Per esempi di utilizzo, consultare Utilizzo di Amazon Comprehend con il cluster database Aurora MySQL.

Concessione dell'accesso alle funzioni di SageMaker AI

Per fornire agli utenti del database l'accesso alle funzioni di SageMaker AI, utilizza l'istruzione appropriata per la versione di Aurora MySQL.

- Aurora MySQL versione 3 (compatibile con MySQL 8.0)

```
GRANT AWS_SAGEMAKER_ACCESS TO user@domain-or-ip-address;
```

- Aurora MySQL versione 2 (compatibile con MySQL 5.7)

```
GRANT INVOKE SAGEMAKER ON *.* TO user@domain-or-ip-address;
```

Per utilizzare SageMaker AI, gli utenti del database devono inoltre disporre di autorizzazioni `EXECUTE` per le funzioni che hai creato. Supponi di aver creato due funzioni, `db1.anomoly_score` e `db2.company_forecasts`, per invocare i servizi del tuo endpoint SageMaker AI. Concedi i privilegi di esecuzione come illustrato nel seguente esempio.

```
GRANT EXECUTE ON FUNCTION db1.anomaly_score TO user1@domain-or-ip-address1;
GRANT EXECUTE ON FUNCTION db2.company_forecasts TO user2@domain-or-ip-address2;
```

Le funzioni di SageMaker AI sono ora disponibili per l'uso. Per esempi di utilizzo, consultare Utilizzo di SageMaker AI con il cluster di database Aurora MySQL.

Utilizzo di Amazon Bedrock con il cluster di database Aurora MySQL

Per utilizzare Amazon Bedrock, crea nel database Aurora MySQL una funzione definita dall'utente (UDF) che invochi un modello. Per ulteriori informazioni, consulta Modelli supportati in Amazon Bedrock nella Guida per l'utente di Amazon Bedrock.

Le UDF utilizzano la sintassi seguente:

```
CREATE FUNCTION function_name (argument type)
        [DEFINER = user]
    RETURNS mysql_data_type
    [SQL SECURITY {DEFINER | INVOKER}]
    ALIAS AWS_BEDROCK_INVOKE_MODEL
    MODEL ID 'model_id'
    [CONTENT_TYPE 'content_type']
    [ACCEPT 'content_type']
    [TIMEOUT_MS timeout_in_milliseconds];
```

- Le funzioni di Amazon Bedrock non supportano `RETURNS JSON`. Se necessario, puoi utilizzare `CONVERT` o `CAST` per eseguire la conversione da `TEXT` a `JSON`.
- Se non specifichi `CONTENT_TYPE` o `ACCEPT`, il valore predefinito è `application/json`.
- Se non specifichi `TIMEOUT_MS`, il valore predefinito è `aurora_ml_inference_timeout`.

Ad esempio, la seguente UDF invoca il modello Amazon Titan Text Express:

```
CREATE FUNCTION invoke_titan (request_body TEXT)
    RETURNS TEXT
    ALIAS AWS_BEDROCK_INVOKE_MODEL
    MODEL ID 'amazon.titan-text-express-v1'
    CONTENT_TYPE 'application/json'
    ACCEPT 'application/json';
```

Per consentire a un utente del database di utilizzare questa funzione, ricorri al seguente comando SQL:

```
GRANT EXECUTE ON FUNCTION database_name.invoke_titan TO user@domain-or-ip-address;
```

Quindi l'utente può chiamare `invoke_titan` come qualsiasi altra funzione, come mostrato nell'esempio seguente. Assicurati di formattare il corpo della richiesta in base ai modelli di testo di Amazon Titan.

```
CREATE TABLE prompts (request varchar(1024));
INSERT INTO prompts VALUES (
'{
    "inputText": "Generate synthetic data for daily product sales in various categories
 - include row number, product name, category, date of sale and price. Produce output
 in JSON format. Count records and ensure there are no more than 5.",
    "textGenerationConfig": {
        "maxTokenCount": 1024,
        "stopSequences": [],
        "temperature":0,
        "topP":1
    }
}');

SELECT invoke_titan(request) FROM prompts;

{"inputTextTokenCount":44,"results":[{"tokenCount":296,"outputText":"
```tabular-data-json
{
    "rows": [
        {
            "Row Number": "1",
            "Product Name": "T-Shirt",
            "Category": "Clothing",
            "Date of Sale": "2024-01-01",
            "Price": "$20"
        },
        {
            "Row Number": "2",
            "Product Name": "Jeans",
            "Category": "Clothing",
            "Date of Sale": "2024-01-02",
            "Price": "$30"
        },
        {
            "Row Number": "3",
            "Product Name": "Hat",
            "Category": "Accessories",
            "Date of Sale": "2024-01-03",
```
```

```
            "Price": "$15"
        },
        {
            "Row Number": "4",
            "Product Name": "Watch",
            "Category": "Accessories",
            "Date of Sale": "2024-01-04",
            "Price": "$40"
        },
        {
            "Row Number": "5",
            "Product Name": "Phone Case",
            "Category": "Accessories",
            "Date of Sale": "2024-01-05",
            "Price": "$25"
        }
    ]
}
```","completionReason":"FINISH"}]}

Per gli altri modelli che utilizzi, assicurati di formattare il corpo della richiesta in modo appropriato. Per ulteriori informazioni, consulta Parametri di inferenza per modelli di fondazione nella Guida per l'utente di Amazon Bedrock.

Utilizzo di Amazon Comprehend con il cluster database Aurora MySQL

Per Aurora MySQL, il machine learning di Aurora fornisce le due funzioni integrate seguenti per l'utilizzo con Amazon Comprehend e i dati di testo. Viene fornito il testo da analizzare (`input_data`) e specificata la lingua (`language_code`).

aws_comprehend_detect_sentiment

Questa funzione identifica il testo come avente un assetto emotivo positivo, negativo, neutro o misto. La documentazione di riferimento di questa funzione è la seguente.

```
aws_comprehend_detect_sentiment(
  input_text,
  language_code
  [,max_batch_size]
)
```

Per ulteriori informazioni, consultare Sentiment nella Guida per gli sviluppatori di Amazon Comprehend.

aws_comprehend_detect_sentiment_confidence

Questa funzione misura il livello di affidabilità del sentiment rilevato per un determinato testo. Restituisce un valore (tipo, `double`) che indica l'affidabilità del sentiment assegnato dalla funzione aws_comprehend_detect_sentiment al testo. L'affidabilità è un parametro statistico compreso tra 0 e 1. Più alto è il livello di affidabilità, maggiore è il peso che è possibile assegnare al risultato. Di seguito è riportato un riepilogo della documentazione della funzione.

```
aws_comprehend_detect_sentiment_confidence(
  input_text,
  language_code
  [,max_batch_size]
)
```

In entrambe le funzioni (aws_comprehend_detect_sentiment_confidence, aws_comprehend_detect_sentiment) il parametro `max_batch_size` utilizza un valore predefinito pari a 25, se uno non è stato specificato. La dimensione del batch deve essere sempre maggiore di zero. `max_batch_size` può essere utilizzato per ottimizzare le prestazioni delle chiamate di funzione Amazon Comprehend. Un batch di grandi dimensioni ha prestazioni più veloci con un maggiore utilizzo della memoria nel cluster database Aurora MySQL. Per ulteriori informazioni, consulta Considerazioni sulle prestazioni per l'utilizzo del machine learning di Aurora con Aurora MySQL.

Per informazioni sui parametri e sui tipi restituiti per le funzioni di rilevamento del sentiment in Amazon Comprehend, consultare DetectSentiment

Example Esempio: una semplice query che utilizza funzioni Amazon Comprehend

Di seguito è riportato un esempio di una semplice query che richiama queste due funzioni per determinare il livello di soddisfazione dei clienti per il team di supporto. Si supponga di avere una tabella di database (`support`) che archivia il feedback dei clienti dopo ogni richiesta di assistenza. Questa query di esempio applica entrambe le funzioni integrate al testo nella colonna `feedback` della tabella e restituisce i risultati. I valori di affidabilità restituiti dalla funzione sono double compresi tra 0,0 e 1,0. Per un output più leggibile, questa query arrotonda i risultati a 6 punti decimali. Per semplificare i confronti, questa query ordina inoltre i risultati in senso decrescente, a partire dal risultato con il massimo grado di affidabilità.

```
SELECT feedback AS 'Customer feedback',
    aws_comprehend_detect_sentiment(feedback, 'en') AS Sentiment,
    ROUND(aws_comprehend_detect_sentiment_confidence(feedback, 'en'), 6)
    AS Confidence FROM support
    ORDER BY Confidence DESC;
+-----------------------------------------------------------+-----------+------------+
| Customer feedback                                         | Sentiment | Confidence |
+-----------------------------------------------------------+-----------+------------+
| Thank you for the excellent customer support!             | POSITIVE  |   0.999771 |
| The latest version of this product stinks!                | NEGATIVE  |   0.999184 |
| Your support team is just awesome! I am blown away.       | POSITIVE  |   0.997774 |
| Your product is too complex, but your support is great.   | MIXED     |   0.957958 |
| Your support tech helped me in fifteen minutes.           | POSITIVE  |   0.949491 |
| My problem was never resolved!                            | NEGATIVE  |   0.920644 |
| When will the new version of this product be released?    | NEUTRAL   |   0.902706 |
| I cannot stand that chatbot.                              | NEGATIVE  |   0.895219 |
| Your support tech talked down to me.                      | NEGATIVE  |   0.868598 |
| It took me way too long to get a real person.             | NEGATIVE  |   0.481805 |
+-----------------------------------------------------------+-----------+------------+
 10 rows in set (0.1898 sec)
```

Example Esempio: determinazione del sentiment medio per testo al di sopra di un livello di affidabilità specifico

Una query Amazon Comprehend tipica cerca le righe in cui il sentiment ha un determinato valore, con un livello di confidenza maggiore di un certo numero. Ad esempio, la query seguente mostra come è possibile determinare il sentiment medio dei documenti nel database. La query considera solo i documenti in cui la confidenza della valutazione è almeno dell'80%.

```
SELECT AVG(CASE aws_comprehend_detect_sentiment(productTable.document, 'en')
  WHEN 'POSITIVE' THEN 1.0
  WHEN 'NEGATIVE' THEN -1.0
  ELSE 0.0 END) AS avg_sentiment, COUNT(*) AS total
FROM productTable
WHERE productTable.productCode = 1302 AND
    aws_comprehend_detect_sentiment_confidence(productTable.document, 'en') >= 0.80;
```

Utilizzo di SageMaker AI con il cluster di database Aurora MySQL

Per utilizzare la funzionalità SageMaker AI dal cluster di database Aurora MySQL, è necessario creare funzioni archiviate che incorporano le chiamate all’endpoint SageMaker AI e alle relative

funzionalità di inferenza. Per eseguire questa operazione, si utilizza `CREATE FUNCTION` di MySQL esattamente come in altre attività di elaborazione sul cluster database Aurora MySQL.

Per utilizzare i modelli implementati in SageMaker AI per l'inferenza, è necessario creare funzioni definite dall'utente utilizzando le dichiarazioni Data Definition Language (DDL) MySQL per le funzioni archiviate. Ogni funzione archiviata rappresenta l'endpoint SageMaker AI che ospita il modello. Quando definisci tale funzione, indichi i parametri di input per il modello, lo specifico endpoint SageMaker AI da invocare e il tipo restituito. La funzione restituisce l'inferenza calcolata dall'endpoint SageMaker AI dopo l'applicazione del modello sui parametri di input.

Tutte le funzioni archiviate Aurora Machine Learning restituiscono tipi numerici o `VARCHAR`. È possibile utilizzare qualsiasi tipo numerico tranne `BIT`. Altri tipi, ad esempio `JSON`, `BLOB`, `TEXT` e `DATE` non sono consentiti.

Nell'esempio seguente viene mostrata la sintassi `CREATE FUNCTION` per utilizzare SageMaker AI.

```
CREATE FUNCTION function_name (
     arg1 type1,
     arg2 type2, ...)
     [DEFINER = user]
  RETURNS mysql_type
  [SQL SECURITY { DEFINER | INVOKER } ]
  ALIAS AWS_SAGEMAKER_INVOKE_ENDPOINT
  ENDPOINT NAME 'endpoint_name'
  [MAX_BATCH_SIZE max_batch_size];
```

Questa è un'estensione della normale istruzione DDL `CREATE FUNCTION`. Nella dichiarazione `CREATE FUNCTION` che definisce la funzione di SageMaker AI, non specificare un corpo della funzione. Specifica invece la parola chiave `ALIAS` al posto del corpo della funzione. Attualmente, Aurora Machine Learning supporta solo `aws_sagemaker_invoke_endpoint` per questa sintassi estesa. È necessario specificare il parametro `endpoint_name`. Un endpoint SageMaker AI può avere caratteristiche diverse per ciascun modello.

> **Note**
>
> Per ulteriori informazioni su `CREATE FUNCTION`, consultare la sezione relativa alle istruzioni CREATE PROCEDURE e CREATE FUNCTION nel manuale di riferimento di MySQL 8.0.

Il parametro `max_batch_size` è facoltativo. Per impostazione predefinita, la dimensione batch massima è 10.000. Questo parametro può essere utilizzato nella funzione per limitare il numero massimo di input elaborati in una richiesta batch a SageMaker AI. Il parametro `max_batch_size` può contribuire a evitare un errore causato da input troppo grandi o a velocizzare la risposta di SageMaker AI. Questo parametro influisce sulla dimensione di un buffer interno utilizzato per l'elaborazione della richiesta SageMaker AI. Se specifichi un valore troppo grande per `max_batch_size`, è possibile che si verifichi un notevole sovraccarico di memoria nell'istanza database.

Ti consigliamo di lasciare l'impostazione `MANIFEST` sul valore predefinito `OFF`. Sebbene sia possibile utilizzare l'opzione `MANIFEST ON`, alcune funzionalità di SageMaker AI non possono utilizzare direttamente il CSV esportato con questa opzione. Il formato manifest non è compatibile con il formato manifest previsto da SageMaker AI.

Crea una funzione archiviata separata per ciascuno dei tuoi modelli SageMaker AI. Questa mappatura delle funzioni sui modelli è necessaria perché un endpoint è associato a un modello specifico e ciascun modello accetta parametri diversi. L'uso dei tipi SQL per gli input del modello e il tipo di output del modello consente di evitare errori di conversione dei tipi che passano i dati avanti e indietro tra i servizi AWS. È possibile controllare chi può applicare il modello. È inoltre possibile controllare le caratteristiche di runtime specificando un parametro che rappresenta la dimensione massima del batch.

Attualmente tutte le funzioni Aurora Machine Learning hanno la proprietà `NOT DETERMINISTIC`. Se non specifichi la proprietà esplicitamente, Aurora imposta `NOT DETERMINISTIC` automaticamente. Questo requisito è dovuto al fatto che il modello SageMaker AI può essere modificato senza alcuna notifica al database. In tal caso, le chiamate a una funzione Aurora Machine Learning potrebbero restituire risultati diversi per lo stesso input all'interno di una singola transazione.

Non puoi usare le caratteristiche `CONTAINS SQL`, `NO SQL`, `READS SQL DATA` o `MODIFIES SQL DATA` nell'istruzione `CREATE FUNCTION`.

Di seguito è riportato un esempio di come invocare un endpoint SageMaker AI per rilevare eventuali anomalie. È presente un endpoint `random-cut-forest-model` di SageMaker AI. Il modello corrispondente è già stato addestrato dall'algoritmo `random-cut-forest`. Per ogni input, il modello restituisce un punteggio di anomalia. Questo esempio mostra i punti dati il cui punteggio è maggiore di 3 deviazioni standard (circa il 99,9 percentile) dal punteggio medio.

```
CREATE FUNCTION anomaly_score(value real) returns real
  alias aws_sagemaker_invoke_endpoint endpoint name 'random-cut-forest-model-demo';
```

```
set @score_cutoff = (select avg(anomaly_score(value)) + 3 * std(anomaly_score(value))
 from nyc_taxi);

select *, anomaly_detection(value) score from nyc_taxi
  where anomaly_detection(value) > @score_cutoff;
```

Requisito del set di caratteri per le funzioni di SageMaker AI che restituiscono stringhe

È consigliabile specificare un set di caratteri `utf8mb4` come tipo restituito per le funzioni SageMaker AI che restituiscono valori stringa. Se non è pratico, utilizza una lunghezza della stringa sufficiente per il tipo restituito per contenere un valore rappresentato nel set di caratteri `utf8mb4`. L'esempio seguente mostra come dichiarare il set di caratteri `utf8mb4` per la tua funzione.

```
CREATE FUNCTION my_ml_func(...) RETURNS VARCHAR(5) CHARSET utf8mb4 ALIAS ...
```

Attualmente, ogni funzione di SageMaker AI che restituisce una stringa utilizza il set di caratteri `utf8mb4` per il valore restituito. Il valore restituito utilizza questo set di caratteri anche se la funzione di SageMaker AI dichiara un set di caratteri diverso per il tipo restituito in modo implicito o esplicito. Se la funzione di SageMaker AI dichiara un set di caratteri diverso per il valore restituito, i dati restituiti archiviati in una colonna della tabella non abbastanza lunga potrebbero essere troncati senza alcun avviso. Ad esempio, una query con una clausola `DISTINCT` crea una tabella temporanea. Pertanto, il risultato della funzione SageMaker AI potrebbe essere troncato a causa del modo in cui le stringhe vengono gestite internamente durante la query.

Esportazione dei dati in Amazon S3 per l’addestramento dei modelli SageMaker AI (avanzato)

Ti consigliamo di iniziare a utilizzare il machine learning di Aurora e SageMaker AI partendo da alcuni degli algoritmi forniti e dagli endpoint SageMaker AI forniti dai data scientist del tuo team che puoi utilizzare con il tuo codice SQL. Di seguito, sono disponibili le informazioni minime relative all’utilizzo del bucket Amazon S3 con i tuoi modelli SageMaker AI e il cluster di database Aurora MySQL.

La funzione machine learning comprende due fasi principali: training e inferenza. Per eseguire l’addestramento dei modelli SageMaker AI, esporta i dati in un bucket Amazon S3. Il bucket Amazon S3 viene utilizzato da un’istanza notebook Jupyter SageMaker AI per eseguire l’addestramento del modello prima della relativa implementazione. Puoi usare l'istruzione `SELECT INTO OUTFILE S3` per eseguire una query sui dati da un cluster di database Aurora MySQL e salvarlo direttamente nei

file di testo archiviati in un bucket Simple Storage Service (Amazon S3). Quindi, l'istanza notebook utilizza i dati dal bucket Simple Storage Service (Amazon S3) per il training.

Aurora Machine Learning estende la sintassi `SELECT INTO OUTFILE` esistente in Aurora MySQL per esportare i dati in formato CSV. Il file CSV generato può essere utilizzato direttamente dai modelli che necessitano di questo formato per il training.

```
SELECT * INTO OUTFILE S3 's3_uri' [FORMAT {CSV|TEXT} [HEADER]] FROM table_name;
```

L'estensione supporta il formato CSV standard.

- Il formato `TEXT` è uguale al formato di esportazione MySQL esistente. Questo è il formato predefinito.
- Il formato `CSV` è un formato appena introdotto che segue le specifiche in RFC-4180.
- Se si specifica la parola chiave facoltativa `HEADER`, il file di output contiene una riga di intestazione. Le etichette nella riga di intestazione corrispondono ai nomi di colonna dell'istruzione `SELECT`.
- Puoi sempre usare le parole chiave `CSV` e `HEADER` come identificatori.

La sintassi estesa e la grammatica di `SELECT INTO` ora sono le seguenti:

```
INTO OUTFILE S3 's3_uri'
[CHARACTER SET charset_name]
[FORMAT {CSV|TEXT} [HEADER]]
[{FIELDS | COLUMNS}
  [TERMINATED BY 'string']
  [[OPTIONALLY] ENCLOSED BY 'char']
  [ESCAPED BY 'char']
]
[LINES
  [STARTING BY 'string']
  [TERMINATED BY 'string']
]
```

Considerazioni sulle prestazioni per l'utilizzo del machine learning di Aurora con Aurora MySQL

I servizi Amazon Bedrock, Amazon Comprehend e SageMaker AI svolgono la maggior parte del lavoro quando vengono invocati da una funzione di machine learning di Aurora. Ciò significa che

è possibile dimensionare tali risorse in base alle esigenze, in modo indipendente. Per il cluster database Aurora MySQL, è possibile rendere le chiamate di funzione il più efficienti possibile. Di seguito sono riportate alcune considerazioni sulle prestazioni da tenere presenti quando si utilizza machine learning di Aurora.

Modello e prompt

Quando utilizzi Amazon Bedrock, le prestazioni dipendono in larga misura dal modello e dal prompt in uso. Scegli un modello e un prompt ottimali per il tuo caso d'uso.

Cache delle query

La cache delle query di Aurora MySQL non funziona per le funzioni di machine learning di Aurora. Aurora MySQL non archivia i risultati della query nella cache delle query per le istruzioni SQL che richiamano le funzioni di machine learning di Aurora.

Ottimizzazione batch per le chiamate di funzione Aurora Machine Learning

L'aspetto principale delle prestazioni Aurora Machine Learning che è possibile applicare dal cluster Aurora è l'impostazione della modalità batch per le chiamate alle funzioni archiviate Aurora Machine Learning. Le funzioni di machine learning in genere richiedono un sovraccarico considerevole, rendendo impossibile chiamare separatamente un servizio esterno per ogni riga. Il machine learning di Aurora può ridurre al minimo questo sovraccarico combinando le chiamate al servizio di machine learning Aurora esterno per molte righe in un singolo batch. Il machine learning di Aurora riceve le risposte per un batch di righe di input e quindi restituisce le risposte alla query in esecuzione una riga alla volta. Questa ottimizzazione migliora la velocità effettiva e la latenza delle query Aurora senza modificare i risultati.

Quando crei una funzione Aurora archiviata connessa a un endpoint SageMaker AI, definisci il parametro della dimensione del batch. Questo parametro influisce sul numero di righe trasferite per ogni chiamata sottostante a SageMaker AI. In caso di query che elaborano un numero elevato di righe, il sovraccarico legato all'esecuzione di una chiamata SageMaker AI separata per ogni riga può essere sostanziale. Maggiore è il set di dati elaborato dalla stored procedure, maggiore è la dimensione del batch.

È possibile verificare se l'ottimizzazione della modalità batch è applicabile a una funzione di SageMaker AI controllando il piano di query prodotto dall'istruzione `EXPLAIN PLAN`. In questo caso, la colonna `extra` nel piano di esecuzione include `Batched machine learning`. Nell'esempio seguente viene mostrata una chiamata a una funzione SageMaker AI che utilizza la modalità batch.

```
mysql> CREATE FUNCTION anomaly_score(val real) returns real alias
 aws_sagemaker_invoke_endpoint endpoint name 'my-rcf-model-20191126';
Query OK, 0 rows affected (0.01 sec)

mysql> explain select timestamp, value, anomaly_score(value) from nyc_taxi;
+----+-------------+----------+------------+------+---------------+------+---------
+------+------+----------+-------------------------+
| id | select_type | table    | partitions | type | possible_keys | key  | key_len |
 ref  | rows | filtered | Extra                   |
+----+-------------+----------+------------+------+---------------+------+---------
+------+------+----------+-------------------------+
|  1 | SIMPLE      | nyc_taxi | NULL       | ALL  | NULL          | NULL | NULL    |
 NULL |   48 |   100.00 | Batched machine learning |
+----+-------------+----------+------------+------+---------------+------+---------
+------+------+----------+-------------------------+
1 row in set, 1 warning (0.01 sec)
```

Quando si chiama una delle funzioni Amazon Comprehend integrate, è possibile controllare la dimensione del batch specificando il parametro facoltativo `max_batch_size` che limita il numero massimo di valori `input_text` elaborati in ogni batch. L'invio di più elementi contemporaneamente riduce il numero di passaggi tra Aurora e Amazon Comprehend. Limitare la dimensione del batch è utile in situazioni come le query con una clausola `LIMIT`. Utilizzando un valore basso per `max_batch_size`, puoi evitare di invocare Amazon Comprehend più volte di quanti sono i testi di input.

L'ottimizzazione batch per la valutazione delle funzioni Aurora Machine Learning si applica nei seguenti casi:

- Chiamate di funzione nell'elenco di selezione o clausola `WHERE` delle istruzioni `SELECT`.
- Chiamate di funzione nell'elenco `VALUES` di istruzioni `INSERT` e `REPLACE`.
- Funzioni di SageMaker AI nei valori `SET` delle istruzioni `UPDATE`:

```
INSERT INTO MY_TABLE (col1, col2, col3) VALUES
  (ML_FUNC(1), ML_FUNC(2), ML_FUNC(3)),
  (ML_FUNC(4), ML_FUNC(5), ML_FUNC(6));
UPDATE MY_TABLE SET col1 = ML_FUNC(col2), SET col3 = ML_FUNC(col4) WHERE ...;
```

Monitoraggio del machine learning di Aurora

Per monitorare le operazioni batch di machine learning di Aurora, puoi eseguire query su diverse variabili globali, come illustrato nell'esempio seguente.

```
show status like 'Aurora_ml%';
```

Lo stato di queste variabili può essere ripristinato utilizzando un'istruzione `FLUSH STATUS`. Pertanto, tutte le cifre rappresentano i totali, le medie e così via, dall'ultima volta che la variabile è stata ripristinata.

`Aurora_ml_logical_request_cnt`

Il numero di richieste logiche valutate dall'istanza database per essere inviate ai servizi di machine learning di Aurora dall'ultimo ripristino dello stato. A seconda del fatto che sia stato utilizzato il batching, questo valore può essere superiore a `Aurora_ml_actual_request_cnt`.

`Aurora_ml_logical_response_cnt`

Il conteggio delle risposte aggregate che Aurora MySQL riceve dai servizi di machine learning di Aurora per tutte le query eseguite dagli utenti dell'istanza database.

`Aurora_ml_actual_request_cnt`

Il conteggio delle richieste di aggregazione effettuate da Aurora MySQL ai servizi di machine learning di Aurora per tutte le query eseguite dagli utenti dell'istanza database.

`Aurora_ml_actual_response_cnt`

Il conteggio delle risposte aggregate che Aurora MySQL riceve dai servizi di machine learning di Aurora per tutte le query eseguite dagli utenti dell'istanza database.

`Aurora_ml_cache_hit_cnt`

Il conteggio degli hit della cache interna aggregata che Aurora MySQL riceve dai servizi di machine learning di Aurora per tutte le query eseguite dagli utenti dell'istanza database.

`Aurora_ml_retry_request_cnt`

Il numero di richieste ripetute inviate dall'istanza database ai servizi di machine learning di Aurora dall'ultimo ripristino dello stato.

`Aurora_ml_single_request_cnt`

Il conteggio aggregato delle funzioni di machine learning di Aurora valutate in modalità non batch per tutte le query eseguite dagli utenti dell'istanza database.

Per informazioni sul monitoraggio delle prestazioni delle operazioni SageMaker AI chiamate dalle funzioni di machine learning di Aurora, consulta Monitor Amazon SageMaker AI.

Utilizzo del machine learning di Amazon Aurora con Aurora PostgreSQL

Utilizzando l'apprendimento automatico di Amazon Aurora con il cluster Aurora PostgreSQL DB, puoi usare Amazon Comprehend o Amazon AI o Amazon Bedrock, a seconda delle tue esigenze. SageMaker Ciascuno di questi servizi supporta casi d'uso di machine learning specifici.

L'apprendimento automatico di Aurora è supportato solo in alcune versioni Regioni AWS e per specifiche di Aurora PostgreSQL. Prima di provare a configurare il machine learning di Aurora, verifica la disponibilità della versione di Aurora PostgreSQL e della tua Regione. Per informazioni dettagliate, vedi Utilizzo della funzionalità Machine Learning di Aurora con Aurora PostgreSQL.

Argomenti

- Requisiti per l'utilizzo del machine learning di Aurora con Aurora PostgreSQL
- Funzionalità e limitazioni supportate del machine learning di Aurora con Aurora PostgreSQL
- Configurazione del cluster database Aurora PostgreSQL per utilizzare il machine learning di Aurora
- Utilizzo di Amazon Bedrock con il cluster di database Aurora PostgreSQL
- Utilizzo di Amazon Comprehend con il cluster database Aurora PostgreSQL
- Utilizzo dell' SageMaker intelligenza artificiale con il cluster Aurora PostgreSQL DB
- Esportazione di dati in Amazon S3 SageMaker per la formazione su modelli di intelligenza artificiale (Advanced)
- Considerazioni sulle prestazioni per l'utilizzo del machine learning di Aurora con Aurora PostgreSQL
- Monitoraggio del machine learning di Aurora

Requisiti per l'utilizzo del machine learning di Aurora con Aurora PostgreSQL

AWS i servizi di machine learning sono servizi gestiti che vengono configurati ed eseguiti nei propri ambienti di produzione. L'apprendimento automatico Aurora supporta l'integrazione con Amazon Comprehend SageMaker , AI e Amazon Bedrock. Prima di provare a configurare il tuo cluster database Aurora PostgreSQL per utilizzare il machine learning di Aurora, assicurati di aver soddisfatto i seguenti requisiti e prerequisiti.

- I servizi Amazon Comprehend, SageMaker AI e Amazon Bedrock devono essere eseguiti nello stesso cluster Aurora Regione AWS PostgreSQL DB. Non puoi utilizzare i servizi Amazon Comprehend o SageMaker AI o Amazon Bedrock da un cluster Aurora PostgreSQL DB in un'altra regione.
- Se il cluster Aurora PostgreSQL DB si trova in un cloud pubblico virtuale (VPC) diverso basato sul servizio Amazon VPC rispetto ai SageMaker servizi Amazon Comprehend e AI, il gruppo Security di VPC deve consentire le connessioni in uscita al servizio di machine learning Aurora di destinazione. Per ulteriori informazioni, consulta Abilitazione delle comunicazioni di rete da Amazon Aurora ad altri servizi AWS.
- Per l' SageMaker intelligenza artificiale, i componenti di machine learning che desideri utilizzare per le inferenze devono essere configurati e pronti all'uso. Durante il processo di configurazione per il cluster Aurora PostgreSQL DB, devi avere a disposizione l'Amazon Resource Name (ARN) dell'endpoint AI. SageMaker I data scientist del tuo team sono probabilmente i più adatti a gestire il lavoro con l' SageMaker intelligenza artificiale per preparare i modelli e gestire le altre attività simili. Per iniziare a usare Amazon SageMaker AI, consulta Get Started with Amazon SageMaker AI. Per ulteriori informazioni su inferenze ed endpoint, consulta Real-time inference (Inferenza in tempo reale).
- Per Amazon Bedrock, è necessario disporre dell'ID del modello dei modelli Bedrock che si desidera utilizzare per le inferenze durante il processo di configurazione del cluster di database Aurora PostgreSQL. I data scientist del tuo team sono probabilmente le persone più preparate per gestire l'utilizzo di Bedrock in modo da decidere quali modelli applicare, di quali eseguire il fine-tuning se necessario ed eseguire le altre attività correlate. Per iniziare a utilizzare Amazon Bedrock, consulta How to setup Bedrock.
- Gli utenti di Amazon Bedrock devono richiedere l'accesso ai modelli prima di poterli utilizzare. Se vuoi aggiungere altri modelli per la generazione di testo, chat e immagini, devi richiedere l'accesso ai modelli in Amazon Bedrock. Per ulteriori informazioni, consulta Accesso ai modelli.

Funzionalità e limitazioni supportate del machine learning di Aurora con Aurora PostgreSQL

L'apprendimento automatico Aurora supporta qualsiasi endpoint di SageMaker intelligenza artificiale in grado di leggere e scrivere il formato di valori separati da virgole (CSV) tramite un valore di. `ContentType` `text/csv` Gli algoritmi di SageMaker intelligenza artificiale integrati che attualmente accettano questo formato sono i seguenti.

- Linear Learner
- Random Cut Forest
- XGBoost

  Per ulteriori informazioni su questi algoritmi, consulta Choose an Algorithm nella Amazon SageMaker AI Developer Guide.

Quando utilizzi Amazon Bedrock con machine learning di Aurora, si applicano le seguenti limitazioni:

- Le funzioni definite dall'utente (UDFs) forniscono un modo nativo per interagire con Amazon Bedrock. UDFs Non hanno requisiti specifici di richiesta o risposta, quindi possono utilizzare qualsiasi modello.
- Puoi utilizzarlo UDFs per creare qualsiasi flusso di lavoro desiderato. Ad esempio, è possibile combinare primitivi di base, come `pg_cron`, per eseguire una query, recuperare dati, generare inferenze e scrivere su tabelle per servire direttamente le query.
- UDFs non supportano chiamate in batch o parallele.
- L'estensione per il machine learning di Aurora non supporta le interfacce vettoriali. Nell'ambito dell'estensione, è disponibile una funzione per l'output degli embedding della risposta del modello in formato `float8[]` che ne consente l'archiviazione in Aurora. Per ulteriori informazioni sull'utilizzo di `float8[]`, consulta Utilizzo di Amazon Bedrock con il cluster di database Aurora PostgreSQL.

Configurazione del cluster database Aurora PostgreSQL per utilizzare il machine learning di Aurora

Affinché l'apprendimento automatico Aurora funzioni con il cluster Aurora PostgreSQL DB, devi creare un ruolo AWS Identity and Access Management (IAM) per ciascuno dei servizi che desideri

utilizzare. Il ruolo IAM consente al cluster database Aurora PostgreSQL di utilizzare il servizio di machine learning di Aurora per conto del cluster. Dovrai inoltre installare l'estensione di machine learning di Aurora. Nei seguenti argomenti sono disponibili le procedure di configurazione per ciascuno di questi servizi di machine learning di Aurora.

Argomenti

- Configurazione di Aurora PostgreSQL per utilizzare Amazon Bedrock
- Configurazione di Aurora PostgreSQL per utilizzare Amazon Comprehend
- Configurazione di Aurora PostgreSQL per utilizzare Amazon AI SageMaker
  - Configurazione di Aurora PostgreSQL per l'utilizzo di Amazon S3 for AI (Advanced) SageMaker
- Installazione dell'estensione di machine learning di Aurora

Configurazione di Aurora PostgreSQL per utilizzare Amazon Bedrock

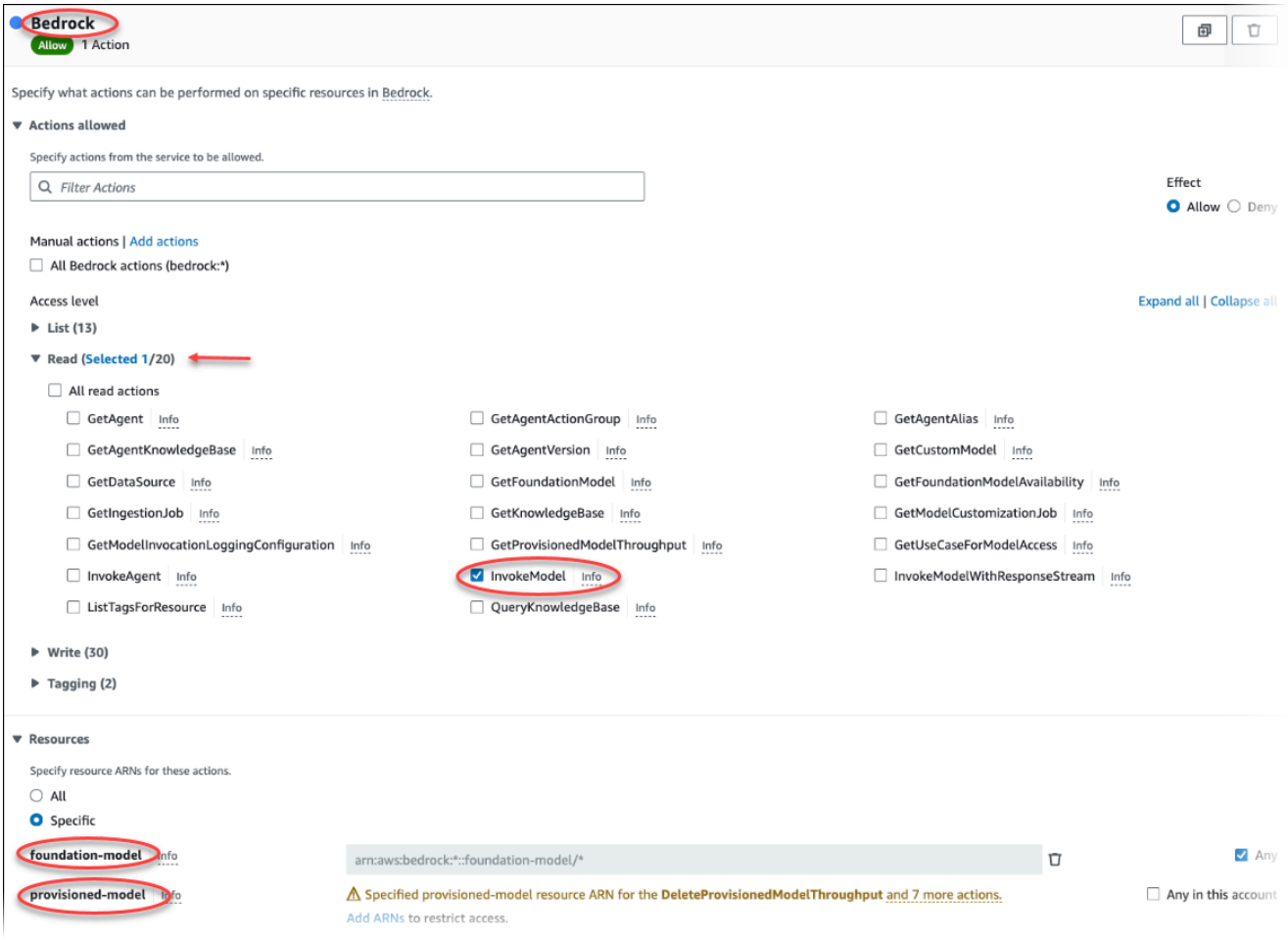
Nella procedura seguente, devi innanzitutto creare il ruolo e la policy IAM che autorizzano Aurora PostgreSQL a utilizzare Amazon Bedrock per conto del cluster. Devi quindi collegare la policy a un ruolo IAM utilizzato dal cluster di database Aurora PostgreSQL per poter eseguire Amazon Bedrock. Per una maggiore semplicità, questa procedura utilizza la Console di gestione AWS per completare tutte le attività.

Per configurare il cluster di database Aurora PostgreSQL in modo che utilizzi Amazon Bedrock

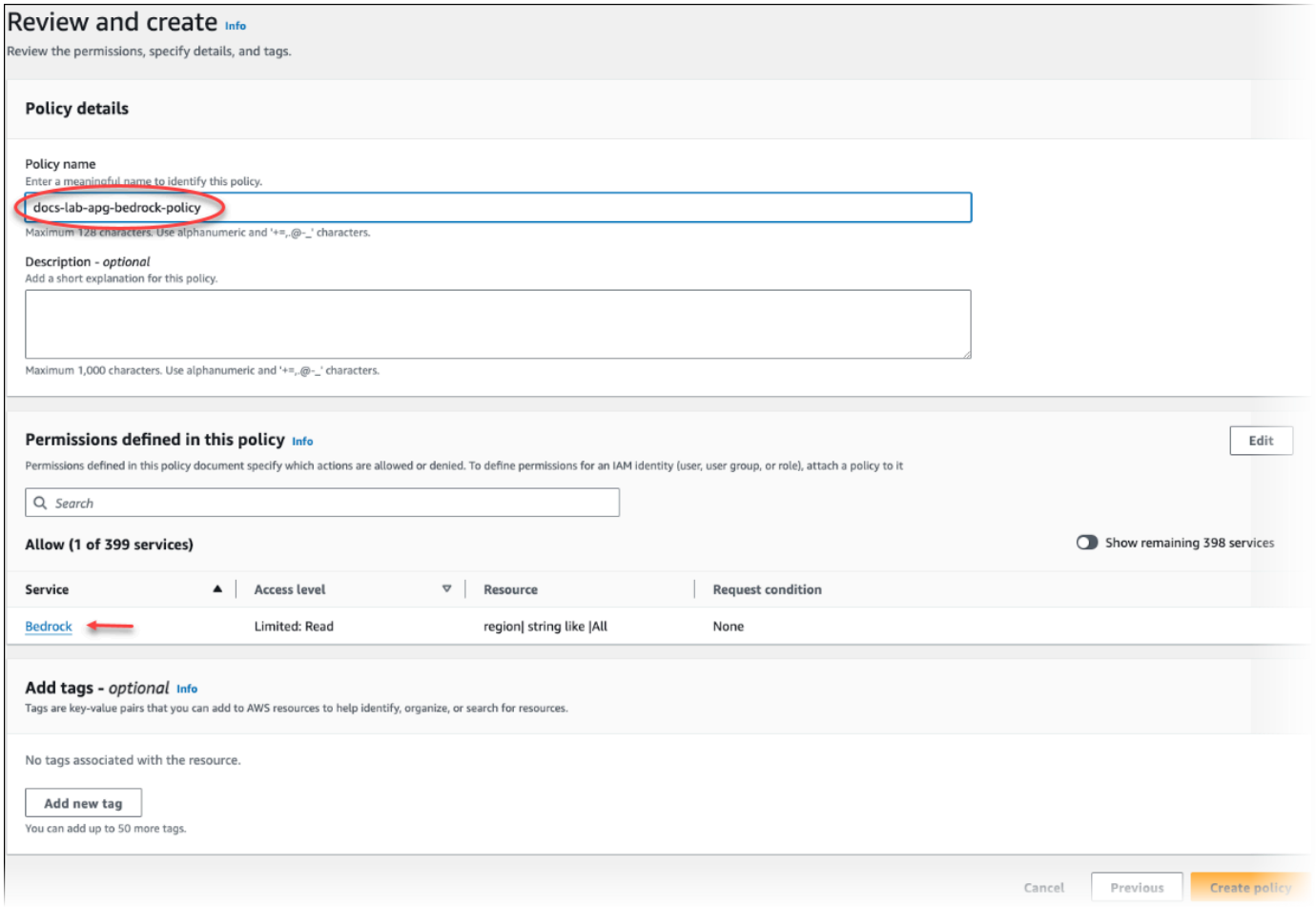
1. Accedi Console di gestione AWS e apri la console IAM all'indirizzo. https://console.aws.amazon.com/iam/
2. Aprire la console IAM all'indirizzo https://console.aws.amazon.com/iam/.
3. Scegli Policies (in Gestione degli accessi) nel menu della console AWS Identity and Access Management (IAM).

   a. Scegli Crea policy. Nella pagina dell’editor visivo, scegli Servizio, quindi immetti Bedrock nel campo Seleziona un servizio. Espandi il livello di accesso Read (Lettura). InvokeModelScegli tra le impostazioni di lettura di Amazon Bedrock.

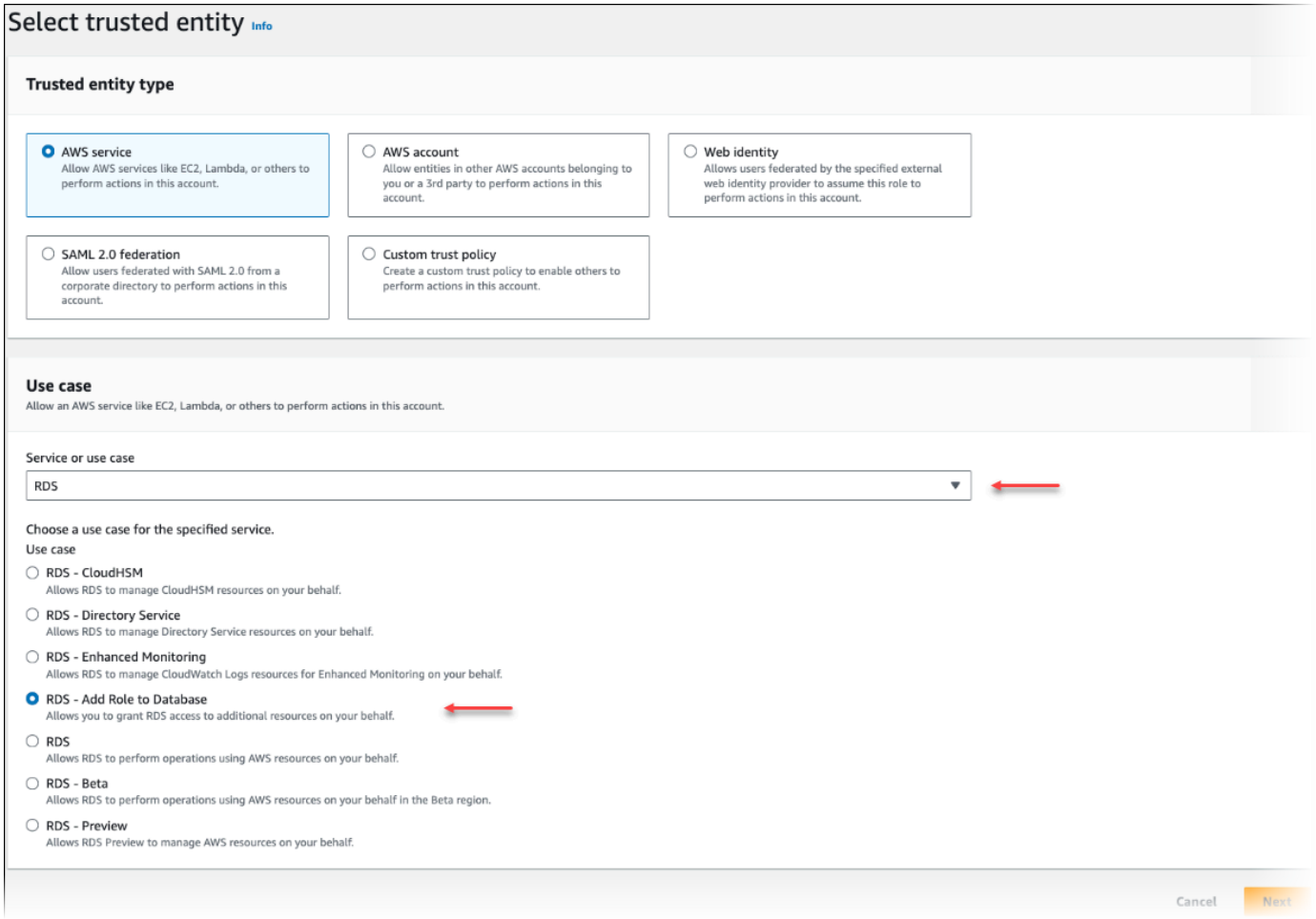
   b. Scegli il Foundation/Provisioned modello a cui desideri concedere l'accesso in lettura tramite la politica.



4. Scegli Next: Tags (Successivo: Tag) e definisci i tag (questo passaggio è facoltativo). Seleziona Next: Review (Successivo: Rivedi). Immetti un nome e una descrizione per la policy, come illustrato nell'immagine.

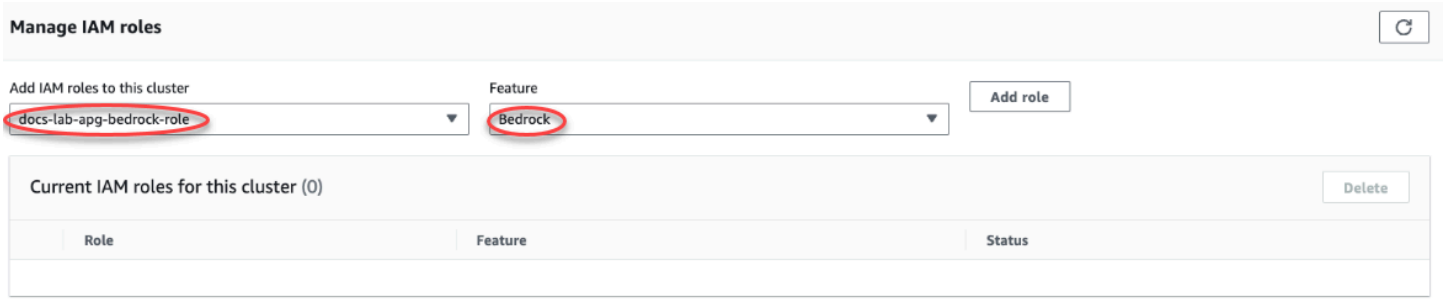


5. Scegli Crea policy. Viene visualizzato un avviso nella console quando la policy viene salvata e diventa disponibile nell'elenco delle policy.
6. Nella console IAM scegli Roles (Ruoli) sotto Access management (Gestione degli accessi).
7. Scegli Crea ruolo.
8. Nella pagina Select trusted entity (Seleziona entità attendibile), scegli il riquadro AWS service (Servizio AWS), quindi seleziona RDS per aprire il selettore.
9. Scegli RDS – Add Role to Database (RDS - Aggiungi ruolo al database).



10. Scegli Next (Successivo). Nella pagina Add permissions (Aggiungi autorizzazioni), trova la policy creata nel passaggio precedente e selezionala tra quelle elencate. Scegli Next (Successivo).
11. Next: Review (Successivo: Rivedi). Immetti un nome e una descrizione per il ruolo IAM.
12. Aprire la console Amazon RDS all'indirizzo https://console.aws.amazon.com/rds/.
13. Passa alla posizione Regione AWS in cui si trova il cluster Aurora PostgreSQL DB.
14. Nel riquadro di navigazione, scegli Database, quindi seleziona il cluster di database Aurora PostgreSQL da utilizzare con Bedrock.
15. Scegli la scheda Connectivity & security (Connettività e sicurezza) e scorri fino a trovare la sezione Manage IAM roles (Gestisci ruoli IAM) della pagina. Nel selettore Add IAM roles to this cluster (Aggiungi i ruoli IAM a questo cluster) scegli il ruolo creato nei passaggi precedenti. Nel riquadro Caratteristica, scegli Bedrock, quindi seleziona Aggiungi ruolo.

Il ruolo e la relativa policy vengono associati al cluster database Aurora PostgreSQL. Al termine del processo, il ruolo viene visualizzato nell'elenco Current IAM roles for this cluster (Ruoli IAM attuali per questo cluster), come illustrato di seguito.



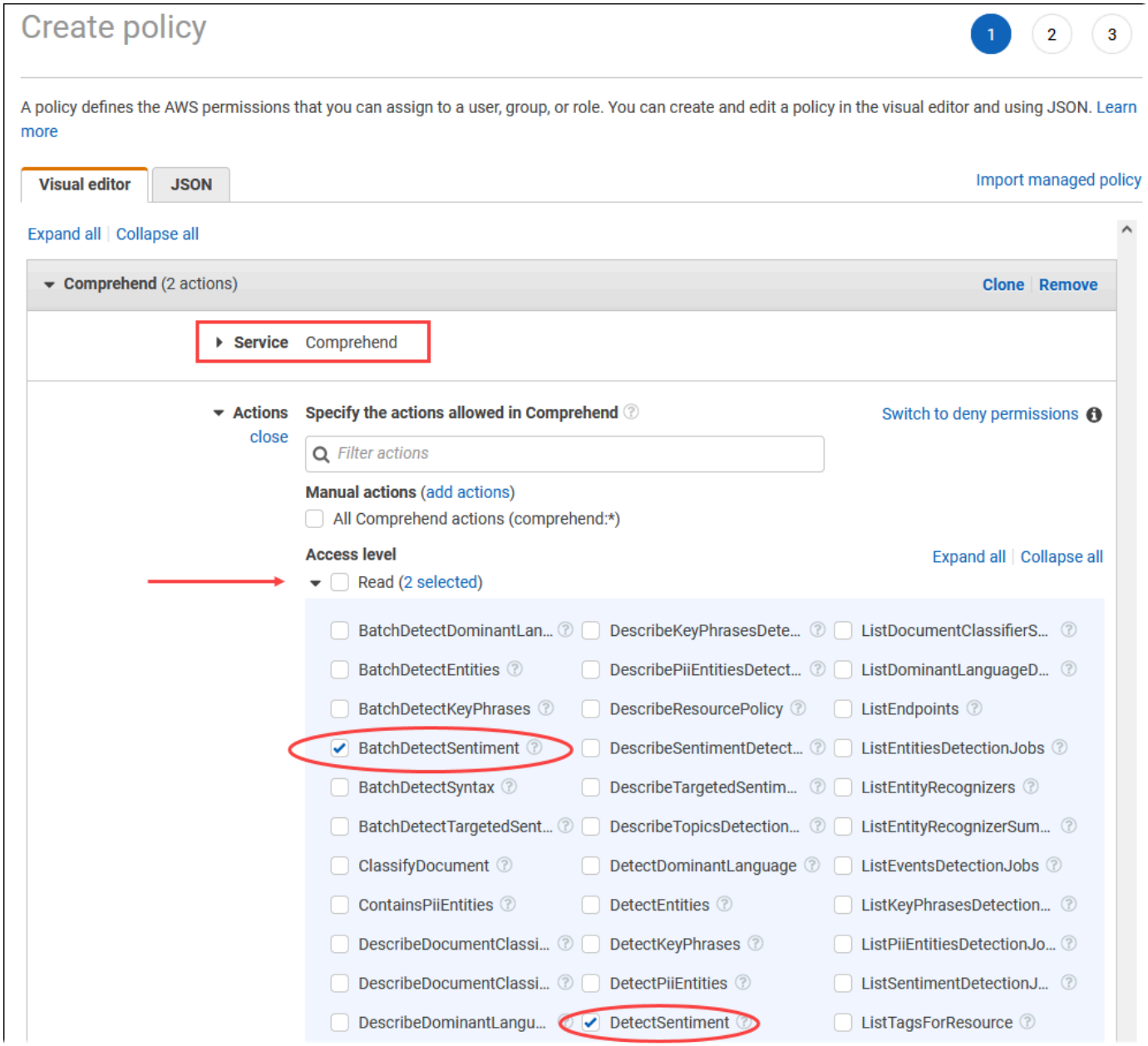
La configurazione di IAM per Amazon Bedrock è stata completata. Ora continua a configurare Aurora PostgreSQL per utilizzare il machine learning di Aurora installando l'estensione come descritto in Installazione dell'estensione di machine learning di Aurora

Configurazione di Aurora PostgreSQL per utilizzare Amazon Comprehend

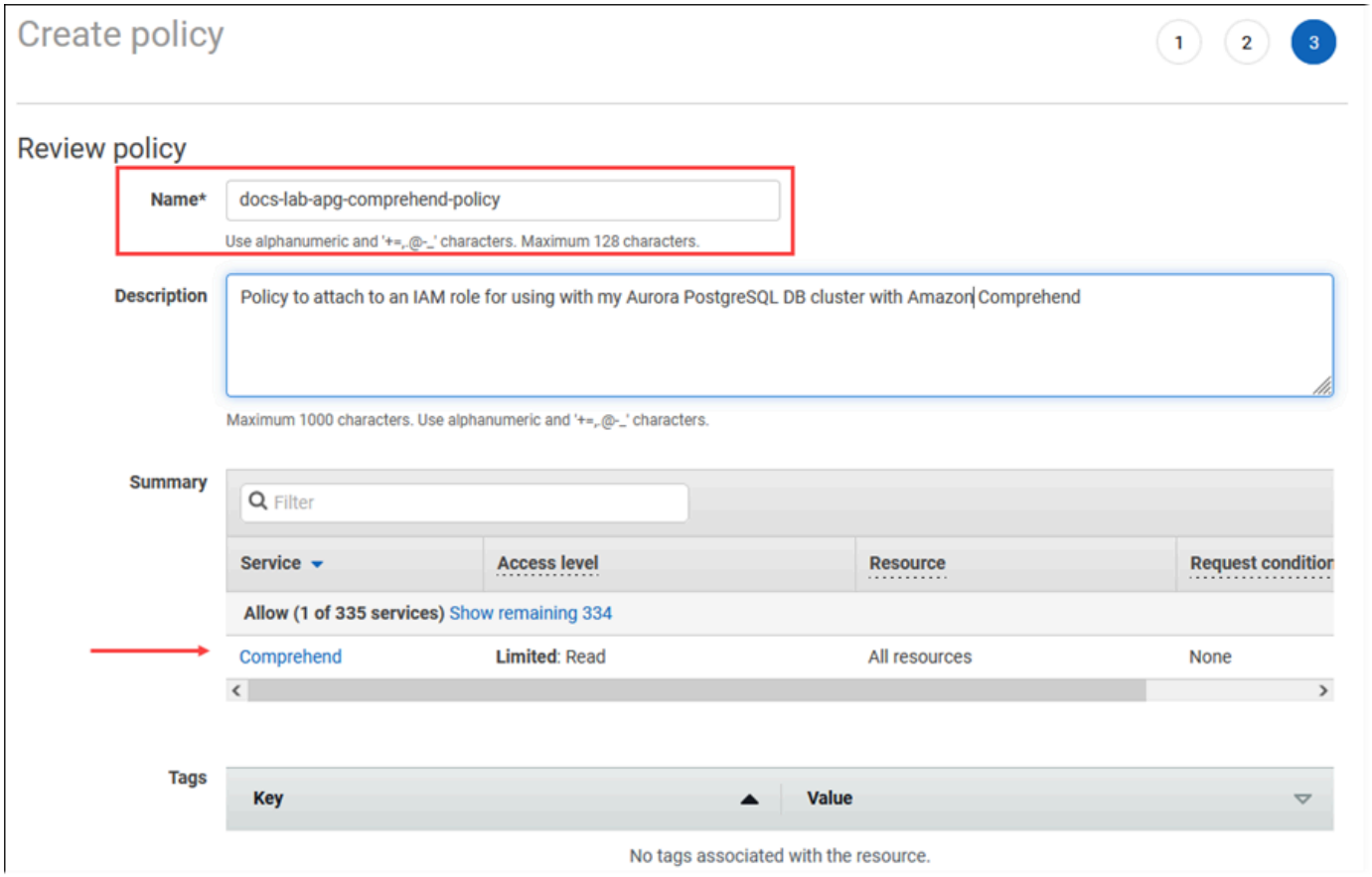
Nella procedura seguente, per prima cosa crei il ruolo e la policy IAM che autorizzano Aurora PostgreSQL a utilizzare Amazon Comprehend per conto del cluster. Quindi colleghi la policy a un ruolo IAM che viene usato dal cluster database Aurora PostgreSQL per poter eseguire Amazon Comprehend. Per motivi di semplicità, si usa la Console di gestione AWS per completare tutte le attività di questa procedura.

Per configurare il cluster database Aurora PostgreSQL per utilizzare Amazon Comprehend

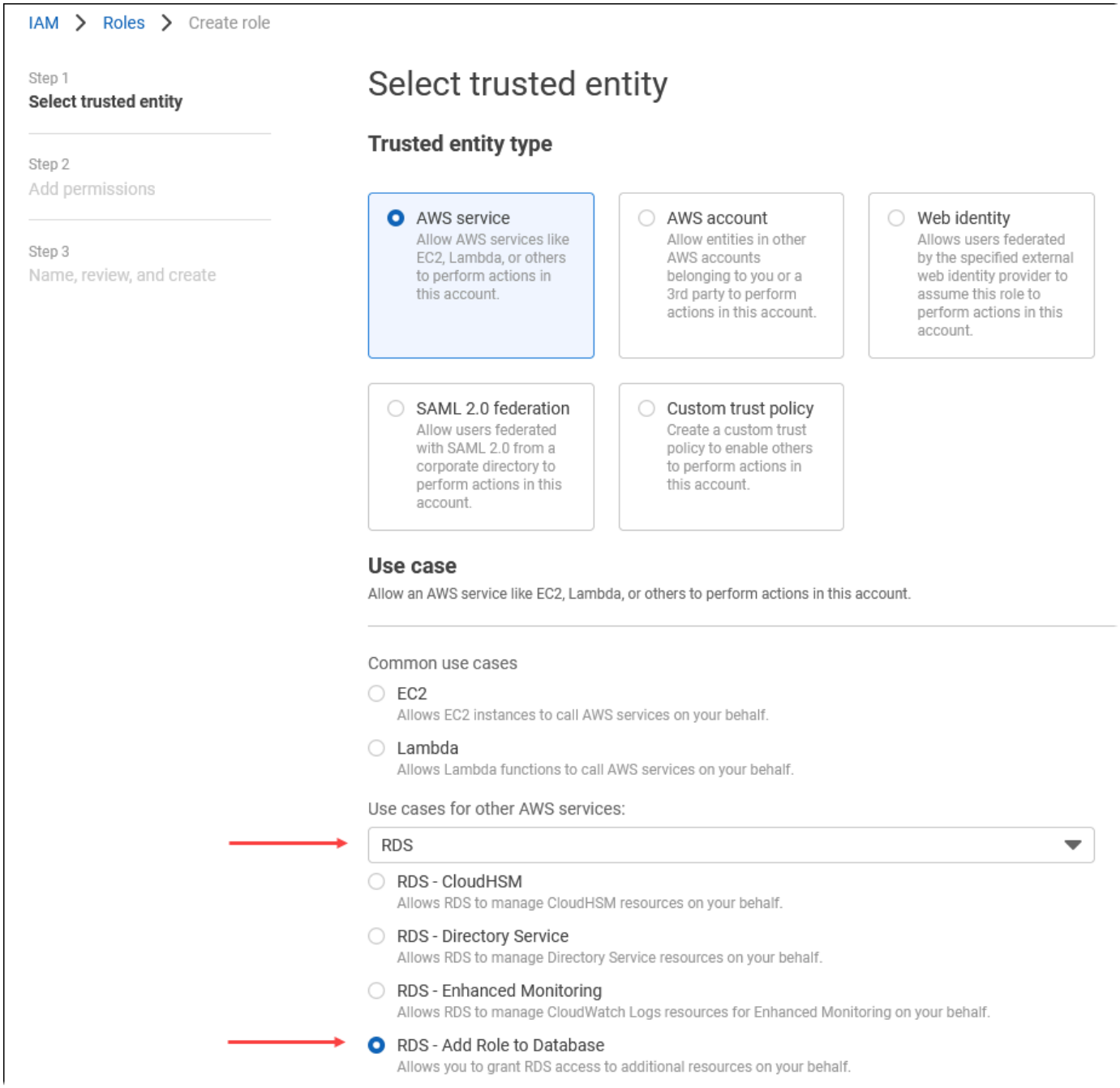
1. Accedi Console di gestione AWS e apri la console IAM all'indirizzo. https://console.aws.amazon.com/iam/
2. Aprire la console IAM all'indirizzo https://console.aws.amazon.com/iam/.
3. Scegli Policies (in Gestione degli accessi) nel menu della console AWS Identity and Access Management (IAM).



4. Scegli Crea policy. Nella pagina Visual editor (Editor visivo) scegli Service (Servizio), quindi immetti Comprehend nel campo Select a service (Seleziona un servizio). Espandi il livello di accesso Read (Lettura). Scegli BatchDetectSentimente tra le DetectSentimentimpostazioni di lettura di Amazon Comprehend.
5. Scegli Next: Tags (Successivo: Tag) e definisci i tag (questo passaggio è facoltativo). Seleziona Next: Review (Successivo: Rivedi). Immetti un nome e una descrizione per la policy, come illustrato nell'immagine.



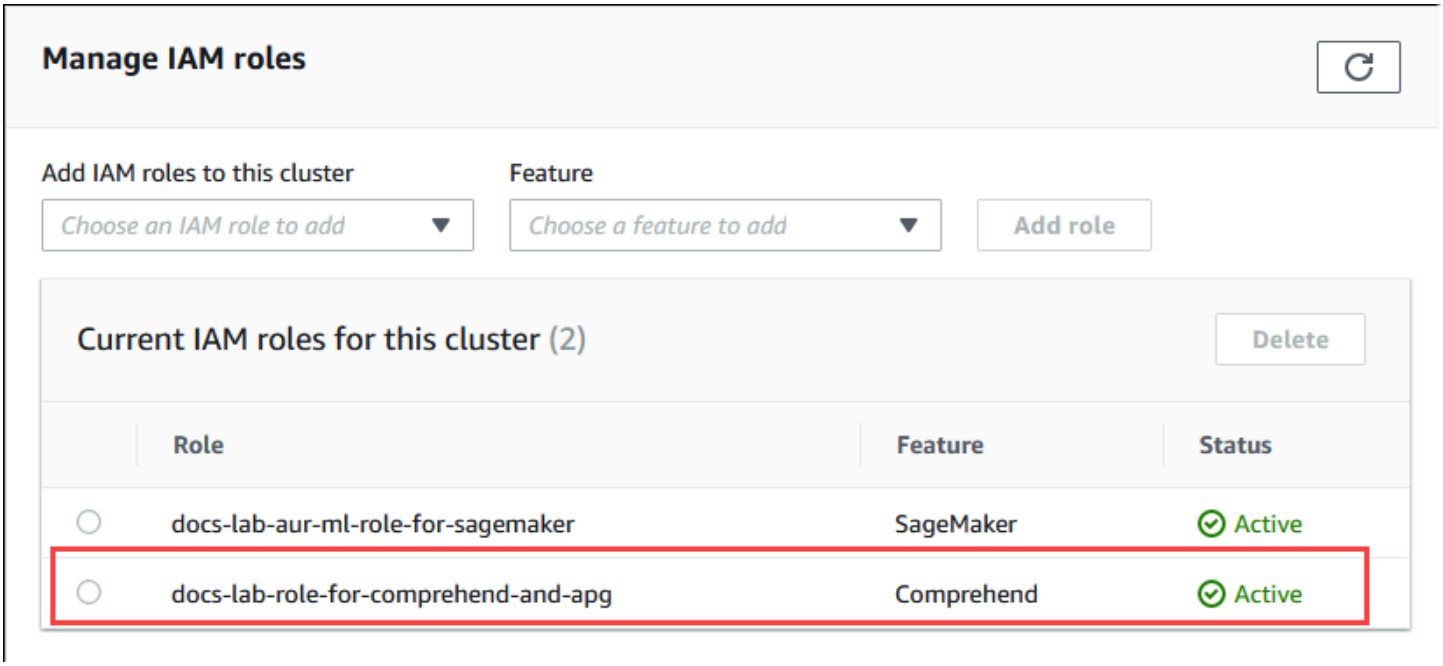
6. Scegli Crea policy. Viene visualizzato un avviso nella console quando la policy viene salvata e diventa disponibile nell'elenco delle policy.
7. Nella console IAM scegli Roles (Ruoli) sotto Access management (Gestione degli accessi).
8. Scegli Crea ruolo.
9. Nella pagina Select trusted entity (Seleziona entità attendibile), scegli il riquadro AWS service (Servizio AWS), quindi seleziona RDS per aprire il selettore.
10. Scegli RDS – Add Role to Database (RDS - Aggiungi ruolo al database).



11. Scegli Next (Successivo). Nella pagina Add permissions (Aggiungi autorizzazioni), trova la policy creata nel passaggio precedente e selezionala tra quelle elencate. Seleziona Next (Successivo).
12. Next: Review (Successivo: Rivedi). Immetti un nome e una descrizione per il ruolo IAM.
13. Aprire la console Amazon RDS all'indirizzo https://console.aws.amazon.com/rds/.
14. Passa alla posizione Regione AWS in cui si trova il cluster Aurora PostgreSQL DB.
15. Nel riquadro di navigazione, scegli Databases (Database), quindi seleziona il cluster database Aurora PostgreSQL da utilizzare con Amazon Comprehend.

16. Scegli la scheda Connectivity & security (Connettività e sicurezza) e scorri fino a trovare la sezione Manage IAM roles (Gestisci ruoli IAM) della pagina. Nel selettore Add IAM roles to this cluster (Aggiungi i ruoli IAM a questo cluster) scegli il ruolo creato nei passaggi precedenti. Nel selettore Caratteristica, scegli Comprehend, quindi seleziona Aggiungi ruolo.

Il ruolo e la relativa policy vengono associati al cluster database Aurora PostgreSQL. Al termine del processo, il ruolo viene visualizzato nell'elenco Current IAM roles for this cluster (Ruoli IAM attuali per questo cluster), come illustrato di seguito.



La configurazione di IAM per Amazon Comprehend è stata completata. Ora continua a configurare Aurora PostgreSQL per utilizzare il machine learning di Aurora installando l'estensione come descritto in Installazione dell'estensione di machine learning di Aurora

Configurazione di Aurora PostgreSQL per utilizzare Amazon AI SageMaker

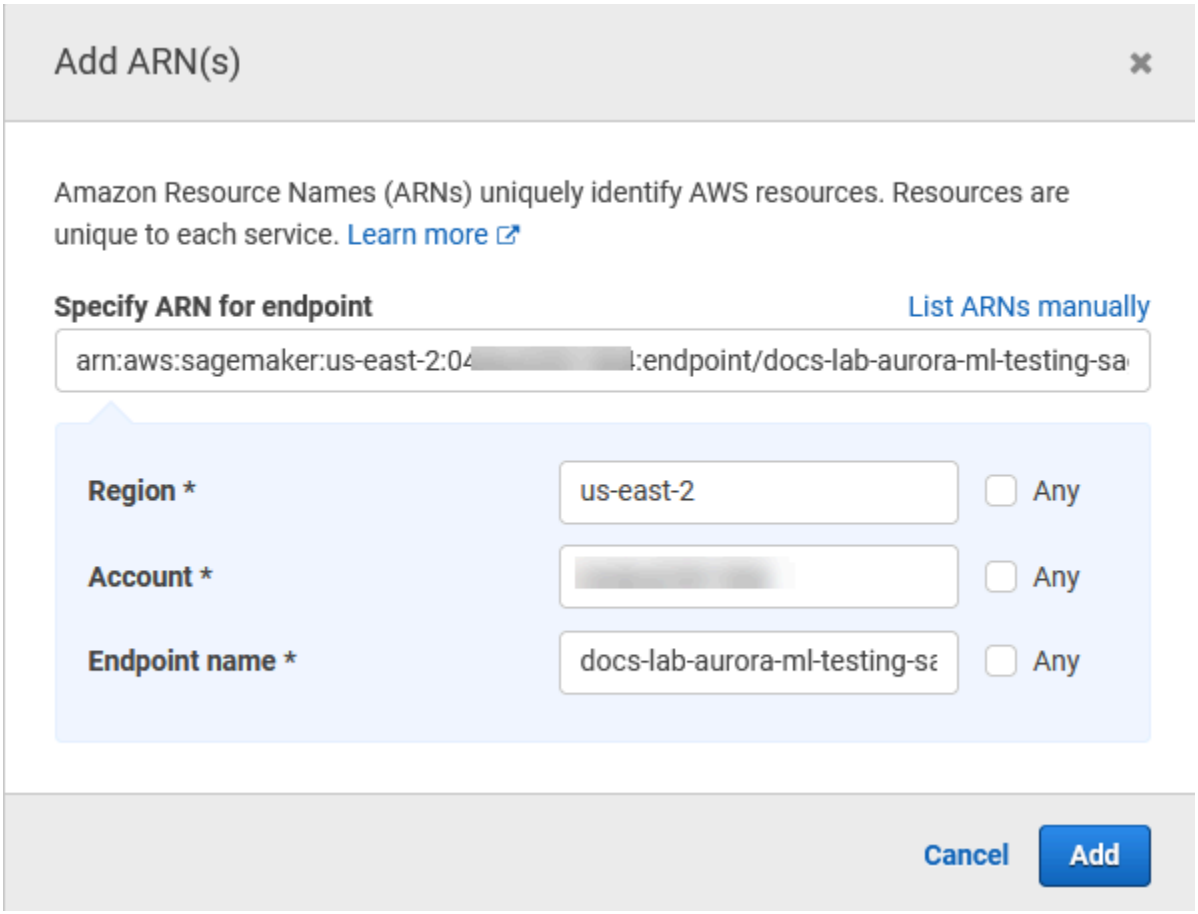
Prima di poter creare la policy e il ruolo IAM per il tuo cluster Aurora PostgreSQL DB, devi avere a disposizione la configurazione del modello di intelligenza artificiale e l'endpoint. SageMaker

Per configurare il cluster Aurora PostgreSQL DB per utilizzare l'intelligenza artificiale SageMaker

1. Accedi Console di gestione AWS e apri la console IAM all'indirizzo. https://console.aws.amazon.com/iam/
2. Scegli Policy (in Gestione degli accessi) nel menu della console AWS Identity and Access Management (IAM), quindi scegli Crea policy. Nell'editor visivo, scegli SageMakeril servizio.

Per Azioni, apri il selettore di lettura (in Livello di accesso) e scegli InvokeEndpoint. Viene visualizzata un'icona di avviso quando si esegue questa operazione.

3. Apri il selettore Risorse e scegli il link Aggiungi ARN per limitare l'accesso sotto Specificare l'ARN della risorsa dell'endpoint per l'azione. InvokeEndpoint
4. Inserisci le Regione AWS tue risorse SageMaker AI e il nome del tuo endpoint. Il tuo AWS account è precompilato.



5. Scegli Add (Aggiungi) per salvare. Scegli Next: Tags (Successivo: Tag) e Next: Review (Successivo: Rivedi) per passare all'ultima pagina del processo di creazione della policy.
6. Immetti un nome e una descrizione per la policy e quindi scegli Create policy (Crea policy). La policy viene creata e aggiunta all'elenco delle policy. Quindi, viene visualizzato un avviso nella console.
7. Nella console IAM scegli Roles (Ruoli).
8. Scegli Crea ruolo.
9. Nella pagina Select trusted entity (Seleziona entità attendibile), scegli il riquadro AWS service (Servizio AWS), quindi seleziona RDS per aprire il selettore.

10. Scegli RDS – Add Role to Database (RDS - Aggiungi ruolo al database).
11. Scegli Next (Successivo). Nella pagina Add permissions (Aggiungi autorizzazioni), trova la policy creata nel passaggio precedente e selezionala tra quelle elencate. Seleziona Next (Successivo).
12. Next: Review (Successivo: Rivedi). Immetti un nome e una descrizione per il ruolo IAM.
13. Aprire la console Amazon RDS all'indirizzo https://console.aws.amazon.com/rds/.
14. Passa alla posizione Regione AWS in cui si trova il cluster Aurora PostgreSQL DB.
15. Nel riquadro di navigazione, scegli Database, quindi scegli il cluster Aurora PostgreSQL DB che desideri utilizzare con l'intelligenza artificiale. SageMaker
16. Scegli la scheda Connectivity & security (Connettività e sicurezza) e scorri fino a trovare la sezione Manage IAM roles (Gestisci ruoli IAM) della pagina. Nel selettore Add IAM roles to this cluster (Aggiungi i ruoli IAM a questo cluster) scegli il ruolo creato nei passaggi precedenti. Nel selettore Feature, scegli SageMaker AI, quindi scegli Aggiungi ruolo.

Il ruolo e la relativa policy vengono associati al cluster database Aurora PostgreSQL. Al termine del processo, il ruolo viene visualizzato nell'elenco Current IAM roles for this cluster (Ruoli IAM attuali per questo cluster).

La configurazione IAM per l' SageMaker IA è completa. Ora continua a configurare Aurora PostgreSQL per utilizzare il machine learning di Aurora installando l'estensione come descritto in Installazione dell'estensione di machine learning di Aurora.

Configurazione di Aurora PostgreSQL per l'utilizzo di Amazon S3 for AI (Advanced) SageMaker

Per utilizzare l' SageMaker intelligenza artificiale con i tuoi modelli anziché utilizzare i componenti predefiniti forniti dall' SageMaker IA, devi configurare un cluster Amazon Simple Storage Service (Amazon S3) per Aurora PostgreSQL DB da utilizzare. Questa è un'attività avanzata non completamente documentata in questa Guida per l'utente di Amazon Aurora. Il processo generale è lo stesso utilizzato per l'integrazione del supporto per l'IA, come segue. SageMaker

1. Crea il ruolo e la policy IAM per Amazon S3.
2. Aggiungi il ruolo IAM e l'importazione o l'esportazione di Amazon S3 come funzionalità nella scheda Connectivity & security (Connettività e sicurezza) del cluster database Aurora PostgreSQL.
3. Aggiungi l'ARN del ruolo al gruppo di parametri personalizzati del cluster database Aurora.

Per informazioni di base sull'utilizzo, consulta Esportazione di dati in Amazon S3 SageMaker per la formazione su modelli di intelligenza artificiale (Advanced).

Installazione dell'estensione di machine learning di Aurora

Le estensioni di machine learning Aurora `aws_ml 1.0` offrono due funzioni che puoi usare per richiamare Amazon Comprehend, i servizi SageMaker AI e `aws_ml 2.0` offrono due funzioni aggiuntive che puoi usare per richiamare i servizi Amazon Bedrock. L'installazione di queste estensioni nel cluster di database Aurora PostgreSQL crea anche un ruolo amministrativo per la funzionalità.

> **Note**
>
> L'utilizzo di queste funzioni dipende dal completamento della configurazione IAM per il servizio di machine learning Aurora (Amazon Comprehend, SageMaker AI, Amazon Bedrock), come dettagliato in. Configurazione del cluster database Aurora PostgreSQL per utilizzare il machine learning di Aurora

- aws_comprehend.detect_sentiment: questa funzione si utilizza per applicare l'analisi del sentiment al testo archiviato nel database del cluster database Aurora PostgreSQL.
- aws_sagemaker.invoke_endpoint: utilizzi questa funzione nel codice SQL per comunicare con l'endpoint AI del tuo cluster. SageMaker
- aws_bedrock.invoke_model: utilizza questa funzione nel codice SQL per comunicare con i modelli Bedrock del cluster. La risposta di questa funzione avrà il formato di un TEXT, quindi se un modello risponde nel formato di un corpo JSON, il formato dell'output della funzione inoltrato all'utente finale sarà una stringa.
- aws_bedrock.invoke_model_get_embeddings: utilizza questa funzione nel codice SQL per invocare modelli Bedrock che restituiscono embedding di output all'interno di una risposta JSON. Questo può risultare utile per estrarre gli embedding direttamente associati alla chiave json per semplificare la risposta con qualsiasi flusso di lavoro autogestito.

Per installare l'estensione di machine learning di Aurora nel cluster database Aurora PostgreSQL

- Utilizza `psql` per connetterti all'istanza di scrittura del cluster database Aurora PostgreSQL. Collega il database specifico in cui installare l'estensione `aws_ml`.

```
psql --host=cluster-instance-1.111122223333.aws-region.rds.amazonaws.com --
port=5432 --username=postgres --password --dbname=labdb
```

```
labdb=> CREATE EXTENSION IF NOT EXISTS aws_ml CASCADE;
NOTICE:  installing required extension "aws_commons"
CREATE EXTENSION
labdb=>
```

L'installazione delle estensioni `aws_ml` crea anche il ruolo amministrativo `aws_ml` e tre nuovi schemi, come indicato di seguito.

- `aws_comprehend`: schema del servizio Amazon Comprehend e origine della funzione `detect_sentiment` (`aws_comprehend.detect_sentiment`).
- `aws_sagemaker`— Schema per il servizio AI e fonte della funzione (). SageMaker `invoke_endpoint` `aws_sagemaker.invoke_endpoint`
- `aws_bedrock`: schema del servizio Amazon Bedrock e origine delle funzioni `invoke_model(aws_bedrock.invoke_model)` e `invoke_model_get_embeddings(aws_bedrock.invoke_model_get_embeddings)`.

Al ruolo `rds_superuser` viene concesso il ruolo amministrativo `aws_ml` e diventa `OWNER` di questi tre schemi di machine learning di Aurora. Per consentire ad altri utenti del database di accedere alle funzioni di machine learning di Aurora, `rds_superuser` deve concedere i privilegi `EXECUTE` per le funzioni di machine learning di Aurora. Per impostazione predefinita, i privilegi `EXECUTE` sono revocati da `PUBLIC` per le funzioni nei due schemi di machine learning di Aurora.

In una configurazione di database multi-tenant, puoi impedire ai tenant di accedere alle funzioni di machine learning di Aurora utilizzando `REVOKE USAGE` nello specifico schema di machine learning di Aurora che desideri proteggere.

Utilizzo di Amazon Bedrock con il cluster di database Aurora PostgreSQL

Per Aurora PostgreSQL, il machine learning di Aurora fornisce la seguente funzione Amazon Bedrock che consente di utilizzare i dati di testo. Questa funzione è disponibile solo dopo aver installato l'estensione `aws_ml` 2.0 e completato tutte le procedure di configurazione. Per ulteriori informazioni, consulta Configurazione del cluster database Aurora PostgreSQL per utilizzare il machine learning di Aurora.

aws_bedrock.invoke_model

Questa funzione accetta il testo formattato in JSON come input, lo elabora per una varietà di modelli ospitati su Amazon Bedrock e recupera la risposta testuale JSON dal modello. Tale

risposta potrebbe contenere testo, immagine o embedding. Di seguito è riportato un riepilogo della documentazione della funzione.

```
aws_bedrock.invoke_model(
   IN model_id      varchar,
   IN content_type  text,
   IN accept_type   text,
   IN model_input   text,
   OUT model_output varchar)
```

Gli input e gli output di questa funzione sono indicati di seguito.

- `model_id`: identificatore del modello.
- `content_type`: tipo di richiesta al modello di Bedrock.
- `accept_type`: tipo di risposta che ci si aspetta dal modello di Bedrock, Di solito application/JSON per la maggior parte dei modelli.
- `model_input`: prompt, un set specifico di input per il modello nel formato specificato da content_type. Per ulteriori informazioni sulla richiesta accettata format/structure dal modello, consulta Parametri di inferenza per i modelli di base.
- `model_output`: output del modello Bedrock in formato testo.

Il seguente esempio mostra come invocare un modello Anthropic Claude 2 per Bedrock utilizzando invoke_model.

Example Esempio: una query semplice che utilizza le funzioni di Amazon Bedrock

```
SELECT aws_bedrock.invoke_model (
    model_id    := 'anthropic.claude-v2',
    content_type:= 'application/json',
    accept_type := 'application/json',
    model_input := '{"prompt": "\n\nHuman: You are a helpful assistant that answers
 questions directly and only using the information provided in the context below.
\nDescribe the answer
    in detail.\n\nContext: %s \n\nQuestion: %s \n
\nAssistant:","max_tokens_to_sample":4096,"temperature":0.5,"top_k":250,"top_p":0.5,"stop_seque
[]}'
);
```

aws_bedrock.invoke_model_get_embeddings

In alcuni casi, l'output del modello può indicare embedding vettoriali. Dato che la risposta varia in base al modello, è possibile ricorrere a un'altra funzione invoke_model_get_embeddings che funziona esattamente come invoke_model, ma restituisce gli embedding specificando la chiave json appropriata.

```
aws_bedrock.invoke_model_get_embeddings(
   IN model_id      varchar,
   IN content_type  text,
   IN json_key      text,
   IN model_input   text,
   OUT model_output float8[])
```

Gli input e gli output di questa funzione sono indicati di seguito.

- `model_id`: identificatore del modello.
- `content_type`: tipo di richiesta al modello di Bedrock. In questo caso, accept_type è impostato sul valore predefinito `application/json`.
- `model_input`: prompt, un set specifico di input per il modello nel formato specificato da content_type. Per ulteriori informazioni sulla richiesta accettata format/structure dal modello, vedere Parametri di inferenza per i modelli di base.
- `json_key`: riferimento al campo da cui estrarre l'embedding. Può variare se il modello di embedding cambia.
- `model_output`: output del modello Bedrock sotto forma di array di embedding con decimali a 16 bit.

L'esempio seguente mostra come generare un incorporamento utilizzando il modello Titan Embeddings G1 — Text embedding per la frase PostgreSQL monitoring views. I/O

Example Esempio: una query semplice che utilizza le funzioni di Amazon Bedrock

```
SELECT aws_bedrock.invoke_model_get_embeddings(
   model_id      := 'amazon.titan-embed-text-v1',
   content_type  := 'application/json',
   json_key      := 'embedding',
   model_input   := '{ "inputText": "PostgreSQL I/O monitoring views"}') AS embedding;
```

Utilizzo di Amazon Comprehend con il cluster database Aurora PostgreSQL

Per Aurora PostgreSQL, il machine learning di Aurora fornisce la seguente funzione Amazon Comprehend per utilizzare i dati di testo. Questa funzione è disponibile solo dopo aver installato l'estensione `aws_ml` e completato tutte le procedure di configurazione. Per ulteriori informazioni, consulta Configurazione del cluster database Aurora PostgreSQL per utilizzare il machine learning di Aurora.

aws_comprehend.detect_sentiment

Questa funzione utilizza il testo come input e valuta se ha un intento emotivo positivo, negativo, neutro o misto. Genera questo sentiment insieme a un livello di confidenza per la valutazione. Di seguito è riportato un riepilogo della documentazione della funzione.

```
aws_comprehend.detect_sentiment(
   IN input_text varchar,
   IN language_code varchar,
   IN max_rows_per_batch int,
   OUT sentiment varchar,
   OUT confidence real)
```

Gli input e gli output di questa funzione sono indicati di seguito.

- `input_text`: il testo da valutare per assegnare il sentiment (negativo, positivo, neutro, misto).
- `language_code`: la lingua di `input_text` identificata utilizzando l'identificatore a 2 lettere ISO 639-1 con etichetta secondaria regionale (se necessaria) o il codice a tre lettere ISO 639-2, a seconda dei casi. Ad esempio, `en` è il codice per l'inglese, `zh` è il codice per il cinese semplificato. Per ulteriori informazioni, consulta Lingue supportate nella Guida per gli sviluppatori di Amazon Comprehend.
- `max_rows_per_batch`: il numero massimo di righe per batch per l'elaborazione in modalità batch. Per ulteriori informazioni, consulta Informazioni sulla modalità batch e sulle funzioni di machine learning di Aurora.
- `sentiment`: il sentiment del testo di input, identificato come POSITIVE (POSITIVO), NEGATIVE (NEGATIVO), NEUTRAL (NEUTRO) o MIXED (MISTO).
- `confidence`: il livello di confidenza della precisione del `sentiment` specificato. I valori sono compresi tra 0,0 e 1,0.

Di seguito sono riportati alcuni esempi di come utilizzare questa funzione.

Example Esempio: una semplice query che utilizza le funzioni di Amazon Comprehend

Di seguito è riportato l'esempio di una semplice query che richiama la funzione per valutare la soddisfazione del cliente nei confronti del team di supporto. Supponi di avere una tabella di database (`support`) che archivia il feedback dei clienti dopo ogni richiesta di assistenza. Questa query di esempio applica la funzione `aws_comprehend.detect_sentiment` al testo nella colonna `feedback` della tabella e restituisce il sentiment e il livello di confidenza del sentiment. Inoltre i risultati vengono restituiti in ordine decrescente.

```
SELECT feedback, s.sentiment,s.confidence
    FROM support,aws_comprehend.detect_sentiment(feedback, 'en') s
    ORDER BY s.confidence DESC;
 feedback                          | sentiment | confidence
 ------------------------------------------------------------+-----------+------------
 Thank you for the excellent customer support!              | POSITIVE  |   0.999771
 The latest version of this product stinks!                 | NEGATIVE  |   0.999184
 Your support team is just awesome! I am blown away.        | POSITIVE  |   0.997774
 Your product is too complex, but your support is great.    | MIXED     |   0.957958
 Your support tech helped me in fifteen minutes.            | POSITIVE  |   0.949491
 My problem was never resolved!                             | NEGATIVE  |   0.920644
 When will the new version of this product be released?     | NEUTRAL   |   0.902706
 I cannot stand that chatbot.                               | NEGATIVE  |   0.895219
 Your support tech talked down to me.                       | NEGATIVE  |   0.868598
 It took me way too long to get a real person.              | NEGATIVE  |   0.481805

 (10 rows)
```

Per evitare che ti venga addebitato il rilevamento del sentiment più di una volta per riga della tabella, puoi materializzare i risultati. Esegui questa operazione sulle righe di interesse. Ad esempio, le note del medico vengono aggiornate in modo che solo quelle in francese (`fr`) utilizzino la funzione di rilevamento del sentiment.

```
UPDATE clinician_notes
SET sentiment = (aws_comprehend.detect_sentiment (french_notes, 'fr')).sentiment,
    confidence = (aws_comprehend.detect_sentiment (french_notes, 'fr')).confidence
WHERE
    clinician_notes.french_notes IS NOT NULL AND
    LENGTH(TRIM(clinician_notes.french_notes)) > 0 AND
```

```
    clinician_notes.sentiment IS NULL;
```

Per ulteriori informazioni sull'ottimizzazione delle chiamate di funzione, consulta Considerazioni sulle prestazioni per l'utilizzo del machine learning di Aurora con Aurora PostgreSQL.

Utilizzo dell' SageMaker intelligenza artificiale con il cluster Aurora PostgreSQL DB

Dopo aver configurato l'ambiente di SageMaker intelligenza artificiale e aver effettuato l'integrazione con Aurora PostgreSQL come Configurazione di Aurora PostgreSQL per utilizzare Amazon AI SageMaker descritto in, puoi richiamare le operazioni utilizzando la funzione. `aws_sagemaker.invoke_endpoint` La funzione `aws_sagemaker.invoke_endpoint` si connette a un solo endpoint del modello nella stessa Regione AWS. Se la tua istanza di database ha repliche multiple, Regioni AWS assicurati di configurare e distribuire ogni modello di intelligenza artificiale su tutti. SageMaker Regione AWS

Le chiamate a `aws_sagemaker.invoke_endpoint` vengono autenticate utilizzando il ruolo IAM che hai impostato per associare il cluster Aurora PostgreSQL DB al servizio AI e all'endpoint che hai SageMaker fornito durante il processo di configurazione. SageMaker Gli endpoint del modello AI sono limitati a un singolo account e non sono pubblici. L'`endpoint_name`URL non contiene l'ID dell'account. SageMaker L'IA determina l'ID dell'account dal token di autenticazione fornito dal ruolo SageMaker AI IAM dell'istanza del database.

aws_sagemaker.invoke_endpoint

Questa funzione accetta l'endpoint SageMaker AI come input e il numero di righe che devono essere elaborate come batch. Prende anche come input i vari parametri previsti dall'endpoint del modello SageMaker AI. La documentazione di riferimento della funzione è indicata di seguito.

```
aws_sagemaker.invoke_endpoint(
  IN endpoint_name varchar,
  IN max_rows_per_batch int,
  VARIADIC model_input "any",
  OUT model_output varchar
  )
```

Gli input e gli output di questa funzione sono indicati di seguito.

- `endpoint_name`— Un URL di endpoint indipendente. Regione AWS

- `max_rows_per_batch`: il numero massimo di righe per batch per l'elaborazione in modalità batch. Per ulteriori informazioni, consulta Informazioni sulla modalità batch e sulle funzioni di machine learning di Aurora.
- `model_input`: uno o più parametri di input per il modello. Questi possono essere qualsiasi tipo di dati necessario al modello di SageMaker intelligenza artificiale. PostgreSQL consente di specificare fino a 100 parametri di input per una funzione. I tipi di dati dell'array devono essere unidimensionali, ma possono contenere tutti gli elementi previsti dal modello di SageMaker intelligenza artificiale. Il numero di input per un modello SageMaker AI è limitato solo dal limite di dimensione dei messaggi SageMaker AI di 6 MB.
- `model_output`— L'output del modello SageMaker AI come testo.

Creazione di una funzione definita dall'utente per richiamare un SageMaker modello di intelligenza artificiale

Crea una funzione definita dall'utente separata da chiamare `aws_sagemaker.invoke_endpoint` per ciascuno dei tuoi SageMaker modelli di intelligenza artificiale. La funzione definita dall'utente rappresenta l'endpoint SageMaker AI che ospita il modello. La funzione `aws_sagemaker.invoke_endpoint` viene eseguita all'interno della funzione definita dall'utente. Le funzioni definite dall'utente offrono molti vantaggi:

- Puoi dare un nome al tuo modello di SageMaker intelligenza artificiale invece di richiamare solo tutti `aws_sagemaker.invoke_endpoint` i tuoi modelli di SageMaker intelligenza artificiale.
- Puoi specificare l'URL dell'endpoint del modello in una sola posizione nel codice dell'applicazione SQL.
- Puoi controllare i privilegi `EXECUTE` per ogni funzione di machine learning di Aurora in modo indipendente.
- Puoi dichiarare i tipi di input e output del modello utilizzando i tipi SQL. SQL impone il numero e il tipo di argomenti passati al modello di SageMaker intelligenza artificiale ed esegue la conversione dei tipi, se necessario. L'utilizzo dei tipi SQL si `SQL NULL` tradurrà anche nel valore predefinito appropriato previsto dal modello di SageMaker intelligenza artificiale.
- Puoi ridurre la dimensione massima del batch se desideri restituire le prime righe un po' più velocemente.

Per specificare una funzione definita dall'utente, utilizza l'istruzione SQL Data Definition Language (DDL) `CREATE FUNCTION`. Quando definisci la funzione, specifica quanto segue:

- I parametri di input per il modello.
- L'endpoint SageMaker AI specifico da richiamare.
- Il tipo restituito.

La funzione definita dall'utente restituisce l'inferenza calcolata dall'endpoint SageMaker AI dopo aver eseguito il modello sui parametri di input. L'esempio seguente crea una funzione definita dall'utente per un modello di SageMaker intelligenza artificiale con due parametri di input.

```
CREATE FUNCTION classify_event (IN arg1 INT, IN arg2 DATE, OUT category INT)
AS $$
    SELECT aws_sagemaker.invoke_endpoint (
        'sagemaker_model_endpoint_name', NULL,
        arg1, arg2                          -- model inputs are separate arguments
        )::INT                              -- cast the output to INT
$$ LANGUAGE SQL PARALLEL SAFE COST 5000;
```

Tenere presente quanto segue:

- L'input della funzione `aws_sagemaker.invoke_endpoint` può essere uno o più parametri di qualsiasi tipo di dati.
- In questo esempio viene utilizzato un tipo di output INT. Se esegui il cast dell'output da un tipo `varchar` a un tipo diverso, è necessario eseguire il cast a un tipo scalare incorporato PostgreSQL come `INTEGER`, `REAL`, `FLOAT` o `NUMERIC`. Per ulteriori informazioni su questi tipi, consulta Tipi di dati nella documentazione PostgreSQL.
- Specifica `PARALLEL SAFE` per abilitare l'esecuzione di query parallele. Per ulteriori informazioni, consulta Miglioramento dei tempi di risposta con l'elaborazione di query parallela.
- Specificare `COST 5000` per stimare il costo di esecuzione della funzione. Utilizza un numero positivo che fornisce i costi di esecuzione stimati della funzione, in unità di `cpu_operator_cost`.

Passare un array come input a un modello SageMaker AI

La funzione `aws_sagemaker.invoke_endpoint` può avere fino a 100 parametri di input, che è il limite per le funzioni PostgreSQL. Se il modello SageMaker AI richiede più di 100 parametri dello stesso tipo, passa i parametri del modello come array.

L'esempio seguente definisce una funzione che passa un array come input al modello di regressione SageMaker AI. L'output viene trasmesso a un valore `REAL`.

```
CREATE FUNCTION regression_model (params REAL[], OUT estimate REAL)
AS $$
    SELECT aws_sagemaker.invoke_endpoint (
      'sagemaker_model_endpoint_name',
      NULL,
      params
      )::REAL
$$ LANGUAGE SQL PARALLEL SAFE COST 5000;
```

Specificazione della dimensione del batch quando si richiama un modello di intelligenza artificiale SageMaker

L'esempio seguente crea una funzione definita dall'utente per un modello di SageMaker intelligenza artificiale che imposta la dimensione del batch predefinita su NULL. La funzione consente inoltre di fornire una dimensione batch diversa quando viene invocata.

```
CREATE FUNCTION classify_event (
    IN event_type INT, IN event_day DATE, IN amount REAL, -- model inputs
    max_rows_per_batch INT DEFAULT NULL,  -- optional batch size limit
    OUT category INT)                     -- model output
AS $$
    SELECT aws_sagemaker.invoke_endpoint (
        'sagemaker_model_endpoint_name', max_rows_per_batch,
        event_type, event_day, COALESCE(amount, 0.0)
        )::INT              -- casts output to type INT
$$ LANGUAGE SQL PARALLEL SAFE COST 5000;
```

Tieni presente quanto segue:

- Utilizza il parametro `max_rows_per_batch` opzionale per fornire il controllo del numero di righe per una chiamata della funzione in modalità batch. Se utilizzi un valore NULL, l'ottimizzatore di query sceglie automaticamente la dimensione batch massima. Per ulteriori informazioni, consulta Informazioni sulla modalità batch e sulle funzioni di machine learning di Aurora.
- Per impostazione predefinita, il passaggio di NULL come valore di un parametro viene tradotto in una stringa vuota prima di passare a AI. SageMaker Per questo esempio i tipi di input sono diversi.
- Se si dispone di un input non di testo o di un input di testo il cui valore predefinito deve essere diverso da una stringa vuota, utilizza l'istruzione `COALESCE`. Utilizza `COALESCE` per convertire NULL nel valore di sostituzione null desiderato nella chiamata a

`aws_sagemaker.invoke_endpoint`. Per il parametro `amount` in questo esempio, un valore NULL viene convertito in 0.0.

Invocare un modello di SageMaker intelligenza artificiale con più output

L'esempio seguente crea una funzione definita dall'utente per un modello di SageMaker intelligenza artificiale che restituisce più output. La funzione deve eseguire il cast dell'output della funzione `aws_sagemaker.invoke_endpoint` a un tipo di dati corrispondente. Ad esempio, puoi utilizzare il tipo di punto PostgreSQL incorporato per coppie (x, y) o un tipo composito definito dall'utente.

Questa funzione definita dall'utente restituisce valori da un modello che restituisce più output utilizzando un tipo composito per gli output.

```
CREATE TYPE company_forecasts AS (
    six_month_estimated_return real,
    one_year_bankruptcy_probability float);
CREATE FUNCTION analyze_company (
    IN free_cash_flow NUMERIC(18, 6),
    IN debt NUMERIC(18,6),
    IN max_rows_per_batch INT DEFAULT NULL,
    OUT prediction company_forecasts)
AS $$
    SELECT (aws_sagemaker.invoke_endpoint('endpt_name',
       max_rows_per_batch,free_cash_flow, debt))::company_forecasts;


$$ LANGUAGE SQL PARALLEL SAFE COST 5000;
```

Per il tipo composito, utilizza i campi nello stesso ordine in cui vengono visualizzati nell'output del modello ed esegui il cast dell'output del `aws_sagemaker.invoke_endpoint` al tipo composito. Il chiamante può estrarre i singoli campi per nome o con notazione PostgreSQL ".*".

Esportazione di dati in Amazon S3 SageMaker per la formazione su modelli di intelligenza artificiale (Advanced)

Ti consigliamo di acquisire familiarità con l'apprendimento automatico e l' SageMaker intelligenza artificiale di Aurora utilizzando gli algoritmi e gli esempi forniti anziché provare ad addestrare i tuoi modelli. Per ulteriori informazioni, consulta Get Started with Amazon SageMaker AI

Per addestrare i modelli di SageMaker intelligenza artificiale, esporti i dati in un bucket Amazon S3. Il bucket Amazon S3 viene utilizzato dall' SageMaker intelligenza artificiale per addestrare il modello prima che venga distribuito. Puoi eseguire query dei dati da un cluster di database Aurora PostgreSQL e salvarli direttamente nei file di testo archiviati in un bucket Amazon S3. Quindi l' SageMaker intelligenza artificiale utilizza i dati del bucket Amazon S3 per la formazione. Per ulteriori informazioni sulla formazione dei modelli di SageMaker intelligenza artificiale, consulta Addestrare un modello con Amazon SageMaker AI.

> **Note**
>
> Quando crei un bucket Amazon S3 per la formazione sui modelli di SageMaker intelligenza artificiale o il punteggio in batch, utilizzalo nel nome del bucket `sagemaker` Amazon S3. Per ulteriori informazioni, consulta Specificare un bucket Amazon S3 per caricare set di dati di formazione e archiviare i dati di output nella Amazon SageMaker AI Developer Guide.

Per ulteriori informazioni sull'esportazione dei dati, consulta Esportazione di dati da del cluster di database Aurora PostgreSQLRDS per PostgreSQL a Amazon S3.

Considerazioni sulle prestazioni per l'utilizzo del machine learning di Aurora con Aurora PostgreSQL

I servizi Amazon Comprehend e SageMaker AI svolgono la maggior parte del lavoro quando vengono richiamati da una funzione di machine learning di Aurora. Ciò significa che è possibile dimensionare tali risorse in base alle esigenze, in modo indipendente. Per il cluster database Aurora PostgreSQL, puoi ottenere il massimo dell'efficienza dalle tue chiamate di funzione. Di seguito sono riportate alcune considerazioni sulle prestazioni da tenere presenti quando si usa il machine learning di Aurora da Aurora PostgreSQL.

Argomenti

- Informazioni sulla modalità batch e sulle funzioni di machine learning di Aurora
- Miglioramento dei tempi di risposta con l'elaborazione di query parallela
- Utilizzo di viste materializzate e colonne materializzate

Informazioni sulla modalità batch e sulle funzioni di machine learning di Aurora

In genere PostgreSQL esegue le funzioni una riga alla volta. Il machine learning di Aurora può ridurre questo sovraccarico combinando in batch le chiamate al servizio di machine learning di Aurora esterno per molte righe con un approccio chiamato esecuzione in modalità batch. In modalità batch, Aurora Machine Learning riceve le risposte per un batch di righe di input e quindi restituisce le risposte alla query in esecuzione una riga alla volta. Questa ottimizzazione migliora il throughput delle query Aurora senza limitare l'ottimizzatore di query PostgreSQL.

Aurora utilizza automaticamente la modalità batch se si fa riferimento alla funzione dall'elenco `SELECT`, da una clausola `WHERE` o da una clausola `HAVING`. Si noti che le espressioni `CASE` semplici di primo livello sono idonee per l'esecuzione in modalità batch. Anche le espressioni `CASE` ricercate di primo livello sono idonee per l'esecuzione in modalità batch, a condizione che la prima clausola `WHEN` sia un predicato semplice con una chiamata alla funzione in modalità batch.

La funzione definita dall'utente deve essere `LANGUAGE SQL` e deve specificare `PARALLEL SAFE` e `COST 5000`.

Migrazione di funzioni dall'istruzione SELECT alla clausola FROM

In genere, una funzione `aws_ml` che è idonea per l'esecuzione in modalità batch viene migrata automaticamente da Aurora alla clausola `FROM`.

La migrazione delle funzioni in modalità batch idonee alla clausola FROM può essere esaminata manualmente a livello di query. A tale scopo, utilizza le istruzioni EXPLAIN (e ANALYZE e VERBOSE) e trova le informazioni "Elaborazione in batch" sotto ogni modalità batch `Function Scan`. Puoi inoltre utilizzare EXPLAIN (con VERBOSE) senza eseguire la query. Osserva quindi se le chiamate alla funzione vengono visualizzate come un `Function Scan` in un join loop nidificato che non è stato specificato nell'istruzione originale.

Nell'esempio seguente, l'operatore join loop nidificato nel piano mostra che Aurora ha migrato la funzione `anomaly_score`. Ha migrato questa funzione dall'elenco SELECT alla clausola FROM, dove è idonea per l'esecuzione in modalità batch.

```
EXPLAIN (VERBOSE, COSTS false)
SELECT anomaly_score(ts.R.description) from ts.R;
                          QUERY PLAN
---------------------------------------------------------------
 Nested Loop
   Output: anomaly_score((r.description)::text)
   ->  Seq Scan on ts.r
```

```
        Output: r.id, r.description, r.score
  ->  Function Scan on public.anomaly_score
        Output: anomaly_score.anomaly_score
        Function Call: anomaly_score((r.description)::text)
```

Per disabilitare l'esecuzione in modalità batch, imposta il parametro `apg_enable_function_migration` su `false`. Ciò impedisce la migrazione delle funzioni `aws_ml` da SELECT alla clausola `FROM`. L'esempio seguente mostra come eseguire questa operazione.

```
SET apg_enable_function_migration = false;
```

Il parametro `apg_enable_function_migration` è di tipo GUC (Grand Unified Configuration) riconosciuto dall'estensione Aurora PostgreSQL `apg_plan_mgmt` per la gestione del piano di query. Per disabilitare la migrazione di funzioni in una sessione, utilizza la gestione del piano di query per salvare il piano risultante come un piano `approved`. In fase di runtime, la gestione del piano di query applica il piano `approved` con la relativa impostazione `apg_enable_function_migration`. Questa imposizione si verifica a prescindere dall'impostazione del parametro GUC `apg_enable_function_migration`. Per ulteriori informazioni, consulta Gestione dei piani di esecuzione delle query per Aurora PostgreSQL.

Utilizzo del parametro max_rows_per_batch

Entrambe le funzioni `aws_comprehend.detect_sentiment` e `aws_sagemaker.invoke_endpoint` hanno un parametro `max_rows_per_batch`. Questo parametro specifica il numero di righe che possono essere inviate al servizio di machine learning di Aurora. Più grande è il set di dati elaborato dalla funzione, maggiore può essere la dimensione del batch.

Le funzioni in modalità batch migliorano l'efficienza creando batch di righe che distribuiscono il costo delle chiamate di funzione Aurora Machine Learning su un numero elevato di righe. Tuttavia, se un'istruzione `SELECT` termina in anticipo a causa di una clausola `LIMIT`, il batch può essere costruito su più righe rispetto a quelle utilizzate dalla query. Questo approccio può comportare costi aggiuntivi sul tuo account. AWS Per ottenere i vantaggi dell'esecuzione in modalità batch, ma evitare di creare batch troppo grandi, utilizza un valore più piccolo per il parametro `max_rows_per_batch` nelle chiamate di funzione.

Se esegui un `EXPLAIN (VERBOSE, ANALYZE)` di una query che utilizza l'esecuzione in modalità batch, viene visualizzato un operatore `FunctionScan` che si trova sotto un join loop nidificato.

Il numero di loop segnalati da EXPLAIN indica il numero di volte che una riga è stata recuperata dall'operatore FunctionScan. Se un'istruzione utilizza una clausola LIMIT, il numero di recuperi è coerente. Per ottimizzare le dimensioni del batch, imposta il parametro max_rows_per_batch su questo valore. Tuttavia, se si fa riferimento alla funzione in modalità batch in un predicato nella clausola WHERE o HAVING, probabilmente non è possibile conoscere il numero di recuperi in anticipo. In questo caso, utilizza i loop come una linea guida ed esegui esperimenti con max_rows_per_batch per trovare un'impostazione che ottimizzi le prestazioni.

Verifica dell'esecuzione in modalità batch

Per verificare se una funzione è stata eseguita in modalità batch, utilizzare EXPLAIN ANALYZE. Se è stata utilizzata l'esecuzione in modalità batch, il piano di query includerà le informazioni in una sezione "Elaborazione in batch".

```
EXPLAIN ANALYZE SELECT user-defined-function();
 Batch Processing: num batches=1 avg/min/max batch size=3333.000/3333.000/3333.000
                                 avg/min/max batch call time=146.273/146.273/146.273
```

Questo esempio prevedeva 1 batch che conteneva 3.333 righe, che richiedevano 146,273 ms per l'elaborazione. La sezione "Elaborazione in batch" mostra quanto segue:

- Il numero di batch per questa operazione di scansione funzione
- La dimensione media, minima e massima del batch
- Il tempo di esecuzione medio, minimo e massimo del batch

In genere le dimensioni del batch finale sono inferiore al resto. Questo si traduce spesso in una dimensione batch minima molto inferiore rispetto alla media.

Per restituire le prime righe più rapidamente, imposta il parametro max_rows_per_batch su un valore più piccolo.

Per ridurre il numero di chiamate in modalità batch al servizio ML quando utilizzi un LIMIT nella funzione definita dall'utente, imposta il parametro max_rows_per_batch su un valore inferiore.

Miglioramento dei tempi di risposta con l'elaborazione di query parallela

Per ottenere risultati il più velocemente possibile da un numero elevato di righe, puoi combinare l'elaborazione di query parallela con l'elaborazione in modalità batch. Puoi utilizzare l'elaborazione

di query parallela per istruzioni `SELECT`, `CREATE TABLE AS SELECT` e `CREATE MATERIALIZED VIEW`.

> **Note**
>
> PostgreSQL non supporta ancora la query parallela per le istruzioni DML (Data Manipulation Language).

L'elaborazione di query parallela avviene sia all'interno del database sia all'interno del servizio ML. Il numero di core nella classe di istanza del database limita il grado di parallelismo che può essere utilizzato durante l'esecuzione di query. Il server di database può costruire un piano di esecuzione di query parallela che partiziona l'attività tra un set di lavoratori paralleli. Quindi ciascuno di questi lavoratori può creare richieste in batch contenenti decine di migliaia di righe (o il numero consentito da ciascun servizio).

Le richieste in batch di tutti i parallel worker vengono inviate all'endpoint SageMaker AI. Il grado di parallelismo supportato dall'endpoint è limitato dal numero e dal tipo di istanze che lo supportano. Per i gradi K di parallelismo, è necessaria una classe di istanza database con almeno i core K. È inoltre necessario configurare l'endpoint SageMaker AI per il modello in modo che abbia K istanze iniziali di una classe di istanze sufficientemente performante.

Per utilizzare l'elaborazione di query parallela, puoi impostare il parametro di archiviazione `parallel_workers` della tabella contenente i dati che intendi passare. Imposta `parallel_workers` su una funzione in modalità batch, ad esempio `aws_comprehend.detect_sentiment`. Se l'ottimizzatore sceglie un piano di query parallelo, i servizi AWS ML possono essere chiamati sia in batch che in parallelo.

Puoi utilizzare i seguenti parametri con la funzione `aws_comprehend.detect_sentiment` per ottenere un piano con parallelismo a quattro vie. Se modifichi uno dei due parametri seguenti, è necessario riavviare l'istanza database per rendere effettive le modifiche.

```
-- SET max_worker_processes to 8;  -- default value is 8
-- SET max_parallel_workers to 8;  -- not greater than max_worker_processes
SET max_parallel_workers_per_gather to 4;  -- not greater than max_parallel_workers

-- You can set the parallel_workers storage parameter on the table that the data
-- for the Aurora machine learning function is coming from in order to manually
 override the degree of
```

```
-- parallelism that would otherwise be chosen by the query optimizer
--
ALTER TABLE yourTable SET (parallel_workers = 4);

-- Example query to exploit both batch-mode execution and parallel query
EXPLAIN (verbose, analyze, buffers, hashes)
SELECT aws_comprehend.detect_sentiment(description, 'en')).*
FROM yourTable
WHERE id < 100;
```

Per ulteriori informazioni sul controllo della query parallela, consulta Parallel plans (Piani paralleli) nella documentazione di PostgreSQL.

Utilizzo di viste materializzate e colonne materializzate

Quando richiami un AWS servizio come SageMaker AI o Amazon Comprehend dal tuo database, il tuo account viene addebitato in base alla politica tariffaria di quel servizio. Per ridurre al minimo gli addebiti sul tuo account, puoi inserire il risultato della chiamata al AWS servizio in una colonna materializzata in modo che il AWS servizio non venga richiamato più di una volta per riga di input. Se lo desideri, puoi aggiungere una colonna timestamp `materializedAt` per registrare l'ora in cui le colonne sono state materializzate.

La latenza di un'istruzione `INSERT` a riga singola ordinaria è in genere molto inferiore rispetto alla latenza di chiamata di una funzione in modalità batch. Pertanto, potrebbe non essere possibile soddisfare i requisiti di latenza dell'applicazione se si invoca la funzione in modalità batch per ogni singola riga `INSERT` eseguita dall'applicazione. Per concretizzare il risultato della chiamata di un AWS servizio in una colonna materializzata, le applicazioni ad alte prestazioni in genere devono compilare le colonne materializzate. A tale scopo, inviano periodicamente un'istruzione `UPDATE` che opera contemporaneamente su un batch di righe di grandi dimensioni.

`UPDATE` accetta un blocco a livello di riga che può influire su un'applicazione in esecuzione. Quindi potrebbe essere necessario utilizzare `SELECT ... FOR UPDATE SKIP LOCKED` o `MATERIALIZED VIEW`.

Le query analitiche che agiscono su un numero elevato di righe in tempo reale possono combinare la materializzazione in modalità batch con l'elaborazione in tempo reale. A tale scopo, queste query assemblano un `UNION ALL` dei risultati pre-materializzati con una query sulle righe che non dispongono ancora di risultati materializzati. In alcuni casi, tale `UNION ALL` è necessario in più posizioni o la query viene generata da un'applicazione di terze parti. In tal caso, puoi creare un `VIEW`

per incapsulare l'operazione `UNION ALL` in modo che questo dettaglio non venga esposto al resto dell'applicazione SQL.

Puoi utilizzare una vista materializzata per materializzare i risultati di un'istruzione `SELECT` arbitraria in una snapshot nel tempo. Puoi inoltre utilizzarla per aggiornare la vista materializzata in qualsiasi momento in futuro. Attualmente PostgreSQL non supporta l'aggiornamento incrementale, quindi la vista materializzata viene completamente ricalcolata ogni volta che viene aggiornata.

Puoi aggiornare le viste materializzate con l'opzione `CONCURRENTLY`, che aggiorna il contenuto della vista materializzata senza acquisire un blocco esclusivo. In questo modo un'applicazione SQL può leggere dalla vista materializzata durante l'aggiornamento.

Monitoraggio del machine learning di Aurora

È possibile monitorare le funzioni `aws_ml` impostando su `all` il parametro `track_functions` nel gruppo di parametri personalizzato del cluster database. Per impostazione predefinita, questo parametro è impostato su `pl`, il che significa che vengono monitorate solo le funzioni del linguaggio procedurale. Modificando questa impostazione su `all`, vengono monitorate anche le funzioni `aws_ml`. Per ulteriori informazioni, consulta Run-time Statistics (Statistiche di runtime) nella documentazione di PostgreSQL.

Per informazioni sul monitoraggio delle prestazioni delle operazioni di SageMaker intelligenza artificiale richiamate dalle funzioni di machine learning di Aurora, consulta Monitor Amazon SageMaker AI nella Amazon SageMaker AI Developer Guide.

Con `track_functions` impostato su `all`, puoi eseguire le query nella visualizzazione `pg_stat_user_functions` per ottenere le statistiche sulle funzioni che definisci e utilizzi per richiamare i servizi di machine learning di Aurora. La visualizzazione fornisce il numero di `calls`, `total_time` e `self_time` per ogni funzione.

Per visualizzare le statistiche per le funzioni `aws_sagemaker.invoke_endpoint` e `aws_comprehend.detect_sentiment` puoi filtrare i risultati in base al nome dello schema utilizzando la seguente query.

```
SELECT * FROM pg_stat_user_functions
    WHERE schemaname
    LIKE 'aws_%';
```

Per cancellare le statistiche, procedi come indicato di seguito.

```
SELECT pg_stat_reset();
```

Puoi ottenere i nomi delle funzioni SQL che chiamano la funzione `aws_sagemaker.invoke_endpoint` eseguendo query nel catalogo di sistema `pg_proc` PostgreSQL. Questo catalogo contiene informazioni su funzioni, procedure e altri elementi. Per ulteriori informazioni, consulta pg_proc nella documentazione di PostgreSQL. Di seguito è riportato un esempio di query nella tabella per ottenere i nomi delle funzioni (`proname`) la cui origine (`prosrc`) include il testo invoke_endpoint.

```
SELECT proname FROM pg_proc WHERE prosrc LIKE '%invoke_endpoint%';
```

Esempi di codice per l'utilizzo di Aurora AWS SDKs

I seguenti esempi di codice mostrano come usare Aurora con un kit di sviluppo AWS software (SDK).

Nozioni di base: esempi di codice che mostrano come eseguire le operazioni essenziali all'interno di un servizio.

Le azioni sono estratti di codice da programmi più grandi e devono essere eseguite nel contesto. Sebbene le azioni mostrino come richiamare le singole funzioni del servizio, è possibile visualizzarle contestualizzate negli scenari correlati.

Scenari: esempi di codice che mostrano come eseguire un'attività specifica chiamando più funzioni all'interno dello stesso servizio o combinate con altri Servizi AWS.

Per un elenco completo delle guide per sviluppatori AWS SDK e degli esempi di codice, consulta. Utilizzo di questo servizio con un AWS SDK Questo argomento include anche informazioni su come iniziare e dettagli sulle versioni precedenti dell'SDK.

Nozioni di base

Hello Aurora

Gli esempi di codice seguenti mostrano come iniziare a utilizzare Aurora.

.NET

SDK per .NET(v4)

> **Note**
>
> C'è altro da fare. GitHub Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice AWS.

```
using Amazon.RDS;
using Amazon.RDS.Model;
using Microsoft.Extensions.DependencyInjection;
using Microsoft.Extensions.Hosting;
```

```
namespace AuroraActions;

public static class HelloAurora
{
    static async Task Main(string[] args)
    {
        // Use the AWS .NET Core Setup package to set up dependency injection for
 the
        // Amazon Relational Database Service (Amazon RDS).
        // Use your AWS profile name, or leave it blank to use the default
 profile.
        using var host = Host.CreateDefaultBuilder(args)
            .ConfigureServices((_, services) =>
                services.AddAWSService<IAmazonRDS>()
            ).Build();

        // Now the client is available for injection. Fetching it directly here
 for example purposes only.
        var rdsClient = host.Services.GetRequiredService<IAmazonRDS>();

        // You can use await and any of the async methods to get a response.
        var response = await rdsClient.DescribeDBClustersAsync(new
 DescribeDBClustersRequest { IncludeShared = true });
        Console.WriteLine($"Hello Amazon RDS Aurora! Let's list some clusters in
 this account:");
        if (response.DBClusters == null)
        {
            Console.WriteLine($"\tNo clusters found.");
        }
        else
        {
            foreach (var cluster in response.DBClusters)
            {
                Console.WriteLine(
                    $"\tCluster: database: {cluster.DatabaseName} identifier:
 {cluster.DBClusterIdentifier}.");
            }
        }
    }
}
```

- Per i dettagli sull'API, consulta Descrivi DBClusters in AWS SDK per .NETAPI Reference.

C++

SDK per C++

> **Note**
>
> C'è altro su GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice AWS.

Codice per il CMake file CMake Lists.txt.

```
# Set the minimum required version of CMake for this project.
cmake_minimum_required(VERSION 3.13)

# Set the AWS service components used by this project.
set(SERVICE_COMPONENTS rds)

# Set this project's name.
project("hello_aurora")

# Set the C++ standard to use to build this target.
# At least C++ 11 is required for the AWS SDK for C++.
set(CMAKE_CXX_STANDARD 11)

# Use the MSVC variable to determine if this is a Windows build.
set(WINDOWS_BUILD ${MSVC})

if (WINDOWS_BUILD) # Set the location where CMake can find the installed
 libraries for the AWS SDK.
    string(REPLACE ";" "/aws-cpp-sdk-all;" SYSTEM_MODULE_PATH
 "${CMAKE_SYSTEM_PREFIX_PATH}/aws-cpp-sdk-all")
    list(APPEND CMAKE_PREFIX_PATH ${SYSTEM_MODULE_PATH})
endif ()

# Find the AWS SDK for C++ package.
find_package(AWSSDK REQUIRED COMPONENTS ${SERVICE_COMPONENTS})

if (WINDOWS_BUILD AND AWSSDK_INSTALL_AS_SHARED_LIBS)
     # Copy relevant AWS SDK for C++ libraries into the current binary directory
 for running and debugging.
```

```
    # set(BIN_SUB_DIR "/Debug") # If you are building from the command line, you
 may need to uncomment this
                                    # and set the proper subdirectory to the
 executables' location.

    AWSSDK_CPY_DYN_LIBS(SERVICE_COMPONENTS ""
 ${CMAKE_CURRENT_BINARY_DIR}${BIN_SUB_DIR})
endif ()

add_executable(${PROJECT_NAME}
        hello_aurora.cpp)

target_link_libraries(${PROJECT_NAME}
        ${AWSSDK_LINK_LIBRARIES})
```

Codice per il file origine hello_aurora.cpp.

```
#include <aws/core/Aws.h>
#include <aws/rds/RDSClient.h>
#include <aws/rds/model/DescribeDBClustersRequest.h>
#include <iostream>

/*
 *  A "Hello Aurora" starter application which initializes an Amazon Relational
 Database Service (Amazon RDS) client
 *  and describes the Amazon Aurora (Aurora) clusters.
 *
 *  main function
 *
 *  Usage: 'hello_aurora'
 *
 */
int main(int argc, char **argv) {
    Aws::SDKOptions options;
    // Optionally change the log level for debugging.
//   options.loggingOptions.logLevel = Utils::Logging::LogLevel::Debug;
    Aws::InitAPI(options); // Should only be called once.
    int result = 0;
    {
        Aws::Client::ClientConfiguration clientConfig;
        // Optional: Set to the AWS Region (overrides config file).
        // clientConfig.region = "us-east-1";
```

```
        Aws::RDS::RDSClient rdsClient(clientConfig);

        Aws::String marker; // Used for pagination.
        std::vector<Aws::String> clusterIds;
        do {
            Aws::RDS::Model::DescribeDBClustersRequest request;

            Aws::RDS::Model::DescribeDBClustersOutcome outcome =
                    rdsClient.DescribeDBClusters(request);

            if (outcome.IsSuccess()) {
                for (auto &cluster: outcome.GetResult().GetDBClusters()) {
                    clusterIds.push_back(cluster.GetDBClusterIdentifier());
                }
                marker = outcome.GetResult().GetMarker();
            } else {
                result = 1;
                std::cerr << "Error with Aurora::GDescribeDBClusters. "
                          << outcome.GetError().GetMessage()
                          << std::endl;
                break;
            }
        } while (!marker.empty());

        std::cout << clusterIds.size() << " Aurora clusters found." << std::endl;
        for (auto &clusterId: clusterIds) {
            std::cout << "  clusterId " << clusterId << std::endl;
        }
    }

    Aws::ShutdownAPI(options); // Should only be called once.
    return 0;
}
```

- Per i dettagli sull'API, consulta Descrivi DBClusters in AWS SDK per C++API Reference.

Go

SDK per Go V2

> **Note**
>
> C'è altro su GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice AWS.

```
package main

import (
 "context"
 "fmt"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/config"
 "github.com/aws/aws-sdk-go-v2/service/rds"
)

// main uses the AWS SDK for Go V2 to create an Amazon Aurora client and list up
 to 20
// DB clusters in your account.
// This example uses the default settings specified in your shared credentials
// and config files.
func main() {
 ctx := context.Background()
 sdkConfig, err := config.LoadDefaultConfig(ctx)
 if err != nil {
  fmt.Println("Couldn't load default configuration. Have you set up your AWS
 account?")
  fmt.Println(err)
  return
 }
 auroraClient := rds.NewFromConfig(sdkConfig)
 const maxClusters = 20
 fmt.Printf("Let's list up to %v DB clusters.\n", maxClusters)
 output, err := auroraClient.DescribeDBClusters(
  ctx, &rds.DescribeDBClustersInput{MaxRecords: aws.Int32(maxClusters)})
 if err != nil {
```

```
  fmt.Printf("Couldn't list DB clusters: %v\n", err)
  return
 }
 if len(output.DBClusters) == 0 {
  fmt.Println("No DB clusters found.")
 } else {
  for _, cluster := range output.DBClusters {
   fmt.Printf("DB cluster %v has database %v.\n", *cluster.DBClusterIdentifier,
    *cluster.DatabaseName)
  }
 }
}
```

- Per i dettagli sull'API, consulta Descrivi DBClusters in AWS SDK per GoAPI Reference.

Java

SDK per Java 2.x

> **Note**
>
> C'è altro su GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice AWS.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.rds.RdsClient;
import software.amazon.awssdk.services.rds.paginators.DescribeDBClustersIterable;

public class DescribeDbClusters {
    public static void main(String[] args) {
        Region region = Region.US_EAST_1;
        RdsClient rdsClient = RdsClient.builder()
                .region(region)
                .build();

        describeClusters(rdsClient);
        rdsClient.close();
    }
```

```
    public static void describeClusters(RdsClient rdsClient) {
        DescribeDBClustersIterable clustersIterable =
 rdsClient.describeDBClustersPaginator();
        clustersIterable.stream()
                .flatMap(r -> r.dbClusters().stream())
                .forEach(cluster -> System.out
                        .println("Database name: " + cluster.databaseName() + "
 Arn = " + cluster.dbClusterArn()));
    }
}
```

- Per i dettagli sull'API, consulta Descrivi DBClusters in AWS SDK for Java 2.xAPI Reference.

Python

SDK per Python (Boto3)

> **Note**
>
> C'è altro su GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice AWS.

```
import boto3

# Create an RDS client
rds = boto3.client("rds")

# Create a paginator for the describe_db_clusters operation
paginator = rds.get_paginator("describe_db_clusters")

# Use the paginator to get a list of DB clusters
response_iterator = paginator.paginate(
    PaginationConfig={
        "PageSize": 50,  # Adjust PageSize as needed
        "StartingToken": None,
    }
)

# Iterate through the pages of the response
clusters_found = False
```

```
for page in response_iterator:
    if "DBClusters" in page and page["DBClusters"]:
        clusters_found = True
        print("Here are your RDS Aurora clusters:")
        for cluster in page["DBClusters"]:
            print(
                f"Cluster ID: {cluster['DBClusterIdentifier']}, Engine:
 {cluster['Engine']}"
            )

if not clusters_found:
    print("No clusters found!")
```

- Per i dettagli sull'API, consulta Descrivi DBClusters in AWSSDK for Python (Boto3) API Reference.

Ruby

SDK per Ruby

> **Note**
>
> C'è altro su. GitHub Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice AWS.

```
require 'aws-sdk-rds'

# Creates an Amazon RDS client for the AWS Region
rds = Aws::RDS::Client.new

puts 'Listing clusters in this AWS account...'

# Calls the describe_db_clusters method to get information about clusters
resp = rds.describe_db_clusters(max_records: 20)

# Checks if any clusters are found and prints the appropriate message
if resp.db_clusters.empty?
  puts 'No clusters found!'
```

```
else
  # Loops through the array of cluster objects and prints the cluster identifier
  resp.db_clusters.each do |cluster|
    puts "Cluster identifier: #{cluster.db_cluster_identifier}"
  end
end
```

- Per i dettagli sull'API, consulta Descrivi DBClusters in AWS SDK per RubyAPI Reference.

Rust

SDK per Rust

> **Note**
>
> C'è altro da fare. GitHub Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice AWS.

```
use aws_sdk_rds::Client;

#[derive(Debug)]
struct Error(String);
impl std::fmt::Display for Error {
    fn fmt(&self, f: &mut std::fmt::Formatter<'_>) -> std::fmt::Result {
        write!(f, "{}", self.0)
    }
}
impl std::error::Error for Error {}

#[tokio::main]
async fn main() -> Result<(), Error> {
    tracing_subscriber::fmt::init();
    let sdk_config = aws_config::from_env().load().await;
    let client = Client::new(&sdk_config);

    let describe_db_clusters_output = client
        .describe_db_clusters()
        .send()
        .await
```

```
        .map_err(|e| Error(e.to_string()))?;
    println!(
        "Found {} clusters:",
        describe_db_clusters_output.db_clusters().len()
    );
    for cluster in describe_db_clusters_output.db_clusters() {
        let name = cluster.database_name().unwrap_or("Unknown");
        let engine = cluster.engine().unwrap_or("Unknown");
        let id = cluster.db_cluster_identifier().unwrap_or("Unknown");
        let class = cluster.db_cluster_instance_class().unwrap_or("Unknown");
        println!("\tDatabase: {name}",);
        println!("\t  Engine: {engine}",);
        println!("\t      ID: {id}",);
        println!("\tInstance: {class}",);
    }

    Ok(())
}
```

- Per i dettagli sull'API, consulta Descrivi DBClusters in AWSSDK for Rust API reference.

Esempi di codice

- Esempi di base per l'utilizzo di Aurora AWS SDKs
  - Hello Aurora
  - Scopri le basi di Aurora con un SDK AWS
  - Azioni per l'utilizzo di Aurora AWS SDKs
    - Utilizzo CreateDBCluster con un AWS SDK
    - Utilizzo CreateDBClusterParameterGroup con un AWS SDK
    - Utilizzo CreateDBClusterSnapshot con un AWS SDK
    - Utilizzo CreateDBInstance con un AWS SDK
    - Utilizzo DeleteDBCluster con un AWS SDK
    - Utilizzo DeleteDBClusterParameterGroup con un AWS SDK
    - Utilizzo DeleteDBInstance con un AWS SDK
    - Utilizzo DescribeDBClusterParameterGroups con un AWS SDK
    - Utilizzo DescribeDBClusterParameters con un AWS SDK
    - Utilizzo DescribeDBClusterSnapshots con un AWS SDK

    - Utilizzo DescribeDBClusters con un AWS SDK
    - Utilizzo DescribeDBEngineVersions con un AWS SDK
    - Utilizzo DescribeDBInstances con un AWS SDK
    - Utilizzo DescribeOrderableDBInstanceOptions con un AWS SDK o una CLI
    - Utilizzo ModifyDBClusterParameterGroup con un AWS SDK
- Scenari per l'utilizzo di Aurora AWS SDKs
    - Creazione di una REST API per la libreria di prestiti
    - Creazione di un tracciatore di elementi di lavoro di Aurora Serverless

Esempi di base per l'utilizzo di Aurora AWS SDKs

I seguenti esempi di codice mostrano come utilizzare le nozioni di base di Amazon Aurora AWS SDKs con.

Esempi

- Hello Aurora
- Scopri le basi di Aurora con un SDK AWS
- Azioni per l'utilizzo di Aurora AWS SDKs
    - Utilizzo CreateDBCluster con un AWS SDK
    - Utilizzo CreateDBClusterParameterGroup con un AWS SDK
    - Utilizzo CreateDBClusterSnapshot con un AWS SDK
    - Utilizzo CreateDBInstance con un AWS SDK
    - Utilizzo DeleteDBCluster con un AWS SDK
    - Utilizzo DeleteDBClusterParameterGroup con un AWS SDK
    - Utilizzo DeleteDBInstance con un AWS SDK
    - Utilizzo DescribeDBClusterParameterGroups con un AWS SDK
    - Utilizzo DescribeDBClusterParameters con un AWS SDK
    - Utilizzo DescribeDBClusterSnapshots con un AWS SDK
    - Utilizzo DescribeDBClusters con un AWS SDK
    - Utilizzo DescribeDBEngineVersions con un AWS SDK
    - Utilizzo DescribeDBInstances con un AWS SDK

- Utilizzo DescribeOrderableDBInstanceOptions con un AWS SDK o una CLI
- Utilizzo ModifyDBClusterParameterGroup con un AWS SDK

Hello Aurora

Gli esempi di codice seguenti mostrano come iniziare a utilizzare Aurora.

.NET

SDK per .NET(v4)

> **Note**
>
> C'è altro da fare. GitHub Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice AWS.

```
using Amazon.RDS;
using Amazon.RDS.Model;
using Microsoft.Extensions.DependencyInjection;
using Microsoft.Extensions.Hosting;

namespace AuroraActions;

public static class HelloAurora
{
    static async Task Main(string[] args)
    {
        // Use the AWS .NET Core Setup package to set up dependency injection for
 the
        // Amazon Relational Database Service (Amazon RDS).
        // Use your AWS profile name, or leave it blank to use the default
 profile.
        using var host = Host.CreateDefaultBuilder(args)
            .ConfigureServices((_, services) =>
                services.AddAWSService<IAmazonRDS>()
            ).Build();

        // Now the client is available for injection. Fetching it directly here
 for example purposes only.
```

```
        var rdsClient = host.Services.GetRequiredService<IAmazonRDS>();

        // You can use await and any of the async methods to get a response.
        var response = await rdsClient.DescribeDBClustersAsync(new
 DescribeDBClustersRequest { IncludeShared = true });
        Console.WriteLine($"Hello Amazon RDS Aurora! Let's list some clusters in
 this account:");
        if (response.DBClusters == null)
        {
            Console.WriteLine($"\tNo clusters found.");
        }
        else
        {
            foreach (var cluster in response.DBClusters)
            {
                Console.WriteLine(
                    $"\tCluster: database: {cluster.DatabaseName} identifier:
 {cluster.DBClusterIdentifier}.");
            }
        }
    }
}
```

- Per i dettagli sull'API, consulta Descrivi DBClusters in AWS SDK per .NETAPI Reference.

C++

SDK per C++

> **Note**
>
> C'è altro su GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice AWS.

Codice per il CMake file CMake Lists.txt.

```
# Set the minimum required version of CMake for this project.
cmake_minimum_required(VERSION 3.13)

# Set the AWS service components used by this project.
```

```
set(SERVICE_COMPONENTS rds)

# Set this project's name.
project("hello_aurora")

# Set the C++ standard to use to build this target.
# At least C++ 11 is required for the AWS SDK for C++.
set(CMAKE_CXX_STANDARD 11)

# Use the MSVC variable to determine if this is a Windows build.
set(WINDOWS_BUILD ${MSVC})

if (WINDOWS_BUILD) # Set the location where CMake can find the installed
 libraries for the AWS SDK.
    string(REPLACE ";" "/aws-cpp-sdk-all;" SYSTEM_MODULE_PATH
 "${CMAKE_SYSTEM_PREFIX_PATH}/aws-cpp-sdk-all")
    list(APPEND CMAKE_PREFIX_PATH ${SYSTEM_MODULE_PATH})
endif ()

# Find the AWS SDK for C++ package.
find_package(AWSSDK REQUIRED COMPONENTS ${SERVICE_COMPONENTS})

if (WINDOWS_BUILD AND AWSSDK_INSTALL_AS_SHARED_LIBS)
     # Copy relevant AWS SDK for C++ libraries into the current binary directory
 for running and debugging.

     # set(BIN_SUB_DIR "/Debug") # If you are building from the command line, you
 may need to uncomment this
                                     # and set the proper subdirectory to the
 executables' location.

     AWSSDK_CPY_DYN_LIBS(SERVICE_COMPONENTS ""
 ${CMAKE_CURRENT_BINARY_DIR}${BIN_SUB_DIR})
endif ()

add_executable(${PROJECT_NAME}
        hello_aurora.cpp)

target_link_libraries(${PROJECT_NAME}
        ${AWSSDK_LINK_LIBRARIES})
```

Codice per il file origine hello_aurora.cpp.

```
#include <aws/core/Aws.h>
#include <aws/rds/RDSClient.h>
#include <aws/rds/model/DescribeDBClustersRequest.h>
#include <iostream>

/*
 *  A "Hello Aurora" starter application which initializes an Amazon Relational
 Database Service (Amazon RDS) client
 *  and describes the Amazon Aurora (Aurora) clusters.
 *
 *  main function
 *
 *  Usage: 'hello_aurora'
 *
 */
int main(int argc, char **argv) {
    Aws::SDKOptions options;
    // Optionally change the log level for debugging.
//   options.loggingOptions.logLevel = Utils::Logging::LogLevel::Debug;
    Aws::InitAPI(options); // Should only be called once.
    int result = 0;
    {
        Aws::Client::ClientConfiguration clientConfig;
        // Optional: Set to the AWS Region (overrides config file).
        // clientConfig.region = "us-east-1";

        Aws::RDS::RDSClient rdsClient(clientConfig);

        Aws::String marker; // Used for pagination.
        std::vector<Aws::String> clusterIds;
        do {
            Aws::RDS::Model::DescribeDBClustersRequest request;

            Aws::RDS::Model::DescribeDBClustersOutcome outcome =
                    rdsClient.DescribeDBClusters(request);

            if (outcome.IsSuccess()) {
                for (auto &cluster: outcome.GetResult().GetDBClusters()) {
                    clusterIds.push_back(cluster.GetDBClusterIdentifier());
                }
                marker = outcome.GetResult().GetMarker();
            } else {
                result = 1;
```

```
                std::cerr << "Error with Aurora::GDescribeDBClusters. "
                          << outcome.GetError().GetMessage()
                          << std::endl;
                break;
            }
        } while (!marker.empty());

        std::cout << clusterIds.size() << " Aurora clusters found." << std::endl;
        for (auto &clusterId: clusterIds) {
            std::cout << "  clusterId " << clusterId << std::endl;
        }
    }

    Aws::ShutdownAPI(options); // Should only be called once.
    return 0;
}
```

- Per i dettagli sull'API, consulta Descrivi DBClusters in AWS SDK per C++API Reference.

Go

SDK per Go V2

> **Note**
>
> C'è altro su GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice AWS.

```
package main

import (
 "context"
 "fmt"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/config"
 "github.com/aws/aws-sdk-go-v2/service/rds"
)
```

```
// main uses the AWS SDK for Go V2 to create an Amazon Aurora client and list up
 to 20
// DB clusters in your account.
// This example uses the default settings specified in your shared credentials
// and config files.
func main() {
 ctx := context.Background()
 sdkConfig, err := config.LoadDefaultConfig(ctx)
 if err != nil {
  fmt.Println("Couldn't load default configuration. Have you set up your AWS
 account?")
  fmt.Println(err)
  return
 }
 auroraClient := rds.NewFromConfig(sdkConfig)
 const maxClusters = 20
 fmt.Printf("Let's list up to %v DB clusters.\n", maxClusters)
 output, err := auroraClient.DescribeDBClusters(
  ctx, &rds.DescribeDBClustersInput{MaxRecords: aws.Int32(maxClusters)})
 if err != nil {
  fmt.Printf("Couldn't list DB clusters: %v\n", err)
  return
 }
 if len(output.DBClusters) == 0 {
  fmt.Println("No DB clusters found.")
 } else {
  for _, cluster := range output.DBClusters {
   fmt.Printf("DB cluster %v has database %v.\n", *cluster.DBClusterIdentifier,
    *cluster.DatabaseName)
  }
 }
}
```

- Per i dettagli sull'API, consulta Descrivi DBClusters in AWS SDK per GoAPI Reference.

Java

SDK per Java 2.x

> **Note**
>
> C'è altro su GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice AWS.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.rds.RdsClient;
import software.amazon.awssdk.services.rds.paginators.DescribeDBClustersIterable;

public class DescribeDbClusters {
    public static void main(String[] args) {
        Region region = Region.US_EAST_1;
        RdsClient rdsClient = RdsClient.builder()
                .region(region)
                .build();

        describeClusters(rdsClient);
        rdsClient.close();
    }

    public static void describeClusters(RdsClient rdsClient) {
        DescribeDBClustersIterable clustersIterable =
 rdsClient.describeDBClustersPaginator();
        clustersIterable.stream()
                .flatMap(r -> r.dbClusters().stream())
                .forEach(cluster -> System.out
                        .println("Database name: " + cluster.databaseName() + "
 Arn = " + cluster.dbClusterArn()));
    }
}
```

- Per i dettagli sull'API, consulta Descrivi DBClusters in AWS SDK for Java 2.xAPI Reference.

Python

SDK per Python (Boto3)

> **Note**
>
> C'è altro su GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice AWS.

```
import boto3

# Create an RDS client
rds = boto3.client("rds")

# Create a paginator for the describe_db_clusters operation
paginator = rds.get_paginator("describe_db_clusters")

# Use the paginator to get a list of DB clusters
response_iterator = paginator.paginate(
    PaginationConfig={
        "PageSize": 50,  # Adjust PageSize as needed
        "StartingToken": None,
    }
)

# Iterate through the pages of the response
clusters_found = False
for page in response_iterator:
    if "DBClusters" in page and page["DBClusters"]:
        clusters_found = True
        print("Here are your RDS Aurora clusters:")
        for cluster in page["DBClusters"]:
            print(
                f"Cluster ID: {cluster['DBClusterIdentifier']}, Engine:
 {cluster['Engine']}"
            )

if not clusters_found:
    print("No clusters found!")
```

- Per i dettagli sull'API, consulta Descrivi DBClusters in AWSSDK for Python (Boto3) API Reference.

Ruby

SDK per Ruby

> **Note**
>
> C'è di più su. GitHub Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice AWS.

```
require 'aws-sdk-rds'

# Creates an Amazon RDS client for the AWS Region
rds = Aws::RDS::Client.new

puts 'Listing clusters in this AWS account...'

# Calls the describe_db_clusters method to get information about clusters
resp = rds.describe_db_clusters(max_records: 20)

# Checks if any clusters are found and prints the appropriate message
if resp.db_clusters.empty?
  puts 'No clusters found!'
else
  # Loops through the array of cluster objects and prints the cluster identifier
  resp.db_clusters.each do |cluster|
    puts "Cluster identifier: #{cluster.db_cluster_identifier}"
  end
end
```

- Per i dettagli sull'API, consulta Descrivi DBClusters in AWS SDK per RubyAPI Reference.

Rust

SDK per Rust

> **Note**
>
> C'è altro da fare. GitHub Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice AWS.

```
use aws_sdk_rds::Client;

#[derive(Debug)]
struct Error(String);
impl std::fmt::Display for Error {
    fn fmt(&self, f: &mut std::fmt::Formatter<'_>) -> std::fmt::Result {
        write!(f, "{}", self.0)
    }
}
impl std::error::Error for Error {}

#[tokio::main]
async fn main() -> Result<(), Error> {
    tracing_subscriber::fmt::init();
    let sdk_config = aws_config::from_env().load().await;
    let client = Client::new(&sdk_config);

    let describe_db_clusters_output = client
        .describe_db_clusters()
        .send()
        .await
        .map_err(|e| Error(e.to_string()))?;
    println!(
        "Found {} clusters:",
        describe_db_clusters_output.db_clusters().len()
    );
    for cluster in describe_db_clusters_output.db_clusters() {
        let name = cluster.database_name().unwrap_or("Unknown");
        let engine = cluster.engine().unwrap_or("Unknown");
        let id = cluster.db_cluster_identifier().unwrap_or("Unknown");
        let class = cluster.db_cluster_instance_class().unwrap_or("Unknown");
        println!("\tDatabase: {name}",);
```

```
        println!("\t  Engine: {engine}",);
        println!("\t      ID: {id}",);
        println!("\tInstance: {class}",);
    }

    Ok(())
}
```

- Per i dettagli sull'API, consulta Descrivi DBClusters nella AWSguida di riferimento all'API SDK for Rust.

Per un elenco completo delle guide per sviluppatori AWS SDK e degli esempi di codice, consulta. Utilizzo di questo servizio con un AWS SDK Questo argomento include anche informazioni su come iniziare e dettagli sulle versioni precedenti dell'SDK.

Scopri le basi di Aurora con un SDK AWS

Gli esempi di codice seguenti mostrano come:

- Crea un gruppo di parametri del cluster di database Aurora personalizzati e imposta i relativi valori.
- Crea un cluster di database che utilizza il gruppo di parametri.
- Crea un'istanza database che contiene un database.
- Acquisisci uno snapshot del cluster di database, quindi elimina le risorse.

.NET

SDK per .NET(v4)

> **Note**
>
> C'è altro da fare. GitHub Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice AWS.

Esegui uno scenario interattivo al prompt dei comandi.

```
using Amazon.RDS;
```

```
using Amazon.RDS.Model;
using AuroraActions;
using Microsoft.Extensions.DependencyInjection;
using Microsoft.Extensions.Hosting;
using Microsoft.Extensions.Logging;
using Microsoft.Extensions.Logging.Console;
using Microsoft.Extensions.Logging.Debug;

namespace AuroraScenario;

/// <summary>
/// Scenario for Amazon Aurora examples.
/// </summary>
public class AuroraScenario
{
    /*
    Before running this .NET code example, set up your development environment,
 including your credentials.

    This .NET example performs the following tasks:
    1.  Return a list of the available DB engine families for Aurora MySql using
 the DescribeDBEngineVersionsAsync method.
    2.  Select an engine family and create a custom DB cluster parameter group
 using the CreateDBClusterParameterGroupAsync method.
    3.  Get the parameter group using the DescribeDBClusterParameterGroupsAsync
 method.
    4.  Get some parameters in the group using the
 DescribeDBClusterParametersAsync method.
    5.  Parse and display some parameters in the group.
    6.  Modify the auto_increment_offset and auto_increment_increment parameters
        using the ModifyDBClusterParameterGroupAsync method.
    7.  Get and display the updated parameters using the
 DescribeDBClusterParametersAsync method with a source of "user".
    8.  Get a list of allowed engine versions using the
 DescribeDBEngineVersionsAsync method.
    9.  Create an Aurora DB cluster that contains a MySql database and uses the
 parameter group.
        using the CreateDBClusterAsync method.
    10. Wait for the DB cluster to be ready using the DescribeDBClustersAsync
 method.
    11. Display and select from a list of instance classes available for the
 selected engine and version
        using the paginated DescribeOrderableDBInstanceOptions method.
```

```
   12. Create a database instance in the cluster using the CreateDBInstanceAsync
method.
   13. Wait for the DB instance to be ready using the DescribeDBInstances
method.
   14. Display the connection endpoint string for the new DB cluster.
   15. Create a snapshot of the DB cluster using the
CreateDBClusterSnapshotAsync method.
   16. Wait for DB snapshot to be ready using the
DescribeDBClusterSnapshotsAsync method.
   17. Delete the DB instance using the DeleteDBInstanceAsync method.
   18. Delete the DB cluster using the DeleteDBClusterAsync method.
   19. Wait for DB cluster to be deleted using the DescribeDBClustersAsync
methods.
   20. Delete the cluster parameter group using the
DeleteDBClusterParameterGroupAsync.
   */

   private static readonly string sepBar = new('-', 80);
   private static AuroraWrapper auroraWrapper = null!;
   private static ILogger logger = null!;
   private static readonly string engine = "aurora-mysql";
   static async Task Main(string[] args)
   {
       // Set up dependency injection for the Amazon Relational Database Service
(Amazon RDS).
       using var host = Host.CreateDefaultBuilder(args)
           .ConfigureLogging(logging =>
               logging.AddFilter("System", LogLevel.Debug)
                   .AddFilter<DebugLoggerProvider>("Microsoft",
LogLevel.Information)
                   .AddFilter<ConsoleLoggerProvider>("Microsoft",
LogLevel.Trace))
           .ConfigureServices((_, services) =>
               services.AddAWSService<IAmazonRDS>()
                   .AddTransient<AuroraWrapper>()
           )
           .Build();

       logger = LoggerFactory.Create(builder =>
       {
           builder.AddConsole();
       }).CreateLogger<AuroraScenario>();

       auroraWrapper = host.Services.GetRequiredService<AuroraWrapper>();
```

```
        Console.WriteLine(sepBar);
        Console.WriteLine(
            "Welcome to the Amazon Aurora: get started with DB clusters
example.");
        Console.WriteLine(sepBar);

        DBClusterParameterGroup parameterGroup = null!;
        DBCluster? newCluster = null;
        DBInstance? newInstance = null;

        try
        {
            var parameterGroupFamily = await ChooseParameterGroupFamilyAsync();

            parameterGroup = await
CreateDBParameterGroupAsync(parameterGroupFamily);

            var parameters = await
DescribeParametersInGroupAsync(parameterGroup.DBClusterParameterGroupName,
                new List<string> { "auto_increment_offset",
"auto_increment_increment" });

            await
ModifyParametersAsync(parameterGroup.DBClusterParameterGroupName, parameters);

            await
DescribeUserSourceParameters(parameterGroup.DBClusterParameterGroupName);

            var engineVersionChoice = await
ChooseDBEngineVersionAsync(parameterGroupFamily);

            var newClusterIdentifier = "Example-Cluster-" + DateTime.Now.Ticks;

            newCluster = await CreateNewCluster
            (
                parameterGroup,
                engine,
                engineVersionChoice.EngineVersion,
                newClusterIdentifier
            );

            var instanceClassChoice = await ChooseDBInstanceClass(engine,
engineVersionChoice.EngineVersion);
```

```
            var newInstanceIdentifier = "Example-Instance-" + DateTime.Now.Ticks;

            newInstance = await CreateNewInstance(
                newClusterIdentifier,
                engine,
                engineVersionChoice.EngineVersion,
                instanceClassChoice.DBInstanceClass,
                newInstanceIdentifier
            );

            DisplayConnectionString(newCluster!);
            await CreateSnapshot(newCluster!);
            await CleanupResources(newInstance, newCluster, parameterGroup);

            Console.WriteLine("Scenario complete.");
            Console.WriteLine(sepBar);
        }
        catch (Exception ex)
        {
            await CleanupResources(newInstance, newCluster, parameterGroup);
            logger.LogError(ex, "There was a problem executing the scenario.");
        }
    }

    /// <summary>
    /// Choose the Aurora DB parameter group family from a list of available
options.
    /// </summary>
    /// <returns>The selected parameter group family.</returns>
    public static async Task<string> ChooseParameterGroupFamilyAsync()
    {
        Console.WriteLine(sepBar);
        // 1. Get a list of available engines.
        var engines = await
auroraWrapper.DescribeDBEngineVersionsForEngineAsync(engine);

        Console.WriteLine($"1. The following is a list of available DB parameter
group families for engine {engine}:");

        var parameterGroupFamilies =
            engines.GroupBy(e => e.DBParameterGroupFamily).ToList();
        for (var i = 1; i <= parameterGroupFamilies.Count; i++)
        {
```

```
            var parameterGroupFamily = parameterGroupFamilies[i - 1];
            // List the available parameter group families.
            Console.WriteLine(
                $"\t{i}. Family: {parameterGroupFamily.Key}");
        }

        var choiceNumber = 0;
        while (choiceNumber < 1 || choiceNumber > parameterGroupFamilies.Count)
        {
            Console.WriteLine("2. Select an available DB parameter group family
by entering a number from the preceding list:");
            var choice = Console.ReadLine();
            Int32.TryParse(choice, out choiceNumber);
        }
        var parameterGroupFamilyChoice = parameterGroupFamilies[choiceNumber -
1];
        Console.WriteLine(sepBar);
        return parameterGroupFamilyChoice.Key;
    }

    /// <summary>
    /// Create and get information on a DB parameter group.
    /// </summary>
    /// <param name="dbParameterGroupFamily">The DBParameterGroupFamily for the
new DB parameter group.</param>
    /// <returns>The new DBParameterGroup.</returns>
    public static async Task<DBClusterParameterGroup>
CreateDBParameterGroupAsync(string dbParameterGroupFamily)
    {
        Console.WriteLine(sepBar);
        Console.WriteLine($"2. Create new DB parameter group with family
{dbParameterGroupFamily}:");

        var parameterGroup = await
auroraWrapper.CreateCustomClusterParameterGroupAsync(
            dbParameterGroupFamily,
            "ExampleParameterGroup-" + DateTime.Now.Ticks,
            "New example parameter group");

        var groupInfo =
            await
auroraWrapper.DescribeCustomDBClusterParameterGroupAsync(parameterGroup.DBClusterParamet

        Console.WriteLine(
```

```
            $"3. New DB parameter group created: \n\t{groupInfo?.Description}, \n
\tARN {groupInfo?.DBClusterParameterGroupName}");
        Console.WriteLine(sepBar);
        return parameterGroup;
    }

    /// <summary>
    /// Get and describe parameters from a DBParameterGroup.
    /// </summary>
    /// <param name="parameterGroupName">The name of the DBParameterGroup.</
param>
    /// <param name="parameterNames">Optional specific names of parameters to
 describe.</param>
    /// <returns>The list of requested parameters.</returns>
    public static async Task<List<Parameter>>
 DescribeParametersInGroupAsync(string parameterGroupName, List<string>?
 parameterNames = null)
    {
        Console.WriteLine(sepBar);
        Console.WriteLine("4. Get some parameters from the group.");
        Console.WriteLine(sepBar);

        var parameters =
            await
 auroraWrapper.DescribeDBClusterParametersInGroupAsync(parameterGroupName);

        var matchingParameters =
            parameters.Where(p => parameterNames == null ||
 parameterNames.Contains(p.ParameterName)).ToList();

        Console.WriteLine("5. Parameter information:");
        matchingParameters.ForEach(p =>
            Console.WriteLine(
                $"\n\tParameter: {p.ParameterName}." +
                $"\n\tDescription: {p.Description}." +
                $"\n\tAllowed Values: {p.AllowedValues}." +
                $"\n\tValue: {p.ParameterValue}."));

        Console.WriteLine(sepBar);

        return matchingParameters;
    }

    /// <summary>
```

```
    /// Modify a parameter from a DBParameterGroup.
    /// </summary>
    /// <param name="parameterGroupName">Name of the DBParameterGroup.</param>
    /// <param name="parameters">The parameters to modify.</param>
    /// <returns>Async task.</returns>
    public static async Task ModifyParametersAsync(string parameterGroupName,
 List<Parameter> parameters)
    {
        Console.WriteLine(sepBar);
        Console.WriteLine("6. Modify some parameters in the group.");

        await
 auroraWrapper.ModifyIntegerParametersInGroupAsync(parameterGroupName,
 parameters);

        Console.WriteLine(sepBar);
    }

    /// <summary>
    /// Describe the user source parameters in the group.
    /// </summary>
    /// <param name="parameterGroupName">The name of the DBParameterGroup.</
param>
    /// <returns>Async task.</returns>
    public static async Task DescribeUserSourceParameters(string
 parameterGroupName)
    {
        Console.WriteLine(sepBar);
        Console.WriteLine("7. Describe updated user source parameters in the
 group.");

        var parameters =
            await
 auroraWrapper.DescribeDBClusterParametersInGroupAsync(parameterGroupName,
 "user");

        parameters.ForEach(p =>
            Console.WriteLine(
                $"\n\tParameter: {p.ParameterName}." +
                $"\n\tDescription: {p.Description}." +
                $"\n\tAllowed Values: {p.AllowedValues}." +
                $"\n\tValue: {p.ParameterValue}."));

        Console.WriteLine(sepBar);
```

```
    }

    /// <summary>
    /// Choose a DB engine version.
    /// </summary>
    /// <param name="dbParameterGroupFamily">DB parameter group family for engine
choice.</param>
    /// <returns>The selected engine version.</returns>
    public static async Task<DBEngineVersion> ChooseDBEngineVersionAsync(string
dbParameterGroupFamily)
    {
        Console.WriteLine(sepBar);
        // Get a list of allowed engines.
        var allowedEngines =
            await auroraWrapper.DescribeDBEngineVersionsForEngineAsync(engine,
dbParameterGroupFamily);

        Console.WriteLine($"Available DB engine versions for parameter group
family {dbParameterGroupFamily}:");
        int i = 1;
        foreach (var version in allowedEngines)
        {
            Console.WriteLine(
                $"\t{i}. {version.DBEngineVersionDescription}.");
            i++;
        }

        var choiceNumber = 0;
        while (choiceNumber < 1 || choiceNumber > allowedEngines.Count)
        {
            Console.WriteLine("8. Select an available DB engine version by
entering a number from the list above:");
            var choice = Console.ReadLine();
            Int32.TryParse(choice, out choiceNumber);
        }

        var engineChoice = allowedEngines[choiceNumber - 1];
        Console.WriteLine(sepBar);
        return engineChoice;
    }

    /// <summary>
    /// Create a new RDS DB cluster.
    /// </summary>
```

```
    /// <param name="parameterGroup">Parameter group to use for the DB cluster.</
param>
    /// <param name="engineName">Engine to use for the DB cluster.</param>
    /// <param name="engineVersion">Engine version to use for the DB cluster.</
param>
    /// <param name="clusterIdentifier">Cluster identifier to use for the DB
 cluster.</param>
    /// <returns>The new DB cluster.</returns>
    public static async Task<DBCluster?> CreateNewCluster(DBClusterParameterGroup
 parameterGroup,
        string engineName, string engineVersion, string clusterIdentifier)
    {
        Console.WriteLine(sepBar);
        Console.WriteLine($"9. Create a new DB cluster with identifier
 {clusterIdentifier}.");

        DBCluster newCluster;
        var clusters = await auroraWrapper.DescribeDBClustersPagedAsync();
        var isClusterCreated = clusters.Any(i => i.DBClusterIdentifier ==
 clusterIdentifier);

        if (isClusterCreated)
        {
            Console.WriteLine("Cluster already created.");
            newCluster = clusters.First(i => i.DBClusterIdentifier ==
 clusterIdentifier);
        }
        else
        {
            Console.WriteLine("Enter an admin username:");
            var username = Console.ReadLine();

            Console.WriteLine("Enter an admin password:");
            var password = Console.ReadLine();

            newCluster = await auroraWrapper.CreateDBClusterWithAdminAsync(
                "ExampleDatabase",
                clusterIdentifier,
                parameterGroup.DBClusterParameterGroupName,
                engineName,
                engineVersion,
                username!,
                password!
            );
```

```
            Console.WriteLine("10. Waiting for DB cluster to be ready...");
            while (newCluster.Status != "available")
            {
                Console.Write(".");
                Thread.Sleep(5000);
                clusters = await
 auroraWrapper.DescribeDBClustersPagedAsync(clusterIdentifier);
                newCluster = clusters.First();
            }
        }

        Console.WriteLine(sepBar);
        return newCluster;
    }

    /// <summary>
    /// Choose a DB instance class for a particular engine and engine version.
    /// </summary>
    /// <param name="engine">DB engine for DB instance choice.</param>
    /// <param name="engineVersion">DB engine version for DB instance choice.</
param>
    /// <returns>The selected orderable DB instance option.</returns>
    public static async Task<OrderableDBInstanceOption>
 ChooseDBInstanceClass(string engine, string engineVersion)
    {
        Console.WriteLine(sepBar);
        // Get a list of allowed DB instance classes.
        var allowedInstances =
            await
 auroraWrapper.DescribeOrderableDBInstanceOptionsPagedAsync(engine,
 engineVersion);

        Console.WriteLine($"Available DB instance classes for engine {engine} and
 version {engineVersion}:");
        int i = 1;

        foreach (var instance in allowedInstances)
        {
            Console.WriteLine(
                $"\t{i}. Instance class: {instance.DBInstanceClass} (storage type
 {instance.StorageType})");
            i++;
        }
```

```
        var choiceNumber = 0;
        while (choiceNumber < 1 || choiceNumber > allowedInstances.Count)
        {
            Console.WriteLine("11. Select an available DB instance class by
 entering a number from the preceding list:");
            var choice = Console.ReadLine();
            Int32.TryParse(choice, out choiceNumber);
        }

        var instanceChoice = allowedInstances[choiceNumber - 1];
        Console.WriteLine(sepBar);
        return instanceChoice;
    }

    /// <summary>
    /// Create a new DB instance.
    /// </summary>
    /// <param name="engineName">Engine to use for the DB instance.</param>
    /// <param name="engineVersion">Engine version to use for the DB instance.</
param>
    /// <param name="instanceClass">Instance class to use for the DB instance.</
param>
    /// <param name="instanceIdentifier">Instance identifier to use for the DB
 instance.</param>
    /// <returns>The new DB instance.</returns>
    public static async Task<DBInstance?> CreateNewInstance(
        string clusterIdentifier,
        string engineName,
        string engineVersion,
        string instanceClass,
        string instanceIdentifier)
    {
        Console.WriteLine(sepBar);
        Console.WriteLine($"12. Create a new DB instance with identifier
 {instanceIdentifier}.");
        bool isInstanceReady = false;
        DBInstance newInstance;
        var instances = await auroraWrapper.DescribeDBInstancesPagedAsync();
        isInstanceReady = instances.FirstOrDefault(i =>
            i.DBInstanceIdentifier == instanceIdentifier)?.DBInstanceStatus ==
 "available";

        if (isInstanceReady)
```

```
        {
            Console.WriteLine("Instance already created.");
            newInstance = instances.First(i => i.DBInstanceIdentifier ==
 instanceIdentifier);
        }
        else
        {
            newInstance = await auroraWrapper.CreateDBInstanceInClusterAsync(
                clusterIdentifier,
                instanceIdentifier,
                engineName,
                engineVersion,
                instanceClass
            );

            Console.WriteLine("13. Waiting for DB instance to be ready...");
            while (!isInstanceReady)
            {
                Console.Write(".");
                Thread.Sleep(5000);
                instances = await
 auroraWrapper.DescribeDBInstancesPagedAsync(instanceIdentifier);
                isInstanceReady = instances.FirstOrDefault()?.DBInstanceStatus ==
 "available";
                newInstance = instances.First();
            }
        }

        Console.WriteLine(sepBar);
        return newInstance;
    }

    /// <summary>
    /// Display a connection string for an Amazon RDS DB cluster.
    /// </summary>
    /// <param name="cluster">The DB cluster to use to get a connection string.</
param>
    public static void DisplayConnectionString(DBCluster cluster)
    {
        Console.WriteLine(sepBar);
        // Display the connection string.
        Console.WriteLine("14. New DB cluster connection string: ");
        Console.WriteLine(
            $"\n{engine} -h {cluster.Endpoint} -P {cluster.Port} "
```

```
            + $"-u {cluster.MasterUsername} -p [YOUR PASSWORD]\n");

        Console.WriteLine(sepBar);
    }

    /// <summary>
    /// Create a snapshot from an Amazon RDS DB cluster.
    /// </summary>
    /// <param name="cluster">DB cluster to use when creating a snapshot.</param>
    /// <returns>The snapshot object.</returns>
    public static async Task<DBClusterSnapshot> CreateSnapshot(DBCluster cluster)
    {
        Console.WriteLine(sepBar);
        // Create a snapshot.
        Console.WriteLine($"15. Creating snapshot from DB cluster
{cluster.DBClusterIdentifier}.");
        var snapshot = await
auroraWrapper.CreateClusterSnapshotByIdentifierAsync(
            cluster.DBClusterIdentifier,
            "ExampleSnapshot-" + DateTime.Now.Ticks);

        // Wait for the snapshot to be available.
        bool isSnapshotReady = false;

        Console.WriteLine($"16. Waiting for snapshot to be ready...");
        while (!isSnapshotReady)
        {
            Console.Write(".");
            Thread.Sleep(5000);
            var snapshots =
                await
auroraWrapper.DescribeDBClusterSnapshotsByIdentifierAsync(cluster.DBClusterIdentifier);
            isSnapshotReady = snapshots.FirstOrDefault()?.Status == "available";
            snapshot = snapshots.First();
        }

        Console.WriteLine(
            $"Snapshot {snapshot.DBClusterSnapshotIdentifier} status is
{snapshot.Status}.");
        Console.WriteLine(sepBar);
        return snapshot;
    }

    /// <summary>
```

```
    /// Clean up resources from the scenario.
    /// </summary>
    /// <param name="newInstance">The instance to clean up.</param>
    /// <param name="newCluster">The cluster to clean up.</param>
    /// <param name="parameterGroup">The parameter group to clean up.</param>
    /// <returns>Async Task.</returns>
    private static async Task CleanupResources(
        DBInstance? newInstance,
        DBCluster? newCluster,
        DBClusterParameterGroup? parameterGroup)
    {
        Console.WriteLine(new string('-', 80));
        Console.WriteLine($"Clean up resources.");

        if (newInstance is not null && GetYesNoResponse($"\tClean up instance
{newInstance.DBInstanceIdentifier}? (y/n)"))
        {
            // Delete the DB instance.
            Console.WriteLine($"17. Deleting the DB instance
{newInstance.DBInstanceIdentifier}.");
            await
auroraWrapper.DeleteDBInstanceByIdentifierAsync(newInstance.DBInstanceIdentifier);
        }

        if (newCluster is not null && GetYesNoResponse($"\tClean up cluster
{newCluster.DBClusterIdentifier}? (y/n)"))
        {
            // Delete the DB cluster.
            Console.WriteLine($"18. Deleting the DB cluster
{newCluster.DBClusterIdentifier}.");
            await
auroraWrapper.DeleteDBClusterByIdentifierAsync(newCluster.DBClusterIdentifier);

            // Wait for the DB cluster to delete.
            Console.WriteLine($"19. Waiting for the DB cluster to delete...");
            bool isClusterDeleted = false;

            while (!isClusterDeleted)
            {
                Console.Write(".");
                Thread.Sleep(5000);
                var cluster = await auroraWrapper.DescribeDBClustersPagedAsync();
                isClusterDeleted = cluster.All(i => i.DBClusterIdentifier !=
newCluster.DBClusterIdentifier);
```

```
            }

            Console.WriteLine("DB cluster deleted.");
        }

        if (parameterGroup is not null && GetYesNoResponse($"\tClean up parameter
 group? (y/n)"))
        {
            Console.WriteLine($"20. Deleting the DB parameter group
 {parameterGroup.DBClusterParameterGroupName}.");
            await
 auroraWrapper.DeleteClusterParameterGroupByNameAsync(parameterGroup.DBClusterParameterGr
            Console.WriteLine("Parameter group deleted.");
        }

        Console.WriteLine(new string('-', 80));
    }

    /// <summary>
    /// Get a yes or no response from the user.
    /// </summary>
    /// <param name="question">The question string to print on the console.</
param>
    /// <returns>True if the user responds with a yes.</returns>
    private static bool GetYesNoResponse(string question)
    {
        Console.WriteLine(question);
        var ynResponse = Console.ReadLine();
        var response = ynResponse != null &&
                       ynResponse.Equals("y",
                           StringComparison.InvariantCultureIgnoreCase);
        return response;
    }
```

Metodi wrapper che vengono richiamati dallo scenario per gestire le operazioni Aurora.

```
using Amazon.RDS;
using Amazon.RDS.Model;

namespace AuroraActions;
```

```
/// <summary>
/// Wrapper for the Amazon Aurora cluster client operations.
/// </summary>
public class AuroraWrapper
{
    private readonly IAmazonRDS _amazonRDS;
    public AuroraWrapper(IAmazonRDS amazonRDS)
    {
        _amazonRDS = amazonRDS;
    }

    /// <summary>
    /// Get a list of DB engine versions for a particular DB engine.
    /// </summary>
    /// <param name="engine">The name of the engine.</param>
    /// <param name="parameterGroupFamily">Optional parameter group family
 name.</param>
    /// <returns>A list of DBEngineVersions.</returns>
    public async Task<List<DBEngineVersion>>
 DescribeDBEngineVersionsForEngineAsync(string engine,
        string? parameterGroupFamily = null)
    {
        var response = await _amazonRDS.DescribeDBEngineVersionsAsync(
            new DescribeDBEngineVersionsRequest()
            {
                Engine = engine,
                DBParameterGroupFamily = parameterGroupFamily
            });
        return response.DBEngineVersions;
    }

    /// <summary>
    /// Create a custom cluster parameter group.
    /// </summary>
    /// <param name="parameterGroupFamily">The family of the parameter group.</
param>
    /// <param name="groupName">The name for the new parameter group.</param>
    /// <param name="description">A description for the new parameter group.</
param>
    /// <returns>The new parameter group object.</returns>
    public async Task<DBClusterParameterGroup>
 CreateCustomClusterParameterGroupAsync(
        string parameterGroupFamily,
        string groupName,
```

```
        string description)
    {
        var request = new CreateDBClusterParameterGroupRequest
        {
            DBParameterGroupFamily = parameterGroupFamily,
            DBClusterParameterGroupName = groupName,
            Description = description,
        };

        var response = await
_amazonRDS.CreateDBClusterParameterGroupAsync(request);
        return response.DBClusterParameterGroup;
    }

    /// <summary>
    /// Describe the cluster parameters in a parameter group.
    /// </summary>
    /// <param name="groupName">The name of the parameter group.</param>
    /// <param name="source">The optional name of the source filter.</param>
    /// <returns>The collection of parameters.</returns>
    public async Task<List<Parameter>>
DescribeDBClusterParametersInGroupAsync(string groupName, string? source = null)
    {
        var paramList = new List<Parameter>();

        DescribeDBClusterParametersResponse response;
        var request = new DescribeDBClusterParametersRequest
        {
            DBClusterParameterGroupName = groupName,
            Source = source,
        };

        // Get the full list if there are multiple pages.
        do
        {
            response = await
_amazonRDS.DescribeDBClusterParametersAsync(request);
            paramList.AddRange(response.Parameters);

            request.Marker = response.Marker;
        }
        while (response.Marker is not null);

        return paramList;
```

```
    }

    /// <summary>
    /// Get the description of a DB cluster parameter group by name.
    /// </summary>
    /// <param name="name">The name of the DB parameter group to describe.</
param>
    /// <returns>The parameter group description.</returns>
    public async Task<DBClusterParameterGroup?>
 DescribeCustomDBClusterParameterGroupAsync(string name)
    {
        var response = await _amazonRDS.DescribeDBClusterParameterGroupsAsync(
            new DescribeDBClusterParameterGroupsRequest()
            {
                DBClusterParameterGroupName = name
            });
        return response.DBClusterParameterGroups.FirstOrDefault();
    }

    /// <summary>
    /// Modify the specified integer parameters with new values from user input.
    /// </summary>
    /// <param name="groupName">The group name for the parameters.</param>
    /// <param name="parameters">The list of integer parameters to modify.</
param>
    /// <param name="newValue">Optional int value to set for parameters.</param>
    /// <returns>The name of the group that was modified.</returns>
    public async Task<string> ModifyIntegerParametersInGroupAsync(string
 groupName, List<Parameter> parameters, int newValue = 0)
    {
        foreach (var p in parameters)
        {
            if (p.IsModifiable.GetValueOrDefault() && p.DataType == "integer")
            {
                while (newValue == 0)
                {
                    Console.WriteLine(
                        $"Enter a new value for {p.ParameterName} from the
 allowed values {p.AllowedValues} ");

                    var choice = Console.ReadLine();
                    int.TryParse(choice, out newValue);
                }
```

```
                p.ParameterValue = newValue.ToString();
            }
        }

        var request = new ModifyDBClusterParameterGroupRequest
        {
            Parameters = parameters,
            DBClusterParameterGroupName = groupName,
        };

        var result = await
_amazonRDS.ModifyDBClusterParameterGroupAsync(request);
        return result.DBClusterParameterGroupName;
    }


    /// <summary>
    /// Get a list of orderable DB instance options for a specific
    /// engine and engine version.
    /// </summary>
    /// <param name="engine">Name of the engine.</param>
    /// <param name="engineVersion">Version of the engine.</param>
    /// <returns>List of OrderableDBInstanceOptions.</returns>
    public async Task<List<OrderableDBInstanceOption>>
DescribeOrderableDBInstanceOptionsPagedAsync(string engine, string
engineVersion)
    {
        // Use a paginator to get a list of DB instance options.
        var results = new List<OrderableDBInstanceOption>();
        var paginateInstanceOptions =
_amazonRDS.Paginators.DescribeOrderableDBInstanceOptions(
            new DescribeOrderableDBInstanceOptionsRequest()
            {
                Engine = engine,
                EngineVersion = engineVersion,
            });
        // Get the entire list using the paginator.
        await foreach (var instanceOptions in
paginateInstanceOptions.OrderableDBInstanceOptions)
        {
            results.Add(instanceOptions);
        }
        return results;
    }
```

```
    /// <summary>
    /// Delete a particular parameter group by name.
    /// </summary>
    /// <param name="groupName">The name of the parameter group.</param>
    /// <returns>True if successful.</returns>
    public async Task<bool> DeleteClusterParameterGroupByNameAsync(string
groupName)
    {
        var request = new DeleteDBClusterParameterGroupRequest
        {
            DBClusterParameterGroupName = groupName,
        };

        var response = await
_amazonRDS.DeleteDBClusterParameterGroupAsync(request);
        return response.HttpStatusCode == System.Net.HttpStatusCode.OK;
    }

    /// <summary>
    /// Create a new cluster and database.
    /// </summary>
    /// <param name="dbName">The name of the new database.</param>
    /// <param name="clusterIdentifier">The identifier of the cluster.</param>
    /// <param name="parameterGroupName">The name of the parameter group.</param>
    /// <param name="dbEngine">The engine to use for the new cluster.</param>
    /// <param name="dbEngineVersion">The version of the engine to use.</param>
    /// <param name="adminName">The admin username.</param>
    /// <param name="adminPassword">The primary admin password.</param>
    /// <returns>The cluster object.</returns>
    public async Task<DBCluster> CreateDBClusterWithAdminAsync(
        string dbName,
        string clusterIdentifier,
        string parameterGroupName,
        string dbEngine,
        string dbEngineVersion,
        string adminName,
        string adminPassword)
    {
        var request = new CreateDBClusterRequest
        {
            DatabaseName = dbName,
            DBClusterIdentifier = clusterIdentifier,
            DBClusterParameterGroupName = parameterGroupName,
```

```
            Engine = dbEngine,
            EngineVersion = dbEngineVersion,
            MasterUsername = adminName,
            MasterUserPassword = adminPassword,
        };

        var response = await _amazonRDS.CreateDBClusterAsync(request);
        return response.DBCluster;
    }

    /// <summary>
    /// Returns a list of DB instances.
    /// </summary>
    /// <param name="dbInstanceIdentifier">Optional name of a specific DB
instance.</param>
    /// <returns>List of DB instances.</returns>
    public async Task<List<DBInstance>> DescribeDBInstancesPagedAsync(string?
dbInstanceIdentifier = null)
    {
        var results = new List<DBInstance>();
        var instancesPaginator = _amazonRDS.Paginators.DescribeDBInstances(
            new DescribeDBInstancesRequest
            {
                DBInstanceIdentifier = dbInstanceIdentifier
            });
        // Get the entire list using the paginator.
        await foreach (var instances in instancesPaginator.DBInstances)
        {
            results.Add(instances);
        }
        return results;
    }

    /// <summary>
    /// Returns a list of DB clusters.
    /// </summary>
    /// <param name="dbInstanceIdentifier">Optional name of a specific DB
cluster.</param>
    /// <returns>List of DB clusters.</returns>
    public async Task<List<DBCluster>> DescribeDBClustersPagedAsync(string?
dbClusterIdentifier = null)
    {
        var results = new List<DBCluster>();
```

```
        DescribeDBClustersResponse response;
        DescribeDBClustersRequest request = new DescribeDBClustersRequest
        {
            DBClusterIdentifier = dbClusterIdentifier
        };
        // Get the full list if there are multiple pages.
        do
        {
            response = await _amazonRDS.DescribeDBClustersAsync(request);
            if (response.DBClusters != null)
            {
                results.AddRange(response.DBClusters);
            }
            request.Marker = response.Marker;
        }
        while (response.Marker is not null);
        return results;
    }

    /// <summary>
    /// Create an Amazon Relational Database Service (Amazon RDS) DB instance
    /// with a particular set of properties. Use the action
DescribeDBInstancesAsync
    /// to determine when the DB instance is ready to use.
    /// </summary>
    /// <param name="dbInstanceIdentifier">DB instance identifier.</param>
    /// <param name="dbClusterIdentifier">DB cluster identifier.</param>
    /// <param name="dbEngine">The engine for the DB instance.</param>
    /// <param name="dbEngineVersion">Version for the DB instance.</param>
    /// <param name="instanceClass">Class for the DB instance.</param>
    /// <returns>DB instance object.</returns>
    public async Task<DBInstance> CreateDBInstanceInClusterAsync(
        string dbClusterIdentifier,
        string dbInstanceIdentifier,
        string dbEngine,
        string dbEngineVersion,
        string instanceClass)
    {
        // When creating the instance within a cluster, do not specify the name
or size.
        var response = await _amazonRDS.CreateDBInstanceAsync(
            new CreateDBInstanceRequest()
            {
                DBClusterIdentifier = dbClusterIdentifier,
```

```
                DBInstanceIdentifier = dbInstanceIdentifier,
                Engine = dbEngine,
                EngineVersion = dbEngineVersion,
                DBInstanceClass = instanceClass
            });

        return response.DBInstance;
    }

    /// <summary>
    /// Create a snapshot of a cluster.
    /// </summary>
    /// <param name="dbClusterIdentifier">DB cluster identifier.</param>
    /// <param name="snapshotIdentifier">Identifier for the snapshot.</param>
    /// <returns>DB snapshot object.</returns>
    public async Task<DBClusterSnapshot>
CreateClusterSnapshotByIdentifierAsync(string dbClusterIdentifier, string
snapshotIdentifier)
    {
        var response = await _amazonRDS.CreateDBClusterSnapshotAsync(
            new CreateDBClusterSnapshotRequest()
            {
                DBClusterIdentifier = dbClusterIdentifier,
                DBClusterSnapshotIdentifier = snapshotIdentifier,
            });

        return response.DBClusterSnapshot;
    }

    /// <summary>
    /// Return a list of DB snapshots for a particular DB cluster.
    /// </summary>
    /// <param name="dbClusterIdentifier">DB cluster identifier.</param>
    /// <returns>List of DB snapshots.</returns>
    public async Task<List<DBClusterSnapshot>>
DescribeDBClusterSnapshotsByIdentifierAsync(string dbClusterIdentifier)
    {
        var results = new List<DBClusterSnapshot>();

        DescribeDBClusterSnapshotsResponse response;
        DescribeDBClusterSnapshotsRequest request = new
DescribeDBClusterSnapshotsRequest
        {
            DBClusterIdentifier = dbClusterIdentifier
```

```
        };
        // Get the full list if there are multiple pages.
        do
        {
            response = await _amazonRDS.DescribeDBClusterSnapshotsAsync(request);
            results.AddRange(response.DBClusterSnapshots);
            request.Marker = response.Marker;
        }
        while (response.Marker is not null);
        return results;
    }

    /// <summary>
    /// Delete a particular DB cluster.
    /// </summary>
    /// <param name="dbClusterIdentifier">DB cluster identifier.</param>
    /// <returns>DB cluster object.</returns>
    public async Task<DBCluster> DeleteDBClusterByIdentifierAsync(string
dbClusterIdentifier)
    {
        var response = await _amazonRDS.DeleteDBClusterAsync(
            new DeleteDBClusterRequest()
            {
                DBClusterIdentifier = dbClusterIdentifier,
                SkipFinalSnapshot = true
            });

        return response.DBCluster;
    }

    /// <summary>
    /// Delete a particular DB instance.
    /// </summary>
    /// <param name="dbInstanceIdentifier">DB instance identifier.</param>
    /// <returns>DB instance object.</returns>
    public async Task<DBInstance> DeleteDBInstanceByIdentifierAsync(string
dbInstanceIdentifier)
    {
        var response = await _amazonRDS.DeleteDBInstanceAsync(
            new DeleteDBInstanceRequest()
            {
                DBInstanceIdentifier = dbInstanceIdentifier,
                SkipFinalSnapshot = true,
                DeleteAutomatedBackups = true
```

```
            });

        return response.DBInstance;
    }
}
```

- Per informazioni dettagliate sull'API, consulta i seguenti argomenti nella documentazione di riferimento dell'API AWS SDK per .NET.
  - CreaDBCluster
  - CreaDBClusterParameterGroup
  - Crea DBCluster istantanea
  - CreaDBInstance
  - EliminaDBCluster
  - EliminaDBClusterParameterGroup
  - EliminaDBInstance
  - Descriva DBCluster ParameterGroups
  - DBClusterDescrivi parametri
  - Descrivi le DBCluster istantanee
  - Descriva DBClusters
  - Descrivi DBEngine versioni
  - Descriva DBInstances
  - DescribeOrderableDBInstanceOpzioni
  - ModificaDBClusterParameterGroup

C++

SDK per C++

> **Note**
>
> C'è altro da fare GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice AWS.

```
        Aws::Client::ClientConfiguration clientConfig;
        // Optional: Set to the AWS Region (overrides config file).
        // clientConfig.region = "us-east-1";

//! Routine which creates an Amazon Aurora DB cluster and demonstrates several
 operations
//! on that cluster.
/*!
 \sa gettingStartedWithDBClusters()
 \param clientConfiguration: AWS client configuration.
 \return bool: Successful completion.
 */
bool AwsDoc::Aurora::gettingStartedWithDBClusters(
        const Aws::Client::ClientConfiguration &clientConfig) {
    Aws::RDS::RDSClient client(clientConfig);

    printAsterisksLine();
    std::cout << "Welcome to the Amazon Relational Database Service (Amazon
 Aurora)"
              << std::endl;
    std::cout << "get started with DB clusters demo." << std::endl;
    printAsterisksLine();

    std::cout << "Checking for an existing DB cluster parameter group named '" <<
              CLUSTER_PARAMETER_GROUP_NAME << "'." << std::endl;
    Aws::String dbParameterGroupFamily("Undefined");
    bool parameterGroupFound = true;
    {
        // 1. Check if the DB cluster parameter group already exists.
        Aws::RDS::Model::DescribeDBClusterParameterGroupsRequest request;
        request.SetDBClusterParameterGroupName(CLUSTER_PARAMETER_GROUP_NAME);

        Aws::RDS::Model::DescribeDBClusterParameterGroupsOutcome outcome =
                client.DescribeDBClusterParameterGroups(request);

        if (outcome.IsSuccess()) {
            std::cout << "DB cluster parameter group named '" <<
                      CLUSTER_PARAMETER_GROUP_NAME << "' already exists." <<
 std::endl;
            dbParameterGroupFamily =
 outcome.GetResult().GetDBClusterParameterGroups()
[0].GetDBParameterGroupFamily();
        }
```

```
        else if (outcome.GetError().GetErrorType() ==
                 Aws::RDS::RDSErrors::D_B_PARAMETER_GROUP_NOT_FOUND_FAULT) {
            std::cout << "DB cluster parameter group named '" <<
                      CLUSTER_PARAMETER_GROUP_NAME << "' does not exist." <<
std::endl;
            parameterGroupFound = false;
        }
        else {
            std::cerr << "Error with Aurora::DescribeDBClusterParameterGroups. "
                      << outcome.GetError().GetMessage()
                      << std::endl;
            return false;
        }
    }

    if (!parameterGroupFound) {
        Aws::Vector<Aws::RDS::Model::DBEngineVersion> engineVersions;

        // 2. Get available parameter group families for the specified engine.
        if (!getDBEngineVersions(DB_ENGINE, NO_PARAMETER_GROUP_FAMILY,
                                 engineVersions, client)) {
            return false;
        }

        std::cout << "Getting available parameter group families for " <<
DB_ENGINE
                  << "."
                  << std::endl;
        std::vector<Aws::String> families;
        for (const Aws::RDS::Model::DBEngineVersion &version: engineVersions) {
            Aws::String family = version.GetDBParameterGroupFamily();
            if (std::find(families.begin(), families.end(), family) ==
                families.end()) {
                families.push_back(family);
                std::cout << "  " << families.size() << ": " << family <<
std::endl;
            }
        }

        int choice = askQuestionForIntRange("Which family do you want to use? ",
1,
                                            static_cast<int>(families.size()));
        dbParameterGroupFamily = families[choice - 1];
    }
```

```
    if (!parameterGroupFound) {
        // 3.  Create a DB cluster parameter group.
        Aws::RDS::Model::CreateDBClusterParameterGroupRequest request;
        request.SetDBClusterParameterGroupName(CLUSTER_PARAMETER_GROUP_NAME);
        request.SetDBParameterGroupFamily(dbParameterGroupFamily);
        request.SetDescription("Example cluster parameter group.");

        Aws::RDS::Model::CreateDBClusterParameterGroupOutcome outcome =
                client.CreateDBClusterParameterGroup(request);

        if (outcome.IsSuccess()) {
            std::cout << "The DB cluster parameter group was successfully
created."
                      << std::endl;
        }
        else {
            std::cerr << "Error with Aurora::CreateDBClusterParameterGroup. "
                      << outcome.GetError().GetMessage()
                      << std::endl;
            return false;
        }
    }

    printAsterisksLine();
    std::cout << "Let's set some parameter values in your cluster parameter
group."
              << std::endl;

    Aws::Vector<Aws::RDS::Model::Parameter> autoIncrementParameters;
    // 4.  Get the parameters in the DB cluster parameter group.
    if (!getDBClusterParameters(CLUSTER_PARAMETER_GROUP_NAME,
AUTO_INCREMENT_PREFIX,
                                NO_SOURCE,
                                autoIncrementParameters,
                                client)) {
        cleanUpResources(CLUSTER_PARAMETER_GROUP_NAME, "", "", client);
        return false;
    }

    Aws::Vector<Aws::RDS::Model::Parameter> updateParameters;

    for (Aws::RDS::Model::Parameter &autoIncParameter: autoIncrementParameters) {
        if (autoIncParameter.GetIsModifiable() &&
            (autoIncParameter.GetDataType() == "integer")) {
```

```
            std::cout << "The " << autoIncParameter.GetParameterName()
                      << " is described as: " <<
                      autoIncParameter.GetDescription() << "." << std::endl;
            if (autoIncParameter.ParameterValueHasBeenSet()) {
                std::cout << "The current value is "
                          << autoIncParameter.GetParameterValue()
                          << "." << std::endl;
            }
            std::vector<int> splitValues = splitToInts(
                    autoIncParameter.GetAllowedValues(), '-');
            if (splitValues.size() == 2) {
                int newValue = askQuestionForIntRange(
                        Aws::String("Enter a new value between ") +
                        autoIncParameter.GetAllowedValues() + ": ",
                        splitValues[0], splitValues[1]);
                autoIncParameter.SetParameterValue(std::to_string(newValue));
                updateParameters.push_back(autoIncParameter);

            }
            else {
                std::cerr << "Error parsing " <<
autoIncParameter.GetAllowedValues()
                          << std::endl;
            }
        }
    }

    {
        // 5.  Modify the auto increment parameters in the DB cluster parameter
group.
        Aws::RDS::Model::ModifyDBClusterParameterGroupRequest request;
        request.SetDBClusterParameterGroupName(CLUSTER_PARAMETER_GROUP_NAME);
        request.SetParameters(updateParameters);

        Aws::RDS::Model::ModifyDBClusterParameterGroupOutcome outcome =
                client.ModifyDBClusterParameterGroup(request);

        if (outcome.IsSuccess()) {
            std::cout << "The DB cluster parameter group was successfully
modified."
                      << std::endl;
        }
        else {
            std::cerr << "Error with Aurora::ModifyDBClusterParameterGroup. "
```

```
                    << outcome.GetError().GetMessage()
                    << std::endl;
        }
    }

    std::cout
            << "You can get a list of parameters you've set by specifying a
source of 'user'."
            << std::endl;

    Aws::Vector<Aws::RDS::Model::Parameter> userParameters;
    // 6.  Display the modified parameters in the DB cluster parameter group.
    if (!getDBClusterParameters(CLUSTER_PARAMETER_GROUP_NAME, NO_NAME_PREFIX,
"user",
                                userParameters,
                                client)) {
        cleanUpResources(CLUSTER_PARAMETER_GROUP_NAME, "", "", client);
        return false;
    }

    for (const auto &userParameter: userParameters) {
        std::cout << "  " << userParameter.GetParameterName() << ", " <<
                  userParameter.GetDescription() << ", parameter value - "
                  << userParameter.GetParameterValue() << std::endl;
    }

    printAsterisksLine();
    std::cout << "Checking for an existing DB Cluster." << std::endl;

    Aws::RDS::Model::DBCluster dbCluster;
    // 7.  Check if the DB cluster already exists.
    if (!describeDBCluster(DB_CLUSTER_IDENTIFIER, dbCluster, client)) {
        cleanUpResources(CLUSTER_PARAMETER_GROUP_NAME, "", "", client);
        return false;
    }

    Aws::String engineVersionName;
    Aws::String engineName;
    if (dbCluster.DBClusterIdentifierHasBeenSet()) {
        std::cout << "The DB cluster already exists." << std::endl;
        engineVersionName = dbCluster.GetEngineVersion();
        engineName = dbCluster.GetEngine();

    }
```

```
   else {
       std::cout << "Let's create a DB cluster." << std::endl;
       const Aws::String administratorName = askQuestion(
               "Enter an administrator username for the database: ");
       const Aws::String administratorPassword = askQuestion(
               "Enter a password for the administrator (at least 8 characters):
");
       Aws::Vector<Aws::RDS::Model::DBEngineVersion> engineVersions;

       // 8.  Get a list of engine versions for the parameter group family.
       if (!getDBEngineVersions(DB_ENGINE, dbParameterGroupFamily,
engineVersions,
                                client)) {
           cleanUpResources(CLUSTER_PARAMETER_GROUP_NAME, "", "", client);
           return false;
       }

       std::cout << "The available engines for your parameter group family are:"
                 << std::endl;

       int index = 1;
       for (const Aws::RDS::Model::DBEngineVersion &engineVersion:
engineVersions) {
           std::cout << "  " << index << ": " <<
engineVersion.GetEngineVersion()
                     << std::endl;
           ++index;
       }
       int choice = askQuestionForIntRange("Which engine do you want to use? ",
1,

static_cast<int>(engineVersions.size()));
       const Aws::RDS::Model::DBEngineVersion engineVersion =
engineVersions[choice -
                                                                              1];

       engineName = engineVersion.GetEngine();
       engineVersionName = engineVersion.GetEngineVersion();
       std::cout << "Creating a DB cluster named '" << DB_CLUSTER_IDENTIFIER
                 << "' and database '" << DB_NAME << "'.\n"
                 << "The DB cluster is configured to use your custom cluster
parameter group '"
                 << CLUSTER_PARAMETER_GROUP_NAME << "', and \n"
```

```
                  << "selected engine version " <<
engineVersion.GetEngineVersion()
                  << ".\nThis typically takes several minutes." << std::endl;

        Aws::RDS::Model::CreateDBClusterRequest request;
        request.SetDBClusterIdentifier(DB_CLUSTER_IDENTIFIER);
        request.SetDBClusterParameterGroupName(CLUSTER_PARAMETER_GROUP_NAME);
        request.SetEngine(engineName);
        request.SetEngineVersion(engineVersionName);
        request.SetMasterUsername(administratorName);
        request.SetMasterUserPassword(administratorPassword);

        Aws::RDS::Model::CreateDBClusterOutcome outcome =
                client.CreateDBCluster(request);

        if (outcome.IsSuccess()) {
            std::cout << "The DB cluster creation has started."
                      << std::endl;
        }
        else {
            std::cerr << "Error with Aurora::CreateDBCluster. "
                      << outcome.GetError().GetMessage()
                      << std::endl;
            cleanUpResources(CLUSTER_PARAMETER_GROUP_NAME, "", "", client);
            return false;
        }
    }

    std::cout << "Waiting for the DB cluster to become available." << std::endl;

    int counter = 0;
    // 11. Wait for the DB cluster to become available.
    do {
        std::this_thread::sleep_for(std::chrono::seconds(1));
        ++counter;
        if (counter > 900) {
            std::cerr << "Wait for cluster to become available timed out ofter "
                      << counter
                      << " seconds." << std::endl;
            cleanUpResources(CLUSTER_PARAMETER_GROUP_NAME,
                             DB_CLUSTER_IDENTIFIER, "", client);
            return false;
        }
```

```
        dbCluster = Aws::RDS::Model::DBCluster();
        if (!describeDBCluster(DB_CLUSTER_IDENTIFIER, dbCluster, client)) {
            cleanUpResources(CLUSTER_PARAMETER_GROUP_NAME,
                             DB_CLUSTER_IDENTIFIER, "", client);
            return false;
        }

        if ((counter % 20) == 0) {
            std::cout << "Current DB cluster status is '"
                      << dbCluster.GetStatus()
                      << "' after " << counter << " seconds." << std::endl;
        }
    } while (dbCluster.GetStatus() != "available");

    if (dbCluster.GetStatus() == "available") {
        std::cout << "The DB cluster has been created." << std::endl;
    }

    printAsterisksLine();
    Aws::RDS::Model::DBInstance dbInstance;
    // 11.  Check if the DB instance already exists.
    if (!describeDBInstance(DB_INSTANCE_IDENTIFIER, dbInstance, client)) {
        cleanUpResources(CLUSTER_PARAMETER_GROUP_NAME, DB_CLUSTER_IDENTIFIER, "",
                         client);
        return false;
    }

    if (dbInstance.DbInstancePortHasBeenSet()) {
        std::cout << "The DB instance already exists." << std::endl;
    }
    else {
        std::cout << "Let's create a DB instance." << std::endl;

        Aws::String dbInstanceClass;
        // 12.  Get a list of instance classes.
        if (!chooseDBInstanceClass(engineName,
                                   engineVersionName,
                                   dbInstanceClass,
                                   client)) {
            cleanUpResources(CLUSTER_PARAMETER_GROUP_NAME, DB_CLUSTER_IDENTIFIER,
""",
                             client);
            return false;
        }
```

```
        std::cout << "Creating a DB instance named '" << DB_INSTANCE_IDENTIFIER
                  << "' with selected DB instance class '" << dbInstanceClass
                  << "'.\nThis typically takes several minutes." << std::endl;

        // 13. Create a DB instance.
        Aws::RDS::Model::CreateDBInstanceRequest request;
        request.SetDBInstanceIdentifier(DB_INSTANCE_IDENTIFIER);
        request.SetDBClusterIdentifier(DB_CLUSTER_IDENTIFIER);
        request.SetEngine(engineName);
        request.SetDBInstanceClass(dbInstanceClass);

        Aws::RDS::Model::CreateDBInstanceOutcome outcome =
                client.CreateDBInstance(request);

        if (outcome.IsSuccess()) {
            std::cout << "The DB instance creation has started."
                      << std::endl;
        }
        else {
            std::cerr << "Error with RDS::CreateDBInstance. "
                      << outcome.GetError().GetMessage()
                      << std::endl;
            cleanUpResources(CLUSTER_PARAMETER_GROUP_NAME, DB_CLUSTER_IDENTIFIER,
"",
                             client);
            return false;
        }
    }

    std::cout << "Waiting for the DB instance to become available." << std::endl;

    counter = 0;
    // 14. Wait for the DB instance to become available.
    do {
        std::this_thread::sleep_for(std::chrono::seconds(1));
        ++counter;
        if (counter > 900) {
            std::cerr << "Wait for instance to become available timed out ofter "
                      << counter
                      << " seconds." << std::endl;
            cleanUpResources(CLUSTER_PARAMETER_GROUP_NAME,
                             DB_CLUSTER_IDENTIFIER, DB_INSTANCE_IDENTIFIER,
client);
```

```
            return false;
        }

        dbInstance = Aws::RDS::Model::DBInstance();
        if (!describeDBInstance(DB_INSTANCE_IDENTIFIER, dbInstance, client)) {
            cleanUpResources(CLUSTER_PARAMETER_GROUP_NAME,
                             DB_CLUSTER_IDENTIFIER, DB_INSTANCE_IDENTIFIER,
client);
            return false;
        }

        if ((counter % 20) == 0) {
            std::cout << "Current DB instance status is '"
                      << dbInstance.GetDBInstanceStatus()
                      << "' after " << counter << " seconds." << std::endl;
        }
    } while (dbInstance.GetDBInstanceStatus() != "available");

    if (dbInstance.GetDBInstanceStatus() == "available") {
        std::cout << "The DB instance has been created." << std::endl;
    }

    // 15. Display the connection string that can be used to connect a 'mysql'
shell to the database.
    displayConnection(dbCluster);

    printAsterisksLine();

    if (askYesNoQuestion(
            "Do you want to create a snapshot of your DB cluster (y/n)? ")) {
        Aws::String snapshotID(DB_CLUSTER_IDENTIFIER + "-" +
                               Aws::String(Aws::Utils::UUID::RandomUUID()));
        {
            std::cout << "Creating a snapshot named " << snapshotID << "." <<
std::endl;
            std::cout << "This typically takes a few minutes." << std::endl;

            // 16. Create a snapshot of the DB cluster. (CreateDBClusterSnapshot)
            Aws::RDS::Model::CreateDBClusterSnapshotRequest request;
            request.SetDBClusterIdentifier(DB_CLUSTER_IDENTIFIER);
            request.SetDBClusterSnapshotIdentifier(snapshotID);

            Aws::RDS::Model::CreateDBClusterSnapshotOutcome outcome =
                    client.CreateDBClusterSnapshot(request);
```

```
            if (outcome.IsSuccess()) {
                std::cout << "Snapshot creation has started."
                          << std::endl;
            }
            else {
                std::cerr << "Error with Aurora::CreateDBClusterSnapshot. "
                          << outcome.GetError().GetMessage()
                          << std::endl;
                cleanUpResources(CLUSTER_PARAMETER_GROUP_NAME,
                                 DB_CLUSTER_IDENTIFIER, DB_INSTANCE_IDENTIFIER,
client);
                return false;
            }
        }

        std::cout << "Waiting for the snapshot to become available." <<
std::endl;

        Aws::RDS::Model::DBClusterSnapshot snapshot;
        counter = 0;
        do {
            std::this_thread::sleep_for(std::chrono::seconds(1));
            ++counter;
            if (counter > 600) {
                std::cerr << "Wait for snapshot to be available timed out ofter "
                          << counter
                          << " seconds." << std::endl;
                cleanUpResources(CLUSTER_PARAMETER_GROUP_NAME,
                                 DB_CLUSTER_IDENTIFIER, DB_INSTANCE_IDENTIFIER,
client);
                return false;
            }

            // 17. Wait for the snapshot to become available.
            Aws::RDS::Model::DescribeDBClusterSnapshotsRequest request;
            request.SetDBClusterSnapshotIdentifier(snapshotID);

            Aws::RDS::Model::DescribeDBClusterSnapshotsOutcome outcome =
                    client.DescribeDBClusterSnapshots(request);

            if (outcome.IsSuccess()) {
                snapshot = outcome.GetResult().GetDBClusterSnapshots()[0];
            }
```

```
            else {
                std::cerr << "Error with Aurora::DescribeDBClusterSnapshots. "
                          << outcome.GetError().GetMessage()
                          << std::endl;
                cleanUpResources(CLUSTER_PARAMETER_GROUP_NAME,
                                 DB_CLUSTER_IDENTIFIER, DB_INSTANCE_IDENTIFIER,
 client);
                return false;
            }

            if ((counter % 20) == 0) {
                std::cout << "Current snapshot status is '"
                          << snapshot.GetStatus()
                          << "' after " << counter << " seconds." << std::endl;
            }
        } while (snapshot.GetStatus() != "available");

        if (snapshot.GetStatus() != "available") {
            std::cout << "A snapshot has been created." << std::endl;
        }
    }

    printAsterisksLine();

    bool result = true;
    if (askYesNoQuestion(
            "Do you want to delete the DB cluster, DB instance, and parameter
 group (y/n)? ")) {
        result = cleanUpResources(CLUSTER_PARAMETER_GROUP_NAME,
                                  DB_CLUSTER_IDENTIFIER, DB_INSTANCE_IDENTIFIER,
                                  client);
    }

    return result;
}

//! Routine which gets a DB cluster description.
/*!
 \sa describeDBCluster()
 \param dbClusterIdentifier: A DB cluster identifier.
 \param clusterResult: The 'DBCluster' object containing the description.
 \param client: 'RDSClient' instance.
 \return bool: Successful completion.
 */
```

```
bool AwsDoc::Aurora::describeDBCluster(const Aws::String &dbClusterIdentifier,
                                       Aws::RDS::Model::DBCluster &clusterResult,
                                       const Aws::RDS::RDSClient &client) {
    Aws::RDS::Model::DescribeDBClustersRequest request;
    request.SetDBClusterIdentifier(dbClusterIdentifier);

    Aws::RDS::Model::DescribeDBClustersOutcome outcome =
            client.DescribeDBClusters(request);

    bool result = true;
    if (outcome.IsSuccess()) {
        clusterResult = outcome.GetResult().GetDBClusters()[0];
    }
    else if (outcome.GetError().GetErrorType() !=
             Aws::RDS::RDSErrors::D_B_CLUSTER_NOT_FOUND_FAULT) {
        result = false;
        std::cerr << "Error with Aurora::GDescribeDBClusters. "
                  << outcome.GetError().GetMessage()
                  << std::endl;
    }
        // This example does not log an error if the DB cluster does not exist.
        // Instead, clusterResult is set to empty.
    else {
        clusterResult = Aws::RDS::Model::DBCluster();
    }

    return result;

}


//! Routine which gets DB parameters using the 'DescribeDBClusterParameters' api.
/*!
 \sa getDBCLusterParameters()
 \param parameterGroupName: The name of the cluster parameter group.
 \param namePrefix: Prefix string to filter results by parameter name.
 \param source: A source such as 'user', ignored if empty.
 \param parametersResult: Vector of 'Parameter' objects returned by the routine.
 \param client: 'RDSClient' instance.
 \return bool: Successful completion.
 */
bool AwsDoc::Aurora::getDBCLusterParameters(const Aws::String
 &parameterGroupName,
                                            const Aws::String &namePrefix,
```

```
                                            const Aws::String &source,

 Aws::Vector<Aws::RDS::Model::Parameter> &parametersResult,
                                            const Aws::RDS::RDSClient &client) {
    Aws::String marker; // The marker is used for pagination.
    do {
        Aws::RDS::Model::DescribeDBClusterParametersRequest request;
        request.SetDBClusterParameterGroupName(CLUSTER_PARAMETER_GROUP_NAME);
        if (!marker.empty()) {
            request.SetMarker(marker);
        }
        if (!source.empty()) {
            request.SetSource(source);
        }

        Aws::RDS::Model::DescribeDBClusterParametersOutcome outcome =
                client.DescribeDBClusterParameters(request);

        if (outcome.IsSuccess()) {
            const Aws::Vector<Aws::RDS::Model::Parameter> &parameters =
                    outcome.GetResult().GetParameters();
            for (const Aws::RDS::Model::Parameter &parameter: parameters) {
                if (!namePrefix.empty()) {
                    if (parameter.GetParameterName().find(namePrefix) == 0) {
                        parametersResult.push_back(parameter);
                    }
                }
                else {
                    parametersResult.push_back(parameter);
                }
            }

            marker = outcome.GetResult().GetMarker();
        }
        else {
            std::cerr << "Error with Aurora::DescribeDBClusterParameters. "
                      << outcome.GetError().GetMessage()
                      << std::endl;
            return false;
        }
    } while (!marker.empty());

    return true;
}
```

```
//! Routine which gets available DB engine versions for an engine name and
//! an optional parameter group family.
/*!
 \sa getDBEngineVersions()
 \param engineName: A DB engine name.
 \param parameterGroupFamily: A parameter group family name, ignored if empty.
 \param engineVersionsResult: Vector of 'DBEngineVersion' objects returned by the
 routine.
 \param client: 'RDSClient' instance.
 \return bool: Successful completion.
 */
bool AwsDoc::Aurora::getDBEngineVersions(const Aws::String &engineName,
                                         const Aws::String &parameterGroupFamily,

 Aws::Vector<Aws::RDS::Model::DBEngineVersion> &engineVersionsResult,
                                         const Aws::RDS::RDSClient &client) {
    Aws::RDS::Model::DescribeDBEngineVersionsRequest request;
    request.SetEngine(engineName);
    if (!parameterGroupFamily.empty()) {
        request.SetDBParameterGroupFamily(parameterGroupFamily);
    }

    engineVersionsResult.clear();
    Aws::String marker; // The marker is used for pagination.
    do {
        if (!marker.empty()) {
            request.SetMarker(marker);
        }

        Aws::RDS::Model::DescribeDBEngineVersionsOutcome outcome =
                client.DescribeDBEngineVersions(request);

        if (outcome.IsSuccess()) {
            const Aws::Vector<Aws::RDS::Model::DBEngineVersion> &engineVersions =
                    outcome.GetResult().GetDBEngineVersions();

            engineVersionsResult.insert(engineVersionsResult.end(),
                                        engineVersions.begin(),
 engineVersions.end());
            marker  = outcome.GetResult().GetMarker();
        }
        else {
```

```
            std::cerr << "Error with Aurora::DescribeDBEngineVersionsRequest. "
                      << outcome.GetError().GetMessage()
                      << std::endl;
        }
    } while (!marker.empty());

    return true;
}


//! Routine which gets a DB instance description.
/*!
 \sa describeDBCluster()
 \param dbInstanceIdentifier: A DB instance identifier.
 \param instanceResult: The 'DBInstance' object containing the description.
 \param client: 'RDSClient' instance.
 \return bool: Successful completion.
 */
bool AwsDoc::Aurora::describeDBInstance(const Aws::String &dbInstanceIdentifier,
                                        Aws::RDS::Model::DBInstance
 &instanceResult,
                                        const Aws::RDS::RDSClient &client) {
    Aws::RDS::Model::DescribeDBInstancesRequest request;
    request.SetDBInstanceIdentifier(dbInstanceIdentifier);

    Aws::RDS::Model::DescribeDBInstancesOutcome outcome =
            client.DescribeDBInstances(request);

    bool result = true;
    if (outcome.IsSuccess()) {
        instanceResult = outcome.GetResult().GetDBInstances()[0];
    }
    else if (outcome.GetError().GetErrorType() !=
             Aws::RDS::RDSErrors::D_B_INSTANCE_NOT_FOUND_FAULT) {
        result = false;
        std::cerr << "Error with Aurora::DescribeDBInstances. "
                  << outcome.GetError().GetMessage()
                  << std::endl;
    }
        // This example does not log an error if the DB instance does not exist.
        // Instead, instanceResult is set to empty.
    else {
        instanceResult = Aws::RDS::Model::DBInstance();
    }
```

```
    return result;
}


//! Routine which gets available DB instance classes, displays the list
//! to the user, and returns the user selection.
/*!
 \sa chooseDBInstanceClass()
 \param engineName: The DB engine name.
 \param engineVersion: The DB engine version.
 \param dbInstanceClass: String for DB instance class chosen by the user.
 \param client: 'RDSClient' instance.
 \return bool: Successful completion.
 */
bool AwsDoc::Aurora::chooseDBInstanceClass(const Aws::String &engine,
                                           const Aws::String &engineVersion,
                                           Aws::String &dbInstanceClass,
                                           const Aws::RDS::RDSClient &client) {
    std::vector<Aws::String> instanceClasses;
    Aws::String marker; // The marker is used for pagination.
    do {
        Aws::RDS::Model::DescribeOrderableDBInstanceOptionsRequest request;
        request.SetEngine(engine);
        request.SetEngineVersion(engineVersion);
        if (!marker.empty()) {
            request.SetMarker(marker);
        }

        Aws::RDS::Model::DescribeOrderableDBInstanceOptionsOutcome outcome =
                client.DescribeOrderableDBInstanceOptions(request);

        if (outcome.IsSuccess()) {
            const Aws::Vector<Aws::RDS::Model::OrderableDBInstanceOption>
 &options =
                    outcome.GetResult().GetOrderableDBInstanceOptions();
            for (const Aws::RDS::Model::OrderableDBInstanceOption &option:
 options) {
                const Aws::String &instanceClass = option.GetDBInstanceClass();
                if (std::find(instanceClasses.begin(), instanceClasses.end(),
                              instanceClass) == instanceClasses.end()) {
                    instanceClasses.push_back(instanceClass);
                }
            }
```

```
            marker = outcome.GetResult().GetMarker();
        }
        else {
            std::cerr << "Error with Aurora::DescribeOrderableDBInstanceOptions.
 "
                      << outcome.GetError().GetMessage()
                      << std::endl;
            return false;
        }
    } while (!marker.empty());

    std::cout << "The available DB instance classes for your database engine
 are:"
              << std::endl;
    for (int i = 0; i < instanceClasses.size(); ++i) {
        std::cout << "   " << i + 1 << ": " << instanceClasses[i] << std::endl;
    }

    int choice = askQuestionForIntRange(
            "Which DB instance class do you want to use? ",
            1, static_cast<int>(instanceClasses.size()));
    dbInstanceClass = instanceClasses[choice - 1];
    return true;
}

//! Routine which deletes resources created by the scenario.
/*!
\sa cleanUpResources()
\param parameterGroupName: A parameter group name, this may be empty.
\param dbInstanceIdentifier: A DB instance identifier, this may be empty.
\param client: 'RDSClient' instance.
\return bool: Successful completion.
*/
bool AwsDoc::Aurora::cleanUpResources(const Aws::String &parameterGroupName,
                                      const Aws::String &dbClusterIdentifier,
                                      const Aws::String &dbInstanceIdentifier,
                                      const Aws::RDS::RDSClient &client) {
    bool result = true;
    bool instanceDeleting = false;
    bool clusterDeleting = false;
    if (!dbInstanceIdentifier.empty()) {
        {
            // 18. Delete the DB instance.
            Aws::RDS::Model::DeleteDBInstanceRequest request;
```

```
            request.SetDBInstanceIdentifier(dbInstanceIdentifier);
            request.SetSkipFinalSnapshot(true);
            request.SetDeleteAutomatedBackups(true);

            Aws::RDS::Model::DeleteDBInstanceOutcome outcome =
                    client.DeleteDBInstance(request);

            if (outcome.IsSuccess()) {
                std::cout << "DB instance deletion has started."
                          << std::endl;
                instanceDeleting = true;
                std::cout
                        << "Waiting for DB instance to delete before deleting the
parameter group."
                        << std::endl;
            }
            else {
                std::cerr << "Error with Aurora::DeleteDBInstance. "
                          << outcome.GetError().GetMessage()
                          << std::endl;
                result = false;
            }
        }
    }

    if (!dbClusterIdentifier.empty()) {
        {
            // 19. Delete the DB cluster.
            Aws::RDS::Model::DeleteDBClusterRequest request;
            request.SetDBClusterIdentifier(dbClusterIdentifier);
            request.SetSkipFinalSnapshot(true);

            Aws::RDS::Model::DeleteDBClusterOutcome outcome =
                    client.DeleteDBCluster(request);

            if (outcome.IsSuccess()) {
                std::cout << "DB cluster deletion has started."
                          << std::endl;
                clusterDeleting = true;
                std::cout
                        << "Waiting for DB cluster to delete before deleting the
parameter group."
                        << std::endl;
                std::cout << "This may take a while." << std::endl;
```

```
            }
            else {
                std::cerr << "Error with Aurora::DeleteDBCluster. "
                          << outcome.GetError().GetMessage()
                          << std::endl;
                result = false;
            }
        }
    }
    int counter = 0;

    while (clusterDeleting || instanceDeleting) {
        // 20. Wait for the DB cluster and instance to be deleted.
        std::this_thread::sleep_for(std::chrono::seconds(1));
        ++counter;
        if (counter > 800) {
            std::cerr << "Wait for instance to delete timed out ofter " <<
counter
                      << " seconds." << std::endl;
            return false;
        }

        Aws::RDS::Model::DBInstance dbInstance = Aws::RDS::Model::DBInstance();
        if (instanceDeleting) {
            if (!describeDBInstance(dbInstanceIdentifier, dbInstance, client)) {
                return false;
            }
            instanceDeleting = dbInstance.DBInstanceIdentifierHasBeenSet();
        }

        Aws::RDS::Model::DBCluster dbCluster = Aws::RDS::Model::DBCluster();
        if (clusterDeleting) {
            if (!describeDBCluster(dbClusterIdentifier, dbCluster, client)) {
                return false;
            }

            clusterDeleting = dbCluster.DBClusterIdentifierHasBeenSet();
        }

        if ((counter % 20) == 0) {
            if (instanceDeleting) {
                std::cout << "Current DB instance status is '"
                          << dbInstance.GetDBInstanceStatus() << "." <<
std::endl;
```

```
            }

            if (clusterDeleting) {
                std::cout << "Current DB cluster status is '"
                          << dbCluster.GetStatus() << "." << std::endl;
            }
        }
    }

    if (!parameterGroupName.empty()) {
        // 21. Delete the DB cluster parameter group.
        Aws::RDS::Model::DeleteDBClusterParameterGroupRequest request;
        request.SetDBClusterParameterGroupName(parameterGroupName);

        Aws::RDS::Model::DeleteDBClusterParameterGroupOutcome outcome =
                client.DeleteDBClusterParameterGroup(request);

        if (outcome.IsSuccess()) {
            std::cout << "The DB parameter group was successfully deleted."
                      << std::endl;
        }
        else {
            std::cerr << "Error with Aurora::DeleteDBClusterParameterGroup. "
                      << outcome.GetError().GetMessage()
                      << std::endl;
            result = false;
        }
    }

    return result;
}
```

- Per informazioni dettagliate sull'API, consulta i seguenti argomenti nella documentazione di riferimento dell'API AWS SDK per C++.
  - CreaDBCluster
  - CreaDBClusterParameterGroup
  - Crea DBCluster istantanea
  - CreaDBInstance
  - EliminaDBCluster
  - EliminaDBClusterParameterGroup

- EliminaDBInstance
- Descriva DBCluster ParameterGroups
- DBClusterDescrivi parametri
- Descrivi le DBCluster istantanee
- Descriva DBClusters
- Descrivi DBEngine versioni
- Descriva DBInstances
- DescribeOrderableDBInstanceOpzioni
- ModificaDBClusterParameterGroup

Go

SDK per Go V2

> **Note**
>
> C'è altro da fare GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice AWS.

Esegui uno scenario interattivo al prompt dei comandi.

```
import (
 "aurora/actions"
 "context"
 "fmt"
 "log"
 "slices"
 "sort"
 "strconv"
 "strings"
 "time"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/service/rds"
 "github.com/aws/aws-sdk-go-v2/service/rds/types"
 "github.com/awsdocs/aws-doc-sdk-examples/gov2/demotools"
```

```
 "github.com/google/uuid"
)

// GetStartedClusters is an interactive example that shows you how to use the AWS
 SDK for Go
// with Amazon Aurora to do the following:
//
// 1. Create a custom DB cluster parameter group and set parameter values.
// 2. Create an Aurora DB cluster that is configured to use the parameter group.
// 3. Create a DB instance in the DB cluster that contains a database.
// 4. Take a snapshot of the DB cluster.
// 5. Delete the DB instance, DB cluster, and parameter group.
type GetStartedClusters struct {
 sdkConfig  aws.Config
 dbClusters actions.DbClusters
 questioner demotools.IQuestioner
 helper     IScenarioHelper
 isTestRun  bool
}

// NewGetStartedClusters constructs a GetStartedClusters instance from a
 configuration.
// It uses the specified config to get an Amazon Relational Database Service
 (Amazon RDS)
// client and create wrappers for the actions used in the scenario.
func NewGetStartedClusters(sdkConfig aws.Config, questioner
 demotools.IQuestioner,
 helper IScenarioHelper) GetStartedClusters {
 auroraClient := rds.NewFromConfig(sdkConfig)
 return GetStartedClusters{
  sdkConfig:  sdkConfig,
  dbClusters: actions.DbClusters{AuroraClient: auroraClient},
  questioner: questioner,
  helper:     helper,
 }
}

// Run runs the interactive scenario.
func (scenario GetStartedClusters) Run(ctx context.Context, dbEngine string,
 parameterGroupName string,
 clusterName string, dbName string) {
 defer func() {
  if r := recover(); r != nil {
   log.Println("Something went wrong with the demo.")
```

```
  }
 }()

 log.Println(strings.Repeat("-", 88))
 log.Println("Welcome to the Amazon Aurora DB Cluster demo.")
 log.Println(strings.Repeat("-", 88))

 parameterGroup := scenario.CreateParameterGroup(ctx, dbEngine,
 parameterGroupName)
 scenario.SetUserParameters(ctx, parameterGroupName)
 cluster := scenario.CreateCluster(ctx, clusterName, dbEngine, dbName,
 parameterGroup)
 scenario.helper.Pause(5)
 dbInstance := scenario.CreateInstance(ctx, cluster)
 scenario.DisplayConnection(cluster)
 scenario.CreateSnapshot(ctx, clusterName)
 scenario.Cleanup(ctx, dbInstance, cluster, parameterGroup)

 log.Println(strings.Repeat("-", 88))
 log.Println("Thanks for watching!")
 log.Println(strings.Repeat("-", 88))
}

// CreateParameterGroup shows how to get available engine versions for a
 specified
// database engine and create a DB cluster parameter group that is compatible
 with a
// selected engine family.
func (scenario GetStartedClusters) CreateParameterGroup(ctx context.Context,
 dbEngine string,
 parameterGroupName string) *types.DBClusterParameterGroup {

 log.Printf("Checking for an existing DB cluster parameter group named %v.\n",
  parameterGroupName)
 parameterGroup, err := scenario.dbClusters.GetParameterGroup(ctx,
 parameterGroupName)
 if err != nil {
  panic(err)
 }
 if parameterGroup == nil {
  log.Printf("Getting available database engine versions for %v.\n", dbEngine)
  engineVersions, err := scenario.dbClusters.GetEngineVersions(ctx, dbEngine, "")
  if err != nil {
   panic(err)
```

```
  }

  familySet := map[string]struct{}{}
  for _, family := range engineVersions {
   familySet[*family.DBParameterGroupFamily] = struct{}{}
  }
  var families []string
  for family := range familySet {
   families = append(families, family)
  }
  sort.Strings(families)
  familyIndex := scenario.questioner.AskChoice("Which family do you want to use?
\n", families)
  log.Println("Creating a DB cluster parameter group.")
  _, err = scenario.dbClusters.CreateParameterGroup(
   ctx, parameterGroupName, families[familyIndex], "Example parameter group.")
  if err != nil {
   panic(err)
  }
  parameterGroup, err = scenario.dbClusters.GetParameterGroup(ctx,
 parameterGroupName)
  if err != nil {
   panic(err)
  }
 }
 log.Printf("Parameter group %v:\n", *parameterGroup.DBParameterGroupFamily)
 log.Printf("\tName: %v\n", *parameterGroup.DBClusterParameterGroupName)
 log.Printf("\tARN: %v\n", *parameterGroup.DBClusterParameterGroupArn)
 log.Printf("\tFamily: %v\n", *parameterGroup.DBParameterGroupFamily)
 log.Printf("\tDescription: %v\n", *parameterGroup.Description)
 log.Println(strings.Repeat("-", 88))
 return parameterGroup

}

// SetUserParameters shows how to get the parameters contained in a custom
 parameter
// group and update some of the parameter values in the group.
func (scenario GetStartedClusters) SetUserParameters(ctx context.Context,
 parameterGroupName string) {
 log.Println("Let's set some parameter values in your parameter group.")
 dbParameters, err := scenario.dbClusters.GetParameters(ctx, parameterGroupName,
 "")
 if err != nil {
```

```
  panic(err)
 }
 var updateParams []types.Parameter
 for _, dbParam := range dbParameters {
  if strings.HasPrefix(*dbParam.ParameterName, "auto_increment") &&
   *dbParam.IsModifiable && *dbParam.DataType == "integer" {
   log.Printf("The %v parameter is described as:\n\t%v",
    *dbParam.ParameterName, *dbParam.Description)
   rangeSplit := strings.Split(*dbParam.AllowedValues, "-")
   lower, _ := strconv.Atoi(rangeSplit[0])
   upper, _ := strconv.Atoi(rangeSplit[1])
   newValue := scenario.questioner.AskInt(
    fmt.Sprintf("Enter a value between %v and %v:", lower, upper),
    demotools.InIntRange{Lower: lower, Upper: upper})
   dbParam.ParameterValue = aws.String(strconv.Itoa(newValue))
   updateParams = append(updateParams, dbParam)
  }
 }
 err = scenario.dbClusters.UpdateParameters(ctx, parameterGroupName,
 updateParams)
 if err != nil {
  panic(err)
 }
 log.Println("You can get a list of parameters you've set by specifying a source
 of 'user'.")
 userParameters, err := scenario.dbClusters.GetParameters(ctx,
 parameterGroupName, "user")
 if err != nil {
  panic(err)
 }
 log.Println("Here are the parameters you've set:")
 for _, param := range userParameters {
  log.Printf("\t%v: %v\n", *param.ParameterName, *param.ParameterValue)
 }
 log.Println(strings.Repeat("-", 88))
}

// CreateCluster shows how to create an Aurora DB cluster that contains a
 database
// of a specified type. The database is also configured to use a custom DB
 cluster
// parameter group.
func (scenario GetStartedClusters) CreateCluster(ctx context.Context, clusterName
 string, dbEngine string,
```

```
 dbName string, parameterGroup *types.DBClusterParameterGroup) *types.DBCluster {

 log.Println("Checking for an existing DB cluster.")
 cluster, err := scenario.dbClusters.GetDbCluster(ctx, clusterName)
 if err != nil {
  panic(err)
 }
 if cluster == nil {
  adminUsername := scenario.questioner.Ask(
   "Enter an administrator user name for the database: ", demotools.NotEmpty{})
  adminPassword := scenario.questioner.Ask(
   "Enter a password for the administrator (at least 8 characters): ",
 demotools.NotEmpty{})
  engineVersions, err := scenario.dbClusters.GetEngineVersions(ctx, dbEngine,
 *parameterGroup.DBParameterGroupFamily)
  if err != nil {
   panic(err)
  }
  var engineChoices []string
  for _, engine := range engineVersions {
   engineChoices = append(engineChoices, *engine.EngineVersion)
  }
  log.Println("The available engines for your parameter group are:")
  engineIndex := scenario.questioner.AskChoice("Which engine do you want to use?
\n", engineChoices)
  log.Printf("Creating DB cluster %v and database %v.\n", clusterName, dbName)
  log.Printf("The DB cluster is configured to use\nyour custom parameter group %v
\n",
   *parameterGroup.DBClusterParameterGroupName)
  log.Printf("and selected engine %v.\n", engineChoices[engineIndex])
  log.Println("This typically takes several minutes.")
  cluster, err = scenario.dbClusters.CreateDbCluster(
   ctx, clusterName, *parameterGroup.DBClusterParameterGroupName, dbName,
 dbEngine,
   engineChoices[engineIndex], adminUsername, adminPassword)
  if err != nil {
   panic(err)
  }
  for *cluster.Status != "available" {
   scenario.helper.Pause(30)
   cluster, err = scenario.dbClusters.GetDbCluster(ctx, clusterName)
   if err != nil {
    panic(err)
   }
```

```
    log.Println("Cluster created and available.")
   }
  }
  log.Println("Cluster data:")
  log.Printf("\tDBClusterIdentifier: %v\n", *cluster.DBClusterIdentifier)
  log.Printf("\tARN: %v\n", *cluster.DBClusterArn)
  log.Printf("\tStatus: %v\n", *cluster.Status)
  log.Printf("\tEngine: %v\n", *cluster.Engine)
  log.Printf("\tEngine version: %v\n", *cluster.EngineVersion)
  log.Printf("\tDBClusterParameterGroup: %v\n", *cluster.DBClusterParameterGroup)
  log.Printf("\tEngineMode: %v\n", *cluster.EngineMode)
  log.Println(strings.Repeat("-", 88))
  return cluster
}

// CreateInstance shows how to create a DB instance in an existing Aurora DB
 cluster.
// A new DB cluster contains no DB instances, so you must add one. The first DB
 instance
// that is added to a DB cluster defaults to a read-write DB instance.
func (scenario GetStartedClusters) CreateInstance(ctx context.Context, cluster
 *types.DBCluster) *types.DBInstance {
  log.Println("Checking for an existing database instance.")
  dbInstance, err := scenario.dbClusters.GetInstance(ctx,
  *cluster.DBClusterIdentifier)
  if err != nil {
   panic(err)
  }
  if dbInstance == nil {
   log.Println("Let's create a database instance in your DB cluster.")
   log.Println("First, choose a DB instance type:")
   instOpts, err := scenario.dbClusters.GetOrderableInstances(
    ctx, *cluster.Engine, *cluster.EngineVersion)
   if err != nil {
    panic(err)
   }
   var instChoices []string
   for _, opt := range instOpts {
    instChoices = append(instChoices, *opt.DBInstanceClass)
   }
   slices.Sort(instChoices)
   instChoices = slices.Compact(instChoices)
   instIndex := scenario.questioner.AskChoice(
    "Which DB instance class do you want to use?\n", instChoices)
```

```
  log.Println("Creating a database instance. This typically takes several
 minutes.")
  dbInstance, err = scenario.dbClusters.CreateInstanceInCluster(
   ctx, *cluster.DBClusterIdentifier, *cluster.DBClusterIdentifier,
 *cluster.Engine,
   instChoices[instIndex])
  if err != nil {
   panic(err)
  }
  for *dbInstance.DBInstanceStatus != "available" {
   scenario.helper.Pause(30)
   dbInstance, err = scenario.dbClusters.GetInstance(ctx,
 *cluster.DBClusterIdentifier)
   if err != nil {
    panic(err)
   }
  }
 }
 log.Println("Instance data:")
 log.Printf("\tDBInstanceIdentifier: %v\n", *dbInstance.DBInstanceIdentifier)
 log.Printf("\tARN: %v\n", *dbInstance.DBInstanceArn)
 log.Printf("\tStatus: %v\n", *dbInstance.DBInstanceStatus)
 log.Printf("\tEngine: %v\n", *dbInstance.Engine)
 log.Printf("\tEngine version: %v\n", *dbInstance.EngineVersion)
 log.Println(strings.Repeat("-", 88))
 return dbInstance
}

// DisplayConnection displays connection information about an Aurora DB cluster
 and tips
// on how to connect to it.
func (scenario GetStartedClusters) DisplayConnection(cluster *types.DBCluster) {
 log.Println(
  "You can now connect to your database using your favorite MySql client.\n" +
   "One way to connect is by using the 'mysql' shell on an Amazon EC2 instance\n"
 +
   "that is running in the same VPC as your database cluster. Pass the endpoint,
\n" +
   "port, and administrator user name to 'mysql' and enter your password\n" +
   "when prompted:")
 log.Printf("\n\tmysql -h %v -P %v -u %v -p\n",
  *cluster.Endpoint, *cluster.Port, *cluster.MasterUsername)
 log.Println("For more information, see the User Guide for Aurora:\n" +
```

```
  "\thttps://docs.aws.amazon.com/AmazonRDS/latest/AuroraUserGuide/
CHAP_GettingStartedAurora.CreatingConnecting.Aurora.html#CHAP_GettingStartedAurora.Aurora
 log.Println(strings.Repeat("-", 88))
}

// CreateSnapshot shows how to create a DB cluster snapshot and wait until it's
 available.
func (scenario GetStartedClusters) CreateSnapshot(ctx context.Context,
 clusterName string) {
 if scenario.questioner.AskBool(
  "Do you want to create a snapshot of your DB cluster (y/n)? ", "y") {
  snapshotId := fmt.Sprintf("%v-%v", clusterName, scenario.helper.UniqueId())
  log.Printf("Creating a snapshot named %v. This typically takes a few minutes.
\n", snapshotId)
  snapshot, err := scenario.dbClusters.CreateClusterSnapshot(ctx, clusterName,
 snapshotId)
  if err != nil {
   panic(err)
  }
  for *snapshot.Status != "available" {
   scenario.helper.Pause(30)
   snapshot, err = scenario.dbClusters.GetClusterSnapshot(ctx, snapshotId)
   if err != nil {
    panic(err)
   }
  }
  log.Println("Snapshot data:")
  log.Printf("\tDBClusterSnapshotIdentifier: %v\n",
 *snapshot.DBClusterSnapshotIdentifier)
  log.Printf("\tARN: %v\n", *snapshot.DBClusterSnapshotArn)
  log.Printf("\tStatus: %v\n", *snapshot.Status)
  log.Printf("\tEngine: %v\n", *snapshot.Engine)
  log.Printf("\tEngine version: %v\n", *snapshot.EngineVersion)
  log.Printf("\tDBClusterIdentifier: %v\n", *snapshot.DBClusterIdentifier)
  log.Printf("\tSnapshotCreateTime: %v\n", *snapshot.SnapshotCreateTime)
  log.Println(strings.Repeat("-", 88))
 }
}

// Cleanup shows how to clean up a DB instance, DB cluster, and DB cluster
 parameter group.
// Before the DB cluster parameter group can be deleted, all associated DB
 instances and
// DB clusters must first be deleted.
```

```
func (scenario GetStartedClusters) Cleanup(ctx context.Context, dbInstance
 *types.DBInstance, cluster *types.DBCluster,
 parameterGroup *types.DBClusterParameterGroup) {

 if scenario.questioner.AskBool(
  "\nDo you want to delete the database instance, DB cluster, and parameter group
 (y/n)? ", "y") {
  log.Printf("Deleting database instance %v.\n",
 *dbInstance.DBInstanceIdentifier)
  err := scenario.dbClusters.DeleteInstance(ctx,
 *dbInstance.DBInstanceIdentifier)
  if err != nil {
   panic(err)
  }
  log.Printf("Deleting database cluster %v.\n", *cluster.DBClusterIdentifier)
  err = scenario.dbClusters.DeleteDbCluster(ctx, *cluster.DBClusterIdentifier)
  if err != nil {
   panic(err)
  }
  log.Println(
   "Waiting for the DB instance and DB cluster to delete. This typically takes
 several minutes.")
  for dbInstance != nil || cluster != nil {
   scenario.helper.Pause(30)
   if dbInstance != nil {
    dbInstance, err = scenario.dbClusters.GetInstance(ctx,
 *dbInstance.DBInstanceIdentifier)
    if err != nil {
     panic(err)
    }
   }
   if cluster != nil {
    cluster, err = scenario.dbClusters.GetDbCluster(ctx,
 *cluster.DBClusterIdentifier)
    if err != nil {
     panic(err)
    }
   }
  }
  log.Printf("Deleting parameter group %v.",
 *parameterGroup.DBClusterParameterGroupName)
  err = scenario.dbClusters.DeleteParameterGroup(ctx,
 *parameterGroup.DBClusterParameterGroupName)
  if err != nil {
```

```
   panic(err)
  }
 }
}

// IScenarioHelper abstracts the function from a scenario so that it
// can be mocked for unit testing.
type IScenarioHelper interface {
 Pause(secs int)
 UniqueId() string
}
type ScenarioHelper struct{}

// Pause waits for the specified number of seconds.
func (helper ScenarioHelper) Pause(secs int) {
 time.Sleep(time.Duration(secs) * time.Second)
}

// UniqueId returns a new UUID.
func (helper ScenarioHelper) UniqueId() string {
 return uuid.New().String()
}
```

Definisci le funzioni richiamate dallo scenario per gestire le operazioni Aurora.

```
import (
 "context"
 "errors"
 "log"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/service/rds"
 "github.com/aws/aws-sdk-go-v2/service/rds/types"
)

type DbClusters struct {
 AuroraClient *rds.Client
}
```

```
// GetParameterGroup gets a DB cluster parameter group by name.
func (clusters *DbClusters) GetParameterGroup(ctx context.Context,
 parameterGroupName string) (
 *types.DBClusterParameterGroup, error) {
 output, err := clusters.AuroraClient.DescribeDBClusterParameterGroups(
  ctx, &rds.DescribeDBClusterParameterGroupsInput{
   DBClusterParameterGroupName: aws.String(parameterGroupName),
  })
 if err != nil {
  var notFoundError *types.DBParameterGroupNotFoundFault
  if errors.As(err, &notFoundError) {
   log.Printf("Parameter group %v does not exist.\n", parameterGroupName)
   err = nil
  } else {
   log.Printf("Error getting parameter group %v: %v\n", parameterGroupName, err)
  }
  return nil, err
 } else {
  return &output.DBClusterParameterGroups[0], err
 }
}


// CreateParameterGroup creates a DB cluster parameter group that is based on the
 specified
// parameter group family.
func (clusters *DbClusters) CreateParameterGroup(
 ctx context.Context, parameterGroupName string, parameterGroupFamily string,
 description string) (
 *types.DBClusterParameterGroup, error) {

 output, err := clusters.AuroraClient.CreateDBClusterParameterGroup(ctx,
  &rds.CreateDBClusterParameterGroupInput{
   DBClusterParameterGroupName: aws.String(parameterGroupName),
   DBParameterGroupFamily:      aws.String(parameterGroupFamily),
   Description:                 aws.String(description),
  })
 if err != nil {
  log.Printf("Couldn't create parameter group %v: %v\n", parameterGroupName, err)
  return nil, err
 } else {
  return output.DBClusterParameterGroup, err
 }
}
```

```
// DeleteParameterGroup deletes the named DB cluster parameter group.
func (clusters *DbClusters) DeleteParameterGroup(ctx context.Context,
 parameterGroupName string) error {
 _, err := clusters.AuroraClient.DeleteDBClusterParameterGroup(ctx,
  &rds.DeleteDBClusterParameterGroupInput{
   DBClusterParameterGroupName: aws.String(parameterGroupName),
  })
 if err != nil {
  log.Printf("Couldn't delete parameter group %v: %v\n", parameterGroupName, err)
  return err
 } else {
  return nil
 }
}



// GetParameters gets the parameters that are contained in a DB cluster parameter
 group.
func (clusters *DbClusters) GetParameters(ctx context.Context, parameterGroupName
 string, source string) (
 []types.Parameter, error) {

 var output *rds.DescribeDBClusterParametersOutput
 var params []types.Parameter
 var err error
 parameterPaginator :=
 rds.NewDescribeDBClusterParametersPaginator(clusters.AuroraClient,
  &rds.DescribeDBClusterParametersInput{
   DBClusterParameterGroupName: aws.String(parameterGroupName),
   Source:                      aws.String(source),
  })
 for parameterPaginator.HasMorePages() {
  output, err = parameterPaginator.NextPage(ctx)
  if err != nil {
   log.Printf("Couldn't get paramaeters for %v: %v\n", parameterGroupName, err)
   break
  } else {
   params = append(params, output.Parameters...)
  }
 }
```

```
 return params, err
}



// UpdateParameters updates parameters in a named DB cluster parameter group.
func (clusters *DbClusters) UpdateParameters(ctx context.Context,
 parameterGroupName string, params []types.Parameter) error {
 _, err := clusters.AuroraClient.ModifyDBClusterParameterGroup(ctx,
  &rds.ModifyDBClusterParameterGroupInput{
   DBClusterParameterGroupName: aws.String(parameterGroupName),
   Parameters:                  params,
  })
 if err != nil {
  log.Printf("Couldn't update parameters in %v: %v\n", parameterGroupName, err)
  return err
 } else {
  return nil
 }
}



// GetDbCluster gets data about an Aurora DB cluster.
func (clusters *DbClusters) GetDbCluster(ctx context.Context, clusterName string)
 (*types.DBCluster, error) {
 output, err := clusters.AuroraClient.DescribeDBClusters(ctx,
  &rds.DescribeDBClustersInput{
   DBClusterIdentifier: aws.String(clusterName),
  })
 if err != nil {
  var notFoundError *types.DBClusterNotFoundFault
  if errors.As(err, &notFoundError) {
   log.Printf("DB cluster %v does not exist.\n", clusterName)
   err = nil
  } else {
   log.Printf("Couldn't get DB cluster %v: %v\n", clusterName, err)
  }
  return nil, err
 } else {
  return &output.DBClusters[0], err
 }
}
```

```
// CreateDbCluster creates a DB cluster that is configured to use the specified
 parameter group.
// The newly created DB cluster contains a database that uses the specified
 engine and
// engine version.
func (clusters *DbClusters) CreateDbCluster(ctx context.Context, clusterName
 string, parameterGroupName string,
 dbName string, dbEngine string, dbEngineVersion string, adminName string,
 adminPassword string) (
 *types.DBCluster, error) {

 output, err := clusters.AuroraClient.CreateDBCluster(ctx,
 &rds.CreateDBClusterInput{
  DBClusterIdentifier:         aws.String(clusterName),
  Engine:                      aws.String(dbEngine),
  DBClusterParameterGroupName: aws.String(parameterGroupName),
  DatabaseName:                aws.String(dbName),
  EngineVersion:               aws.String(dbEngineVersion),
  MasterUserPassword:          aws.String(adminPassword),
  MasterUsername:              aws.String(adminName),
 })
 if err != nil {
  log.Printf("Couldn't create DB cluster %v: %v\n", clusterName, err)
  return nil, err
 } else {
  return output.DBCluster, err
 }
}



// DeleteDbCluster deletes a DB cluster without keeping a final snapshot.
func (clusters *DbClusters) DeleteDbCluster(ctx context.Context, clusterName
 string) error {
 _, err := clusters.AuroraClient.DeleteDBCluster(ctx, &rds.DeleteDBClusterInput{
  DBClusterIdentifier: aws.String(clusterName),
  SkipFinalSnapshot:   aws.Bool(true),
 })
 if err != nil {
  log.Printf("Couldn't delete DB cluster %v: %v\n", clusterName, err)
  return err
 } else {
```

```
  return nil
 }
}



// CreateClusterSnapshot creates a snapshot of a DB cluster.
func (clusters *DbClusters) CreateClusterSnapshot(ctx context.Context,
 clusterName string, snapshotName string) (
 *types.DBClusterSnapshot, error) {
 output, err := clusters.AuroraClient.CreateDBClusterSnapshot(ctx,
 &rds.CreateDBClusterSnapshotInput{
  DBClusterIdentifier:         aws.String(clusterName),
  DBClusterSnapshotIdentifier: aws.String(snapshotName),
 })
 if err != nil {
  log.Printf("Couldn't create snapshot %v: %v\n", snapshotName, err)
  return nil, err
 } else {
  return output.DBClusterSnapshot, nil
 }
}



// GetClusterSnapshot gets a DB cluster snapshot.
func (clusters *DbClusters) GetClusterSnapshot(ctx context.Context, snapshotName
 string) (*types.DBClusterSnapshot, error) {
 output, err := clusters.AuroraClient.DescribeDBClusterSnapshots(ctx,
  &rds.DescribeDBClusterSnapshotsInput{
   DBClusterSnapshotIdentifier: aws.String(snapshotName),
  })
 if err != nil {
  log.Printf("Couldn't get snapshot %v: %v\n", snapshotName, err)
  return nil, err
 } else {
  return &output.DBClusterSnapshots[0], nil
 }
}



// CreateInstanceInCluster creates a database instance in an existing DB cluster.
 The first database that is
```

```
// created defaults to a read-write DB instance.
func (clusters *DbClusters) CreateInstanceInCluster(ctx context.Context,
 clusterName string, instanceName string,
 dbEngine string, dbInstanceClass string) (*types.DBInstance, error) {
 output, err := clusters.AuroraClient.CreateDBInstance(ctx,
 &rds.CreateDBInstanceInput{
  DBInstanceIdentifier: aws.String(instanceName),
  DBClusterIdentifier:  aws.String(clusterName),
  Engine:               aws.String(dbEngine),
  DBInstanceClass:      aws.String(dbInstanceClass),
 })
 if err != nil {
  log.Printf("Couldn't create instance %v: %v\n", instanceName, err)
  return nil, err
 } else {
  return output.DBInstance, nil
 }
}



// GetInstance gets data about a DB instance.
func (clusters *DbClusters) GetInstance(ctx context.Context, instanceName string)
 (
 *types.DBInstance, error) {
 output, err := clusters.AuroraClient.DescribeDBInstances(ctx,
  &rds.DescribeDBInstancesInput{
   DBInstanceIdentifier: aws.String(instanceName),
  })
 if err != nil {
  var notFoundError *types.DBInstanceNotFoundFault
  if errors.As(err, &notFoundError) {
   log.Printf("DB instance %v does not exist.\n", instanceName)
   err = nil
  } else {
   log.Printf("Couldn't get instance %v: %v\n", instanceName, err)
  }
  return nil, err
 } else {
  return &output.DBInstances[0], nil
 }
}
```

```
// DeleteInstance deletes a DB instance.
func (clusters *DbClusters) DeleteInstance(ctx context.Context, instanceName
 string) error {
 _, err := clusters.AuroraClient.DeleteDBInstance(ctx,
 &rds.DeleteDBInstanceInput{
  DBInstanceIdentifier:   aws.String(instanceName),
  SkipFinalSnapshot:      aws.Bool(true),
  DeleteAutomatedBackups: aws.Bool(true),
 })
 if err != nil {
  log.Printf("Couldn't delete instance %v: %v\n", instanceName, err)
  return err
 } else {
  return nil
 }
}



// GetEngineVersions gets database engine versions that are available for the
 specified engine
// and parameter group family.
func (clusters *DbClusters) GetEngineVersions(ctx context.Context, engine string,
 parameterGroupFamily string) (
 []types.DBEngineVersion, error) {
 output, err := clusters.AuroraClient.DescribeDBEngineVersions(ctx,
  &rds.DescribeDBEngineVersionsInput{
   Engine:                 aws.String(engine),
   DBParameterGroupFamily: aws.String(parameterGroupFamily),
  })
 if err != nil {
  log.Printf("Couldn't get engine versions for %v: %v\n", engine, err)
  return nil, err
 } else {
  return output.DBEngineVersions, nil
 }
}



// GetOrderableInstances uses a paginator to get DB instance options that can be
 used to create DB instances that are
// compatible with a set of specifications.
```

```
func (clusters *DbClusters) GetOrderableInstances(ctx context.Context, engine
 string, engineVersion string) (
 []types.OrderableDBInstanceOption, error) {

 var output *rds.DescribeOrderableDBInstanceOptionsOutput
 var instances []types.OrderableDBInstanceOption
 var err error
 orderablePaginator :=
 rds.NewDescribeOrderableDBInstanceOptionsPaginator(clusters.AuroraClient,
  &rds.DescribeOrderableDBInstanceOptionsInput{
   Engine:        aws.String(engine),
   EngineVersion: aws.String(engineVersion),
  })
 for orderablePaginator.HasMorePages() {
  output, err = orderablePaginator.NextPage(ctx)
  if err != nil {
   log.Printf("Couldn't get orderable DB instances: %v\n", err)
   break
  } else {
   instances = append(instances, output.OrderableDBInstanceOptions...)
  }
 }
 return instances, err
}
```

- Per informazioni dettagliate sull'API, consulta i seguenti argomenti nella documentazione di riferimento dell'API AWS SDK per Go.
  - CreaDBCluster
  - CreaDBClusterParameterGroup
  - Crea DBCluster istantanea
  - CreaDBInstance
  - EliminaDBCluster
  - EliminaDBClusterParameterGroup
  - EliminaDBInstance
  - Descriva DBCluster ParameterGroups
  - DBClusterDescrivi parametri
  - Descrivi le DBCluster istantanee

- Descriva DBClusters
- Descrivi DBEngine versioni
- Descriva DBInstances
- DescribeOrderableDBInstanceOpzioni
- ModificaDBClusterParameterGroup

Java

SDK per Java 2.x

> **Note**
>
> C'è altro da fare GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice AWS.

```
/**
 * Before running this Java (v2) code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 *
 * This example requires an AWS Secrets Manager secret that contains the
 * database credentials. If you do not create a
 * secret, this example will not work. For details, see:
 *
 * https://docs.aws.amazon.com/secretsmanager/latest/userguide/integrating_how-
services-use-secrets_RS.html
 *
 * This Java example performs the following tasks:
 *
 * 1. Gets available engine families for Amazon Aurora MySQL-Compatible Edition
 * by calling the DescribeDbEngineVersions(Engine='aurora-mysql') method.
 * 2. Selects an engine family and creates a custom DB cluster parameter group
 * by invoking the describeDBClusterParameters method.
 * 3. Gets the parameter groups by invoking the describeDBClusterParameterGroups
 * method.
```

```
 * 4. Gets parameters in the group by invoking the describeDBClusterParameters
 * method.
 * 5. Modifies the auto_increment_offset parameter by invoking the
 * modifyDbClusterParameterGroupRequest method.
 * 6. Gets and displays the updated parameters.
 * 7. Gets a list of allowed engine versions by invoking the
 * describeDbEngineVersions method.
 * 8. Creates an Aurora DB cluster database cluster that contains a MySQL
 * database.
 * 9. Waits for DB instance to be ready.
 * 10. Gets a list of instance classes available for the selected engine.
 * 11. Creates a database instance in the cluster.
 * 12. Waits for DB instance to be ready.
 * 13. Creates a snapshot.
 * 14. Waits for DB snapshot to be ready.
 * 15. Deletes the DB cluster.
 * 16. Deletes the DB cluster group.
 */
public class AuroraScenario {
    public static long sleepTime = 20;
    public static final String DASHES = new String(new char[80]).replace("\0",
 "-");

    public static void main(String[] args) throws InterruptedException {
        final String usage = "\n" +
                "Usage:\n" +
                "    <dbClusterGroupName> <dbParameterGroupFamily>
 <dbInstanceClusterIdentifier> <dbInstanceIdentifier> <dbName>
 <dbSnapshotIdentifier><secretName>"
                +
                "Where:\n" +
                "    dbClusterGroupName - The name of the DB cluster parameter
 group. \n" +
                "    dbParameterGroupFamily - The DB cluster parameter group
 family name (for example, aurora-mysql5.7). \n"
                +
                "    dbInstanceClusterIdentifier - The instance cluster
 identifier value.\n" +
                "    dbInstanceIdentifier - The database instance identifier.\n"
 +
                "    dbName - The database name.\n" +
                "    dbSnapshotIdentifier - The snapshot identifier.\n" +
                "    secretName - The name of the AWS Secrets Manager secret that
 contains the database credentials\"\n";
```

```
        ;

        if (args.length != 7) {
            System.out.println(usage);
            System.exit(1);
        }

        String dbClusterGroupName = args[0];
        String dbParameterGroupFamily = args[1];
        String dbInstanceClusterIdentifier = args[2];
        String dbInstanceIdentifier = args[3];
        String dbName = args[4];
        String dbSnapshotIdentifier = args[5];
        String secretName = args[6];

        // Retrieve the database credentials using AWS Secrets Manager.
        Gson gson = new Gson();
        User user = gson.fromJson(String.valueOf(getSecretValues(secretName)),
User.class);
        String username = user.getUsername();
        String userPassword = user.getPassword();

        Region region = Region.US_WEST_2;
        RdsClient rdsClient = RdsClient.builder()
                .region(region)
                .build();

        System.out.println(DASHES);
        System.out.println("Welcome to the Amazon Aurora example scenario.");
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("1. Return a list of the available DB engines");
        describeDBEngines(rdsClient);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("2. Create a custom parameter group");
        createDBClusterParameterGroup(rdsClient, dbClusterGroupName,
dbParameterGroupFamily);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("3. Get the parameter group");
```

```
        describeDbClusterParameterGroups(rdsClient, dbClusterGroupName);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("4. Get the parameters in the group");
        describeDbClusterParameters(rdsClient, dbClusterGroupName, 0);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("5. Modify the auto_increment_offset parameter");
        modifyDBClusterParas(rdsClient, dbClusterGroupName);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("6. Display the updated parameter value");
        describeDbClusterParameters(rdsClient, dbClusterGroupName, -1);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("7. Get a list of allowed engine versions");
        getAllowedEngines(rdsClient, dbParameterGroupFamily);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("8. Create an Aurora DB cluster database");
        String arnClusterVal = createDBCluster(rdsClient, dbClusterGroupName,
dbName, dbInstanceClusterIdentifier,
                username, userPassword);
        System.out.println("The ARN of the cluster is " + arnClusterVal);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("9. Wait for DB instance to be ready");
        waitForInstanceReady(rdsClient, dbInstanceClusterIdentifier);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("10. Get a list of instance classes available for the
selected engine");
        String instanceClass = getListInstanceClasses(rdsClient);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("11. Create a database instance in the cluster.");
```

```
        String clusterDBARN = createDBInstanceCluster(rdsClient,
dbInstanceIdentifier, dbInstanceClusterIdentifier,
                instanceClass);
        System.out.println("The ARN of the database is " + clusterDBARN);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("12. Wait for DB instance to be ready");
        waitDBInstanceReady(rdsClient, dbInstanceIdentifier);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("13. Create a snapshot");
        createDBClusterSnapshot(rdsClient, dbInstanceClusterIdentifier,
dbSnapshotIdentifier);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("14. Wait for DB snapshot to be ready");
        waitForSnapshotReady(rdsClient, dbSnapshotIdentifier,
dbInstanceClusterIdentifier);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("14. Delete the DB instance");
        deleteDatabaseInstance(rdsClient, dbInstanceIdentifier);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("15. Delete the DB cluster");
        deleteCluster(rdsClient, dbInstanceClusterIdentifier);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("16. Delete the DB cluster group");
        deleteDBClusterGroup(rdsClient, dbClusterGroupName, clusterDBARN);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("The Scenario has successfully completed.");
        System.out.println(DASHES);
        rdsClient.close();
    }
```

```
   private static SecretsManagerClient getSecretClient() {
       Region region = Region.US_WEST_2;
       return SecretsManagerClient.builder()
               .region(region)

.credentialsProvider(EnvironmentVariableCredentialsProvider.create())
               .build();
   }

   private static String getSecretValues(String secretName) {
       SecretsManagerClient secretClient = getSecretClient();
       GetSecretValueRequest valueRequest = GetSecretValueRequest.builder()
               .secretId(secretName)
               .build();

       GetSecretValueResponse valueResponse =
secretClient.getSecretValue(valueRequest);
       return valueResponse.secretString();
   }

   public static void deleteDBClusterGroup(RdsClient rdsClient, String
dbClusterGroupName, String clusterDBARN)
           throws InterruptedException {
       try {
           boolean isDataDel = false;
           boolean didFind;
           String instanceARN;

           // Make sure that the database has been deleted.
           while (!isDataDel) {
               DescribeDbInstancesResponse response =
rdsClient.describeDBInstances();
               List<DBInstance> instanceList = response.dbInstances();
               int listSize = instanceList.size();
               didFind = false;
               int index = 1;
               for (DBInstance instance : instanceList) {
                   instanceARN = instance.dbInstanceArn();
                   if (instanceARN.compareTo(clusterDBARN) == 0) {
                       System.out.println(clusterDBARN + " still exists");
                       didFind = true;
                   }
                   if ((index == listSize) && (!didFind)) {
```

```
                        // Went through the entire list and did not find the
database ARN.
                        isDataDel = true;
                    }
                    Thread.sleep(sleepTime * 1000);
                    index++;
                }
            }

            DeleteDbClusterParameterGroupRequest clusterParameterGroupRequest =
DeleteDbClusterParameterGroupRequest
                    .builder()
                    .dbClusterParameterGroupName(dbClusterGroupName)
                    .build();


rdsClient.deleteDBClusterParameterGroup(clusterParameterGroupRequest);
            System.out.println(dbClusterGroupName + " was deleted.");

        } catch (RdsException e) {
            System.out.println(e.getLocalizedMessage());
            System.exit(1);
        }
    }

    public static void deleteCluster(RdsClient rdsClient, String
dbInstanceClusterIdentifier) {
        try {
            DeleteDbClusterRequest deleteDbClusterRequest =
DeleteDbClusterRequest.builder()
                    .dbClusterIdentifier(dbInstanceClusterIdentifier)
                    .skipFinalSnapshot(true)
                    .build();

            rdsClient.deleteDBCluster(deleteDbClusterRequest);
            System.out.println(dbInstanceClusterIdentifier + " was deleted!");

        } catch (RdsException e) {
            System.out.println(e.getLocalizedMessage());
            System.exit(1);
        }
    }
```

```
   public static void deleteDatabaseInstance(RdsClient rdsClient, String
dbInstanceIdentifier) {
       try {
           DeleteDbInstanceRequest deleteDbInstanceRequest =
DeleteDbInstanceRequest.builder()
                   .dbInstanceIdentifier(dbInstanceIdentifier)
                   .deleteAutomatedBackups(true)
                   .skipFinalSnapshot(true)
                   .build();

           DeleteDbInstanceResponse response =
rdsClient.deleteDBInstance(deleteDbInstanceRequest);
           System.out.println("The status of the database is " +
response.dbInstance().dbInstanceStatus());

       } catch (RdsException e) {
           System.out.println(e.getLocalizedMessage());
           System.exit(1);
       }
   }

   public static void waitForSnapshotReady(RdsClient rdsClient, String
dbSnapshotIdentifier,
           String dbInstanceClusterIdentifier) {
       try {
           boolean snapshotReady = false;
           String snapshotReadyStr;
           System.out.println("Waiting for the snapshot to become available.");

           DescribeDbClusterSnapshotsRequest snapshotsRequest =
DescribeDbClusterSnapshotsRequest.builder()
                   .dbClusterSnapshotIdentifier(dbSnapshotIdentifier)
                   .dbClusterIdentifier(dbInstanceClusterIdentifier)
                   .build();

           while (!snapshotReady) {
               DescribeDbClusterSnapshotsResponse response =
rdsClient.describeDBClusterSnapshots(snapshotsRequest);
               List<DBClusterSnapshot> snapshotList =
response.dbClusterSnapshots();
               for (DBClusterSnapshot snapshot : snapshotList) {
                   snapshotReadyStr = snapshot.status();
                   if (snapshotReadyStr.contains("available")) {
                       snapshotReady = true;
```

```
                    } else {
                        System.out.println(".");
                        Thread.sleep(sleepTime * 5000);
                    }
                }
            }

            System.out.println("The Snapshot is available!");

        } catch (RdsException | InterruptedException e) {
            System.out.println(e.getLocalizedMessage());
            System.exit(1);
        }
    }

    public static void createDBClusterSnapshot(RdsClient rdsClient, String
dbInstanceClusterIdentifier,
            String dbSnapshotIdentifier) {
        try {
            CreateDbClusterSnapshotRequest snapshotRequest =
CreateDbClusterSnapshotRequest.builder()
                    .dbClusterIdentifier(dbInstanceClusterIdentifier)
                    .dbClusterSnapshotIdentifier(dbSnapshotIdentifier)
                    .build();

            CreateDbClusterSnapshotResponse response =
rdsClient.createDBClusterSnapshot(snapshotRequest);
            System.out.println("The Snapshot ARN is " +
response.dbClusterSnapshot().dbClusterSnapshotArn());

        } catch (RdsException e) {
            System.out.println(e.getLocalizedMessage());
            System.exit(1);
        }
    }

    public static void waitDBInstanceReady(RdsClient rdsClient, String
dbInstanceIdentifier) {
        boolean instanceReady = false;
        String instanceReadyStr;
        System.out.println("Waiting for instance to become available.");
        try {
            DescribeDbInstancesRequest instanceRequest =
DescribeDbInstancesRequest.builder()
```

```
                    .dbInstanceIdentifier(dbInstanceIdentifier)
                    .build();

            String endpoint = "";
            while (!instanceReady) {
                DescribeDbInstancesResponse response =
rdsClient.describeDBInstances(instanceRequest);
                List<DBInstance> instanceList = response.dbInstances();
                for (DBInstance instance : instanceList) {
                    instanceReadyStr = instance.dbInstanceStatus();
                    if (instanceReadyStr.contains("available")) {
                        endpoint = instance.endpoint().address();
                        instanceReady = true;
                    } else {
                        System.out.print(".");
                        Thread.sleep(sleepTime * 1000);
                    }
                }
            }
            System.out.println("Database instance is available! The connection
endpoint is " + endpoint);

        } catch (RdsException | InterruptedException e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
    }

    public static String createDBInstanceCluster(RdsClient rdsClient,
            String dbInstanceIdentifier,
            String dbInstanceClusterIdentifier,
            String instanceClass) {
        try {
            CreateDbInstanceRequest instanceRequest =
CreateDbInstanceRequest.builder()
                    .dbInstanceIdentifier(dbInstanceIdentifier)
                    .dbClusterIdentifier(dbInstanceClusterIdentifier)
                    .engine("aurora-mysql")
                    .dbInstanceClass(instanceClass)
                    .build();

            CreateDbInstanceResponse response =
rdsClient.createDBInstance(instanceRequest);
```

```
            System.out.print("The status is " +
response.dbInstance().dbInstanceStatus());
            return response.dbInstance().dbInstanceArn();

        } catch (RdsException e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
        return "";
    }

    public static String getListInstanceClasses(RdsClient rdsClient) {
        try {
            DescribeOrderableDbInstanceOptionsRequest optionsRequest =
DescribeOrderableDbInstanceOptionsRequest
                    .builder()
                    .engine("aurora-mysql")
                    .maxRecords(20)
                    .build();

            DescribeOrderableDbInstanceOptionsResponse response = rdsClient
                    .describeOrderableDBInstanceOptions(optionsRequest);
            List<OrderableDBInstanceOption> instanceOptions =
response.orderableDBInstanceOptions();
            String instanceClass = "";
            for (OrderableDBInstanceOption instanceOption : instanceOptions) {
                instanceClass = instanceOption.dbInstanceClass();
                System.out.println("The instance class is " +
instanceOption.dbInstanceClass());
                System.out.println("The engine version is " +
instanceOption.engineVersion());
            }
            return instanceClass;

        } catch (RdsException e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
        return "";
    }

    // Waits until the database instance is available.
    public static void waitForInstanceReady(RdsClient rdsClient, String
dbClusterIdentifier) {
```

```
        boolean instanceReady = false;
        String instanceReadyStr;
        System.out.println("Waiting for instance to become available.");
        try {
            DescribeDbClustersRequest instanceRequest =
DescribeDbClustersRequest.builder()
                    .dbClusterIdentifier(dbClusterIdentifier)
                    .build();

            while (!instanceReady) {
                DescribeDbClustersResponse response =
rdsClient.describeDBClusters(instanceRequest);
                List<DBCluster> clusterList = response.dbClusters();
                for (DBCluster cluster : clusterList) {
                    instanceReadyStr = cluster.status();
                    if (instanceReadyStr.contains("available")) {
                        instanceReady = true;
                    } else {
                        System.out.print(".");
                        Thread.sleep(sleepTime * 1000);
                    }
                }
            }
            System.out.println("Database cluster is available!");

        } catch (RdsException | InterruptedException e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
    }

    public static String createDBCluster(RdsClient rdsClient, String
dbParameterGroupFamily, String dbName,
            String dbClusterIdentifier, String userName, String password) {
        try {
            CreateDbClusterRequest clusterRequest =
CreateDbClusterRequest.builder()
                    .databaseName(dbName)
                    .dbClusterIdentifier(dbClusterIdentifier)
                    .dbClusterParameterGroupName(dbParameterGroupFamily)
                    .engine("aurora-mysql")
                    .masterUsername(userName)
                    .masterUserPassword(password)
                    .build();
```

```
            CreateDbClusterResponse response =
rdsClient.createDBCluster(clusterRequest);
            return response.dbCluster().dbClusterArn();

        } catch (RdsException e) {
            System.out.println(e.getLocalizedMessage());
            System.exit(1);
        }
        return "";
    }

    // Get a list of allowed engine versions.
    public static void getAllowedEngines(RdsClient rdsClient, String
dbParameterGroupFamily) {
        try {
            DescribeDbEngineVersionsRequest versionsRequest =
DescribeDbEngineVersionsRequest.builder()
                    .dbParameterGroupFamily(dbParameterGroupFamily)
                    .engine("aurora-mysql")
                    .build();

            DescribeDbEngineVersionsResponse response =
rdsClient.describeDBEngineVersions(versionsRequest);
            List<DBEngineVersion> dbEngines = response.dbEngineVersions();
            for (DBEngineVersion dbEngine : dbEngines) {
                System.out.println("The engine version is " +
dbEngine.engineVersion());
                System.out.println("The engine description is " +
dbEngine.dbEngineDescription());
            }

        } catch (RdsException e) {
            System.out.println(e.getLocalizedMessage());
            System.exit(1);
        }
    }

    // Modify the auto_increment_offset parameter.
    public static void modifyDBClusterParas(RdsClient rdsClient, String
dClusterGroupName) {
        try {
            Parameter parameter1 = Parameter.builder()
                    .parameterName("auto_increment_offset")
```

```
                    .applyMethod("immediate")
                    .parameterValue("5")
                    .build();

            List<Parameter> paraList = new ArrayList<>();
            paraList.add(parameter1);
            ModifyDbClusterParameterGroupRequest groupRequest =
ModifyDbClusterParameterGroupRequest.builder()
                    .dbClusterParameterGroupName(dClusterGroupName)
                    .parameters(paraList)
                    .build();

            ModifyDbClusterParameterGroupResponse response =
rdsClient.modifyDBClusterParameterGroup(groupRequest);
            System.out.println(
                    "The parameter group " +
response.dbClusterParameterGroupName() + " was successfully modified");

        } catch (RdsException e) {
            System.out.println(e.getLocalizedMessage());
            System.exit(1);
        }
    }

    public static void describeDbClusterParameters(RdsClient rdsClient, String
dbCLusterGroupName, int flag) {
        try {
            DescribeDbClusterParametersRequest dbParameterGroupsRequest;
            if (flag == 0) {
                dbParameterGroupsRequest =
DescribeDbClusterParametersRequest.builder()
                        .dbClusterParameterGroupName(dbCLusterGroupName)
                        .build();
            } else {
                dbParameterGroupsRequest =
DescribeDbClusterParametersRequest.builder()
                        .dbClusterParameterGroupName(dbCLusterGroupName)
                        .source("user")
                        .build();
            }

            DescribeDbClusterParametersResponse response = rdsClient
                    .describeDBClusterParameters(dbParameterGroupsRequest);
            List<Parameter> dbParameters = response.parameters();
```

```
            String paraName;
            for (Parameter para : dbParameters) {
                // Only print out information about either auto_increment_offset
or
                // auto_increment_increment.
                paraName = para.parameterName();
                if ((paraName.compareTo("auto_increment_offset") == 0)
                        || (paraName.compareTo("auto_increment_increment ") ==
0)) {
                    System.out.println("*** The parameter name is  " + paraName);
                    System.out.println("*** The parameter value is  " +
para.parameterValue());
                    System.out.println("*** The parameter data type is " +
para.dataType());
                    System.out.println("*** The parameter description is " +
para.description());
                    System.out.println("*** The parameter allowed values  is " +
para.allowedValues());
                }
            }

        } catch (RdsException e) {
            System.out.println(e.getLocalizedMessage());
            System.exit(1);
        }
    }

    public static void describeDbClusterParameterGroups(RdsClient rdsClient,
String dbClusterGroupName) {
        try {
            DescribeDbClusterParameterGroupsRequest groupsRequest =
DescribeDbClusterParameterGroupsRequest.builder()
                    .dbClusterParameterGroupName(dbClusterGroupName)
                    .maxRecords(20)
                    .build();

            List<DBClusterParameterGroup> groups =
rdsClient.describeDBClusterParameterGroups(groupsRequest)
                    .dbClusterParameterGroups();
            for (DBClusterParameterGroup group : groups) {
                System.out.println("The group name is " +
group.dbClusterParameterGroupName());
                System.out.println("The group ARN is " +
group.dbClusterParameterGroupArn());
```

```
            }

        } catch (RdsException e) {
            System.out.println(e.getLocalizedMessage());
            System.exit(1);
        }
    }

    public static void createDBClusterParameterGroup(RdsClient rdsClient, String
dbClusterGroupName,
            String dbParameterGroupFamily) {
        try {
            CreateDbClusterParameterGroupRequest groupRequest =
CreateDbClusterParameterGroupRequest.builder()
                    .dbClusterParameterGroupName(dbClusterGroupName)
                    .dbParameterGroupFamily(dbParameterGroupFamily)
                    .description("Created by using the AWS SDK for Java")
                    .build();

            CreateDbClusterParameterGroupResponse response =
rdsClient.createDBClusterParameterGroup(groupRequest);
            System.out.println("The group name is " +
response.dbClusterParameterGroup().dbClusterParameterGroupName());

        } catch (RdsException e) {
            System.out.println(e.getLocalizedMessage());
            System.exit(1);
        }
    }

    public static void describeDBEngines(RdsClient rdsClient) {
        try {
            DescribeDbEngineVersionsRequest engineVersionsRequest =
DescribeDbEngineVersionsRequest.builder()
                    .engine("aurora-mysql")
                    .defaultOnly(true)
                    .maxRecords(20)
                    .build();

            DescribeDbEngineVersionsResponse response =
rdsClient.describeDBEngineVersions(engineVersionsRequest);
            List<DBEngineVersion> engines = response.dbEngineVersions();

            // Get all DBEngineVersion objects.
```

```
            for (DBEngineVersion engineOb : engines) {
                System.out.println("The name of the DB parameter group family for
 the database engine is "
                        + engineOb.dbParameterGroupFamily());
                System.out.println("The name of the database engine " +
 engineOb.engine());
                System.out.println("The version number of the database engine " +
 engineOb.engineVersion());
            }

        } catch (RdsException e) {
            System.out.println(e.getLocalizedMessage());
            System.exit(1);
        }
    }
}
```

- Per informazioni dettagliate sull'API, consulta i seguenti argomenti nella documentazione di riferimento dell'API AWS SDK for Java 2.x.
  - CreaDBCluster
  - CreaDBClusterParameterGroup
  - Crea DBCluster istantanea
  - CreaDBInstance
  - EliminaDBCluster
  - EliminaDBClusterParameterGroup
  - EliminaDBInstance
  - Descriva DBCluster ParameterGroups
  - DBClusterDescrivi parametri
  - Descrivi le DBCluster istantanee
  - Descriva DBClusters
  - Descrivi DBEngine versioni
  - Descriva DBInstances
  - DescribeOrderableDBInstanceOpzioni
  - ModificaDBClusterParameterGroup

Kotlin

SDK per Kotlin

> **Note**
>
> C'è altro da fare GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice AWS.

```
/**
Before running this Kotlin code example, set up your development environment,
 including your credentials.

For more information, see the following documentation topic:

https://docs.aws.amazon.com/sdk-for-kotlin/latest/developer-guide/setup.html

This example requires an AWS Secrets Manager secret that contains the database
 credentials. If you do not create a
secret, this example will not work. For more details, see:

https://docs.aws.amazon.com/secretsmanager/latest/userguide/integrating_how-
services-use-secrets_RS.html

This Kotlin example performs the following tasks:

1. Returns a list of the available DB engines.
2. Creates a custom DB parameter group.
3. Gets the parameter groups.
4. Gets the parameters in the group.
5. Modifies the auto_increment_increment parameter.
6. Displays the updated parameter value.
7. Gets a list of allowed engine versions.
8. Creates an Aurora DB cluster database.
9. Waits for DB instance to be ready.
10. Gets a list of instance classes available for the selected engine.
11. Creates a database instance in the cluster.
12. Waits for the database instance in the cluster to be ready.
13. Creates a snapshot.
14. Waits for DB snapshot to be ready.
```

```
15. Deletes the DB instance.
16. Deletes the DB cluster.
17. Deletes the DB cluster group.
 */

var slTime: Long = 20

suspend fun main(args: Array<String>) {
    val usage = """
        Usage:
            <dbClusterGroupName> <dbParameterGroupFamily>
 <dbInstanceClusterIdentifier> <dbName> <dbSnapshotIdentifier> <secretName>
        Where:
            dbClusterGroupName - The database group name.
            dbParameterGroupFamily - The database parameter group name.
            dbInstanceClusterIdentifier - The database instance identifier.
            dbName -  The database name.
            dbSnapshotIdentifier - The snapshot identifier.
            secretName - The name of the AWS Secrets Manager secret that contains
 the database credentials.
    """

    if (args.size != 7) {
        println(usage)
        exitProcess(1)
    }

    val dbClusterGroupName = args[0]
    val dbParameterGroupFamily = args[1]
    val dbInstanceClusterIdentifier = args[2]
    val dbInstanceIdentifier = args[3]
    val dbName = args[4]
    val dbSnapshotIdentifier = args[5]
    val secretName = args[6]

    val gson = Gson()
    val user = gson.fromJson(getSecretValues(secretName).toString(),
 User::class.java)
    val username = user.username
    val userPassword = user.password

    println("1. Return a list of the available DB engines")
    describeAuroraDBEngines()
```

```
    println("2. Create a custom parameter group")
    createDBClusterParameterGroup(dbClusterGroupName, dbParameterGroupFamily)

    println("3. Get the parameter group")
    describeDbClusterParameterGroups(dbClusterGroupName)

    println("4. Get the parameters in the group")
    describeDbClusterParameters(dbClusterGroupName, 0)

    println("5. Modify the auto_increment_offset parameter")
    modifyDBClusterParas(dbClusterGroupName)

    println("6. Display the updated parameter value")
    describeDbClusterParameters(dbClusterGroupName, -1)

    println("7. Get a list of allowed engine versions")
    getAllowedClusterEngines(dbParameterGroupFamily)

    println("8. Create an Aurora DB cluster database")
    val arnClusterVal = createDBCluster(dbClusterGroupName, dbName,
dbInstanceClusterIdentifier, username, userPassword)
    println("The ARN of the cluster is $arnClusterVal")

    println("9. Wait for DB instance to be ready")
    waitForClusterInstanceReady(dbInstanceClusterIdentifier)

    println("10. Get a list of instance classes available for the selected
engine")
    val instanceClass = getListInstanceClasses()

    println("11. Create a database instance in the cluster.")
    val clusterDBARN = createDBInstanceCluster(dbInstanceIdentifier,
dbInstanceClusterIdentifier, instanceClass)
    println("The ARN of the database is $clusterDBARN")

    println("12. Wait for DB instance to be ready")
    waitDBAuroraInstanceReady(dbInstanceIdentifier)

    println("13. Create a snapshot")
    createDBClusterSnapshot(dbInstanceClusterIdentifier, dbSnapshotIdentifier)

    println("14. Wait for DB snapshot to be ready")
    waitSnapshotReady(dbSnapshotIdentifier, dbInstanceClusterIdentifier)
```

```
    println("15. Delete the DB instance")
    deleteDBInstance(dbInstanceIdentifier)

    println("16. Delete the DB cluster")
    deleteCluster(dbInstanceClusterIdentifier)

    println("17. Delete the DB cluster group")
    if (clusterDBARN != null) {
        deleteDBClusterGroup(dbClusterGroupName, clusterDBARN)
    }
    println("The Scenario has successfully completed.")
}

@Throws(InterruptedException::class)
suspend fun deleteDBClusterGroup(
    dbClusterGroupName: String,
    clusterDBARN: String,
) {
    var isDataDel = false
    var didFind: Boolean
    var instanceARN: String

    RdsClient.fromEnvironment { region = "us-west-2" }.use { rdsClient ->
        // Make sure that the database has been deleted.
        while (!isDataDel) {
            val response = rdsClient.describeDbInstances()
            val instanceList = response.dbInstances
            val listSize = instanceList?.size
            isDataDel = false
            didFind = false
            var index = 1
            if (instanceList != null) {
                for (instance in instanceList) {
                    instanceARN = instance.dbInstanceArn.toString()
                    if (instanceARN.compareTo(clusterDBARN) == 0) {
                        println("$clusterDBARN still exists")
                        didFind = true
                    }
                    if (index == listSize && !didFind) {
                        // Went through the entire list and did not find the
 database ARN.
                        isDataDel = true
                    }
                    delay(slTime * 1000)
```

```
                    index++
                }
            }
        }
        val clusterParameterGroupRequest =
            DeleteDbClusterParameterGroupRequest {
                dbClusterParameterGroupName = dbClusterGroupName
            }

        rdsClient.deleteDbClusterParameterGroup(clusterParameterGroupRequest)
        println("$dbClusterGroupName was deleted.")
    }
}

suspend fun deleteCluster(dbInstanceClusterIdentifier: String) {
    val deleteDbClusterRequest =
        DeleteDbClusterRequest {
            dbClusterIdentifier = dbInstanceClusterIdentifier
            skipFinalSnapshot = true
        }

    RdsClient.fromEnvironment { region = "us-west-2" }.use { rdsClient ->
        rdsClient.deleteDbCluster(deleteDbClusterRequest)
        println("$dbInstanceClusterIdentifier was deleted!")
    }
}

suspend fun deleteDBInstance(dbInstanceIdentifierVal: String) {
    val deleteDbInstanceRequest =
        DeleteDbInstanceRequest {
            dbInstanceIdentifier = dbInstanceIdentifierVal
            deleteAutomatedBackups = true
            skipFinalSnapshot = true
        }

    RdsClient.fromEnvironment { region = "us-west-2" }.use { rdsClient ->
        val response = rdsClient.deleteDbInstance(deleteDbInstanceRequest)
        print("The status of the database is
 ${response.dbInstance?.dbInstanceStatus}")
    }
}

suspend fun waitSnapshotReady(
    dbSnapshotIdentifier: String?,
```

```
    dbInstanceClusterIdentifier: String?,
) {
    var snapshotReady = false
    var snapshotReadyStr: String
    println("Waiting for the snapshot to become available.")

    val snapshotsRequest =
        DescribeDbClusterSnapshotsRequest {
            dbClusterSnapshotIdentifier = dbSnapshotIdentifier
            dbClusterIdentifier = dbInstanceClusterIdentifier
        }

    RdsClient.fromEnvironment { region = "us-west-2" }.use { rdsClient ->
        while (!snapshotReady) {
            val response = rdsClient.describeDbClusterSnapshots(snapshotsRequest)
            val snapshotList = response.dbClusterSnapshots
            if (snapshotList != null) {
                for (snapshot in snapshotList) {
                    snapshotReadyStr = snapshot.status.toString()
                    if (snapshotReadyStr.contains("available")) {
                        snapshotReady = true
                    } else {
                        println(".")
                        delay(slTime * 5000)
                    }
                }
            }
        }
    }
    println("The Snapshot is available!")
}

suspend fun createDBClusterSnapshot(
    dbInstanceClusterIdentifier: String?,
    dbSnapshotIdentifier: String?,
) {
    val snapshotRequest =
        CreateDbClusterSnapshotRequest {
            dbClusterIdentifier = dbInstanceClusterIdentifier
            dbClusterSnapshotIdentifier = dbSnapshotIdentifier
        }

    RdsClient.fromEnvironment { region = "us-west-2" }.use { rdsClient ->
        val response = rdsClient.createDbClusterSnapshot(snapshotRequest)
```

```
        println("The Snapshot ARN is
 ${response.dbClusterSnapshot?.dbClusterSnapshotArn}")
    }
}

suspend fun waitDBAuroraInstanceReady(dbInstanceIdentifierVal: String?) {
    var instanceReady = false
    var instanceReadyStr: String
    println("Waiting for instance to become available.")
    val instanceRequest =
        DescribeDbInstancesRequest {
            dbInstanceIdentifier = dbInstanceIdentifierVal
        }

    var endpoint = ""
    RdsClient.fromEnvironment { region = "us-west-2" }.use { rdsClient ->
        while (!instanceReady) {
            val response = rdsClient.describeDbInstances(instanceRequest)
            response.dbInstances?.forEach { instance ->
                instanceReadyStr = instance.dbInstanceStatus.toString()
                if (instanceReadyStr.contains("available")) {
                    endpoint = instance.endpoint?.address.toString()
                    instanceReady = true
                } else {
                    print(".")
                    delay(sleepTime * 1000)
                }
            }
        }
    }
    println("Database instance is available! The connection endpoint is
 $endpoint")
}

suspend fun createDBInstanceCluster(
    dbInstanceIdentifierVal: String?,
    dbInstanceClusterIdentifierVal: String?,
    instanceClassVal: String?,
): String? {
    val instanceRequest =
        CreateDbInstanceRequest {
            dbInstanceIdentifier = dbInstanceIdentifierVal
            dbClusterIdentifier = dbInstanceClusterIdentifierVal
            engine = "aurora-mysql"
```

```
            dbInstanceClass = instanceClassVal
        }

    RdsClient.fromEnvironment { region = "us-west-2" }.use { rdsClient ->
        val response = rdsClient.createDbInstance(instanceRequest)
        print("The status is ${response.dbInstance?.dbInstanceStatus}")
        return response.dbInstance?.dbInstanceArn
    }
}

suspend fun getListInstanceClasses(): String {
    val optionsRequest =
        DescribeOrderableDbInstanceOptionsRequest {
            engine = "aurora-mysql"
            maxRecords = 20
        }
    var instanceClass = ""
    RdsClient.fromEnvironment { region = "us-west-2" }.use { rdsClient ->
        val response =
 rdsClient.describeOrderableDbInstanceOptions(optionsRequest)
        response.orderableDbInstanceOptions?.forEach { instanceOption ->
            instanceClass = instanceOption.dbInstanceClass.toString()
            println("The instance class is ${instanceOption.dbInstanceClass}")
            println("The engine version is ${instanceOption.engineVersion}")
        }
    }
    return instanceClass
}

// Waits until the database instance is available.
suspend fun waitForClusterInstanceReady(dbClusterIdentifierVal: String?) {
    var instanceReady = false
    var instanceReadyStr: String
    println("Waiting for instance to become available.")

    val instanceRequest =
        DescribeDbClustersRequest {
            dbClusterIdentifier = dbClusterIdentifierVal
        }

    RdsClient.fromEnvironment { region = "us-west-2" }.use { rdsClient ->
        while (!instanceReady) {
            val response = rdsClient.describeDbClusters(instanceRequest)
            response.dbClusters?.forEach { cluster ->
```

```
                instanceReadyStr = cluster.status.toString()
                if (instanceReadyStr.contains("available")) {
                    instanceReady = true
                } else {
                    print(".")
                    delay(sleepTime * 1000)
                }
            }
        }
    }
    println("Database cluster is available!")
}

suspend fun createDBCluster(
    dbParameterGroupFamilyVal: String?,
    dbName: String?,
    dbClusterIdentifierVal: String?,
    userName: String?,
    password: String?,
): String? {
    val clusterRequest =
        CreateDbClusterRequest {
            databaseName = dbName
            dbClusterIdentifier = dbClusterIdentifierVal
            dbClusterParameterGroupName = dbParameterGroupFamilyVal
            engine = "aurora-mysql"
            masterUsername = userName
            masterUserPassword = password
        }

    RdsClient.fromEnvironment { region = "us-west-2" }.use { rdsClient ->
        val response = rdsClient.createDbCluster(clusterRequest)
        return response.dbCluster?.dbClusterArn
    }
}

// Get a list of allowed engine versions.
suspend fun getAllowedClusterEngines(dbParameterGroupFamilyVal: String?) {
    val versionsRequest =
        DescribeDbEngineVersionsRequest {
            dbParameterGroupFamily = dbParameterGroupFamilyVal
            engine = "aurora-mysql"
        }
```

```
    RdsClient.fromEnvironment { region = "us-west-2" }.use { rdsClient ->
        val response = rdsClient.describeDbEngineVersions(versionsRequest)
        response.dbEngineVersions?.forEach { dbEngine ->
            println("The engine version is ${dbEngine.engineVersion}")
            println("The engine description is ${dbEngine.dbEngineDescription}")
        }
    }
}

// Modify the auto_increment_offset parameter.
suspend fun modifyDBClusterParas(dClusterGroupName: String?) {
    val parameter1 =
        Parameter {
            parameterName = "auto_increment_offset"
            applyMethod = ApplyMethod.fromValue("immediate")
            parameterValue = "5"
        }

    val paraList = ArrayList<Parameter>()
    paraList.add(parameter1)
    val groupRequest =
        ModifyDbClusterParameterGroupRequest {
            dbClusterParameterGroupName = dClusterGroupName
            parameters = paraList
        }

    RdsClient.fromEnvironment { region = "us-west-2" }.use { rdsClient ->
        val response = rdsClient.modifyDbClusterParameterGroup(groupRequest)
        println("The parameter group ${response.dbClusterParameterGroupName} was
 successfully modified")
    }
}

suspend fun describeDbClusterParameters(
    dbCLusterGroupName: String?,
    flag: Int,
) {
    val dbParameterGroupsRequest: DescribeDbClusterParametersRequest
    dbParameterGroupsRequest =
        if (flag == 0) {
            DescribeDbClusterParametersRequest {
                dbClusterParameterGroupName = dbCLusterGroupName
            }
        } else {
```

```
            DescribeDbClusterParametersRequest {
                dbClusterParameterGroupName = dbCLusterGroupName
                source = "user"
            }
        }

    RdsClient.fromEnvironment { region = "us-west-2" }.use { rdsClient ->
        val response =
 rdsClient.describeDbClusterParameters(dbParameterGroupsRequest)
        response.parameters?.forEach { para ->
            // Only print out information about either auto_increment_offset or
 auto_increment_increment.
            val paraName = para.parameterName
            if (paraName != null) {
                if (paraName.compareTo("auto_increment_offset") == 0 ||
 paraName.compareTo("auto_increment_increment ") == 0) {
                    println("*** The parameter name is  $paraName")
                    println("*** The parameter value is  ${para.parameterValue}")
                    println("*** The parameter data type is ${para.dataType}")
                    println("*** The parameter description is
 ${para.description}")
                    println("*** The parameter allowed values  is
 ${para.allowedValues}")
                }
            }
        }
    }
}

suspend fun describeDbClusterParameterGroups(dbClusterGroupName: String?) {
    val groupsRequest =
        DescribeDbClusterParameterGroupsRequest {
            dbClusterParameterGroupName = dbClusterGroupName
            maxRecords = 20
        }

    RdsClient.fromEnvironment { region = "us-west-2" }.use { rdsClient ->
        val response = rdsClient.describeDbClusterParameterGroups(groupsRequest)
        response.dbClusterParameterGroups?.forEach { group ->
            println("The group name is ${group.dbClusterParameterGroupName}")
            println("The group ARN is ${group.dbClusterParameterGroupArn}")
        }
    }
}
```

```
suspend fun createDBClusterParameterGroup(
    dbClusterGroupNameVal: String?,
    dbParameterGroupFamilyVal: String?,
) {
    val groupRequest =
        CreateDbClusterParameterGroupRequest {
            dbClusterParameterGroupName = dbClusterGroupNameVal
            dbParameterGroupFamily = dbParameterGroupFamilyVal
            description = "Created by using the AWS SDK for Kotlin"
        }

    RdsClient.fromEnvironment { region = "us-west-2" }.use { rdsClient ->
        val response = rdsClient.createDbClusterParameterGroup(groupRequest)
        println("The group name is
 ${response.dbClusterParameterGroup?.dbClusterParameterGroupName}")
    }
}

suspend fun describeAuroraDBEngines() {
    val engineVersionsRequest =
        DescribeDbEngineVersionsRequest {
            engine = "aurora-mysql"
            defaultOnly = true
            maxRecords = 20
        }

    RdsClient.fromEnvironment { region = "us-west-2" }.use { rdsClient ->
        val response = rdsClient.describeDbEngineVersions(engineVersionsRequest)
        response.dbEngineVersions?.forEach { engineOb ->
            println("The name of the DB parameter group family for the database
 engine is ${engineOb.dbParameterGroupFamily}")
            println("The name of the database engine ${engineOb.engine}")
            println("The version number of the database engine
 ${engineOb.engineVersion}")
        }
    }
}
```

- Per informazioni dettagliate sull'API, consulta i seguenti argomenti nella documentazione di riferimento dell'API AWS SDK per Kotlin.

- CreaDBCluster
- CreaDBClusterParameterGroup
- Crea DBCluster istantanea
- CreaDBInstance
- EliminaDBCluster
- EliminaDBClusterParameterGroup
- EliminaDBInstance
- Descriva DBCluster ParameterGroups
- DBClusterDescrivi parametri
- Descrivi le DBCluster istantanee
- Descriva DBClusters
- Descrivi DBEngine versioni
- Descriva DBInstances
- DescribeOrderableDBInstanceOpzioni
- ModificaDBClusterParameterGroup

Python

SDK per Python (Boto3)

> **Note**
>
> C'è altro da fare GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice AWS.

Esegui uno scenario interattivo al prompt dei comandi.

```
class AuroraClusterScenario:
    """Runs a scenario that shows how to get started using Aurora DB clusters."""

    def __init__(self, aurora_wrapper):
        """
        :param aurora_wrapper: An object that wraps Aurora DB cluster actions.
        """
```

```
        self.aurora_wrapper = aurora_wrapper

    def create_parameter_group(self, db_engine, parameter_group_name):
        """
        Shows how to get available engine versions for a specified database
engine and
        create a DB cluster parameter group that is compatible with a selected
engine family.

        :param db_engine: The database engine to use as a basis.
        :param parameter_group_name: The name given to the newly created
parameter group.
        :return: The newly created parameter group.
        """
        print(
            f"Checking for an existing DB cluster parameter group named
{parameter_group_name}."
        )
        parameter_group =
self.aurora_wrapper.get_parameter_group(parameter_group_name)
        if parameter_group is None:
            print(f"Getting available database engine versions for {db_engine}.")
            engine_versions = self.aurora_wrapper.get_engine_versions(db_engine)
            families = list({ver["DBParameterGroupFamily"] for ver in
engine_versions})
            family_index = q.choose("Which family do you want to use? ",
families)
            print(f"Creating a DB cluster parameter group.")
            self.aurora_wrapper.create_parameter_group(
                parameter_group_name, families[family_index], "Example parameter
group."
            )
            parameter_group = self.aurora_wrapper.get_parameter_group(
                parameter_group_name
            )
        print(f"Parameter group
{parameter_group['DBClusterParameterGroupName']}:")
        pp(parameter_group)
        print("-" * 88)
        return parameter_group

    def set_user_parameters(self, parameter_group_name):
        """
        Shows how to get the parameters contained in a custom parameter group and
```

```
        update some of the parameter values in the group.

        :param parameter_group_name: The name of the parameter group to query and
modify.
        """
        print("Let's set some parameter values in your parameter group.")
        auto_inc_parameters = self.aurora_wrapper.get_parameters(
            parameter_group_name, name_prefix="auto_increment"
        )
        update_params = []
        for auto_inc in auto_inc_parameters:
            if auto_inc["IsModifiable"] and auto_inc["DataType"] == "integer":
                print(f"The {auto_inc['ParameterName']} parameter is described
as:")
                print(f"\t{auto_inc['Description']}")
                param_range = auto_inc["AllowedValues"].split("-")
                auto_inc["ParameterValue"] = str(
                    q.ask(
                        f"Enter a value between {param_range[0]} and
{param_range[1]}: ",
                        q.is_int,
                        q.in_range(int(param_range[0]), int(param_range[1])),
                    )
                )
                update_params.append(auto_inc)
        self.aurora_wrapper.update_parameters(parameter_group_name,
update_params)
        print(
            "You can get a list of parameters you've set by specifying a source
of 'user'."
        )
        user_parameters = self.aurora_wrapper.get_parameters(
            parameter_group_name, source="user"
        )
        pp(user_parameters)
        print("-" * 88)

    def create_cluster(self, cluster_name, db_engine, db_name, parameter_group):
        """
        Shows how to create an Aurora DB cluster that contains a database of a
specified
        type. The database is also configured to use a custom DB cluster
parameter group.
```

```
        :param cluster_name: The name given to the newly created DB cluster.
        :param db_engine: The engine of the created database.
        :param db_name: The name given to the created database.
        :param parameter_group: The parameter group that is associated with the
DB cluster.
        :return: The newly created DB cluster.
        """
        print("Checking for an existing DB cluster.")
        cluster = self.aurora_wrapper.get_db_cluster(cluster_name)
        if cluster is None:
            admin_username = q.ask(
                "Enter an administrator user name for the database: ",
q.non_empty
            )
            admin_password = q.ask(
                "Enter a password for the administrator (at least 8 characters):
",
                q.non_empty,
            )
            engine_versions = self.aurora_wrapper.get_engine_versions(
                db_engine, parameter_group["DBParameterGroupFamily"]
            )
            engine_choices = [
                ver["EngineVersionDescription"] for ver in engine_versions
            ]
            print("The available engines for your parameter group are:")
            engine_index = q.choose("Which engine do you want to use? ",
engine_choices)
            print(
                f"Creating DB cluster {cluster_name} and database {db_name}.\n"
                f"The DB cluster is configured to use\n"
                f"your custom parameter group
{parameter_group['DBClusterParameterGroupName']}\n"
                f"and selected engine {engine_choices[engine_index]}.\n"
                f"This typically takes several minutes."
            )
            cluster = self.aurora_wrapper.create_db_cluster(
                cluster_name,
                parameter_group["DBClusterParameterGroupName"],
                db_name,
                db_engine,
                engine_versions[engine_index]["EngineVersion"],
                admin_username,
                admin_password,
```

```
            )
            while cluster.get("Status") != "available":
                wait(30)
                cluster = self.aurora_wrapper.get_db_cluster(cluster_name)
            print("Cluster created and available.\n")
        print("Cluster data:")
        pp(cluster)
        print("-" * 88)
        return cluster

    def create_instance(self, cluster):
        """
        Shows how to create a DB instance in an existing Aurora DB cluster. A new
DB cluster
        contains no DB instances, so you must add one. The first DB instance that
is added
        to a DB cluster defaults to a read-write DB instance.

        :param cluster: The DB cluster where the DB instance is added.
        :return: The newly created DB instance.
        """
        print("Checking for an existing database instance.")
        cluster_name = cluster["DBClusterIdentifier"]
        db_inst = self.aurora_wrapper.get_db_instance(cluster_name)
        if db_inst is None:
            print("Let's create a database instance in your DB cluster.")
            print("First, choose a DB instance type:")
            inst_opts = self.aurora_wrapper.get_orderable_instances(
                cluster["Engine"], cluster["EngineVersion"]
            )
            inst_choices = list(
                {
                    opt["DBInstanceClass"] + ", storage type: " +
opt["StorageType"]
                    for opt in inst_opts
                }
            )
            inst_index = q.choose(
                "Which DB instance class do you want to use? ", inst_choices
            )
            print(
                f"Creating a database instance. This typically takes several
minutes."
            )
```

```
            db_inst = self.aurora_wrapper.create_instance_in_cluster(
                cluster_name,
                cluster_name,
                cluster["Engine"],
                inst_opts[inst_index]["DBInstanceClass"],
            )
            while db_inst.get("DBInstanceStatus") != "available":
                wait(30)
                db_inst = self.aurora_wrapper.get_db_instance(cluster_name)
        print("Instance data:")
        pp(db_inst)
        print("-" * 88)
        return db_inst

    @staticmethod
    def display_connection(cluster):
        """
        Displays connection information about an Aurora DB cluster and tips on
 how to
        connect to it.

        :param cluster: The DB cluster to display.
        """
        print(
            "You can now connect to your database using your favorite MySql
 client.\n"
            "One way to connect is by using the 'mysql' shell on an Amazon EC2
 instance\n"
            "that is running in the same VPC as your database cluster. Pass the
 endpoint,\n"
            "port, and administrator user name to 'mysql' and enter your password
\n"
            "when prompted:\n"
        )
        print(
            f"\n\tmysql -h {cluster['Endpoint']} -P {cluster['Port']} -u
 {cluster['MasterUsername']} -p\n"
        )
        print(
            "For more information, see the User Guide for Aurora:\n"
            "\thttps://docs.aws.amazon.com/AmazonRDS/latest/AuroraUserGuide/
CHAP_GettingStartedAurora.CreatingConnecting.Aurora.html#CHAP_GettingStartedAurora.Aurora
        )
        print("-" * 88)
```

```
   def create_snapshot(self, cluster_name):
       """
       Shows how to create a DB cluster snapshot and wait until it's available.

       :param cluster_name: The name of a DB cluster to snapshot.
       """
       if q.ask(
           "Do you want to create a snapshot of your DB cluster (y/n)? ",
q.is_yesno
       ):
           snapshot_id = f"{cluster_name}-{uuid.uuid4()}"
           print(
               f"Creating a snapshot named {snapshot_id}. This typically takes a
few minutes."
           )
           snapshot = self.aurora_wrapper.create_cluster_snapshot(
               snapshot_id, cluster_name
           )
           while snapshot.get("Status") != "available":
               wait(30)
               snapshot = self.aurora_wrapper.get_cluster_snapshot(snapshot_id)
           pp(snapshot)
           print("-" * 88)

   def cleanup(self, db_inst, cluster, parameter_group):
       """
       Shows how to clean up a DB instance, DB cluster, and DB cluster parameter
group.
       Before the DB cluster parameter group can be deleted, all associated DB
instances and
       DB clusters must first be deleted.

       :param db_inst: The DB instance to delete.
       :param cluster: The DB cluster to delete.
       :param parameter_group: The DB cluster parameter group to delete.
       """
       cluster_name = cluster["DBClusterIdentifier"]
       parameter_group_name = parameter_group["DBClusterParameterGroupName"]
       if q.ask(
           "\nDo you want to delete the database instance, DB cluster, and
parameter "
           "group (y/n)? ",
           q.is_yesno,
```

```
        ):
            print(f"Deleting database instance
{db_inst['DBInstanceIdentifier']}.")

self.aurora_wrapper.delete_db_instance(db_inst["DBInstanceIdentifier"])
            print(f"Deleting database cluster {cluster_name}.")
            self.aurora_wrapper.delete_db_cluster(cluster_name)
            print(
                "Waiting for the DB instance and DB cluster to delete.\n"
                "This typically takes several minutes."
            )
            while db_inst is not None or cluster is not None:
                wait(30)
                if db_inst is not None:
                    db_inst = self.aurora_wrapper.get_db_instance(
                        db_inst["DBInstanceIdentifier"]
                    )
                if cluster is not None:
                    cluster = self.aurora_wrapper.get_db_cluster(
                        cluster["DBClusterIdentifier"]
                    )
            print(f"Deleting parameter group {parameter_group_name}.")
            self.aurora_wrapper.delete_parameter_group(parameter_group_name)

    def run_scenario(self, db_engine, parameter_group_name, cluster_name,
db_name):
        print("-" * 88)
        print(
            "Welcome to the Amazon Relational Database Service (Amazon RDS) get
started\n"
            "with Aurora DB clusters demo."
        )
        print("-" * 88)

        parameter_group = self.create_parameter_group(db_engine,
parameter_group_name)
        self.set_user_parameters(parameter_group_name)
        cluster = self.create_cluster(cluster_name, db_engine, db_name,
parameter_group)
        wait(5)
        db_inst = self.create_instance(cluster)
        self.display_connection(cluster)
        self.create_snapshot(cluster_name)
        self.cleanup(db_inst, cluster, parameter_group)
```

```
        print("\nThanks for watching!")
        print("-" * 88)


if __name__ == "__main__":
    logging.basicConfig(level=logging.INFO, format="%(levelname)s: %(message)s")
    try:
        scenario = AuroraClusterScenario(AuroraWrapper.from_client())
        scenario.run_scenario(
            "aurora-mysql",
            "doc-example-cluster-parameter-group",
            "doc-example-aurora",
            "docexampledb",
        )
    except Exception:
        logging.exception("Something went wrong with the demo.")
```

Definisci le funzioni richiamate dallo scenario per gestire le operazioni Aurora.

```
class AuroraWrapper:
    """Encapsulates Aurora DB cluster actions."""

    def __init__(self, rds_client):
        """
        :param rds_client: A Boto3 Amazon Relational Database Service (Amazon
 RDS) client.
        """
        self.rds_client = rds_client

    @classmethod
    def from_client(cls):
        """
        Instantiates this class from a Boto3 client.
        """
        rds_client = boto3.client("rds")
        return cls(rds_client)


    def get_parameter_group(self, parameter_group_name):
        """
        Gets a DB cluster parameter group.
```

```
        :param parameter_group_name: The name of the parameter group to retrieve.
        :return: The requested parameter group.
        """
        try:
            response = self.rds_client.describe_db_cluster_parameter_groups(
                DBClusterParameterGroupName=parameter_group_name
            )
            parameter_group = response["DBClusterParameterGroups"][0]
        except ClientError as err:
            if err.response["Error"]["Code"] == "DBParameterGroupNotFound":
                logger.info("Parameter group %s does not exist.",
parameter_group_name)
            else:
                logger.error(
                    "Couldn't get parameter group %s. Here's why: %s: %s",
                    parameter_group_name,
                    err.response["Error"]["Code"],
                    err.response["Error"]["Message"],
                )
                raise
        else:
            return parameter_group


    def create_parameter_group(
        self, parameter_group_name, parameter_group_family, description
    ):
        """
        Creates a DB cluster parameter group that is based on the specified
parameter group
        family.

        :param parameter_group_name: The name of the newly created parameter
group.
        :param parameter_group_family: The family that is used as the basis of
the new
                                       parameter group.
        :param description: A description given to the parameter group.
        :return: Data about the newly created parameter group.
        """
        try:
            response = self.rds_client.create_db_cluster_parameter_group(
                DBClusterParameterGroupName=parameter_group_name,
```

```
                DBParameterGroupFamily=parameter_group_family,
                Description=description,
            )
        except ClientError as err:
            logger.error(
                "Couldn't create parameter group %s. Here's why: %s: %s",
                parameter_group_name,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
        else:
            return response


    def delete_parameter_group(self, parameter_group_name):
        """
        Deletes a DB cluster parameter group.

        :param parameter_group_name: The name of the parameter group to delete.
        :return: Data about the parameter group.
        """
        try:
            response = self.rds_client.delete_db_cluster_parameter_group(
                DBClusterParameterGroupName=parameter_group_name
            )
        except ClientError as err:
            logger.error(
                "Couldn't delete parameter group %s. Here's why: %s: %s",
                parameter_group_name,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
        else:
            return response


    def get_parameters(self, parameter_group_name, name_prefix="", source=None):
        """
        Gets the parameters that are contained in a DB cluster parameter group.

        :param parameter_group_name: The name of the parameter group to query.
```

```
        :param name_prefix: When specified, the retrieved list of parameters is
filtered
                            to contain only parameters that start with this
prefix.
        :param source: When specified, only parameters from this source are
retrieved.
                       For example, a source of 'user' retrieves only parameters
that
                       were set by a user.
        :return: The list of requested parameters.
        """
        try:
            kwargs = {"DBClusterParameterGroupName": parameter_group_name}
            if source is not None:
                kwargs["Source"] = source
            parameters = []
            paginator =
self.rds_client.get_paginator("describe_db_cluster_parameters")
            for page in paginator.paginate(**kwargs):
                parameters += [
                    p
                    for p in page["Parameters"]
                    if p["ParameterName"].startswith(name_prefix)
                ]
        except ClientError as err:
            logger.error(
                "Couldn't get parameters for %s. Here's why: %s: %s",
                parameter_group_name,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
        else:
            return parameters


    def update_parameters(self, parameter_group_name, update_parameters):
        """
        Updates parameters in a custom DB cluster parameter group.

        :param parameter_group_name: The name of the parameter group to update.
        :param update_parameters: The parameters to update in the group.
        :return: Data about the modified parameter group.
        """
```

```
        try:
            response = self.rds_client.modify_db_cluster_parameter_group(
                DBClusterParameterGroupName=parameter_group_name,
                Parameters=update_parameters,
            )
        except ClientError as err:
            logger.error(
                "Couldn't update parameters in %s. Here's why: %s: %s",
                parameter_group_name,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
        else:
            return response


    def get_db_cluster(self, cluster_name):
        """
        Gets data about an Aurora DB cluster.

        :param cluster_name: The name of the DB cluster to retrieve.
        :return: The retrieved DB cluster.
        """
        try:
            response = self.rds_client.describe_db_clusters(
                DBClusterIdentifier=cluster_name
            )
            cluster = response["DBClusters"][0]
        except ClientError as err:
            if err.response["Error"]["Code"] == "DBClusterNotFoundFault":
                logger.info("Cluster %s does not exist.", cluster_name)
            else:
                logger.error(
                    "Couldn't verify the existence of DB cluster %s. Here's why:
%s: %s",
                    cluster_name,
                    err.response["Error"]["Code"],
                    err.response["Error"]["Message"],
                )
                raise
        else:
            return cluster
```

```
   def create_db_cluster(
       self,
       cluster_name,
       parameter_group_name,
       db_name,
       db_engine,
       db_engine_version,
       admin_name,
       admin_password,
   ):
       """
       Creates a DB cluster that is configured to use the specified parameter
group.
       The newly created DB cluster contains a database that uses the specified
engine and
       engine version.

       :param cluster_name: The name of the DB cluster to create.
       :param parameter_group_name: The name of the parameter group to associate
with
                                    the DB cluster.
       :param db_name: The name of the database to create.
       :param db_engine: The database engine of the database that is created,
such as MySql.
       :param db_engine_version: The version of the database engine.
       :param admin_name: The user name of the database administrator.
       :param admin_password: The password of the database administrator.
       :return: The newly created DB cluster.
       """
       try:
           response = self.rds_client.create_db_cluster(
               DatabaseName=db_name,
               DBClusterIdentifier=cluster_name,
               DBClusterParameterGroupName=parameter_group_name,
               Engine=db_engine,
               EngineVersion=db_engine_version,
               MasterUsername=admin_name,
               MasterUserPassword=admin_password,
           )
           cluster = response["DBCluster"]
       except ClientError as err:
           logger.error(
               "Couldn't create database %s. Here's why: %s: %s",
```

```
                db_name,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
        else:
            return cluster


    def delete_db_cluster(self, cluster_name):
        """
        Deletes a DB cluster.

        :param cluster_name: The name of the DB cluster to delete.
        """
        try:
            self.rds_client.delete_db_cluster(
                DBClusterIdentifier=cluster_name, SkipFinalSnapshot=True
            )
            logger.info("Deleted DB cluster %s.", cluster_name)
        except ClientError:
            logger.exception("Couldn't delete DB cluster %s.", cluster_name)
            raise


    def create_cluster_snapshot(self, snapshot_id, cluster_id):
        """
        Creates a snapshot of a DB cluster.

        :param snapshot_id: The ID to give the created snapshot.
        :param cluster_id: The DB cluster to snapshot.
        :return: Data about the newly created snapshot.
        """
        try:
            response = self.rds_client.create_db_cluster_snapshot(
                DBClusterSnapshotIdentifier=snapshot_id,
DBClusterIdentifier=cluster_id
            )
            snapshot = response["DBClusterSnapshot"]
        except ClientError as err:
            logger.error(
                "Couldn't create snapshot of %s. Here's why: %s: %s",
                cluster_id,
                err.response["Error"]["Code"],
```

```
                err.response["Error"]["Message"],
            )
            raise
        else:
            return snapshot


    def get_cluster_snapshot(self, snapshot_id):
        """
        Gets a DB cluster snapshot.

        :param snapshot_id: The ID of the snapshot to retrieve.
        :return: The retrieved snapshot.
        """
        try:
            response = self.rds_client.describe_db_cluster_snapshots(
                DBClusterSnapshotIdentifier=snapshot_id
            )
            snapshot = response["DBClusterSnapshots"][0]
        except ClientError as err:
            logger.error(
                "Couldn't get DB cluster snapshot %s. Here's why: %s: %s",
                snapshot_id,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
        else:
            return snapshot


    def create_instance_in_cluster(
        self, instance_id, cluster_id, db_engine, instance_class
    ):
        """
        Creates a database instance in an existing DB cluster. The first database
that is
        created defaults to a read-write DB instance.

        :param instance_id: The ID to give the newly created DB instance.
        :param cluster_id: The ID of the DB cluster where the DB instance is
created.
        :param db_engine: The database engine of a database to create in the DB
instance.
```

```
                          This must be compatible with the configured parameter
group
                          of the DB cluster.
       :param instance_class: The DB instance class for the newly created DB
instance.
       :return: Data about the newly created DB instance.
       """
       try:
           response = self.rds_client.create_db_instance(
               DBInstanceIdentifier=instance_id,
               DBClusterIdentifier=cluster_id,
               Engine=db_engine,
               DBInstanceClass=instance_class,
           )
           db_inst = response["DBInstance"]
       except ClientError as err:
           logger.error(
               "Couldn't create DB instance %s. Here's why: %s: %s",
               instance_id,
               err.response["Error"]["Code"],
               err.response["Error"]["Message"],
           )
           raise
       else:
           return db_inst


   def get_engine_versions(self, engine, parameter_group_family=None):
       """
       Gets database engine versions that are available for the specified engine
       and parameter group family.

       :param engine: The database engine to look up.
       :param parameter_group_family: When specified, restricts the returned
list of
                                      engine versions to those that are
compatible with
                                      this parameter group family.
       :return: The list of database engine versions.
       """
       try:
           kwargs = {"Engine": engine}
           if parameter_group_family is not None:
               kwargs["DBParameterGroupFamily"] = parameter_group_family
```

```
            response = self.rds_client.describe_db_engine_versions(**kwargs)
            versions = response["DBEngineVersions"]
        except ClientError as err:
            logger.error(
                "Couldn't get engine versions for %s. Here's why: %s: %s",
                engine,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
        else:
            return versions


    def get_orderable_instances(self, db_engine, db_engine_version):
        """
        Gets DB instance options that can be used to create DB instances that are
        compatible with a set of specifications.

        :param db_engine: The database engine that must be supported by the DB
instance.
        :param db_engine_version: The engine version that must be supported by
the DB instance.
        :return: The list of DB instance options that can be used to create a
compatible DB instance.
        """
        try:
            inst_opts = []
            paginator = self.rds_client.get_paginator(
                "describe_orderable_db_instance_options"
            )
            for page in paginator.paginate(
                Engine=db_engine, EngineVersion=db_engine_version
            ):
                inst_opts += page["OrderableDBInstanceOptions"]
        except ClientError as err:
            logger.error(
                "Couldn't get orderable DB instances. Here's why: %s: %s",
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
        else:
            return inst_opts
```

```
    def get_db_instance(self, instance_id):
        """
        Gets data about a DB instance.

        :param instance_id: The ID of the DB instance to retrieve.
        :return: The retrieved DB instance.
        """
        try:
            response = self.rds_client.describe_db_instances(
                DBInstanceIdentifier=instance_id
            )
            db_inst = response["DBInstances"][0]
        except ClientError as err:
            if err.response["Error"]["Code"] == "DBInstanceNotFound":
                logger.info("Instance %s does not exist.", instance_id)
            else:
                logger.error(
                    "Couldn't get DB instance %s. Here's why: %s: %s",
                    instance_id,
                    err.response["Error"]["Code"],
                    err.response["Error"]["Message"],
                )
                raise
        else:
            return db_inst


    def delete_db_instance(self, instance_id):
        """
        Deletes a DB instance.

        :param instance_id: The ID of the DB instance to delete.
        :return: Data about the deleted DB instance.
        """
        try:
            response = self.rds_client.delete_db_instance(
                DBInstanceIdentifier=instance_id,
                SkipFinalSnapshot=True,
                DeleteAutomatedBackups=True,
            )
            db_inst = response["DBInstance"]
        except ClientError as err:
```

```
            logger.error(
                "Couldn't delete DB instance %s. Here's why: %s: %s",
                instance_id,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
        else:
            return db_inst
```

- Per informazioni dettagliate sull'API, consulta i seguenti argomenti nella documentazione di riferimento dell'API AWS SDK per Python (Boto3).
  - CreaDBCluster
  - CreaDBClusterParameterGroup
  - Crea DBCluster istantanea
  - CreaDBInstance
  - EliminaDBCluster
  - EliminaDBClusterParameterGroup
  - EliminaDBInstance
  - Descriva DBCluster ParameterGroups
  - DBClusterDescrivi parametri
  - Descrivi le DBCluster istantanee
  - Descriva DBClusters
  - Descrivi DBEngine versioni
  - Descriva DBInstances
  - DescribeOrderableDBInstanceOpzioni
  - ModificaDBClusterParameterGroup

Rust

SDK per Rust

> **Note**
>
> C'è altro da fare. GitHub Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice AWS.

Una libreria contenente le funzioni specifiche per lo scenario Aurora.

```
use phf::{phf_set, Set};
use secrecy::SecretString;
use std::{collections::HashMap, fmt::Display, time::Duration};

use aws_sdk_rds::{
    error::ProvideErrorMetadata,

 operation::create_db_cluster_parameter_group::CreateDbClusterParameterGroupOutput,
    types::{DbCluster, DbClusterParameterGroup, DbClusterSnapshot, DbInstance,
 Parameter},
};
use sdk_examples_test_utils::waiter::Waiter;
use tracing::{info, trace, warn};

const DB_ENGINE: &str = "aurora-mysql";
const DB_CLUSTER_PARAMETER_GROUP_NAME: &str =
 "RustSDKCodeExamplesDBParameterGroup";
const DB_CLUSTER_PARAMETER_GROUP_DESCRIPTION: &str =
    "Parameter Group created by Rust SDK Code Example";
const DB_CLUSTER_IDENTIFIER: &str = "RustSDKCodeExamplesDBCluster";
const DB_INSTANCE_IDENTIFIER: &str = "RustSDKCodeExamplesDBInstance";

static FILTER_PARAMETER_NAMES: Set<&'static str> = phf_set! {
    "auto_increment_offset",
    "auto_increment_increment",
};

#[derive(Debug, PartialEq, Eq)]
struct MetadataError {
    message: Option<String>,
```

```
    code: Option<String>,
}

impl MetadataError {
    fn from(err: &dyn ProvideErrorMetadata) -> Self {
        MetadataError {
            message: err.message().map(String::from),
            code: err.code().map(String::from),
        }
    }
}

impl Display for MetadataError {
    fn fmt(&self, f: &mut std::fmt::Formatter<'_>) -> std::fmt::Result {
        let display = match (&self.message, &self.code) {
            (None, None) => "Unknown".to_string(),
            (None, Some(code)) => format!("({code})"),
            (Some(message), None) => message.to_string(),
            (Some(message), Some(code)) => format!("{message} ({code})"),
        };
        write!(f, "{display}")
    }
}

#[derive(Debug, PartialEq, Eq)]
pub struct ScenarioError {
    message: String,
    context: Option<MetadataError>,
}

impl ScenarioError {
    pub fn with(message: impl Into<String>) -> Self {
        ScenarioError {
            message: message.into(),
            context: None,
        }
    }

    pub fn new(message: impl Into<String>, err: &dyn ProvideErrorMetadata) ->
 Self {
        ScenarioError {
            message: message.into(),
            context: Some(MetadataError::from(err)),
        }
```

```
    }
}

impl std::error::Error for ScenarioError {}
impl Display for ScenarioError {
    fn fmt(&self, f: &mut std::fmt::Formatter<'_>) -> std::fmt::Result {
        match &self.context {
            Some(c) => write!(f, "{}: {}", self.message, c),
            None => write!(f, "{}", self.message),
        }
    }
}

// Parse the ParameterName, Description, and AllowedValues values and display
 them.
#[derive(Debug)]
pub struct AuroraScenarioParameter {
    name: String,
    allowed_values: String,
    current_value: String,
}

impl Display for AuroraScenarioParameter {
    fn fmt(&self, f: &mut std::fmt::Formatter<'_>) -> std::fmt::Result {
        write!(
            f,
            "{}: {} (allowed: {})",
            self.name, self.current_value, self.allowed_values
        )
    }
}

impl From<aws_sdk_rds::types::Parameter> for AuroraScenarioParameter {
    fn from(value: aws_sdk_rds::types::Parameter) -> Self {
        AuroraScenarioParameter {
            name: value.parameter_name.unwrap_or_default(),
            allowed_values: value.allowed_values.unwrap_or_default(),
            current_value: value.parameter_value.unwrap_or_default(),
        }
    }
}

pub struct AuroraScenario {
    rds: crate::rds::Rds,
```

```
    engine_family: Option<String>,
    engine_version: Option<String>,
    instance_class: Option<String>,
    db_cluster_parameter_group: Option<DbClusterParameterGroup>,
    db_cluster_identifier: Option<String>,
    db_instance_identifier: Option<String>,
    username: Option<String>,
    password: Option<SecretString>,
}

impl AuroraScenario {
    pub fn new(client: crate::rds::Rds) -> Self {
        AuroraScenario {
            rds: client,
            engine_family: None,
            engine_version: None,
            instance_class: None,
            db_cluster_parameter_group: None,
            db_cluster_identifier: None,
            db_instance_identifier: None,
            username: None,
            password: None,
        }
    }

    // Get available engine families for Aurora MySql.
 rds.DescribeDbEngineVersions(Engine='aurora-mysql') and build a set of the
 'DBParameterGroupFamily' field values. I get {aurora-mysql8.0, aurora-mysql5.7}.
    pub async fn get_engines(&self) -> Result<HashMap<String, Vec<String>>,
 ScenarioError> {
        let describe_db_engine_versions =
 self.rds.describe_db_engine_versions(DB_ENGINE).await;
        trace!(versions=?describe_db_engine_versions, "full list of versions");

        if let Err(err) = describe_db_engine_versions {
            return Err(ScenarioError::new(
                "Failed to retrieve DB Engine Versions",
                &err,
            ));
        };

        let version_count = describe_db_engine_versions
            .as_ref()
            .map(|o| o.db_engine_versions().len())
```

```
            .unwrap_or_default();
        info!(version_count, "got list of versions");

        // Create a map of engine families to their available versions.
        let mut versions = HashMap::<String, Vec<String>>::new();
        describe_db_engine_versions
            .unwrap()
            .db_engine_versions()
            .iter()
            .filter_map(
                |v| match (&v.db_parameter_group_family, &v.engine_version) {
                    (Some(family), Some(version)) => Some((family.clone(),
version.clone())),
                    _ => None,
                },
            )
            .for_each(|(family, version)|
versions.entry(family).or_default().push(version));

        Ok(versions)
    }

    pub async fn get_instance_classes(&self) -> Result<Vec<String>,
ScenarioError> {
        let describe_orderable_db_instance_options_items = self
            .rds
            .describe_orderable_db_instance_options(
                DB_ENGINE,
                self.engine_version
                    .as_ref()
                    .expect("engine version for db instance options")
                    .as_str(),
            )
            .await;

        describe_orderable_db_instance_options_items
            .map(|options| {
                options
                    .iter()
                    .filter(|o| o.storage_type() == Some("aurora"))
                    .map(|o|
o.db_instance_class().unwrap_or_default().to_string())
                    .collect::<Vec<String>>()
            })
```

```
            .map_err(|err| ScenarioError::new("Could not get available instance
classes", &err))
    }

    // Select an engine family and create a custom DB cluster parameter group.
rds.CreateDbClusterParameterGroup(DBParameterGroupFamily='aurora-mysql8.0')
    pub async fn set_engine(&mut self, engine: &str, version: &str) -> Result<(),
ScenarioError> {
        self.engine_family = Some(engine.to_string());
        self.engine_version = Some(version.to_string());
        let create_db_cluster_parameter_group = self
            .rds
            .create_db_cluster_parameter_group(
                DB_CLUSTER_PARAMETER_GROUP_NAME,
                DB_CLUSTER_PARAMETER_GROUP_DESCRIPTION,
                engine,
            )
            .await;

        match create_db_cluster_parameter_group {
            Ok(CreateDbClusterParameterGroupOutput {
                db_cluster_parameter_group: None,
                ..
            }) => {
                return Err(ScenarioError::with(
                    "CreateDBClusterParameterGroup had empty response",
                ));
            }
            Err(error) => {
                if error.code() == Some("DBParameterGroupAlreadyExists") {
                    info!("Cluster Parameter Group already exists, nothing to
do");
                } else {
                    return Err(ScenarioError::new(
                        "Could not create Cluster Parameter Group",
                        &error,
                    ));
                }
            }
            _ => {
                info!("Created Cluster Parameter Group");
            }
        }
```

```
        Ok(())
    }

    pub fn set_instance_class(&mut self, instance_class: Option<String>) {
        self.instance_class = instance_class;
    }

    pub fn set_login(&mut self, username: Option<String>, password:
Option<SecretString>) {
        self.username = username;
        self.password = password;
    }

    pub async fn connection_string(&self) -> Result<String, ScenarioError> {
        let cluster = self.get_cluster().await?;
        let endpoint = cluster.endpoint().unwrap_or_default();
        let port = cluster.port().unwrap_or_default();
        let username = cluster.master_username().unwrap_or_default();
        Ok(format!("mysql -h {endpoint} -P {port} -u {username} -p"))
    }

    pub async fn get_cluster(&self) -> Result<DbCluster, ScenarioError> {
        let describe_db_clusters_output = self
            .rds
            .describe_db_clusters(
                self.db_cluster_identifier
                    .as_ref()
                    .expect("cluster identifier")
                    .as_str(),
            )
            .await;
        if let Err(err) = describe_db_clusters_output {
            return Err(ScenarioError::new("Failed to get cluster", &err));
        }

        let db_cluster = describe_db_clusters_output
            .unwrap()
            .db_clusters
            .and_then(|output| output.first().cloned());

        db_cluster.ok_or_else(|| ScenarioError::with("Did not find the cluster"))
    }

    // Get the parameter group. rds.DescribeDbClusterParameterGroups
```

```
   // Get parameters in the group. This is a long list so you will have to
paginate. Find the auto_increment_offset and auto_increment_increment parameters
(by ParameterName). rds.DescribeDbClusterParameters
   // Parse the ParameterName, Description, and AllowedValues values and display
them.
   pub async fn cluster_parameters(&self) ->
Result<Vec<AuroraScenarioParameter>, ScenarioError> {
       let parameters_output = self
           .rds
           .describe_db_cluster_parameters(DB_CLUSTER_PARAMETER_GROUP_NAME)
           .await;

       if let Err(err) = parameters_output {
           return Err(ScenarioError::new(
               format!("Failed to retrieve parameters for
{DB_CLUSTER_PARAMETER_GROUP_NAME}"),
               &err,
           ));
       }

       let parameters = parameters_output
           .unwrap()
           .into_iter()
           .flat_map(|p| p.parameters.unwrap_or_default().into_iter())
           .filter(|p|
FILTER_PARAMETER_NAMES.contains(p.parameter_name().unwrap_or_default()))
           .map(AuroraScenarioParameter::from)
           .collect::<Vec<_>>();

       Ok(parameters)
   }

   // Modify both the auto_increment_offset and auto_increment_increment
parameters in one call in the custom parameter group. Set their ParameterValue
fields to a new allowable value. rds.ModifyDbClusterParameterGroup.
   pub async fn update_auto_increment(
       &self,
       offset: u8,
       increment: u8,
   ) -> Result<(), ScenarioError> {
       let modify_db_cluster_parameter_group = self
           .rds
           .modify_db_cluster_parameter_group(
               DB_CLUSTER_PARAMETER_GROUP_NAME,
```

```
                vec![
                    Parameter::builder()
                        .parameter_name("auto_increment_offset")
                        .parameter_value(format!("{offset}"))
                        .apply_method(aws_sdk_rds::types::ApplyMethod::Immediate)
                        .build(),
                    Parameter::builder()
                        .parameter_name("auto_increment_increment")
                        .parameter_value(format!("{increment}"))
                        .apply_method(aws_sdk_rds::types::ApplyMethod::Immediate)
                        .build(),
                ],
            )
            .await;

        if let Err(error) = modify_db_cluster_parameter_group {
            return Err(ScenarioError::new(
                "Failed to modify cluster parameter group",
                &error,
            ));
        }

        Ok(())
    }

    // Get a list of allowed engine versions.
rds.DescribeDbEngineVersions(Engine='aurora-mysql', DBParameterGroupFamily=<the
family used to create your parameter group in step 2>)
    // Create an Aurora DB cluster database cluster that contains a MySql
database and uses the parameter group you created.
    // Wait for DB cluster to be ready. Call rds.DescribeDBClusters and check for
Status == 'available'.
    // Get a list of instance classes available for the selected engine
and engine version. rds.DescribeOrderableDbInstanceOptions(Engine='mysql',
EngineVersion=).

    // Create a database instance in the cluster.
    // Wait for DB instance to be ready. Call rds.DescribeDbInstances and check
for DBInstanceStatus == 'available'.
    pub async fn start_cluster_and_instance(&mut self) -> Result<(),
ScenarioError> {
        if self.password.is_none() {
            return Err(ScenarioError::with(
                "Must set Secret Password before starting a cluster",
```

```
            ));
        }
        let create_db_cluster = self
            .rds
            .create_db_cluster(
                DB_CLUSTER_IDENTIFIER,
                DB_CLUSTER_PARAMETER_GROUP_NAME,
                DB_ENGINE,
                self.engine_version.as_deref().expect("engine version"),
                self.username.as_deref().expect("username"),
                self.password
                    .replace(SecretString::new("".to_string()))
                    .expect("password"),
            )
            .await;
        if let Err(err) = create_db_cluster {
            return Err(ScenarioError::new(
                "Failed to create DB Cluster with cluster group",
                &err,
            ));
        }

        self.db_cluster_identifier = create_db_cluster
            .unwrap()
            .db_cluster
            .and_then(|c| c.db_cluster_identifier);

        if self.db_cluster_identifier.is_none() {
            return Err(ScenarioError::with("Created DB Cluster missing
Identifier"));
        }

        info!(
            "Started a db cluster: {}",
            self.db_cluster_identifier
                .as_deref()
                .unwrap_or("Missing ARN")
        );

        let create_db_instance = self
            .rds
            .create_db_instance(
                self.db_cluster_identifier.as_deref().expect("cluster name"),
                DB_INSTANCE_IDENTIFIER,
```

```
                self.instance_class.as_deref().expect("instance class"),
                DB_ENGINE,
            )
            .await;
        if let Err(err) = create_db_instance {
            return Err(ScenarioError::new(
                "Failed to create Instance in DB Cluster",
                &err,
            ));
        }

        self.db_instance_identifier = create_db_instance
            .unwrap()
            .db_instance
            .and_then(|i| i.db_instance_identifier);

        // Cluster creation can take up to 20 minutes to become available
        let cluster_max_wait = Duration::from_secs(20 * 60);
        let waiter = Waiter::builder().max(cluster_max_wait).build();
        while waiter.sleep().await.is_ok() {
            let cluster = self
                .rds
                .describe_db_clusters(
                    self.db_cluster_identifier
                        .as_deref()
                        .expect("cluster identifier"),
                )
                .await;

            if let Err(err) = cluster {
                warn!(?err, "Failed to describe cluster while waiting for
ready");
                continue;
            }

            let instance = self
                .rds
                .describe_db_instance(
                    self.db_instance_identifier
                        .as_deref()
                        .expect("instance identifier"),
                )
                .await;
            if let Err(err) = instance {
```

```
                return Err(ScenarioError::new(
                    "Failed to find instance for cluster",
                    &err,
                ));
            }

            let instances_available = instance
                .unwrap()
                .db_instances()
                .iter()
                .all(|instance| instance.db_instance_status() ==
Some("Available"));

            let endpoints = self
                .rds
                .describe_db_cluster_endpoints(
                    self.db_cluster_identifier
                        .as_deref()
                        .expect("cluster identifier"),
                )
                .await;

            if let Err(err) = endpoints {
                return Err(ScenarioError::new(
                    "Failed to find endpoint for cluster",
                    &err,
                ));
            }

            let endpoints_available = endpoints
                .unwrap()
                .db_cluster_endpoints()
                .iter()
                .all(|endpoint| endpoint.status() == Some("available"));

            if instances_available && endpoints_available {
                return Ok(());
            }
        }

        Err(ScenarioError::with("timed out waiting for cluster"))
    }

    // Create a snapshot of the DB cluster. rds.CreateDbClusterSnapshot.
```

```
   // Wait for the snapshot to create. rds.DescribeDbClusterSnapshots until
Status == 'available'.
   pub async fn snapshot(&self, name: &str) -> Result<DbClusterSnapshot,
ScenarioError> {
       let id = self.db_cluster_identifier.as_deref().unwrap_or_default();
       let snapshot = self
           .rds
           .snapshot_cluster(id, format!("{id}_{name}").as_str())
           .await;
       match snapshot {
           Ok(output) => match output.db_cluster_snapshot {
               Some(snapshot) => Ok(snapshot),
               None => Err(ScenarioError::with("Missing Snapshot")),
           },
           Err(err) => Err(ScenarioError::new("Failed to create snapshot",
&err)),
       }
   }

   pub async fn clean_up(self) -> Result<(), Vec<ScenarioError>> {
       let mut clean_up_errors: Vec<ScenarioError> = vec![];

       // Delete the instance. rds.DeleteDbInstance.
       let delete_db_instance = self
           .rds
           .delete_db_instance(
               self.db_instance_identifier
                   .as_deref()
                   .expect("instance identifier"),
           )
           .await;
       if let Err(err) = delete_db_instance {
           let identifier = self
               .db_instance_identifier
               .as_deref()
               .unwrap_or("Missing Instance Identifier");
           let message = format!("failed to delete db instance {identifier}");
           clean_up_errors.push(ScenarioError::new(message, &err));
       } else {
           // Wait for the instance to delete
           let waiter = Waiter::default();
           while waiter.sleep().await.is_ok() {
               let describe_db_instances =
self.rds.describe_db_instances().await;
```

```
                if let Err(err) = describe_db_instances {
                    clean_up_errors.push(ScenarioError::new(
                        "Failed to check instance state during deletion",
                        &err,
                    ));
                    break;
                }
                let db_instances = describe_db_instances
                    .unwrap()
                    .db_instances()
                    .iter()
                    .filter(|instance| instance.db_cluster_identifier ==
self.db_cluster_identifier)
                    .cloned()
                    .collect::<Vec<DbInstance>>();

                if db_instances.is_empty() {
                    trace!("Delete Instance waited and no instances were found");
                    break;
                }
                match db_instances.first().unwrap().db_instance_status() {
                    Some("Deleting") => continue,
                    Some(status) => {
                        info!("Attempting to delete but instances is in
{status}");
                        continue;
                    }
                    None => {
                        warn!("No status for DB instance");
                        break;
                    }
                }
            }
        }

        // Delete the DB cluster. rds.DeleteDbCluster.
        let delete_db_cluster = self
            .rds
            .delete_db_cluster(
                self.db_cluster_identifier
                    .as_deref()
                    .expect("cluster identifier"),
            )
            .await;
```

```
        if let Err(err) = delete_db_cluster {
            let identifier = self
                .db_cluster_identifier
                .as_deref()
                .unwrap_or("Missing DB Cluster Identifier");
            let message = format!("failed to delete db cluster {identifier}");
            clean_up_errors.push(ScenarioError::new(message, &err));
        } else {
            // Wait for the instance and cluster to fully delete.
rds.DescribeDbInstances and rds.DescribeDbClusters until both are not found.
            let waiter = Waiter::default();
            while waiter.sleep().await.is_ok() {
                let describe_db_clusters = self
                    .rds
                    .describe_db_clusters(
                        self.db_cluster_identifier
                            .as_deref()
                            .expect("cluster identifier"),
                    )
                    .await;
                if let Err(err) = describe_db_clusters {
                    clean_up_errors.push(ScenarioError::new(
                        "Failed to check cluster state during deletion",
                        &err,
                    ));
                    break;
                }
                let describe_db_clusters = describe_db_clusters.unwrap();
                let db_clusters = describe_db_clusters.db_clusters();
                if db_clusters.is_empty() {
                    trace!("Delete cluster waited and no clusters were found");
                    break;
                }
                match db_clusters.first().unwrap().status() {
                    Some("Deleting") => continue,
                    Some(status) => {
                        info!("Attempting to delete but clusters is in
{status}");
                        continue;
                    }
                    None => {
                        warn!("No status for DB cluster");
                        break;
```

```
                    }
                }
            }
        }

        // Delete the DB cluster parameter group.
 rds.DeleteDbClusterParameterGroup.
        let delete_db_cluster_parameter_group = self
            .rds
            .delete_db_cluster_parameter_group(
                self.db_cluster_parameter_group
                    .map(|g| {
                        g.db_cluster_parameter_group_name
                            .unwrap_or_else(||
 DB_CLUSTER_PARAMETER_GROUP_NAME.to_string())
                    })
                    .as_deref()
                    .expect("cluster parameter group name"),
            )
            .await;
        if let Err(error) = delete_db_cluster_parameter_group {
            clean_up_errors.push(ScenarioError::new(
                "Failed to delete the db cluster parameter group",
                &error,
            ))
        }

        if clean_up_errors.is_empty() {
            Ok(())
        } else {
            Err(clean_up_errors)
        }
    }
}

#[cfg(test)]
pub mod tests;
```

Test per la libreria attraverso automock per il wrapper del client RDS.

```
use crate::rds::MockRdsImpl;
```

```
use super::*;

use std::io::{Error, ErrorKind};

use assert_matches::assert_matches;
use aws_sdk_rds::{
    error::SdkError,
    operation::{
        create_db_cluster::{CreateDBClusterError, CreateDbClusterOutput},
        create_db_cluster_parameter_group::CreateDBClusterParameterGroupError,
        create_db_cluster_snapshot::{CreateDBClusterSnapshotError,
 CreateDbClusterSnapshotOutput},
        create_db_instance::{CreateDBInstanceError, CreateDbInstanceOutput},
        delete_db_cluster::DeleteDbClusterOutput,
        delete_db_cluster_parameter_group::DeleteDbClusterParameterGroupOutput,
        delete_db_instance::DeleteDbInstanceOutput,
        describe_db_cluster_endpoints::DescribeDbClusterEndpointsOutput,
        describe_db_cluster_parameters::{
            DescribeDBClusterParametersError, DescribeDbClusterParametersOutput,
        },
        describe_db_clusters::{DescribeDBClustersError,
 DescribeDbClustersOutput},
        describe_db_engine_versions::{
            DescribeDBEngineVersionsError, DescribeDbEngineVersionsOutput,
        },
        describe_db_instances::{DescribeDBInstancesError,
 DescribeDbInstancesOutput},

 describe_orderable_db_instance_options::DescribeOrderableDBInstanceOptionsError,
        modify_db_cluster_parameter_group::{
            ModifyDBClusterParameterGroupError,
 ModifyDbClusterParameterGroupOutput,
        },
    },
    types::{
        error::DbParameterGroupAlreadyExistsFault, DbClusterEndpoint,
 DbEngineVersion,
        OrderableDbInstanceOption,
    },
};
use aws_smithy_runtime_api::http::{Response, StatusCode};
use aws_smithy_types::body::SdkBody;
use mockall::predicate::eq;
```

```
use secrecy::ExposeSecret;

#[tokio::test]
async fn test_scenario_set_engine() {
    let mut mock_rds = MockRdsImpl::default();

    mock_rds
        .expect_create_db_cluster_parameter_group()
        .with(
            eq("RustSDKCodeExamplesDBParameterGroup"),
            eq("Parameter Group created by Rust SDK Code Example"),
            eq("aurora-mysql"),
        )
        .return_once(|_, _, _| {
            Ok(CreateDbClusterParameterGroupOutput::builder()

 .db_cluster_parameter_group(DbClusterParameterGroup::builder().build())
                .build())
        });

    let mut scenario = AuroraScenario::new(mock_rds);

    let set_engine = scenario.set_engine("aurora-mysql", "aurora-
mysql8.0").await;

    assert_eq!(set_engine, Ok(()));
    assert_eq!(Some("aurora-mysql"), scenario.engine_family.as_deref());
    assert_eq!(Some("aurora-mysql8.0"), scenario.engine_version.as_deref());
}

#[tokio::test]
async fn test_scenario_set_engine_not_create() {
    let mut mock_rds = MockRdsImpl::default();

    mock_rds
        .expect_create_db_cluster_parameter_group()
        .with(
            eq("RustSDKCodeExamplesDBParameterGroup"),
            eq("Parameter Group created by Rust SDK Code Example"),
            eq("aurora-mysql"),
        )
        .return_once(|_, _, _|
 Ok(CreateDbClusterParameterGroupOutput::builder().build()));
```

```
    let mut scenario = AuroraScenario::new(mock_rds);

    let set_engine = scenario.set_engine("aurora-mysql", "aurora-
mysql8.0").await;

    assert!(set_engine.is_err());
}

#[tokio::test]
async fn test_scenario_set_engine_param_group_exists() {
    let mut mock_rds = MockRdsImpl::default();

    mock_rds
        .expect_create_db_cluster_parameter_group()
        .withf(|_, _, _| true)
        .return_once(|_, _, _| {
            Err(SdkError::service_error(

 CreateDBClusterParameterGroupError::DbParameterGroupAlreadyExistsFault(
                    DbParameterGroupAlreadyExistsFault::builder().build(),
                ),
                Response::new(StatusCode::try_from(400).unwrap(),
 SdkBody::empty()),
            ))
        });

    let mut scenario = AuroraScenario::new(mock_rds);

    let set_engine = scenario.set_engine("aurora-mysql", "aurora-
mysql8.0").await;

    assert!(set_engine.is_err());
}

#[tokio::test]
async fn test_scenario_get_engines() {
    let mut mock_rds = MockRdsImpl::default();

    mock_rds
        .expect_describe_db_engine_versions()
        .with(eq("aurora-mysql"))
        .return_once(|_| {
            Ok(DescribeDbEngineVersionsOutput::builder()
                .db_engine_versions(
```

```
                    DbEngineVersion::builder()
                        .db_parameter_group_family("f1")
                        .engine_version("f1a")
                        .build(),
                )
                .db_engine_versions(
                    DbEngineVersion::builder()
                        .db_parameter_group_family("f1")
                        .engine_version("f1b")
                        .build(),
                )
                .db_engine_versions(
                    DbEngineVersion::builder()
                        .db_parameter_group_family("f2")
                        .engine_version("f2a")
                        .build(),
                )
                .db_engine_versions(DbEngineVersion::builder().build())
                .build())
        });

    let scenario = AuroraScenario::new(mock_rds);

    let versions_map = scenario.get_engines().await;

    assert_eq!(
        versions_map,
        Ok(HashMap::from([
            ("f1".into(), vec!["f1a".into(), "f1b".into()]),
            ("f2".into(), vec!["f2a".into()])
        ]))
    );
}

#[tokio::test]
async fn test_scenario_get_engines_failed() {
    let mut mock_rds = MockRdsImpl::default();

    mock_rds
        .expect_describe_db_engine_versions()
        .with(eq("aurora-mysql"))
        .return_once(|_| {
            Err(SdkError::service_error(
                DescribeDBEngineVersionsError::unhandled(Box::new(Error::new(
```

```
                    ErrorKind::Other,
                    "describe_db_engine_versions error",
                ))),
                Response::new(StatusCode::try_from(400).unwrap(),
 SdkBody::empty()),
            ))
        });

    let scenario = AuroraScenario::new(mock_rds);

    let versions_map = scenario.get_engines().await;
    assert_matches!(
        versions_map,
        Err(ScenarioError { message, context: _ }) if message == "Failed to
 retrieve DB Engine Versions"
    );
}

#[tokio::test]
async fn test_scenario_get_instance_classes() {
    let mut mock_rds = MockRdsImpl::default();

    mock_rds
        .expect_create_db_cluster_parameter_group()
        .return_once(|_, _, _| {
            Ok(CreateDbClusterParameterGroupOutput::builder()

 .db_cluster_parameter_group(DbClusterParameterGroup::builder().build())
                .build())
        });

    mock_rds
        .expect_describe_orderable_db_instance_options()
        .with(eq("aurora-mysql"), eq("aurora-mysql8.0"))
        .return_once(|_, _| {
            Ok(vec![
                OrderableDbInstanceOption::builder()
                    .db_instance_class("t1")
                    .storage_type("aurora")
                    .build(),
                OrderableDbInstanceOption::builder()
                    .db_instance_class("t1")
                    .storage_type("aurora-iopt1")
                    .build(),
```

```
                OrderableDbInstanceOption::builder()
                    .db_instance_class("t2")
                    .storage_type("aurora")
                    .build(),
                OrderableDbInstanceOption::builder()
                    .db_instance_class("t3")
                    .storage_type("aurora")
                    .build(),
            ])
        });

    let mut scenario = AuroraScenario::new(mock_rds);
    scenario
        .set_engine("aurora-mysql", "aurora-mysql8.0")
        .await
        .expect("set engine");

    let instance_classes = scenario.get_instance_classes().await;

    assert_eq!(
        instance_classes,
        Ok(vec!["t1".into(), "t2".into(), "t3".into()])
    );
}

#[tokio::test]
async fn test_scenario_get_instance_classes_error() {
    let mut mock_rds = MockRdsImpl::default();

    mock_rds
        .expect_describe_orderable_db_instance_options()
        .with(eq("aurora-mysql"), eq("aurora-mysql8.0"))
        .return_once(|_, _| {
            Err(SdkError::service_error(

 DescribeOrderableDBInstanceOptionsError::unhandled(Box::new(Error::new(
                    ErrorKind::Other,
                    "describe_orderable_db_instance_options_error",
                ))),
                Response::new(StatusCode::try_from(400).unwrap(),
 SdkBody::empty()),
            ))
        });
```

```
    let mut scenario = AuroraScenario::new(mock_rds);
    scenario.engine_family = Some("aurora-mysql".into());
    scenario.engine_version = Some("aurora-mysql8.0".into());

    let instance_classes = scenario.get_instance_classes().await;

    assert_matches!(
        instance_classes,
        Err(ScenarioError {message, context: _}) if message == "Could not get
 available instance classes"
    );
}

#[tokio::test]
async fn test_scenario_get_cluster() {
    let mut mock_rds = MockRdsImpl::default();

    mock_rds
        .expect_describe_db_clusters()
        .with(eq("RustSDKCodeExamplesDBCluster"))
        .return_once(|_| {
            Ok(DescribeDbClustersOutput::builder()
                .db_clusters(DbCluster::builder().build())
                .build())
        });

    let mut scenario = AuroraScenario::new(mock_rds);
    scenario.db_cluster_identifier = Some("RustSDKCodeExamplesDBCluster".into());
    let cluster = scenario.get_cluster().await;

    assert!(cluster.is_ok());
}

#[tokio::test]
async fn test_scenario_get_cluster_missing_cluster() {
    let mut mock_rds = MockRdsImpl::default();

    mock_rds
        .expect_create_db_cluster_parameter_group()
        .return_once(|_, _, _| {
            Ok(CreateDbClusterParameterGroupOutput::builder()

 .db_cluster_parameter_group(DbClusterParameterGroup::builder().build())
                .build())
```

```
        });

    mock_rds
        .expect_describe_db_clusters()
        .with(eq("RustSDKCodeExamplesDBCluster"))
        .return_once(|_| Ok(DescribeDbClustersOutput::builder().build()));

    let mut scenario = AuroraScenario::new(mock_rds);
    scenario.db_cluster_identifier = Some("RustSDKCodeExamplesDBCluster".into());
    let cluster = scenario.get_cluster().await;

    assert_matches!(cluster, Err(ScenarioError { message, context: _ }) if
 message == "Did not find the cluster");
}

#[tokio::test]
async fn test_scenario_get_cluster_error() {
    let mut mock_rds = MockRdsImpl::default();

    mock_rds
        .expect_create_db_cluster_parameter_group()
        .return_once(|_, _, _| {
            Ok(CreateDbClusterParameterGroupOutput::builder()

 .db_cluster_parameter_group(DbClusterParameterGroup::builder().build())
                .build())
        });

    mock_rds
        .expect_describe_db_clusters()
        .with(eq("RustSDKCodeExamplesDBCluster"))
        .return_once(|_| {
            Err(SdkError::service_error(
                DescribeDBClustersError::unhandled(Box::new(Error::new(
                    ErrorKind::Other,
                    "describe_db_clusters_error",
                ))),
                Response::new(StatusCode::try_from(400).unwrap(),
 SdkBody::empty()),
            ))
        });

    let mut scenario = AuroraScenario::new(mock_rds);
    scenario.db_cluster_identifier = Some("RustSDKCodeExamplesDBCluster".into());
```

```
    let cluster = scenario.get_cluster().await;

    assert_matches!(cluster, Err(ScenarioError { message, context: _ }) if
 message == "Failed to get cluster");
}

#[tokio::test]
async fn test_scenario_connection_string() {
    let mut mock_rds = MockRdsImpl::default();

    mock_rds
        .expect_describe_db_clusters()
        .with(eq("RustSDKCodeExamplesDBCluster"))
        .return_once(|_| {
            Ok(DescribeDbClustersOutput::builder()
                .db_clusters(
                    DbCluster::builder()
                        .endpoint("test_endpoint")
                        .port(3306)
                        .master_username("test_username")
                        .build(),
                )
                .build())
        });

    let mut scenario = AuroraScenario::new(mock_rds);
    scenario.db_cluster_identifier = Some("RustSDKCodeExamplesDBCluster".into());
    let connection_string = scenario.connection_string().await;

    assert_eq!(
        connection_string,
        Ok("mysql -h test_endpoint -P 3306 -u test_username -p".into())
    );
}

#[tokio::test]
async fn test_scenario_cluster_parameters() {
    let mut mock_rds = MockRdsImpl::default();

    mock_rds
        .expect_describe_db_cluster_parameters()
        .with(eq("RustSDKCodeExamplesDBParameterGroup"))
        .return_once(|_| {
            Ok(vec![DescribeDbClusterParametersOutput::builder()
```

```
                .parameters(Parameter::builder().parameter_name("a").build())
                .parameters(Parameter::builder().parameter_name("b").build())
                .parameters(
                    Parameter::builder()
                        .parameter_name("auto_increment_offset")
                        .build(),
                )
                .parameters(Parameter::builder().parameter_name("c").build())
                .parameters(
                    Parameter::builder()
                        .parameter_name("auto_increment_increment")
                        .build(),
                )
                .parameters(Parameter::builder().parameter_name("d").build())
                .build()])
        });

    let mut scenario = AuroraScenario::new(mock_rds);
    scenario.db_cluster_identifier = Some("RustSDKCodeExamplesDBCluster".into());

    let params = scenario.cluster_parameters().await.expect("cluster params");
    let names: Vec<String> = params.into_iter().map(|p| p.name).collect();
    assert_eq!(
        names,
        vec!["auto_increment_offset", "auto_increment_increment"]
    );
}

#[tokio::test]
async fn test_scenario_cluster_parameters_error() {
    let mut mock_rds = MockRdsImpl::default();

    mock_rds
        .expect_describe_db_cluster_parameters()
        .with(eq("RustSDKCodeExamplesDBParameterGroup"))
        .return_once(|_| {
            Err(SdkError::service_error(
                DescribeDBClusterParametersError::unhandled(Box::new(Error::new(
                    ErrorKind::Other,
                    "describe_db_cluster_parameters_error",
                ))),
                Response::new(StatusCode::try_from(400).unwrap(),
 SdkBody::empty()),
            ))
```

```
        });

    let mut scenario = AuroraScenario::new(mock_rds);
    scenario.db_cluster_identifier = Some("RustSDKCodeExamplesDBCluster".into());
    let params = scenario.cluster_parameters().await;
    assert_matches!(params, Err(ScenarioError { message, context: _ }) if message
 == "Failed to retrieve parameters for RustSDKCodeExamplesDBParameterGroup");
}

#[tokio::test]
async fn test_scenario_update_auto_increment() {
    let mut mock_rds = MockRdsImpl::default();

    mock_rds
        .expect_modify_db_cluster_parameter_group()
        .withf(|name, params| {
            assert_eq!(name, "RustSDKCodeExamplesDBParameterGroup");
            assert_eq!(
                params,
                &vec![
                    Parameter::builder()
                        .parameter_name("auto_increment_offset")
                        .parameter_value("10")
                        .apply_method(aws_sdk_rds::types::ApplyMethod::Immediate)
                        .build(),
                    Parameter::builder()
                        .parameter_name("auto_increment_increment")
                        .parameter_value("20")
                        .apply_method(aws_sdk_rds::types::ApplyMethod::Immediate)
                        .build(),
                ]
            );
            true
        })
        .return_once(|_, _|
 Ok(ModifyDbClusterParameterGroupOutput::builder().build()));

    let scenario = AuroraScenario::new(mock_rds);

    scenario
        .update_auto_increment(10, 20)
        .await
        .expect("update auto increment");
}
```

```
#[tokio::test]
async fn test_scenario_update_auto_increment_error() {
    let mut mock_rds = MockRdsImpl::default();

    mock_rds
        .expect_modify_db_cluster_parameter_group()
        .return_once(|_, _| {
            Err(SdkError::service_error(

 ModifyDBClusterParameterGroupError::unhandled(Box::new(Error::new(
                    ErrorKind::Other,
                    "modify_db_cluster_parameter_group_error",
                ))),
                Response::new(StatusCode::try_from(400).unwrap(),
 SdkBody::empty()),
            ))
        });

    let scenario = AuroraScenario::new(mock_rds);

    let update = scenario.update_auto_increment(10, 20).await;
    assert_matches!(update, Err(ScenarioError { message, context: _}) if message
 == "Failed to modify cluster parameter group");
}

#[tokio::test]
async fn test_start_cluster_and_instance() {
    let mut mock_rds = MockRdsImpl::default();

    mock_rds
        .expect_create_db_cluster()
        .withf(|id, params, engine, version, username, password| {
            assert_eq!(id, "RustSDKCodeExamplesDBCluster");
            assert_eq!(params, "RustSDKCodeExamplesDBParameterGroup");
            assert_eq!(engine, "aurora-mysql");
            assert_eq!(version, "aurora-mysql8.0");
            assert_eq!(username, "test username");
            assert_eq!(password.expose_secret(), "test password");
            true
        })
        .return_once(|id, _, _, _, _, _| {
            Ok(CreateDbClusterOutput::builder()
```

```
.db_cluster(DbCluster::builder().db_cluster_identifier(id).build())
                .build())
        });

    mock_rds
        .expect_create_db_instance()
        .withf(|cluster, name, class, engine| {
            assert_eq!(cluster, "RustSDKCodeExamplesDBCluster");
            assert_eq!(name, "RustSDKCodeExamplesDBInstance");
            assert_eq!(class, "m5.large");
            assert_eq!(engine, "aurora-mysql");
            true
        })
        .return_once(|cluster, name, class, _| {
            Ok(CreateDbInstanceOutput::builder()
                .db_instance(
                    DbInstance::builder()
                        .db_cluster_identifier(cluster)
                        .db_instance_identifier(name)
                        .db_instance_class(class)
                        .build(),
                )
                .build())
        });

    mock_rds
        .expect_describe_db_clusters()
        .with(eq("RustSDKCodeExamplesDBCluster"))
        .return_once(|id| {
            Ok(DescribeDbClustersOutput::builder()

.db_clusters(DbCluster::builder().db_cluster_identifier(id).build())
                .build())
        });

    mock_rds
        .expect_describe_db_instance()
        .with(eq("RustSDKCodeExamplesDBInstance"))
        .return_once(|name| {
            Ok(DescribeDbInstancesOutput::builder()
                .db_instances(
                    DbInstance::builder()
                        .db_instance_identifier(name)
```

```
                        .db_instance_status("Available")
                        .build(),
                )
                .build())
        });

    mock_rds
        .expect_describe_db_cluster_endpoints()
        .with(eq("RustSDKCodeExamplesDBCluster"))
        .return_once(|_| {
            Ok(DescribeDbClusterEndpointsOutput::builder()

 .db_cluster_endpoints(DbClusterEndpoint::builder().status("available").build())
                .build())
        });

    let mut scenario = AuroraScenario::new(mock_rds);
    scenario.engine_version = Some("aurora-mysql8.0".into());
    scenario.instance_class = Some("m5.large".into());
    scenario.username = Some("test username".into());
    scenario.password = Some(SecretString::new("test password".into()));

    tokio::time::pause();
    let assertions = tokio::spawn(async move {
        let create = scenario.start_cluster_and_instance().await;
        assert!(create.is_ok());
        assert!(scenario
            .password
            .replace(SecretString::new("BAD SECRET".into()))
            .unwrap()
            .expose_secret()
            .is_empty());
        assert_eq!(
            scenario.db_cluster_identifier,
            Some("RustSDKCodeExamplesDBCluster".into())
        );
    });
    tokio::time::advance(Duration::from_secs(1)).await;
    tokio::time::resume();
    let _ = assertions.await;
}

#[tokio::test]
async fn test_start_cluster_and_instance_cluster_create_error() {
```

```
    let mut mock_rds = MockRdsImpl::default();

    mock_rds
        .expect_create_db_cluster()
        .return_once(|_, _, _, _, _, _| {
            Err(SdkError::service_error(
                CreateDBClusterError::unhandled(Box::new(Error::new(
                    ErrorKind::Other,
                    "create db cluster error",
                ))),
                Response::new(StatusCode::try_from(400).unwrap(),
 SdkBody::empty()),
            ))
        });

    let mut scenario = AuroraScenario::new(mock_rds);
    scenario.engine_version = Some("aurora-mysql8.0".into());
    scenario.instance_class = Some("m5.large".into());
    scenario.username = Some("test username".into());
    scenario.password = Some(SecretString::new("test password".into()));

    let create = scenario.start_cluster_and_instance().await;
    assert_matches!(create, Err(ScenarioError { message, context: _}) if message
 == "Failed to create DB Cluster with cluster group")
}

#[tokio::test]
async fn test_start_cluster_and_instance_cluster_create_missing_id() {
    let mut mock_rds = MockRdsImpl::default();

    mock_rds
        .expect_create_db_cluster()
        .return_once(|_, _, _, _, _, _| {
            Ok(CreateDbClusterOutput::builder()
                .db_cluster(DbCluster::builder().build())
                .build())
        });

    let mut scenario = AuroraScenario::new(mock_rds);
    scenario.engine_version = Some("aurora-mysql8.0".into());
    scenario.instance_class = Some("m5.large".into());
    scenario.username = Some("test username".into());
    scenario.password = Some(SecretString::new("test password".into()));
```

```
    let create = scenario.start_cluster_and_instance().await;
    assert_matches!(create, Err(ScenarioError { message, context:_ }) if message
 == "Created DB Cluster missing Identifier");
}

#[tokio::test]
async fn test_start_cluster_and_instance_instance_create_error() {
    let mut mock_rds = MockRdsImpl::default();

    mock_rds
        .expect_create_db_cluster()
        .withf(|id, params, engine, version, username, password| {
            assert_eq!(id, "RustSDKCodeExamplesDBCluster");
            assert_eq!(params, "RustSDKCodeExamplesDBParameterGroup");
            assert_eq!(engine, "aurora-mysql");
            assert_eq!(version, "aurora-mysql8.0");
            assert_eq!(username, "test username");
            assert_eq!(password.expose_secret(), "test password");
            true
        })
        .return_once(|id, _, _, _, _, _| {
            Ok(CreateDbClusterOutput::builder()

 .db_cluster(DbCluster::builder().db_cluster_identifier(id).build())
                .build())
        });

    mock_rds
        .expect_create_db_instance()
        .return_once(|_, _, _, _| {
            Err(SdkError::service_error(
                CreateDBInstanceError::unhandled(Box::new(Error::new(
                    ErrorKind::Other,
                    "create db instance error",
                ))),
                Response::new(StatusCode::try_from(400).unwrap(),
 SdkBody::empty()),
            ))
        });

    let mut scenario = AuroraScenario::new(mock_rds);
    scenario.engine_version = Some("aurora-mysql8.0".into());
    scenario.instance_class = Some("m5.large".into());
    scenario.username = Some("test username".into());
```

```
    scenario.password = Some(SecretString::new("test password".into()));

    let create = scenario.start_cluster_and_instance().await;
    assert_matches!(create, Err(ScenarioError { message, context: _ }) if message
 == "Failed to create Instance in DB Cluster")
}

#[tokio::test]
async fn test_start_cluster_and_instance_wait_hiccup() {
    let mut mock_rds = MockRdsImpl::default();

    mock_rds
        .expect_create_db_cluster()
        .withf(|id, params, engine, version, username, password| {
            assert_eq!(id, "RustSDKCodeExamplesDBCluster");
            assert_eq!(params, "RustSDKCodeExamplesDBParameterGroup");
            assert_eq!(engine, "aurora-mysql");
            assert_eq!(version, "aurora-mysql8.0");
            assert_eq!(username, "test username");
            assert_eq!(password.expose_secret(), "test password");
            true
        })
        .return_once(|id, _, _, _, _, _| {
            Ok(CreateDbClusterOutput::builder()

 .db_cluster(DbCluster::builder().db_cluster_identifier(id).build())
                .build())
        });

    mock_rds
        .expect_create_db_instance()
        .withf(|cluster, name, class, engine| {
            assert_eq!(cluster, "RustSDKCodeExamplesDBCluster");
            assert_eq!(name, "RustSDKCodeExamplesDBInstance");
            assert_eq!(class, "m5.large");
            assert_eq!(engine, "aurora-mysql");
            true
        })
        .return_once(|cluster, name, class, _| {
            Ok(CreateDbInstanceOutput::builder()
                .db_instance(
                    DbInstance::builder()
                        .db_cluster_identifier(cluster)
                        .db_instance_identifier(name)
```

```
                        .db_instance_class(class)
                        .build(),
                )
                .build())
        });

    mock_rds
        .expect_describe_db_clusters()
        .with(eq("RustSDKCodeExamplesDBCluster"))
        .times(1)
        .returning(|_| {
            Err(SdkError::service_error(
                DescribeDBClustersError::unhandled(Box::new(Error::new(
                    ErrorKind::Other,
                    "describe cluster error",
                ))),
                Response::new(StatusCode::try_from(400).unwrap(),
SdkBody::empty()),
            ))
        })
        .with(eq("RustSDKCodeExamplesDBCluster"))
        .times(1)
        .returning(|id| {
            Ok(DescribeDbClustersOutput::builder()

.db_clusters(DbCluster::builder().db_cluster_identifier(id).build())
                .build())
        });

    mock_rds.expect_describe_db_instance().return_once(|name| {
        Ok(DescribeDbInstancesOutput::builder()
            .db_instances(
                DbInstance::builder()
                    .db_instance_identifier(name)
                    .db_instance_status("Available")
                    .build(),
            )
            .build())
    });

    mock_rds
        .expect_describe_db_cluster_endpoints()
        .return_once(|_| {
            Ok(DescribeDbClusterEndpointsOutput::builder()
```

```
 .db_cluster_endpoints(DbClusterEndpoint::builder().status("available").build())
                .build())
        });

    let mut scenario = AuroraScenario::new(mock_rds);
    scenario.engine_version = Some("aurora-mysql8.0".into());
    scenario.instance_class = Some("m5.large".into());
    scenario.username = Some("test username".into());
    scenario.password = Some(SecretString::new("test password".into()));

    tokio::time::pause();
    let assertions = tokio::spawn(async move {
        let create = scenario.start_cluster_and_instance().await;
        assert!(create.is_ok());
    });

    tokio::time::advance(Duration::from_secs(1)).await;
    tokio::time::advance(Duration::from_secs(1)).await;
    tokio::time::resume();
    let _ = assertions.await;
}

#[tokio::test]
async fn test_scenario_clean_up() {
    let mut mock_rds = MockRdsImpl::default();

    mock_rds
        .expect_delete_db_instance()
        .with(eq("MockInstance"))
        .return_once(|_| Ok(DeleteDbInstanceOutput::builder().build()));

    mock_rds
        .expect_describe_db_instances()
        .with()
        .times(1)
        .returning(|| {
            Ok(DescribeDbInstancesOutput::builder()
                .db_instances(
                    DbInstance::builder()
                        .db_cluster_identifier("MockCluster")
                        .db_instance_status("Deleting")
                        .build(),
                )
```

```
                .build())
        })
        .with()
        .times(1)
        .returning(|| Ok(DescribeDbInstancesOutput::builder().build()));

    mock_rds
        .expect_delete_db_cluster()
        .with(eq("MockCluster"))
        .return_once(|_| Ok(DeleteDbClusterOutput::builder().build()));

    mock_rds
        .expect_describe_db_clusters()
        .with(eq("MockCluster"))
        .times(1)
        .returning(|id| {
            Ok(DescribeDbClustersOutput::builder()
                .db_clusters(
                    DbCluster::builder()
                        .db_cluster_identifier(id)
                        .status("Deleting")
                        .build(),
                )
                .build())
        })
        .with(eq("MockCluster"))
        .times(1)
        .returning(|_| Ok(DescribeDbClustersOutput::builder().build()));

    mock_rds
        .expect_delete_db_cluster_parameter_group()
        .with(eq("MockParamGroup"))
        .return_once(|_|
Ok(DeleteDbClusterParameterGroupOutput::builder().build()));

    let mut scenario = AuroraScenario::new(mock_rds);
    scenario.db_cluster_identifier = Some(String::from("MockCluster"));
    scenario.db_instance_identifier = Some(String::from("MockInstance"));
    scenario.db_cluster_parameter_group = Some(
        DbClusterParameterGroup::builder()
            .db_cluster_parameter_group_name("MockParamGroup")
            .build(),
    );
```

```
    tokio::time::pause();
    let assertions = tokio::spawn(async move {
        let clean_up = scenario.clean_up().await;
        assert!(clean_up.is_ok());
    });

    tokio::time::advance(Duration::from_secs(1)).await; // Wait for first
 Describe Instances
    tokio::time::advance(Duration::from_secs(1)).await; // Wait for second
 Describe Instances
    tokio::time::advance(Duration::from_secs(1)).await; // Wait for first
 Describe Cluster
    tokio::time::advance(Duration::from_secs(1)).await; // Wait for second
 Describe Cluster
    tokio::time::resume();
    let _ = assertions.await;
}

#[tokio::test]
async fn test_scenario_clean_up_errors() {
    let mut mock_rds = MockRdsImpl::default();

    mock_rds
        .expect_delete_db_instance()
        .with(eq("MockInstance"))
        .return_once(|_| Ok(DeleteDbInstanceOutput::builder().build()));

    mock_rds
        .expect_describe_db_instances()
        .with()
        .times(1)
        .returning(|| {
            Ok(DescribeDbInstancesOutput::builder()
                .db_instances(
                    DbInstance::builder()
                        .db_cluster_identifier("MockCluster")
                        .db_instance_status("Deleting")
                        .build(),
                )
                .build())
        })
        .with()
        .times(1)
        .returning(|| {
```

```
            Err(SdkError::service_error(
                DescribeDBInstancesError::unhandled(Box::new(Error::new(
                    ErrorKind::Other,
                    "describe db instances error",
                ))),
                Response::new(StatusCode::try_from(400).unwrap(),
SdkBody::empty()),
            ))
        });

    mock_rds
        .expect_delete_db_cluster()
        .with(eq("MockCluster"))
        .return_once(|_| Ok(DeleteDbClusterOutput::builder().build()));

    mock_rds
        .expect_describe_db_clusters()
        .with(eq("MockCluster"))
        .times(1)
        .returning(|id| {
            Ok(DescribeDbClustersOutput::builder()
                .db_clusters(
                    DbCluster::builder()
                        .db_cluster_identifier(id)
                        .status("Deleting")
                        .build(),
                )
                .build())
        })
        .with(eq("MockCluster"))
        .times(1)
        .returning(|_| {
            Err(SdkError::service_error(
                DescribeDBClustersError::unhandled(Box::new(Error::new(
                    ErrorKind::Other,
                    "describe db clusters error",
                ))),
                Response::new(StatusCode::try_from(400).unwrap(),
SdkBody::empty()),
            ))
        });

    mock_rds
        .expect_delete_db_cluster_parameter_group()
```

```
        .with(eq("MockParamGroup"))
        .return_once(|_|
 Ok(DeleteDbClusterParameterGroupOutput::builder().build()));

    let mut scenario = AuroraScenario::new(mock_rds);
    scenario.db_cluster_identifier = Some(String::from("MockCluster"));
    scenario.db_instance_identifier = Some(String::from("MockInstance"));
    scenario.db_cluster_parameter_group = Some(
        DbClusterParameterGroup::builder()
            .db_cluster_parameter_group_name("MockParamGroup")
            .build(),
    );

    tokio::time::pause();
    let assertions = tokio::spawn(async move {
        let clean_up = scenario.clean_up().await;
        assert!(clean_up.is_err());
        let errs = clean_up.unwrap_err();
        assert_eq!(errs.len(), 2);
        assert_matches!(errs.first(), Some(ScenarioError {message, context: _})
 if message == "Failed to check instance state during deletion");
        assert_matches!(errs.get(1), Some(ScenarioError {message, context: _}) if
 message == "Failed to check cluster state during deletion");
    });

    tokio::time::advance(Duration::from_secs(1)).await; // Wait for first
 Describe Instances
    tokio::time::advance(Duration::from_secs(1)).await; // Wait for second
 Describe Instances
    tokio::time::advance(Duration::from_secs(1)).await; // Wait for first
 Describe Cluster
    tokio::time::advance(Duration::from_secs(1)).await; // Wait for second
 Describe Cluster
    tokio::time::resume();
    let _ = assertions.await;
}

#[tokio::test]
async fn test_scenario_snapshot() {
    let mut mock_rds = MockRdsImpl::default();

    mock_rds
        .expect_snapshot_cluster()
        .with(eq("MockCluster"), eq("MockCluster_MockSnapshot"))
```

```
        .times(1)
        .return_once(|_, _| {
            Ok(CreateDbClusterSnapshotOutput::builder()
                .db_cluster_snapshot(
                    DbClusterSnapshot::builder()
                        .db_cluster_identifier("MockCluster")

 .db_cluster_snapshot_identifier("MockCluster_MockSnapshot")
                        .build(),
                )
                .build())
        });

    let mut scenario = AuroraScenario::new(mock_rds);
    scenario.db_cluster_identifier = Some("MockCluster".into());
    let create_snapshot = scenario.snapshot("MockSnapshot").await;
    assert!(create_snapshot.is_ok());
}

#[tokio::test]
async fn test_scenario_snapshot_error() {
    let mut mock_rds = MockRdsImpl::default();

    mock_rds
        .expect_snapshot_cluster()
        .with(eq("MockCluster"), eq("MockCluster_MockSnapshot"))
        .times(1)
        .return_once(|_, _| {
            Err(SdkError::service_error(
                CreateDBClusterSnapshotError::unhandled(Box::new(Error::new(
                    ErrorKind::Other,
                    "create snapshot error",
                ))),
                Response::new(StatusCode::try_from(400).unwrap(),
 SdkBody::empty()),
            ))
        });

    let mut scenario = AuroraScenario::new(mock_rds);
    scenario.db_cluster_identifier = Some("MockCluster".into());
    let create_snapshot = scenario.snapshot("MockSnapshot").await;
    assert_matches!(create_snapshot, Err(ScenarioError { message, context: _}) if
 message == "Failed to create snapshot");
}
```

```
#[tokio::test]
async fn test_scenario_snapshot_invalid() {
    let mut mock_rds = MockRdsImpl::default();

    mock_rds
        .expect_snapshot_cluster()
        .with(eq("MockCluster"), eq("MockCluster_MockSnapshot"))
        .times(1)
        .return_once(|_, _|
 Ok(CreateDbClusterSnapshotOutput::builder().build()));

    let mut scenario = AuroraScenario::new(mock_rds);
    scenario.db_cluster_identifier = Some("MockCluster".into());
    let create_snapshot = scenario.snapshot("MockSnapshot").await;
    assert_matches!(create_snapshot, Err(ScenarioError { message, context: _}) if
 message == "Missing Snapshot");
}
```

Un file binario per eseguire lo scenario dall'inizio alla fine, utilizzando un inquirer per consentire all'utente di prendere decisioni.

```
use std::fmt::Display;

use anyhow::anyhow;
use aurora_code_examples::{
    aurora_scenario::{AuroraScenario, ScenarioError},
    rds::Rds as RdsClient,
};
use aws_sdk_rds::Client;
use inquire::{validator::StringValidator, CustomUserError};
use secrecy::SecretString;
use tracing::warn;

#[derive(Default, Debug)]
struct Warnings(Vec<String>);

impl Warnings {
    fn new() -> Self {
        Warnings(Vec::with_capacity(5))
    }
```

```
    fn push(&mut self, warning: &str, error: ScenarioError) {
        let formatted = format!("{warning}: {error}");
        warn!("{formatted}");
        self.0.push(formatted);
    }

    fn is_empty(&self) -> bool {
        self.0.is_empty()
    }
}

impl Display for Warnings {
    fn fmt(&self, f: &mut std::fmt::Formatter<'_>) -> std::fmt::Result {
        writeln!(f, "Warnings:")?;
        for warning in &self.0 {
            writeln!(f, "{: >4}- {warning}", "")?;
        }
        Ok(())
    }
}

fn select(
    prompt: &str,
    choices: Vec<String>,
    error_message: &str,
) -> Result<String, anyhow::Error> {
    inquire::Select::new(prompt, choices)
        .prompt()
        .map_err(|error| anyhow!("{error_message}: {error}"))
}

// Prepare the Aurora Scenario. Prompt for several settings that are optional to
 the Scenario, but that the user should choose for the demo.
// This includes the engine, engine version, and instance class.
async fn prepare_scenario(rds: RdsClient) -> Result<AuroraScenario,
 anyhow::Error> {
    let mut scenario = AuroraScenario::new(rds);

    // Get available engine families for Aurora MySql.
 rds.DescribeDbEngineVersions(Engine='aurora-mysql') and build a set of the
 'DBParameterGroupFamily' field values. I get {aurora-mysql8.0, aurora-mysql5.7}.
    let available_engines = scenario.get_engines().await;
    if let Err(error) = available_engines {
```

```
        return Err(anyhow!("Failed to get available engines: {}", error));
    }
    let available_engines = available_engines.unwrap();

    // Select an engine family and create a custom DB cluster parameter group.
 rds.CreateDbClusterParameterGroup(DBParameterGroupFamily='aurora-mysql8.0')
    let engine = select(
        "Select an Aurora engine family",
        available_engines.keys().cloned().collect::<Vec<String>>(),
        "Invalid engine selection",
    )?;

    let version = select(
        format!("Select an Aurora engine version for {engine}").as_str(),
        available_engines.get(&engine).cloned().unwrap_or_default(),
        "Invalid engine version selection",
    )?;

    let set_engine = scenario.set_engine(engine.as_str(),
 version.as_str()).await;
    if let Err(error) = set_engine {
        return Err(anyhow!("Could not set engine: {}", error));
    }

    let instance_classes = scenario.get_instance_classes().await;
    match instance_classes {
        Ok(classes) => {
            let instance_class = select(
                format!("Select an Aurora instance class for {engine}").as_str(),
                classes,
                "Invalid instance class selection",
            )?;
            scenario.set_instance_class(Some(instance_class))
        }
        Err(err) => return Err(anyhow!("Failed to get instance classes for
 engine: {err}")),
    }

    Ok(scenario)
}

// Prepare the cluster, creating a custom parameter group overriding some group
 parameters based on user input.
```

```
async fn prepare_cluster(scenario: &mut AuroraScenario, warnings: &mut Warnings)
 -> Result<(), ()> {
    show_parameters(scenario, warnings).await;

    let offset = prompt_number_or_default(warnings, "auto_increment_offset", 5);
    let increment = prompt_number_or_default(warnings,
 "auto_increment_increment", 3);

    // Modify both the auto_increment_offset and auto_increment_increment
 parameters in one call in the custom parameter group. Set their ParameterValue
 fields to a new allowable value. rds.ModifyDbClusterParameterGroup.
    let update_auto_increment = scenario.update_auto_increment(offset,
 increment).await;

    if let Err(error) = update_auto_increment {
        warnings.push("Failed to update auto increment", error);
        return Err(());
    }

    // Get and display the updated parameters. Specify Source of 'user' to get
 just the modified parameters. rds.DescribeDbClusterParameters(Source='user')
    show_parameters(scenario, warnings).await;

    let username = inquire::Text::new("Username for the database (default
'testuser')")
        .with_default("testuser")
        .with_initial_value("testuser")
        .prompt();

    if let Err(error) = username {
        warnings.push(
            "Failed to get username, using default",
            ScenarioError::with(format!("Error from inquirer: {error}")),
        );
        return Err(());
    }
    let username = username.unwrap();

    let password = inquire::Text::new("Password for the database (minimum 8
 characters)")
        .with_validator(|i: &str| {
            if i.len() >= 8 {
                Ok(inquire::validator::Validation::Valid)
            } else {
```

```
                Ok(inquire::validator::Validation::Invalid(
                    "Password must be at least 8 characters".into(),
                ))
            }
        })
        .prompt();

    let password: Option<SecretString> = match password {
        Ok(password) => Some(SecretString::from(password)),
        Err(error) => {
            warnings.push(
                "Failed to get password, using none (and not starting a DB)",
                ScenarioError::with(format!("Error from inquirer: {error}")),
            );
            return Err(());
        }
    };

    scenario.set_login(Some(username), password);

    Ok(())
}

// Start a single instance in the cluster,
async fn run_instance(scenario: &mut AuroraScenario) -> Result<(), ScenarioError>
 {
    // Create an Aurora DB cluster database cluster that contains a MySql
 database and uses the parameter group you created.
    // Create a database instance in the cluster.
    // Wait for DB instance to be ready. Call rds.DescribeDbInstances and check
 for DBInstanceStatus == 'available'.
    scenario.start_cluster_and_instance().await?;

    let connection_string = scenario.connection_string().await?;

    println!("Database ready: {connection_string}",);

    let _ = inquire::Text::new("Use the database with the connection string. When
 you're finished, press enter key to continue.").prompt();

    // Create a snapshot of the DB cluster. rds.CreateDbClusterSnapshot.
    // Wait for the snapshot to create. rds.DescribeDbClusterSnapshots until
 Status == 'available'.
    let snapshot_name = inquire::Text::new("Provide a name for the snapshot")
```

```
        .prompt()
        .unwrap_or(String::from("ScenarioRun"));
    let snapshot = scenario.snapshot(snapshot_name.as_str()).await?;
    println!(
        "Snapshot is available: {}",
        snapshot.db_cluster_snapshot_arn().unwrap_or("Missing ARN")
    );

    Ok(())
}

#[tokio::main]
async fn main() -> Result<(), anyhow::Error> {
    tracing_subscriber::fmt::init();
    let sdk_config = aws_config::from_env().load().await;
    let client = Client::new(&sdk_config);
    let rds = RdsClient::new(client);
    let mut scenario = prepare_scenario(rds).await?;

    // At this point, the scenario has things in AWS and needs to get cleaned up.
    let mut warnings = Warnings::new();

    if prepare_cluster(&mut scenario, &mut warnings).await.is_ok() {
        println!("Configured database cluster, starting an instance.");
        if let Err(err) = run_instance(&mut scenario).await {
            warnings.push("Problem running instance", err);
        }
    }

    // Clean up the instance, cluster, and parameter group, waiting for the
 instance and cluster to delete before moving on.
    let clean_up = scenario.clean_up().await;
    if let Err(errors) = clean_up {
        for error in errors {
            warnings.push("Problem cleaning up scenario", error);
        }
    }

    if warnings.is_empty() {
        Ok(())
    } else {
        println!("There were problems running the scenario:");
        println!("{warnings}");
        Err(anyhow!("There were problems running the scenario"))
```

```
    }
}

#[derive(Clone)]
struct U8Validator {}
impl StringValidator for U8Validator {
    fn validate(&self, input: &str) -> Result<inquire::validator::Validation,
 CustomUserError> {
        if input.parse::<u8>().is_err() {
            Ok(inquire::validator::Validation::Invalid(
                "Can't parse input as number".into(),
            ))
        } else {
            Ok(inquire::validator::Validation::Valid)
        }
    }
}

async fn show_parameters(scenario: &AuroraScenario, warnings: &mut Warnings) {
    let parameters = scenario.cluster_parameters().await;

    match parameters {
        Ok(parameters) => {
            println!("Current parameters");
            for parameter in parameters {
                println!("\t{parameter}");
            }
        }
        Err(error) => warnings.push("Could not find cluster parameters", error),
    }
}

fn prompt_number_or_default(warnings: &mut Warnings, name: &str, default: u8) ->
 u8 {
    let input = inquire::Text::new(format!("Updated {name}:").as_str())
        .with_validator(U8Validator {})
        .prompt();

    match input {
        Ok(increment) => match increment.parse::<u8>() {
            Ok(increment) => increment,
            Err(error) => {
                warnings.push(
```

```
                    format!("Invalid updated {name} (using {default}
 instead)").as_str(),
                    ScenarioError::with(format!("{error}")),
                );
                default
            }
        },
        Err(error) => {
            warnings.push(
                format!("Invalid updated {name} (using {default}
 instead)").as_str(),
                ScenarioError::with(format!("{error}")),
            );
            default
        }
    }
}
```

Un wrapper per il servizio Amazon RDS che consente l'automocking per i test.

```
use aws_sdk_rds::{
    error::SdkError,
    operation::{
        create_db_cluster::{CreateDBClusterError, CreateDbClusterOutput},
        create_db_cluster_parameter_group::CreateDBClusterParameterGroupError,
        create_db_cluster_parameter_group::CreateDbClusterParameterGroupOutput,
        create_db_cluster_snapshot::{CreateDBClusterSnapshotError,
 CreateDbClusterSnapshotOutput},
        create_db_instance::{CreateDBInstanceError, CreateDbInstanceOutput},
        delete_db_cluster::{DeleteDBClusterError, DeleteDbClusterOutput},
        delete_db_cluster_parameter_group::{
            DeleteDBClusterParameterGroupError,
 DeleteDbClusterParameterGroupOutput,
        },
        delete_db_instance::{DeleteDBInstanceError, DeleteDbInstanceOutput},
        describe_db_cluster_endpoints::{
            DescribeDBClusterEndpointsError, DescribeDbClusterEndpointsOutput,
        },
        describe_db_cluster_parameters::{
            DescribeDBClusterParametersError, DescribeDbClusterParametersOutput,
        },
```

```
        describe_db_clusters::{DescribeDBClustersError,
 DescribeDbClustersOutput},
        describe_db_engine_versions::{
            DescribeDBEngineVersionsError, DescribeDbEngineVersionsOutput,
        },
        describe_db_instances::{DescribeDBInstancesError,
 DescribeDbInstancesOutput},

 describe_orderable_db_instance_options::DescribeOrderableDBInstanceOptionsError,
        modify_db_cluster_parameter_group::{
            ModifyDBClusterParameterGroupError,
 ModifyDbClusterParameterGroupOutput,
        },
    },
    types::{OrderableDbInstanceOption, Parameter},
    Client as RdsClient,
};
use secrecy::{ExposeSecret, SecretString};

#[cfg(test)]
use mockall::automock;

#[cfg(test)]
pub use MockRdsImpl as Rds;
#[cfg(not(test))]
pub use RdsImpl as Rds;

pub struct RdsImpl {
    pub inner: RdsClient,
}

#[cfg_attr(test, automock)]
impl RdsImpl {
    pub fn new(inner: RdsClient) -> Self {
        RdsImpl { inner }
    }

    pub async fn describe_db_engine_versions(
        &self,
        engine: &str,
    ) -> Result<DescribeDbEngineVersionsOutput,
 SdkError<DescribeDBEngineVersionsError>> {
        self.inner
            .describe_db_engine_versions()
```

```
            .engine(engine)
            .send()
            .await
    }

    pub async fn describe_orderable_db_instance_options(
        &self,
        engine: &str,
        engine_version: &str,
    ) -> Result<Vec<OrderableDbInstanceOption>,
SdkError<DescribeOrderableDBInstanceOptionsError>>
    {
        self.inner
            .describe_orderable_db_instance_options()
            .engine(engine)
            .engine_version(engine_version)
            .into_paginator()
            .items()
            .send()
            .try_collect()
            .await
    }

    pub async fn create_db_cluster_parameter_group(
        &self,
        name: &str,
        description: &str,
        family: &str,
    ) -> Result<CreateDbClusterParameterGroupOutput,
SdkError<CreateDBClusterParameterGroupError>>
    {
        self.inner
            .create_db_cluster_parameter_group()
            .db_cluster_parameter_group_name(name)
            .description(description)
            .db_parameter_group_family(family)
            .send()
            .await
    }

    pub async fn describe_db_clusters(
        &self,
        id: &str,
    ) -> Result<DescribeDbClustersOutput, SdkError<DescribeDBClustersError>> {
```

```
        self.inner
            .describe_db_clusters()
            .db_cluster_identifier(id)
            .send()
            .await
    }

    pub async fn describe_db_cluster_parameters(
        &self,
        name: &str,
    ) -> Result<Vec<DescribeDbClusterParametersOutput>,
SdkError<DescribeDBClusterParametersError>>
    {
        self.inner
            .describe_db_cluster_parameters()
            .db_cluster_parameter_group_name(name)
            .into_paginator()
            .send()
            .try_collect()
            .await
    }

    pub async fn modify_db_cluster_parameter_group(
        &self,
        name: &str,
        parameters: Vec<Parameter>,
    ) -> Result<ModifyDbClusterParameterGroupOutput,
SdkError<ModifyDBClusterParameterGroupError>>
    {
        self.inner
            .modify_db_cluster_parameter_group()
            .db_cluster_parameter_group_name(name)
            .set_parameters(Some(parameters))
            .send()
            .await
    }

    pub async fn create_db_cluster(
        &self,
        name: &str,
        parameter_group: &str,
        engine: &str,
        version: &str,
        username: &str,
```

```
        password: SecretString,
    ) -> Result<CreateDbClusterOutput, SdkError<CreateDBClusterError>> {
        self.inner
            .create_db_cluster()
            .db_cluster_identifier(name)
            .db_cluster_parameter_group_name(parameter_group)
            .engine(engine)
            .engine_version(version)
            .master_username(username)
            .master_user_password(password.expose_secret())
            .send()
            .await
    }

    pub async fn create_db_instance(
        &self,
        cluster_name: &str,
        instance_name: &str,
        instance_class: &str,
        engine: &str,
    ) -> Result<CreateDbInstanceOutput, SdkError<CreateDBInstanceError>> {
        self.inner
            .create_db_instance()
            .db_cluster_identifier(cluster_name)
            .db_instance_identifier(instance_name)
            .db_instance_class(instance_class)
            .engine(engine)
            .send()
            .await
    }

    pub async fn describe_db_instance(
        &self,
        instance_identifier: &str,
    ) -> Result<DescribeDbInstancesOutput, SdkError<DescribeDBInstancesError>> {
        self.inner
            .describe_db_instances()
            .db_instance_identifier(instance_identifier)
            .send()
            .await
    }

    pub async fn snapshot_cluster(
        &self,
```

```
        db_cluster_identifier: &str,
        snapshot_name: &str,
    ) -> Result<CreateDbClusterSnapshotOutput,
SdkError<CreateDBClusterSnapshotError>> {
        self.inner
            .create_db_cluster_snapshot()
            .db_cluster_identifier(db_cluster_identifier)
            .db_cluster_snapshot_identifier(snapshot_name)
            .send()
            .await
    }

    pub async fn describe_db_instances(
        &self,
    ) -> Result<DescribeDbInstancesOutput, SdkError<DescribeDBInstancesError>> {
        self.inner.describe_db_instances().send().await
    }

    pub async fn describe_db_cluster_endpoints(
        &self,
        cluster_identifier: &str,
    ) -> Result<DescribeDbClusterEndpointsOutput,
SdkError<DescribeDBClusterEndpointsError>> {
        self.inner
            .describe_db_cluster_endpoints()
            .db_cluster_identifier(cluster_identifier)
            .send()
            .await
    }

    pub async fn delete_db_instance(
        &self,
        instance_identifier: &str,
    ) -> Result<DeleteDbInstanceOutput, SdkError<DeleteDBInstanceError>> {
        self.inner
            .delete_db_instance()
            .db_instance_identifier(instance_identifier)
            .skip_final_snapshot(true)
            .send()
            .await
    }

    pub async fn delete_db_cluster(
        &self,
```

```
        cluster_identifier: &str,
    ) -> Result<DeleteDbClusterOutput, SdkError<DeleteDBClusterError>> {
        self.inner
            .delete_db_cluster()
            .db_cluster_identifier(cluster_identifier)
            .skip_final_snapshot(true)
            .send()
            .await
    }

    pub async fn delete_db_cluster_parameter_group(
        &self,
        name: &str,
    ) -> Result<DeleteDbClusterParameterGroupOutput,
 SdkError<DeleteDBClusterParameterGroupError>>
    {
        self.inner
            .delete_db_cluster_parameter_group()
            .db_cluster_parameter_group_name(name)
            .send()
            .await
    }
}
```

Il file Cargo.toml con dipendenze utilizzato in questo scenario.

```
[package]
name = "aurora-code-examples"
authors = [
  "David Souther <dpsouth@amazon.com>",
]
edition = "2021"
version = "0.1.0"

# See more keys and their definitions at https://doc.rust-lang.org/cargo/
reference/manifest.html

[dependencies]
anyhow = "1.0.75"
assert_matches = "1.5.0"
aws-config = { version = "1.0.1", features = ["behavior-version-latest"] }
aws-smithy-types = { version = "1.0.1" }
```

```
aws-smithy-runtime-api = { version = "1.0.1" }
aws-sdk-rds = { version = "1.3.0" }
inquire = "0.6.2"
mockall = "0.11.4"
phf = { version = "0.11.2", features = ["std", "macros"] }
sdk-examples-test-utils = { path = "../../test-utils" }
secrecy = "0.8.0"
tokio = { version = "1.20.1", features = ["full", "test-util"] }
tracing = "0.1.37"
tracing-subscriber = { version = "0.3.15", features = ["env-filter"] }
```

- Per informazioni dettagliate sull'API, consulta i seguenti argomenti nella documentazione di riferimento dell'API SDK AWS per Rust.
  - CreaDBCluster
  - CreaDBClusterParameterGroup
  - Crea DBCluster istantanea
  - CreaDBInstance
  - EliminaDBCluster
  - EliminaDBClusterParameterGroup
  - EliminaDBInstance
  - Descriva DBCluster ParameterGroups
  - DBClusterDescrivi parametri
  - Descrivi le DBCluster istantanee
  - Descriva DBClusters
  - Descrivi DBEngine versioni
  - Descriva DBInstances
  - DescribeOrderableDBInstanceOpzioni
  - ModificaDBClusterParameterGroup

Per un elenco completo delle guide per sviluppatori AWS SDK e degli esempi di codice, consultaUtilizzo di questo servizio con un AWS SDK. Questo argomento include anche informazioni su come iniziare e dettagli sulle versioni precedenti dell'SDK.

Azioni per l'utilizzo di Aurora AWS SDKs

I seguenti esempi di codice mostrano come eseguire singole azioni Aurora con. AWS SDKs Ogni esempio include un collegamento a GitHub, dove è possibile trovare le istruzioni per la configurazione e l'esecuzione del codice.

Questi estratti chiamano l'API Aurora e sono estratti di codice da programmi più grandi che devono essere eseguiti in modo contestuale. È possibile visualizzare le azioni nel contesto in Scenari per l'utilizzo di Aurora AWS SDKs .

Gli esempi seguenti includono solo le azioni più comunemente utilizzate. Per l'elenco completo, consulta la documentazione di riferimento dell'API Amazon Aurora.

Esempi

- Utilizzo CreateDBCluster con un AWS SDK
- Utilizzo CreateDBClusterParameterGroup con un AWS SDK
- Utilizzo CreateDBClusterSnapshot con un AWS SDK
- Utilizzo CreateDBInstance con un AWS SDK
- Utilizzo DeleteDBCluster con un AWS SDK
- Utilizzo DeleteDBClusterParameterGroup con un AWS SDK
- Utilizzo DeleteDBInstance con un AWS SDK
- Utilizzo DescribeDBClusterParameterGroups con un AWS SDK
- Utilizzo DescribeDBClusterParameters con un AWS SDK
- Utilizzo DescribeDBClusterSnapshots con un AWS SDK
- Utilizzo DescribeDBClusters con un AWS SDK
- Utilizzo DescribeDBEngineVersions con un AWS SDK
- Utilizzo DescribeDBInstances con un AWS SDK
- Utilizzo DescribeOrderableDBInstanceOptions con un AWS SDK o una CLI
- Utilizzo ModifyDBClusterParameterGroup con un AWS SDK

Utilizzo **`CreateDBCluster`** con un AWS SDK

Gli esempi di codice seguenti mostrano come utilizzare `CreateDBCluster`.

Gli esempi di operazioni sono estratti di codice da programmi più grandi e devono essere eseguiti nel contesto. È possibile visualizzare questa operazione nel contesto nel seguente esempio di codice:

- Informazioni di base

.NET

SDK per .NET(v4)

> **Note**
>
> C'è altro da fare. GitHub Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice AWS.

```
/// <summary>
/// Create a new cluster and database.
/// </summary>
/// <param name="dbName">The name of the new database.</param>
/// <param name="clusterIdentifier">The identifier of the cluster.</param>
/// <param name="parameterGroupName">The name of the parameter group.</param>
/// <param name="dbEngine">The engine to use for the new cluster.</param>
/// <param name="dbEngineVersion">The version of the engine to use.</param>
/// <param name="adminName">The admin username.</param>
/// <param name="adminPassword">The primary admin password.</param>
/// <returns>The cluster object.</returns>
public async Task<DBCluster> CreateDBClusterWithAdminAsync(
    string dbName,
    string clusterIdentifier,
    string parameterGroupName,
    string dbEngine,
    string dbEngineVersion,
    string adminName,
    string adminPassword)
{
    var request = new CreateDBClusterRequest
    {
        DatabaseName = dbName,
        DBClusterIdentifier = clusterIdentifier,
        DBClusterParameterGroupName = parameterGroupName,
        Engine = dbEngine,
```

```
            EngineVersion = dbEngineVersion,
            MasterUsername = adminName,
            MasterUserPassword = adminPassword,
        };

        var response = await _amazonRDS.CreateDBClusterAsync(request);
        return response.DBCluster;
    }
```

- Per i dettagli sull'API, consulta Create DBCluster in AWS SDK per .NETAPI Reference.

C++

SDK per C++

> **Note**
>
> C'è altro su GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice AWS.

```
    Aws::Client::ClientConfiguration clientConfig;
    // Optional: Set to the AWS Region (overrides config file).
    // clientConfig.region = "us-east-1";

Aws::RDS::RDSClient client(clientConfig);

    Aws::RDS::Model::CreateDBClusterRequest request;
    request.SetDBClusterIdentifier(DB_CLUSTER_IDENTIFIER);
    request.SetDBClusterParameterGroupName(CLUSTER_PARAMETER_GROUP_NAME);
    request.SetEngine(engineName);
    request.SetEngineVersion(engineVersionName);
    request.SetMasterUsername(administratorName);
    request.SetMasterUserPassword(administratorPassword);

    Aws::RDS::Model::CreateDBClusterOutcome outcome =
            client.CreateDBCluster(request);

    if (outcome.IsSuccess()) {
        std::cout << "The DB cluster creation has started."
                  << std::endl;
```

```
        }
        else {
            std::cerr << "Error with Aurora::CreateDBCluster. "
                      << outcome.GetError().GetMessage()
                      << std::endl;
            cleanUpResources(CLUSTER_PARAMETER_GROUP_NAME, "", "", client);
            return false;
        }
```

- Per i dettagli sull'API, consulta Create DBCluster in AWS SDK per C++API Reference.

Go

SDK per Go V2

> **Note**
>
> C'è altro su GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice AWS.

```
import (
 "context"
 "errors"
 "log"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/service/rds"
 "github.com/aws/aws-sdk-go-v2/service/rds/types"
)

type DbClusters struct {
 AuroraClient *rds.Client
}



// CreateDbCluster creates a DB cluster that is configured to use the specified
 parameter group.
```

```
// The newly created DB cluster contains a database that uses the specified
 engine and
// engine version.
func (clusters *DbClusters) CreateDbCluster(ctx context.Context, clusterName
 string, parameterGroupName string,
 dbName string, dbEngine string, dbEngineVersion string, adminName string,
 adminPassword string) (
 *types.DBCluster, error) {

 output, err := clusters.AuroraClient.CreateDBCluster(ctx,
 &rds.CreateDBClusterInput{
  DBClusterIdentifier:         aws.String(clusterName),
  Engine:                      aws.String(dbEngine),
  DBClusterParameterGroupName: aws.String(parameterGroupName),
  DatabaseName:                aws.String(dbName),
  EngineVersion:               aws.String(dbEngineVersion),
  MasterUserPassword:          aws.String(adminPassword),
  MasterUsername:              aws.String(adminName),
 })
 if err != nil {
  log.Printf("Couldn't create DB cluster %v: %v\n", clusterName, err)
  return nil, err
 } else {
  return output.DBCluster, err
 }
}
```

- Per i dettagli sull'API, consulta Create DBCluster in AWS SDK per GoAPI Reference.

Java

SDK per Java 2.x

> **Note**
>
> C'è altro su GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice AWS.

```
   public static String createDBCluster(RdsClient rdsClient, String
dbParameterGroupFamily, String dbName,
           String dbClusterIdentifier, String userName, String password) {
       try {
           CreateDbClusterRequest clusterRequest =
CreateDbClusterRequest.builder()
                   .databaseName(dbName)
                   .dbClusterIdentifier(dbClusterIdentifier)
                   .dbClusterParameterGroupName(dbParameterGroupFamily)
                   .engine("aurora-mysql")
                   .masterUsername(userName)
                   .masterUserPassword(password)
                   .build();

           CreateDbClusterResponse response =
rdsClient.createDBCluster(clusterRequest);
           return response.dbCluster().dbClusterArn();

       } catch (RdsException e) {
           System.out.println(e.getLocalizedMessage());
           System.exit(1);
       }
       return "";
   }
```

- Per i dettagli sull'API, consulta Create DBCluster in AWS SDK for Java 2.xAPI Reference.

Kotlin

SDK per Kotlin

> **Note**
>
> C'è altro su GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice AWS.

```
suspend fun createDBCluster(
    dbParameterGroupFamilyVal: String?,
    dbName: String?,
```

```
    dbClusterIdentifierVal: String?,
    userName: String?,
    password: String?,
): String? {
    val clusterRequest =
        CreateDbClusterRequest {
            databaseName = dbName
            dbClusterIdentifier = dbClusterIdentifierVal
            dbClusterParameterGroupName = dbParameterGroupFamilyVal
            engine = "aurora-mysql"
            masterUsername = userName
            masterUserPassword = password
        }

    RdsClient.fromEnvironment { region = "us-west-2" }.use { rdsClient ->
        val response = rdsClient.createDbCluster(clusterRequest)
        return response.dbCluster?.dbClusterArn
    }
}
```

- Per i dettagli sull'API, consulta Create DBCluster in AWSSDK for Kotlin API reference.

Python

SDK per Python (Boto3)

> **Note**
>
> C'è altro su. GitHub Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice AWS.

```
class AuroraWrapper:
    """Encapsulates Aurora DB cluster actions."""

    def __init__(self, rds_client):
        """
        :param rds_client: A Boto3 Amazon Relational Database Service (Amazon
 RDS) client.
        """
        self.rds_client = rds_client
```

```
    @classmethod
    def from_client(cls):
        """
        Instantiates this class from a Boto3 client.
        """
        rds_client = boto3.client("rds")
        return cls(rds_client)


    def create_db_cluster(
        self,
        cluster_name,
        parameter_group_name,
        db_name,
        db_engine,
        db_engine_version,
        admin_name,
        admin_password,
    ):
        """
        Creates a DB cluster that is configured to use the specified parameter
group.
        The newly created DB cluster contains a database that uses the specified
engine and
        engine version.

        :param cluster_name: The name of the DB cluster to create.
        :param parameter_group_name: The name of the parameter group to associate
with
                                     the DB cluster.
        :param db_name: The name of the database to create.
        :param db_engine: The database engine of the database that is created,
such as MySql.
        :param db_engine_version: The version of the database engine.
        :param admin_name: The user name of the database administrator.
        :param admin_password: The password of the database administrator.
        :return: The newly created DB cluster.
        """
        try:
            response = self.rds_client.create_db_cluster(
                DatabaseName=db_name,
                DBClusterIdentifier=cluster_name,
                DBClusterParameterGroupName=parameter_group_name,
```

```
                Engine=db_engine,
                EngineVersion=db_engine_version,
                MasterUsername=admin_name,
                MasterUserPassword=admin_password,
            )
            cluster = response["DBCluster"]
        except ClientError as err:
            logger.error(
                "Couldn't create database %s. Here's why: %s: %s",
                db_name,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
        else:
            return cluster
```

- Per i dettagli sull'API, consulta Create DBCluster in AWSSDK for Python (Boto3) API Reference.

Rust

SDK per Rust

> **Note**
>
> C'è altro da fare. GitHub Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice AWS.

```
   // Get a list of allowed engine versions.
rds.DescribeDbEngineVersions(Engine='aurora-mysql', DBParameterGroupFamily=<the
family used to create your parameter group in step 2>)
   // Create an Aurora DB cluster database cluster that contains a MySql
database and uses the parameter group you created.
   // Wait for DB cluster to be ready. Call rds.DescribeDBClusters and check for
Status == 'available'.
```

```
   // Get a list of instance classes available for the selected engine
and engine version. rds.DescribeOrderableDbInstanceOptions(Engine='mysql',
EngineVersion=).

   // Create a database instance in the cluster.
   // Wait for DB instance to be ready. Call rds.DescribeDbInstances and check
for DBInstanceStatus == 'available'.
   pub async fn start_cluster_and_instance(&mut self) -> Result<(),
ScenarioError> {
       if self.password.is_none() {
           return Err(ScenarioError::with(
               "Must set Secret Password before starting a cluster",
           ));
       }
       let create_db_cluster = self
           .rds
           .create_db_cluster(
               DB_CLUSTER_IDENTIFIER,
               DB_CLUSTER_PARAMETER_GROUP_NAME,
               DB_ENGINE,
               self.engine_version.as_deref().expect("engine version"),
               self.username.as_deref().expect("username"),
               self.password
                   .replace(SecretString::new("".to_string()))
                   .expect("password"),
           )
           .await;
       if let Err(err) = create_db_cluster {
           return Err(ScenarioError::new(
               "Failed to create DB Cluster with cluster group",
               &err,
           ));
       }

       self.db_cluster_identifier = create_db_cluster
           .unwrap()
           .db_cluster
           .and_then(|c| c.db_cluster_identifier);

       if self.db_cluster_identifier.is_none() {
           return Err(ScenarioError::with("Created DB Cluster missing
Identifier"));
       }
```

```
        info!(
            "Started a db cluster: {}",
            self.db_cluster_identifier
                .as_deref()
                .unwrap_or("Missing ARN")
        );

        let create_db_instance = self
            .rds
            .create_db_instance(
                self.db_cluster_identifier.as_deref().expect("cluster name"),
                DB_INSTANCE_IDENTIFIER,
                self.instance_class.as_deref().expect("instance class"),
                DB_ENGINE,
            )
            .await;
        if let Err(err) = create_db_instance {
            return Err(ScenarioError::new(
                "Failed to create Instance in DB Cluster",
                &err,
            ));
        }

        self.db_instance_identifier = create_db_instance
            .unwrap()
            .db_instance
            .and_then(|i| i.db_instance_identifier);

        // Cluster creation can take up to 20 minutes to become available
        let cluster_max_wait = Duration::from_secs(20 * 60);
        let waiter = Waiter::builder().max(cluster_max_wait).build();
        while waiter.sleep().await.is_ok() {
            let cluster = self
                .rds
                .describe_db_clusters(
                    self.db_cluster_identifier
                        .as_deref()
                        .expect("cluster identifier"),
                )
                .await;

            if let Err(err) = cluster {
                warn!(?err, "Failed to describe cluster while waiting for
ready");
```

```
                continue;
            }

            let instance = self
                .rds
                .describe_db_instance(
                    self.db_instance_identifier
                        .as_deref()
                        .expect("instance identifier"),
                )
                .await;
            if let Err(err) = instance {
                return Err(ScenarioError::new(
                    "Failed to find instance for cluster",
                    &err,
                ));
            }

            let instances_available = instance
                .unwrap()
                .db_instances()
                .iter()
                .all(|instance| instance.db_instance_status() ==
Some("Available"));

            let endpoints = self
                .rds
                .describe_db_cluster_endpoints(
                    self.db_cluster_identifier
                        .as_deref()
                        .expect("cluster identifier"),
                )
                .await;

            if let Err(err) = endpoints {
                return Err(ScenarioError::new(
                    "Failed to find endpoint for cluster",
                    &err,
                ));
            }

            let endpoints_available = endpoints
                .unwrap()
                .db_cluster_endpoints()
```

```
                .iter()
                .all(|endpoint| endpoint.status() == Some("available"));

            if instances_available && endpoints_available {
                return Ok(());
            }
        }

        Err(ScenarioError::with("timed out waiting for cluster"))
    }

    pub async fn create_db_cluster(
        &self,
        name: &str,
        parameter_group: &str,
        engine: &str,
        version: &str,
        username: &str,
        password: SecretString,
    ) -> Result<CreateDbClusterOutput, SdkError<CreateDBClusterError>> {
        self.inner
            .create_db_cluster()
            .db_cluster_identifier(name)
            .db_cluster_parameter_group_name(parameter_group)
            .engine(engine)
            .engine_version(version)
            .master_username(username)
            .master_user_password(password.expose_secret())
            .send()
            .await
    }

#[tokio::test]
async fn test_start_cluster_and_instance() {
    let mut mock_rds = MockRdsImpl::default();

    mock_rds
        .expect_create_db_cluster()
        .withf(|id, params, engine, version, username, password| {
            assert_eq!(id, "RustSDKCodeExamplesDBCluster");
            assert_eq!(params, "RustSDKCodeExamplesDBParameterGroup");
            assert_eq!(engine, "aurora-mysql");
            assert_eq!(version, "aurora-mysql8.0");
            assert_eq!(username, "test username");
```

```
            assert_eq!(password.expose_secret(), "test password");
            true
        })
        .return_once(|id, _, _, _, _, _| {
            Ok(CreateDbClusterOutput::builder()

.db_cluster(DbCluster::builder().db_cluster_identifier(id).build())
                .build())
        });

    mock_rds
        .expect_create_db_instance()
        .withf(|cluster, name, class, engine| {
            assert_eq!(cluster, "RustSDKCodeExamplesDBCluster");
            assert_eq!(name, "RustSDKCodeExamplesDBInstance");
            assert_eq!(class, "m5.large");
            assert_eq!(engine, "aurora-mysql");
            true
        })
        .return_once(|cluster, name, class, _| {
            Ok(CreateDbInstanceOutput::builder()
                .db_instance(
                    DbInstance::builder()
                        .db_cluster_identifier(cluster)
                        .db_instance_identifier(name)
                        .db_instance_class(class)
                        .build(),
                )
                .build())
        });

    mock_rds
        .expect_describe_db_clusters()
        .with(eq("RustSDKCodeExamplesDBCluster"))
        .return_once(|id| {
            Ok(DescribeDbClustersOutput::builder()

.db_clusters(DbCluster::builder().db_cluster_identifier(id).build())
                .build())
        });

    mock_rds
        .expect_describe_db_instance()
        .with(eq("RustSDKCodeExamplesDBInstance"))
```

```
        .return_once(|name| {
            Ok(DescribeDbInstancesOutput::builder()
                .db_instances(
                    DbInstance::builder()
                        .db_instance_identifier(name)
                        .db_instance_status("Available")
                        .build(),
                )
                .build())
        });

    mock_rds
        .expect_describe_db_cluster_endpoints()
        .with(eq("RustSDKCodeExamplesDBCluster"))
        .return_once(|_| {
            Ok(DescribeDbClusterEndpointsOutput::builder()

.db_cluster_endpoints(DbClusterEndpoint::builder().status("available").build())
                .build())
        });

    let mut scenario = AuroraScenario::new(mock_rds);
    scenario.engine_version = Some("aurora-mysql8.0".into());
    scenario.instance_class = Some("m5.large".into());
    scenario.username = Some("test username".into());
    scenario.password = Some(SecretString::new("test password".into()));

    tokio::time::pause();
    let assertions = tokio::spawn(async move {
        let create = scenario.start_cluster_and_instance().await;
        assert!(create.is_ok());
        assert!(scenario
            .password
            .replace(SecretString::new("BAD SECRET".into()))
            .unwrap()
            .expose_secret()
            .is_empty());
        assert_eq!(
            scenario.db_cluster_identifier,
            Some("RustSDKCodeExamplesDBCluster".into())
        );
    });
    tokio::time::advance(Duration::from_secs(1)).await;
    tokio::time::resume();
```

```
    let _ = assertions.await;
}

#[tokio::test]
async fn test_start_cluster_and_instance_cluster_create_error() {
    let mut mock_rds = MockRdsImpl::default();

    mock_rds
        .expect_create_db_cluster()
        .return_once(|_, _, _, _, _, _| {
            Err(SdkError::service_error(
                CreateDBClusterError::unhandled(Box::new(Error::new(
                    ErrorKind::Other,
                    "create db cluster error",
                ))),
                Response::new(StatusCode::try_from(400).unwrap(),
 SdkBody::empty()),
            ))
        });

    let mut scenario = AuroraScenario::new(mock_rds);
    scenario.engine_version = Some("aurora-mysql8.0".into());
    scenario.instance_class = Some("m5.large".into());
    scenario.username = Some("test username".into());
    scenario.password = Some(SecretString::new("test password".into()));

    let create = scenario.start_cluster_and_instance().await;
    assert_matches!(create, Err(ScenarioError { message, context: _}) if message
 == "Failed to create DB Cluster with cluster group")
}

#[tokio::test]
async fn test_start_cluster_and_instance_cluster_create_missing_id() {
    let mut mock_rds = MockRdsImpl::default();

    mock_rds
        .expect_create_db_cluster()
        .return_once(|_, _, _, _, _, _| {
            Ok(CreateDbClusterOutput::builder()
                .db_cluster(DbCluster::builder().build())
                .build())
        });

    let mut scenario = AuroraScenario::new(mock_rds);
```

```
    scenario.engine_version = Some("aurora-mysql8.0".into());
    scenario.instance_class = Some("m5.large".into());
    scenario.username = Some("test username".into());
    scenario.password = Some(SecretString::new("test password".into()));

    let create = scenario.start_cluster_and_instance().await;
    assert_matches!(create, Err(ScenarioError { message, context:_ }) if message
 == "Created DB Cluster missing Identifier");
}

#[tokio::test]
async fn test_start_cluster_and_instance_instance_create_error() {
    let mut mock_rds = MockRdsImpl::default();

    mock_rds
        .expect_create_db_cluster()
        .withf(|id, params, engine, version, username, password| {
            assert_eq!(id, "RustSDKCodeExamplesDBCluster");
            assert_eq!(params, "RustSDKCodeExamplesDBParameterGroup");
            assert_eq!(engine, "aurora-mysql");
            assert_eq!(version, "aurora-mysql8.0");
            assert_eq!(username, "test username");
            assert_eq!(password.expose_secret(), "test password");
            true
        })
        .return_once(|id, _, _, _, _, _| {
            Ok(CreateDbClusterOutput::builder()

 .db_cluster(DbCluster::builder().db_cluster_identifier(id).build())
                .build())
        });

    mock_rds
        .expect_create_db_instance()
        .return_once(|_, _, _, _| {
            Err(SdkError::service_error(
                CreateDBInstanceError::unhandled(Box::new(Error::new(
                    ErrorKind::Other,
                    "create db instance error",
                ))),
                Response::new(StatusCode::try_from(400).unwrap(),
 SdkBody::empty()),
            ))
        });
```

```
    let mut scenario = AuroraScenario::new(mock_rds);
    scenario.engine_version = Some("aurora-mysql8.0".into());
    scenario.instance_class = Some("m5.large".into());
    scenario.username = Some("test username".into());
    scenario.password = Some(SecretString::new("test password".into()));

    let create = scenario.start_cluster_and_instance().await;
    assert_matches!(create, Err(ScenarioError { message, context: _ }) if message
 == "Failed to create Instance in DB Cluster")
}

#[tokio::test]
async fn test_start_cluster_and_instance_wait_hiccup() {
    let mut mock_rds = MockRdsImpl::default();

    mock_rds
        .expect_create_db_cluster()
        .withf(|id, params, engine, version, username, password| {
            assert_eq!(id, "RustSDKCodeExamplesDBCluster");
            assert_eq!(params, "RustSDKCodeExamplesDBParameterGroup");
            assert_eq!(engine, "aurora-mysql");
            assert_eq!(version, "aurora-mysql8.0");
            assert_eq!(username, "test username");
            assert_eq!(password.expose_secret(), "test password");
            true
        })
        .return_once(|id, _, _, _, _, _| {
            Ok(CreateDbClusterOutput::builder()

 .db_cluster(DbCluster::builder().db_cluster_identifier(id).build())
                .build())
        });

    mock_rds
        .expect_create_db_instance()
        .withf(|cluster, name, class, engine| {
            assert_eq!(cluster, "RustSDKCodeExamplesDBCluster");
            assert_eq!(name, "RustSDKCodeExamplesDBInstance");
            assert_eq!(class, "m5.large");
            assert_eq!(engine, "aurora-mysql");
            true
        })
        .return_once(|cluster, name, class, _| {
```

```
            Ok(CreateDbInstanceOutput::builder()
                .db_instance(
                    DbInstance::builder()
                        .db_cluster_identifier(cluster)
                        .db_instance_identifier(name)
                        .db_instance_class(class)
                        .build(),
                )
                .build())
        });

    mock_rds
        .expect_describe_db_clusters()
        .with(eq("RustSDKCodeExamplesDBCluster"))
        .times(1)
        .returning(|_| {
            Err(SdkError::service_error(
                DescribeDBClustersError::unhandled(Box::new(Error::new(
                    ErrorKind::Other,
                    "describe cluster error",
                ))),
                Response::new(StatusCode::try_from(400).unwrap(),
SdkBody::empty()),
            ))
        })
        .with(eq("RustSDKCodeExamplesDBCluster"))
        .times(1)
        .returning(|id| {
            Ok(DescribeDbClustersOutput::builder()

.db_clusters(DbCluster::builder().db_cluster_identifier(id).build())
                .build())
        });

    mock_rds.expect_describe_db_instance().return_once(|name| {
        Ok(DescribeDbInstancesOutput::builder()
            .db_instances(
                DbInstance::builder()
                    .db_instance_identifier(name)
                    .db_instance_status("Available")
                    .build(),
            )
            .build())
    });
```

```
    mock_rds
        .expect_describe_db_cluster_endpoints()
        .return_once(|_| {
            Ok(DescribeDbClusterEndpointsOutput::builder()

 .db_cluster_endpoints(DbClusterEndpoint::builder().status("available").build())
                .build())
        });

    let mut scenario = AuroraScenario::new(mock_rds);
    scenario.engine_version = Some("aurora-mysql8.0".into());
    scenario.instance_class = Some("m5.large".into());
    scenario.username = Some("test username".into());
    scenario.password = Some(SecretString::new("test password".into()));

    tokio::time::pause();
    let assertions = tokio::spawn(async move {
        let create = scenario.start_cluster_and_instance().await;
        assert!(create.is_ok());
    });

    tokio::time::advance(Duration::from_secs(1)).await;
    tokio::time::advance(Duration::from_secs(1)).await;
    tokio::time::resume();
    let _ = assertions.await;
}
```

- Per i dettagli sull'API, consulta DBClusterCreate in AWSSDK for Rust API reference.

Per un elenco completo delle guide per sviluppatori AWS SDK e degli esempi di codice, consulta. Utilizzo di questo servizio con un AWS SDK Questo argomento include anche informazioni su come iniziare e dettagli sulle versioni precedenti dell'SDK.

Utilizzo **`CreateDBClusterParameterGroup`** con un AWS SDK

Gli esempi di codice seguenti mostrano come utilizzare `CreateDBClusterParameterGroup`.

Gli esempi di operazioni sono estratti di codice da programmi più grandi e devono essere eseguiti nel contesto. È possibile visualizzare questa operazione nel contesto nel seguente esempio di codice:

- Informazioni di base

.NET

SDK per .NET(v4)

> **Note**
>
> C'è altro da fare. GitHub Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice AWS.

```
    /// <summary>
    /// Create a custom cluster parameter group.
    /// </summary>
    /// <param name="parameterGroupFamily">The family of the parameter group.</
param>
    /// <param name="groupName">The name for the new parameter group.</param>
    /// <param name="description">A description for the new parameter group.</
param>
    /// <returns>The new parameter group object.</returns>
    public async Task<DBClusterParameterGroup>
 CreateCustomClusterParameterGroupAsync(
        string parameterGroupFamily,
        string groupName,
        string description)
    {
        var request = new CreateDBClusterParameterGroupRequest
        {
            DBParameterGroupFamily = parameterGroupFamily,
            DBClusterParameterGroupName = groupName,
            Description = description,
        };

        var response = await
 _amazonRDS.CreateDBClusterParameterGroupAsync(request);
        return response.DBClusterParameterGroup;
    }
```

- Per i dettagli sull'API, consulta Create DBCluster ParameterGroup in AWS SDK per .NETAPI Reference.

C++

SDK per C++

> **Note**
>
> C'è altro su GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice AWS.

```
        Aws::Client::ClientConfiguration clientConfig;
        // Optional: Set to the AWS Region (overrides config file).
        // clientConfig.region = "us-east-1";

    Aws::RDS::RDSClient client(clientConfig);

        Aws::RDS::Model::CreateDBClusterParameterGroupRequest request;
        request.SetDBClusterParameterGroupName(CLUSTER_PARAMETER_GROUP_NAME);
        request.SetDBParameterGroupFamily(dbParameterGroupFamily);
        request.SetDescription("Example cluster parameter group.");

        Aws::RDS::Model::CreateDBClusterParameterGroupOutcome outcome =
                client.CreateDBClusterParameterGroup(request);

        if (outcome.IsSuccess()) {
            std::cout << "The DB cluster parameter group was successfully
created."
                      << std::endl;
        }
        else {
            std::cerr << "Error with Aurora::CreateDBClusterParameterGroup. "
                      << outcome.GetError().GetMessage()
                      << std::endl;
            return false;
        }
```

- Per i dettagli sull'API, consulta Create DBCluster ParameterGroup in AWS SDK per C++API Reference.

Go

SDK per Go V2

> **Note**
>
> C'è altro su GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice AWS.

```
import (
 "context"
 "errors"
 "log"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/service/rds"
 "github.com/aws/aws-sdk-go-v2/service/rds/types"
)

type DbClusters struct {
 AuroraClient *rds.Client
}



// CreateParameterGroup creates a DB cluster parameter group that is based on the
 specified
// parameter group family.
func (clusters *DbClusters) CreateParameterGroup(
 ctx context.Context, parameterGroupName string, parameterGroupFamily string,
 description string) (
 *types.DBClusterParameterGroup, error) {

 output, err := clusters.AuroraClient.CreateDBClusterParameterGroup(ctx,
  &rds.CreateDBClusterParameterGroupInput{
   DBClusterParameterGroupName: aws.String(parameterGroupName),
```

```
    DBParameterGroupFamily:      aws.String(parameterGroupFamily),
    Description:                 aws.String(description),
   })
  if err != nil {
   log.Printf("Couldn't create parameter group %v: %v\n", parameterGroupName, err)
   return nil, err
  } else {
   return output.DBClusterParameterGroup, err
  }
 }
```

- Per i dettagli sull'API, consulta Create DBCluster ParameterGroup in AWS SDK per GoAPI Reference.

Java

SDK per Java 2.x

> **Note**
>
> C'è altro su GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice AWS.

```
    public static void createDBClusterParameterGroup(RdsClient rdsClient, String
dbClusterGroupName,
            String dbParameterGroupFamily) {
        try {
            CreateDbClusterParameterGroupRequest groupRequest =
CreateDbClusterParameterGroupRequest.builder()
                    .dbClusterParameterGroupName(dbClusterGroupName)
                    .dbParameterGroupFamily(dbParameterGroupFamily)
                    .description("Created by using the AWS SDK for Java")
                    .build();

            CreateDbClusterParameterGroupResponse response =
rdsClient.createDBClusterParameterGroup(groupRequest);
            System.out.println("The group name is " +
response.dbClusterParameterGroup().dbClusterParameterGroupName());
```

```
        } catch (RdsException e) {
            System.out.println(e.getLocalizedMessage());
            System.exit(1);
        }
    }
```

- Per i dettagli sull'API, consulta Create DBCluster ParameterGroup in AWS SDK for Java 2.xAPI Reference.

Kotlin

SDK per Kotlin

> **Note**
>
> C'è altro su GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice AWS.

```
suspend fun createDBClusterParameterGroup(
    dbClusterGroupNameVal: String?,
    dbParameterGroupFamilyVal: String?,
) {
    val groupRequest =
        CreateDbClusterParameterGroupRequest {
            dbClusterParameterGroupName = dbClusterGroupNameVal
            dbParameterGroupFamily = dbParameterGroupFamilyVal
            description = "Created by using the AWS SDK for Kotlin"
        }

    RdsClient.fromEnvironment { region = "us-west-2" }.use { rdsClient ->
        val response = rdsClient.createDbClusterParameterGroup(groupRequest)
        println("The group name is
 ${response.dbClusterParameterGroup?.dbClusterParameterGroupName}")
    }
}
```

- Per i dettagli sull'API, consulta Create DBCluster ParameterGroup in AWSSDK for Kotlin API reference.

Python

SDK per Python (Boto3)

> **Note**
>
> C'è altro su. GitHub Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice AWS.

```
class AuroraWrapper:
    """Encapsulates Aurora DB cluster actions."""

    def __init__(self, rds_client):
        """
        :param rds_client: A Boto3 Amazon Relational Database Service (Amazon
 RDS) client.
        """
        self.rds_client = rds_client

    @classmethod
    def from_client(cls):
        """
        Instantiates this class from a Boto3 client.
        """
        rds_client = boto3.client("rds")
        return cls(rds_client)


    def create_parameter_group(
        self, parameter_group_name, parameter_group_family, description
    ):
        """
        Creates a DB cluster parameter group that is based on the specified
 parameter group
        family.
```

```
        :param parameter_group_name: The name of the newly created parameter
group.
        :param parameter_group_family: The family that is used as the basis of
the new
                                       parameter group.
        :param description: A description given to the parameter group.
        :return: Data about the newly created parameter group.
        """
        try:
            response = self.rds_client.create_db_cluster_parameter_group(
                DBClusterParameterGroupName=parameter_group_name,
                DBParameterGroupFamily=parameter_group_family,
                Description=description,
            )
        except ClientError as err:
            logger.error(
                "Couldn't create parameter group %s. Here's why: %s: %s",
                parameter_group_name,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
        else:
            return response
```

- Per i dettagli sull'API, consulta Create DBCluster ParameterGroup in AWSSDK for Python (Boto3) API Reference.

Rust

SDK per Rust

> **Note**
>
> C'è altro da fare. GitHub Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice AWS.

```
   // Select an engine family and create a custom DB cluster parameter group.
rds.CreateDbClusterParameterGroup(DBParameterGroupFamily='aurora-mysql8.0')
   pub async fn set_engine(&mut self, engine: &str, version: &str) -> Result<(),
ScenarioError> {
       self.engine_family = Some(engine.to_string());
       self.engine_version = Some(version.to_string());
       let create_db_cluster_parameter_group = self
           .rds
           .create_db_cluster_parameter_group(
               DB_CLUSTER_PARAMETER_GROUP_NAME,
               DB_CLUSTER_PARAMETER_GROUP_DESCRIPTION,
               engine,
           )
           .await;

       match create_db_cluster_parameter_group {
           Ok(CreateDbClusterParameterGroupOutput {
               db_cluster_parameter_group: None,
               ..
           }) => {
               return Err(ScenarioError::with(
                   "CreateDBClusterParameterGroup had empty response",
               ));
           }
           Err(error) => {
               if error.code() == Some("DBParameterGroupAlreadyExists") {
                   info!("Cluster Parameter Group already exists, nothing to
do");
               } else {
                   return Err(ScenarioError::new(
                       "Could not create Cluster Parameter Group",
                       &error,
                   ));
               }
           }
           _ => {
               info!("Created Cluster Parameter Group");
           }
       }

       Ok(())
   }
```

```
    pub async fn create_db_cluster_parameter_group(
        &self,
        name: &str,
        description: &str,
        family: &str,
    ) -> Result<CreateDbClusterParameterGroupOutput,
 SdkError<CreateDBClusterParameterGroupError>>
    {
        self.inner
            .create_db_cluster_parameter_group()
            .db_cluster_parameter_group_name(name)
            .description(description)
            .db_parameter_group_family(family)
            .send()
            .await
    }

#[tokio::test]
async fn test_scenario_set_engine() {
    let mut mock_rds = MockRdsImpl::default();

    mock_rds
        .expect_create_db_cluster_parameter_group()
        .with(
            eq("RustSDKCodeExamplesDBParameterGroup"),
            eq("Parameter Group created by Rust SDK Code Example"),
            eq("aurora-mysql"),
        )
        .return_once(|_, _, _| {
            Ok(CreateDbClusterParameterGroupOutput::builder()

 .db_cluster_parameter_group(DbClusterParameterGroup::builder().build())
                .build())
        });

    let mut scenario = AuroraScenario::new(mock_rds);

    let set_engine = scenario.set_engine("aurora-mysql", "aurora-
mysql8.0").await;

    assert_eq!(set_engine, Ok(()));
    assert_eq!(Some("aurora-mysql"), scenario.engine_family.as_deref());
    assert_eq!(Some("aurora-mysql8.0"), scenario.engine_version.as_deref());
}
```

```
#[tokio::test]
async fn test_scenario_set_engine_not_create() {
    let mut mock_rds = MockRdsImpl::default();

    mock_rds
        .expect_create_db_cluster_parameter_group()
        .with(
            eq("RustSDKCodeExamplesDBParameterGroup"),
            eq("Parameter Group created by Rust SDK Code Example"),
            eq("aurora-mysql"),
        )
        .return_once(|_, _, _|
 Ok(CreateDbClusterParameterGroupOutput::builder().build()));

    let mut scenario = AuroraScenario::new(mock_rds);

    let set_engine = scenario.set_engine("aurora-mysql", "aurora-
mysql8.0").await;

    assert!(set_engine.is_err());
}

#[tokio::test]
async fn test_scenario_set_engine_param_group_exists() {
    let mut mock_rds = MockRdsImpl::default();

    mock_rds
        .expect_create_db_cluster_parameter_group()
        .withf(|_, _, _| true)
        .return_once(|_, _, _| {
            Err(SdkError::service_error(

 CreateDBClusterParameterGroupError::DbParameterGroupAlreadyExistsFault(
                    DbParameterGroupAlreadyExistsFault::builder().build(),
                ),
                Response::new(StatusCode::try_from(400).unwrap(),
 SdkBody::empty()),
            ))
        });

    let mut scenario = AuroraScenario::new(mock_rds);
```

```
    let set_engine = scenario.set_engine("aurora-mysql", "aurora-
mysql8.0").await;

    assert!(set_engine.is_err());
}
```

- Per i dettagli sull'API, consulta DBClusterParameterGroupCreate in AWSSDK for Rust API reference.

Per un elenco completo delle guide per sviluppatori AWS SDK e degli esempi di codice, consulta. Utilizzo di questo servizio con un AWS SDK Questo argomento include anche informazioni su come iniziare e dettagli sulle versioni precedenti dell'SDK.

Utilizzo **`CreateDBClusterSnapshot`** con un AWS SDK

Gli esempi di codice seguenti mostrano come utilizzare `CreateDBClusterSnapshot`.

Gli esempi di operazioni sono estratti di codice da programmi più grandi e devono essere eseguiti nel contesto. È possibile visualizzare questa operazione nel contesto nel seguente esempio di codice:

- Informazioni di base

.NET

SDK per .NET(v4)

> **Note**
>
> C'è altro da fare. GitHub Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice AWS.

```
    /// <summary>
    /// Create a snapshot of a cluster.
    /// </summary>
    /// <param name="dbClusterIdentifier">DB cluster identifier.</param>
    /// <param name="snapshotIdentifier">Identifier for the snapshot.</param>
    /// <returns>DB snapshot object.</returns>
```

```
    public async Task<DBClusterSnapshot>
CreateClusterSnapshotByIdentifierAsync(string dbClusterIdentifier, string
snapshotIdentifier)
    {
        var response = await _amazonRDS.CreateDBClusterSnapshotAsync(
            new CreateDBClusterSnapshotRequest()
            {
                DBClusterIdentifier = dbClusterIdentifier,
                DBClusterSnapshotIdentifier = snapshotIdentifier,
            });

        return response.DBClusterSnapshot;
    }
```

- Per i dettagli sull'API, consulta Create DBCluster Snapshot in AWS SDK per .NETAPI Reference.

C++

SDK per C++

> **Note**
>
> C'è di più su. GitHub Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice AWS.

```
        Aws::Client::ClientConfiguration clientConfig;
        // Optional: Set to the AWS Region (overrides config file).
        // clientConfig.region = "us-east-1";

    Aws::RDS::RDSClient client(clientConfig);

            Aws::RDS::Model::CreateDBClusterSnapshotRequest request;
            request.SetDBClusterIdentifier(DB_CLUSTER_IDENTIFIER);
            request.SetDBClusterSnapshotIdentifier(snapshotID);

            Aws::RDS::Model::CreateDBClusterSnapshotOutcome outcome =
                    client.CreateDBClusterSnapshot(request);
```

```
            if (outcome.IsSuccess()) {
                std::cout << "Snapshot creation has started."
                          << std::endl;
            }
            else {
                std::cerr << "Error with Aurora::CreateDBClusterSnapshot. "
                          << outcome.GetError().GetMessage()
                          << std::endl;
                cleanUpResources(CLUSTER_PARAMETER_GROUP_NAME,
                                 DB_CLUSTER_IDENTIFIER, DB_INSTANCE_IDENTIFIER,
 client);
                return false;
            }
```

- Per i dettagli sull'API, consulta Create DBCluster Snapshot in AWS SDK per C++API Reference.

Go

SDK per Go V2

> **Note**
>
> C'è di più su. GitHub Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice AWS.

```
import (
 "context"
 "errors"
 "log"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/service/rds"
 "github.com/aws/aws-sdk-go-v2/service/rds/types"
)

type DbClusters struct {
 AuroraClient *rds.Client
```

```
}



// CreateClusterSnapshot creates a snapshot of a DB cluster.
func (clusters *DbClusters) CreateClusterSnapshot(ctx context.Context,
 clusterName string, snapshotName string) (
 *types.DBClusterSnapshot, error) {
 output, err := clusters.AuroraClient.CreateDBClusterSnapshot(ctx,
 &rds.CreateDBClusterSnapshotInput{
  DBClusterIdentifier:         aws.String(clusterName),
  DBClusterSnapshotIdentifier: aws.String(snapshotName),
 })
 if err != nil {
  log.Printf("Couldn't create snapshot %v: %v\n", snapshotName, err)
  return nil, err
 } else {
  return output.DBClusterSnapshot, nil
 }
}
```

- Per i dettagli sull'API, consulta Create DBCluster Snapshot in AWS SDK per GoAPI Reference.

Java

SDK per Java 2.x

> **Note**
>
> C'è di più su. GitHub Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice AWS.

```
   public static void createDBClusterSnapshot(RdsClient rdsClient, String
dbInstanceClusterIdentifier,
           String dbSnapshotIdentifier) {
       try {
```

```
            CreateDbClusterSnapshotRequest snapshotRequest =
CreateDbClusterSnapshotRequest.builder()
                    .dbClusterIdentifier(dbInstanceClusterIdentifier)
                    .dbClusterSnapshotIdentifier(dbSnapshotIdentifier)
                    .build();

            CreateDbClusterSnapshotResponse response =
rdsClient.createDBClusterSnapshot(snapshotRequest);
            System.out.println("The Snapshot ARN is " +
response.dbClusterSnapshot().dbClusterSnapshotArn());

        } catch (RdsException e) {
            System.out.println(e.getLocalizedMessage());
            System.exit(1);
        }
    }
```

- Per i dettagli sull'API, consulta Create DBCluster Snapshot in AWS SDK for Java 2.xAPI Reference.

Kotlin

SDK per Kotlin

> **Note**
>
> C'è di più su. GitHub Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice AWS.

```
suspend fun createDBClusterSnapshot(
    dbInstanceClusterIdentifier: String?,
    dbSnapshotIdentifier: String?,
) {
    val snapshotRequest =
        CreateDbClusterSnapshotRequest {
            dbClusterIdentifier = dbInstanceClusterIdentifier
            dbClusterSnapshotIdentifier = dbSnapshotIdentifier
        }
```

```
    RdsClient.fromEnvironment { region = "us-west-2" }.use { rdsClient ->
        val response = rdsClient.createDbClusterSnapshot(snapshotRequest)
        println("The Snapshot ARN is
 ${response.dbClusterSnapshot?.dbClusterSnapshotArn}")
    }
}
```

- Per i dettagli sull'API, consulta Create DBCluster Snapshot in AWSSDK per il riferimento all'API Kotlin.

Python

SDK per Python (Boto3)

> **Note**
>
> C'è di più su. GitHub Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice AWS.

```
class AuroraWrapper:
    """Encapsulates Aurora DB cluster actions."""

    def __init__(self, rds_client):
        """
        :param rds_client: A Boto3 Amazon Relational Database Service (Amazon
 RDS) client.
        """
        self.rds_client = rds_client

    @classmethod
    def from_client(cls):
        """
        Instantiates this class from a Boto3 client.
        """
        rds_client = boto3.client("rds")
        return cls(rds_client)


    def create_cluster_snapshot(self, snapshot_id, cluster_id):
```

```
        """
        Creates a snapshot of a DB cluster.

        :param snapshot_id: The ID to give the created snapshot.
        :param cluster_id: The DB cluster to snapshot.
        :return: Data about the newly created snapshot.
        """
        try:
            response = self.rds_client.create_db_cluster_snapshot(
                DBClusterSnapshotIdentifier=snapshot_id,
DBClusterIdentifier=cluster_id
            )
            snapshot = response["DBClusterSnapshot"]
        except ClientError as err:
            logger.error(
                "Couldn't create snapshot of %s. Here's why: %s: %s",
                cluster_id,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
        else:
            return snapshot
```

- Per i dettagli sull'API, consulta Create DBCluster Snapshot in AWSSDK for Python (Boto3) API Reference.

Rust

SDK per Rust

> **Note**
>
> C'è altro da fare. GitHub Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice AWS.

```
   // Get a list of allowed engine versions.
rds.DescribeDbEngineVersions(Engine='aurora-mysql', DBParameterGroupFamily=<the
family used to create your parameter group in step 2>)
   // Create an Aurora DB cluster database cluster that contains a MySql
database and uses the parameter group you created.
   // Wait for DB cluster to be ready. Call rds.DescribeDBClusters and check for
Status == 'available'.
   // Get a list of instance classes available for the selected engine
and engine version. rds.DescribeOrderableDbInstanceOptions(Engine='mysql',
EngineVersion=).

   // Create a database instance in the cluster.
   // Wait for DB instance to be ready. Call rds.DescribeDbInstances and check
for DBInstanceStatus == 'available'.
   pub async fn start_cluster_and_instance(&mut self) -> Result<(),
ScenarioError> {
       if self.password.is_none() {
           return Err(ScenarioError::with(
               "Must set Secret Password before starting a cluster",
           ));
       }
       let create_db_cluster = self
           .rds
           .create_db_cluster(
               DB_CLUSTER_IDENTIFIER,
               DB_CLUSTER_PARAMETER_GROUP_NAME,
               DB_ENGINE,
               self.engine_version.as_deref().expect("engine version"),
               self.username.as_deref().expect("username"),
               self.password
                   .replace(SecretString::new("".to_string()))
                   .expect("password"),
           )
           .await;
       if let Err(err) = create_db_cluster {
           return Err(ScenarioError::new(
               "Failed to create DB Cluster with cluster group",
               &err,
           ));
       }

       self.db_cluster_identifier = create_db_cluster
           .unwrap()
```

```
            .db_cluster
            .and_then(|c| c.db_cluster_identifier);

        if self.db_cluster_identifier.is_none() {
            return Err(ScenarioError::with("Created DB Cluster missing
Identifier"));
        }

        info!(
            "Started a db cluster: {}",
            self.db_cluster_identifier
                .as_deref()
                .unwrap_or("Missing ARN")
        );

        let create_db_instance = self
            .rds
            .create_db_instance(
                self.db_cluster_identifier.as_deref().expect("cluster name"),
                DB_INSTANCE_IDENTIFIER,
                self.instance_class.as_deref().expect("instance class"),
                DB_ENGINE,
            )
            .await;
        if let Err(err) = create_db_instance {
            return Err(ScenarioError::new(
                "Failed to create Instance in DB Cluster",
                &err,
            ));
        }

        self.db_instance_identifier = create_db_instance
            .unwrap()
            .db_instance
            .and_then(|i| i.db_instance_identifier);

        // Cluster creation can take up to 20 minutes to become available
        let cluster_max_wait = Duration::from_secs(20 * 60);
        let waiter = Waiter::builder().max(cluster_max_wait).build();
        while waiter.sleep().await.is_ok() {
            let cluster = self
                .rds
                .describe_db_clusters(
                    self.db_cluster_identifier
```

```
                        .as_deref()
                        .expect("cluster identifier"),
                )
                .await;

            if let Err(err) = cluster {
                warn!(?err, "Failed to describe cluster while waiting for
ready");
                continue;
            }

            let instance = self
                .rds
                .describe_db_instance(
                    self.db_instance_identifier
                        .as_deref()
                        .expect("instance identifier"),
                )
                .await;
            if let Err(err) = instance {
                return Err(ScenarioError::new(
                    "Failed to find instance for cluster",
                    &err,
                ));
            }

            let instances_available = instance
                .unwrap()
                .db_instances()
                .iter()
                .all(|instance| instance.db_instance_status() ==
Some("Available"));

            let endpoints = self
                .rds
                .describe_db_cluster_endpoints(
                    self.db_cluster_identifier
                        .as_deref()
                        .expect("cluster identifier"),
                )
                .await;

            if let Err(err) = endpoints {
                return Err(ScenarioError::new(
```

```
                    "Failed to find endpoint for cluster",
                    &err,
                ));
            }

            let endpoints_available = endpoints
                .unwrap()
                .db_cluster_endpoints()
                .iter()
                .all(|endpoint| endpoint.status() == Some("available"));

            if instances_available && endpoints_available {
                return Ok(());
            }
        }

        Err(ScenarioError::with("timed out waiting for cluster"))
    }

    pub async fn snapshot_cluster(
        &self,
        db_cluster_identifier: &str,
        snapshot_name: &str,
    ) -> Result<CreateDbClusterSnapshotOutput,
 SdkError<CreateDBClusterSnapshotError>> {
        self.inner
            .create_db_cluster_snapshot()
            .db_cluster_identifier(db_cluster_identifier)
            .db_cluster_snapshot_identifier(snapshot_name)
            .send()
            .await
    }

#[tokio::test]
async fn test_start_cluster_and_instance() {
    let mut mock_rds = MockRdsImpl::default();

    mock_rds
        .expect_create_db_cluster()
        .withf(|id, params, engine, version, username, password| {
            assert_eq!(id, "RustSDKCodeExamplesDBCluster");
            assert_eq!(params, "RustSDKCodeExamplesDBParameterGroup");
            assert_eq!(engine, "aurora-mysql");
            assert_eq!(version, "aurora-mysql8.0");
```

```
            assert_eq!(username, "test username");
            assert_eq!(password.expose_secret(), "test password");
            true
        })
        .return_once(|id, _, _, _, _, _| {
            Ok(CreateDbClusterOutput::builder()

.db_cluster(DbCluster::builder().db_cluster_identifier(id).build())
                .build())
        });

    mock_rds
        .expect_create_db_instance()
        .withf(|cluster, name, class, engine| {
            assert_eq!(cluster, "RustSDKCodeExamplesDBCluster");
            assert_eq!(name, "RustSDKCodeExamplesDBInstance");
            assert_eq!(class, "m5.large");
            assert_eq!(engine, "aurora-mysql");
            true
        })
        .return_once(|cluster, name, class, _| {
            Ok(CreateDbInstanceOutput::builder()
                .db_instance(
                    DbInstance::builder()
                        .db_cluster_identifier(cluster)
                        .db_instance_identifier(name)
                        .db_instance_class(class)
                        .build(),
                )
                .build())
        });

    mock_rds
        .expect_describe_db_clusters()
        .with(eq("RustSDKCodeExamplesDBCluster"))
        .return_once(|id| {
            Ok(DescribeDbClustersOutput::builder()

.db_clusters(DbCluster::builder().db_cluster_identifier(id).build())
                .build())
        });

    mock_rds
        .expect_describe_db_instance()
```

```
        .with(eq("RustSDKCodeExamplesDBInstance"))
        .return_once(|name| {
            Ok(DescribeDbInstancesOutput::builder()
                .db_instances(
                    DbInstance::builder()
                        .db_instance_identifier(name)
                        .db_instance_status("Available")
                        .build(),
                )
                .build())
        });

    mock_rds
        .expect_describe_db_cluster_endpoints()
        .with(eq("RustSDKCodeExamplesDBCluster"))
        .return_once(|_| {
            Ok(DescribeDbClusterEndpointsOutput::builder()

.db_cluster_endpoints(DbClusterEndpoint::builder().status("available").build())
                .build())
        });

    let mut scenario = AuroraScenario::new(mock_rds);
    scenario.engine_version = Some("aurora-mysql8.0".into());
    scenario.instance_class = Some("m5.large".into());
    scenario.username = Some("test username".into());
    scenario.password = Some(SecretString::new("test password".into()));

    tokio::time::pause();
    let assertions = tokio::spawn(async move {
        let create = scenario.start_cluster_and_instance().await;
        assert!(create.is_ok());
        assert!(scenario
            .password
            .replace(SecretString::new("BAD SECRET".into()))
            .unwrap()
            .expose_secret()
            .is_empty());
        assert_eq!(
            scenario.db_cluster_identifier,
            Some("RustSDKCodeExamplesDBCluster".into())
        );
    });
    tokio::time::advance(Duration::from_secs(1)).await;
```

```
    tokio::time::resume();
    let _ = assertions.await;
}

#[tokio::test]
async fn test_start_cluster_and_instance_cluster_create_error() {
    let mut mock_rds = MockRdsImpl::default();

    mock_rds
        .expect_create_db_cluster()
        .return_once(|_, _, _, _, _, _| {
            Err(SdkError::service_error(
                CreateDBClusterError::unhandled(Box::new(Error::new(
                    ErrorKind::Other,
                    "create db cluster error",
                ))),
                Response::new(StatusCode::try_from(400).unwrap(),
 SdkBody::empty()),
            ))
        });

    let mut scenario = AuroraScenario::new(mock_rds);
    scenario.engine_version = Some("aurora-mysql8.0".into());
    scenario.instance_class = Some("m5.large".into());
    scenario.username = Some("test username".into());
    scenario.password = Some(SecretString::new("test password".into()));

    let create = scenario.start_cluster_and_instance().await;
    assert_matches!(create, Err(ScenarioError { message, context: _}) if message
 == "Failed to create DB Cluster with cluster group")
}

#[tokio::test]
async fn test_start_cluster_and_instance_cluster_create_missing_id() {
    let mut mock_rds = MockRdsImpl::default();

    mock_rds
        .expect_create_db_cluster()
        .return_once(|_, _, _, _, _, _| {
            Ok(CreateDbClusterOutput::builder()
                .db_cluster(DbCluster::builder().build())
                .build())
        });
```

```
    let mut scenario = AuroraScenario::new(mock_rds);
    scenario.engine_version = Some("aurora-mysql8.0".into());
    scenario.instance_class = Some("m5.large".into());
    scenario.username = Some("test username".into());
    scenario.password = Some(SecretString::new("test password".into()));

    let create = scenario.start_cluster_and_instance().await;
    assert_matches!(create, Err(ScenarioError { message, context:_ }) if message
 == "Created DB Cluster missing Identifier");
}

#[tokio::test]
async fn test_start_cluster_and_instance_instance_create_error() {
    let mut mock_rds = MockRdsImpl::default();

    mock_rds
        .expect_create_db_cluster()
        .withf(|id, params, engine, version, username, password| {
            assert_eq!(id, "RustSDKCodeExamplesDBCluster");
            assert_eq!(params, "RustSDKCodeExamplesDBParameterGroup");
            assert_eq!(engine, "aurora-mysql");
            assert_eq!(version, "aurora-mysql8.0");
            assert_eq!(username, "test username");
            assert_eq!(password.expose_secret(), "test password");
            true
        })
        .return_once(|id, _, _, _, _, _| {
            Ok(CreateDbClusterOutput::builder()

 .db_cluster(DbCluster::builder().db_cluster_identifier(id).build())
                .build())
        });

    mock_rds
        .expect_create_db_instance()
        .return_once(|_, _, _, _| {
            Err(SdkError::service_error(
                CreateDBInstanceError::unhandled(Box::new(Error::new(
                    ErrorKind::Other,
                    "create db instance error",
                ))),
                Response::new(StatusCode::try_from(400).unwrap(),
 SdkBody::empty()),
            ))
```

```
        });

    let mut scenario = AuroraScenario::new(mock_rds);
    scenario.engine_version = Some("aurora-mysql8.0".into());
    scenario.instance_class = Some("m5.large".into());
    scenario.username = Some("test username".into());
    scenario.password = Some(SecretString::new("test password".into()));

    let create = scenario.start_cluster_and_instance().await;
    assert_matches!(create, Err(ScenarioError { message, context: _ }) if message
 == "Failed to create Instance in DB Cluster")
}

#[tokio::test]
async fn test_start_cluster_and_instance_wait_hiccup() {
    let mut mock_rds = MockRdsImpl::default();

    mock_rds
        .expect_create_db_cluster()
        .withf(|id, params, engine, version, username, password| {
            assert_eq!(id, "RustSDKCodeExamplesDBCluster");
            assert_eq!(params, "RustSDKCodeExamplesDBParameterGroup");
            assert_eq!(engine, "aurora-mysql");
            assert_eq!(version, "aurora-mysql8.0");
            assert_eq!(username, "test username");
            assert_eq!(password.expose_secret(), "test password");
            true
        })
        .return_once(|id, _, _, _, _, _| {
            Ok(CreateDbClusterOutput::builder()

 .db_cluster(DbCluster::builder().db_cluster_identifier(id).build())
                .build())
        });

    mock_rds
        .expect_create_db_instance()
        .withf(|cluster, name, class, engine| {
            assert_eq!(cluster, "RustSDKCodeExamplesDBCluster");
            assert_eq!(name, "RustSDKCodeExamplesDBInstance");
            assert_eq!(class, "m5.large");
            assert_eq!(engine, "aurora-mysql");
            true
        })
```

```
        .return_once(|cluster, name, class, _| {
            Ok(CreateDbInstanceOutput::builder()
                .db_instance(
                    DbInstance::builder()
                        .db_cluster_identifier(cluster)
                        .db_instance_identifier(name)
                        .db_instance_class(class)
                        .build(),
                )
                .build())
        });

    mock_rds
        .expect_describe_db_clusters()
        .with(eq("RustSDKCodeExamplesDBCluster"))
        .times(1)
        .returning(|_| {
            Err(SdkError::service_error(
                DescribeDBClustersError::unhandled(Box::new(Error::new(
                    ErrorKind::Other,
                    "describe cluster error",
                ))),
                Response::new(StatusCode::try_from(400).unwrap(),
SdkBody::empty()),
            ))
        })
        .with(eq("RustSDKCodeExamplesDBCluster"))
        .times(1)
        .returning(|id| {
            Ok(DescribeDbClustersOutput::builder()

.db_clusters(DbCluster::builder().db_cluster_identifier(id).build())
                .build())
        });

    mock_rds.expect_describe_db_instance().return_once(|name| {
        Ok(DescribeDbInstancesOutput::builder()
            .db_instances(
                DbInstance::builder()
                    .db_instance_identifier(name)
                    .db_instance_status("Available")
                    .build(),
            )
            .build())
```

```
    });

    mock_rds
        .expect_describe_db_cluster_endpoints()
        .return_once(|_| {
            Ok(DescribeDbClusterEndpointsOutput::builder()

 .db_cluster_endpoints(DbClusterEndpoint::builder().status("available").build())
                .build())
        });

    let mut scenario = AuroraScenario::new(mock_rds);
    scenario.engine_version = Some("aurora-mysql8.0".into());
    scenario.instance_class = Some("m5.large".into());
    scenario.username = Some("test username".into());
    scenario.password = Some(SecretString::new("test password".into()));

    tokio::time::pause();
    let assertions = tokio::spawn(async move {
        let create = scenario.start_cluster_and_instance().await;
        assert!(create.is_ok());
    });

    tokio::time::advance(Duration::from_secs(1)).await;
    tokio::time::advance(Duration::from_secs(1)).await;
    tokio::time::resume();
    let _ = assertions.await;
}
```

- Per i dettagli sull'API, consulta Create DBCluster Snapshot in AWSSDK for Rust API reference.

Per un elenco completo delle guide per sviluppatori AWS SDK e degli esempi di codice, consulta. Utilizzo di questo servizio con un AWS SDK Questo argomento include anche informazioni su come iniziare e dettagli sulle versioni precedenti dell'SDK.

Utilizzo **`CreateDBInstance`** con un AWS SDK

Gli esempi di codice seguenti mostrano come utilizzare `CreateDBInstance`.

Gli esempi di operazioni sono estratti di codice da programmi più grandi e devono essere eseguiti nel contesto. È possibile visualizzare questa operazione nel contesto nel seguente esempio di codice:

- Informazioni di base

.NET

SDK per .NET(v4)

> **Note**
>
> C'è altro da fare. GitHub Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice AWS.

```
   /// <summary>
   /// Create an Amazon Relational Database Service (Amazon RDS) DB instance
   /// with a particular set of properties. Use the action
DescribeDBInstancesAsync
   /// to determine when the DB instance is ready to use.
   /// </summary>
   /// <param name="dbInstanceIdentifier">DB instance identifier.</param>
   /// <param name="dbClusterIdentifier">DB cluster identifier.</param>
   /// <param name="dbEngine">The engine for the DB instance.</param>
   /// <param name="dbEngineVersion">Version for the DB instance.</param>
   /// <param name="instanceClass">Class for the DB instance.</param>
   /// <returns>DB instance object.</returns>
   public async Task<DBInstance> CreateDBInstanceInClusterAsync(
       string dbClusterIdentifier,
       string dbInstanceIdentifier,
       string dbEngine,
       string dbEngineVersion,
       string instanceClass)
   {
       // When creating the instance within a cluster, do not specify the name
or size.
       var response = await _amazonRDS.CreateDBInstanceAsync(
           new CreateDBInstanceRequest()
           {
               DBClusterIdentifier = dbClusterIdentifier,
               DBInstanceIdentifier = dbInstanceIdentifier,
```

```
                Engine = dbEngine,
                EngineVersion = dbEngineVersion,
                DBInstanceClass = instanceClass
            });

        return response.DBInstance;
    }
```

- Per i dettagli sull'API, consulta Create DBInstance in AWS SDK per .NETAPI Reference.

C++

SDK per C++

> **Note**
>
> C'è altro su GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice AWS.

```
    Aws::Client::ClientConfiguration clientConfig;
    // Optional: Set to the AWS Region (overrides config file).
    // clientConfig.region = "us-east-1";

Aws::RDS::RDSClient client(clientConfig);

    Aws::RDS::Model::CreateDBInstanceRequest request;
    request.SetDBInstanceIdentifier(DB_INSTANCE_IDENTIFIER);
    request.SetDBClusterIdentifier(DB_CLUSTER_IDENTIFIER);
    request.SetEngine(engineName);
    request.SetDBInstanceClass(dbInstanceClass);

    Aws::RDS::Model::CreateDBInstanceOutcome outcome =
            client.CreateDBInstance(request);

    if (outcome.IsSuccess()) {
        std::cout << "The DB instance creation has started."
                  << std::endl;
    }
    else {
        std::cerr << "Error with RDS::CreateDBInstance. "
```

```
                    << outcome.GetError().GetMessage()
                    << std::endl;
            cleanUpResources(CLUSTER_PARAMETER_GROUP_NAME, DB_CLUSTER_IDENTIFIER,
 "",
                             client);
            return false;
        }
```

- Per i dettagli sull'API, consulta Create DBInstance in AWS SDK per C++API Reference.

Go

SDK per Go V2

> **Note**
>
> C'è altro su GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice AWS.

```
import (
 "context"
 "errors"
 "log"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/service/rds"
 "github.com/aws/aws-sdk-go-v2/service/rds/types"
)

type DbClusters struct {
 AuroraClient *rds.Client
}



// CreateInstanceInCluster creates a database instance in an existing DB cluster.
 The first database that is
// created defaults to a read-write DB instance.
```

```
func (clusters *DbClusters) CreateInstanceInCluster(ctx context.Context,
 clusterName string, instanceName string,
 dbEngine string, dbInstanceClass string) (*types.DBInstance, error) {
 output, err := clusters.AuroraClient.CreateDBInstance(ctx,
 &rds.CreateDBInstanceInput{
  DBInstanceIdentifier: aws.String(instanceName),
  DBClusterIdentifier:  aws.String(clusterName),
  Engine:               aws.String(dbEngine),
  DBInstanceClass:      aws.String(dbInstanceClass),
 })
 if err != nil {
  log.Printf("Couldn't create instance %v: %v\n", instanceName, err)
  return nil, err
 } else {
  return output.DBInstance, nil
 }
}
```

- Per i dettagli sull'API, consulta Create DBInstance in AWS SDK per GoAPI Reference.

Java

SDK per Java 2.x

> **Note**
>
> C'è altro su GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice AWS.

```
    public static String createDBInstanceCluster(RdsClient rdsClient,
            String dbInstanceIdentifier,
            String dbInstanceClusterIdentifier,
            String instanceClass) {
        try {
            CreateDbInstanceRequest instanceRequest =
 CreateDbInstanceRequest.builder()
                    .dbInstanceIdentifier(dbInstanceIdentifier)
                    .dbClusterIdentifier(dbInstanceClusterIdentifier)
                    .engine("aurora-mysql")
```

```
                    .dbInstanceClass(instanceClass)
                    .build();

            CreateDbInstanceResponse response =
 rdsClient.createDBInstance(instanceRequest);
            System.out.print("The status is " +
 response.dbInstance().dbInstanceStatus());
            return response.dbInstance().dbInstanceArn();

        } catch (RdsException e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
        return "";
    }
```

- Per i dettagli sull'API, consulta Create DBInstance in AWS SDK for Java 2.xAPI Reference.

Kotlin

SDK per Kotlin

> **Note**
>
> C'è altro su GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice AWS.

```
suspend fun createDBInstanceCluster(
    dbInstanceIdentifierVal: String?,
    dbInstanceClusterIdentifierVal: String?,
    instanceClassVal: String?,
): String? {
    val instanceRequest =
        CreateDbInstanceRequest {
            dbInstanceIdentifier = dbInstanceIdentifierVal
            dbClusterIdentifier = dbInstanceClusterIdentifierVal
            engine = "aurora-mysql"
            dbInstanceClass = instanceClassVal
        }
```

```
    RdsClient.fromEnvironment { region = "us-west-2" }.use { rdsClient ->
        val response = rdsClient.createDbInstance(instanceRequest)
        print("The status is ${response.dbInstance?.dbInstanceStatus}")
        return response.dbInstance?.dbInstanceArn
    }
}
```

- Per i dettagli sull'API, consulta Create DBInstance in AWSSDK for Kotlin API reference.

Python

SDK per Python (Boto3)

> **Note**
>
> C'è altro su. GitHub Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice AWS.

```
class AuroraWrapper:
    """Encapsulates Aurora DB cluster actions."""

    def __init__(self, rds_client):
        """
        :param rds_client: A Boto3 Amazon Relational Database Service (Amazon
 RDS) client.
        """
        self.rds_client = rds_client

    @classmethod
    def from_client(cls):
        """
        Instantiates this class from a Boto3 client.
        """
        rds_client = boto3.client("rds")
        return cls(rds_client)


    def create_instance_in_cluster(
        self, instance_id, cluster_id, db_engine, instance_class
    ):
```

```
        """
        Creates a database instance in an existing DB cluster. The first database
that is
        created defaults to a read-write DB instance.

        :param instance_id: The ID to give the newly created DB instance.
        :param cluster_id: The ID of the DB cluster where the DB instance is
created.
        :param db_engine: The database engine of a database to create in the DB
instance.
                          This must be compatible with the configured parameter
group
                          of the DB cluster.
        :param instance_class: The DB instance class for the newly created DB
instance.
        :return: Data about the newly created DB instance.
        """
        try:
            response = self.rds_client.create_db_instance(
                DBInstanceIdentifier=instance_id,
                DBClusterIdentifier=cluster_id,
                Engine=db_engine,
                DBInstanceClass=instance_class,
            )
            db_inst = response["DBInstance"]
        except ClientError as err:
            logger.error(
                "Couldn't create DB instance %s. Here's why: %s: %s",
                instance_id,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
        else:
            return db_inst
```

- Per i dettagli sull'API, consulta Create DBInstance in AWSSDK for Python (Boto3) API Reference.

Rust

SDK per Rust

> **Note**
>
> C'è altro da fare. GitHub Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice AWS.

```
   // Get a list of allowed engine versions.
rds.DescribeDbEngineVersions(Engine='aurora-mysql', DBParameterGroupFamily=<the
family used to create your parameter group in step 2>)
   // Create an Aurora DB cluster database cluster that contains a MySql
database and uses the parameter group you created.
   // Wait for DB cluster to be ready. Call rds.DescribeDBClusters and check for
Status == 'available'.
   // Get a list of instance classes available for the selected engine
and engine version. rds.DescribeOrderableDbInstanceOptions(Engine='mysql',
EngineVersion=).

   // Create a database instance in the cluster.
   // Wait for DB instance to be ready. Call rds.DescribeDbInstances and check
for DBInstanceStatus == 'available'.
   pub async fn start_cluster_and_instance(&mut self) -> Result<(),
ScenarioError> {
       if self.password.is_none() {
           return Err(ScenarioError::with(
               "Must set Secret Password before starting a cluster",
           ));
       }
       let create_db_cluster = self
           .rds
           .create_db_cluster(
               DB_CLUSTER_IDENTIFIER,
               DB_CLUSTER_PARAMETER_GROUP_NAME,
               DB_ENGINE,
               self.engine_version.as_deref().expect("engine version"),
               self.username.as_deref().expect("username"),
               self.password
                   .replace(SecretString::new("".to_string()))
                   .expect("password"),
```

```
            )
            .await;
        if let Err(err) = create_db_cluster {
            return Err(ScenarioError::new(
                "Failed to create DB Cluster with cluster group",
                &err,
            ));
        }

        self.db_cluster_identifier = create_db_cluster
            .unwrap()
            .db_cluster
            .and_then(|c| c.db_cluster_identifier);

        if self.db_cluster_identifier.is_none() {
            return Err(ScenarioError::with("Created DB Cluster missing
Identifier"));
        }

        info!(
            "Started a db cluster: {}",
            self.db_cluster_identifier
                .as_deref()
                .unwrap_or("Missing ARN")
        );

        let create_db_instance = self
            .rds
            .create_db_instance(
                self.db_cluster_identifier.as_deref().expect("cluster name"),
                DB_INSTANCE_IDENTIFIER,
                self.instance_class.as_deref().expect("instance class"),
                DB_ENGINE,
            )
            .await;
        if let Err(err) = create_db_instance {
            return Err(ScenarioError::new(
                "Failed to create Instance in DB Cluster",
                &err,
            ));
        }

        self.db_instance_identifier = create_db_instance
            .unwrap()
```

```
            .db_instance
            .and_then(|i| i.db_instance_identifier);

        // Cluster creation can take up to 20 minutes to become available
        let cluster_max_wait = Duration::from_secs(20 * 60);
        let waiter = Waiter::builder().max(cluster_max_wait).build();
        while waiter.sleep().await.is_ok() {
            let cluster = self
                .rds
                .describe_db_clusters(
                    self.db_cluster_identifier
                        .as_deref()
                        .expect("cluster identifier"),
                )
                .await;

            if let Err(err) = cluster {
                warn!(?err, "Failed to describe cluster while waiting for
ready");
                continue;
            }

            let instance = self
                .rds
                .describe_db_instance(
                    self.db_instance_identifier
                        .as_deref()
                        .expect("instance identifier"),
                )
                .await;
            if let Err(err) = instance {
                return Err(ScenarioError::new(
                    "Failed to find instance for cluster",
                    &err,
                ));
            }

            let instances_available = instance
                .unwrap()
                .db_instances()
                .iter()
                .all(|instance| instance.db_instance_status() ==
Some("Available"));
```

```
            let endpoints = self
                .rds
                .describe_db_cluster_endpoints(
                    self.db_cluster_identifier
                        .as_deref()
                        .expect("cluster identifier"),
                )
                .await;

            if let Err(err) = endpoints {
                return Err(ScenarioError::new(
                    "Failed to find endpoint for cluster",
                    &err,
                ));
            }

            let endpoints_available = endpoints
                .unwrap()
                .db_cluster_endpoints()
                .iter()
                .all(|endpoint| endpoint.status() == Some("available"));

            if instances_available && endpoints_available {
                return Ok(());
            }
        }

        Err(ScenarioError::with("timed out waiting for cluster"))
    }

    pub async fn create_db_instance(
        &self,
        cluster_name: &str,
        instance_name: &str,
        instance_class: &str,
        engine: &str,
    ) -> Result<CreateDbInstanceOutput, SdkError<CreateDBInstanceError>> {
        self.inner
            .create_db_instance()
            .db_cluster_identifier(cluster_name)
            .db_instance_identifier(instance_name)
            .db_instance_class(instance_class)
            .engine(engine)
            .send()
```

```
            .await
    }

#[tokio::test]
async fn test_start_cluster_and_instance() {
    let mut mock_rds = MockRdsImpl::default();

    mock_rds
        .expect_create_db_cluster()
        .withf(|id, params, engine, version, username, password| {
            assert_eq!(id, "RustSDKCodeExamplesDBCluster");
            assert_eq!(params, "RustSDKCodeExamplesDBParameterGroup");
            assert_eq!(engine, "aurora-mysql");
            assert_eq!(version, "aurora-mysql8.0");
            assert_eq!(username, "test username");
            assert_eq!(password.expose_secret(), "test password");
            true
        })
        .return_once(|id, _, _, _, _, _| {
            Ok(CreateDbClusterOutput::builder()

 .db_cluster(DbCluster::builder().db_cluster_identifier(id).build())
                .build())
        });

    mock_rds
        .expect_create_db_instance()
        .withf(|cluster, name, class, engine| {
            assert_eq!(cluster, "RustSDKCodeExamplesDBCluster");
            assert_eq!(name, "RustSDKCodeExamplesDBInstance");
            assert_eq!(class, "m5.large");
            assert_eq!(engine, "aurora-mysql");
            true
        })
        .return_once(|cluster, name, class, _| {
            Ok(CreateDbInstanceOutput::builder()
                .db_instance(
                    DbInstance::builder()
                        .db_cluster_identifier(cluster)
                        .db_instance_identifier(name)
                        .db_instance_class(class)
                        .build(),
                )
                .build())
```

```
        });

    mock_rds
        .expect_describe_db_clusters()
        .with(eq("RustSDKCodeExamplesDBCluster"))
        .return_once(|id| {
            Ok(DescribeDbClustersOutput::builder()

.db_clusters(DbCluster::builder().db_cluster_identifier(id).build())
                .build())
        });

    mock_rds
        .expect_describe_db_instance()
        .with(eq("RustSDKCodeExamplesDBInstance"))
        .return_once(|name| {
            Ok(DescribeDbInstancesOutput::builder()
                .db_instances(
                    DbInstance::builder()
                        .db_instance_identifier(name)
                        .db_instance_status("Available")
                        .build(),
                )
                .build())
        });

    mock_rds
        .expect_describe_db_cluster_endpoints()
        .with(eq("RustSDKCodeExamplesDBCluster"))
        .return_once(|_| {
            Ok(DescribeDbClusterEndpointsOutput::builder()

.db_cluster_endpoints(DbClusterEndpoint::builder().status("available").build())
                .build())
        });

    let mut scenario = AuroraScenario::new(mock_rds);
    scenario.engine_version = Some("aurora-mysql8.0".into());
    scenario.instance_class = Some("m5.large".into());
    scenario.username = Some("test username".into());
    scenario.password = Some(SecretString::new("test password".into()));

    tokio::time::pause();
    let assertions = tokio::spawn(async move {
```

```
        let create = scenario.start_cluster_and_instance().await;
        assert!(create.is_ok());
        assert!(scenario
            .password
            .replace(SecretString::new("BAD SECRET".into()))
            .unwrap()
            .expose_secret()
            .is_empty());
        assert_eq!(
            scenario.db_cluster_identifier,
            Some("RustSDKCodeExamplesDBCluster".into())
        );
    });
    tokio::time::advance(Duration::from_secs(1)).await;
    tokio::time::resume();
    let _ = assertions.await;
}

#[tokio::test]
async fn test_start_cluster_and_instance_cluster_create_error() {
    let mut mock_rds = MockRdsImpl::default();

    mock_rds
        .expect_create_db_cluster()
        .return_once(|_, _, _, _, _, _| {
            Err(SdkError::service_error(
                CreateDBClusterError::unhandled(Box::new(Error::new(
                    ErrorKind::Other,
                    "create db cluster error",
                ))),
                Response::new(StatusCode::try_from(400).unwrap(),
 SdkBody::empty()),
            ))
        });

    let mut scenario = AuroraScenario::new(mock_rds);
    scenario.engine_version = Some("aurora-mysql8.0".into());
    scenario.instance_class = Some("m5.large".into());
    scenario.username = Some("test username".into());
    scenario.password = Some(SecretString::new("test password".into()));

    let create = scenario.start_cluster_and_instance().await;
    assert_matches!(create, Err(ScenarioError { message, context: _}) if message
 == "Failed to create DB Cluster with cluster group")
```

```
}

#[tokio::test]
async fn test_start_cluster_and_instance_cluster_create_missing_id() {
    let mut mock_rds = MockRdsImpl::default();

    mock_rds
        .expect_create_db_cluster()
        .return_once(|_, _, _, _, _, _| {
            Ok(CreateDbClusterOutput::builder()
                .db_cluster(DbCluster::builder().build())
                .build())
        });

    let mut scenario = AuroraScenario::new(mock_rds);
    scenario.engine_version = Some("aurora-mysql8.0".into());
    scenario.instance_class = Some("m5.large".into());
    scenario.username = Some("test username".into());
    scenario.password = Some(SecretString::new("test password".into()));

    let create = scenario.start_cluster_and_instance().await;
    assert_matches!(create, Err(ScenarioError { message, context:_ }) if message
 == "Created DB Cluster missing Identifier");
}

#[tokio::test]
async fn test_start_cluster_and_instance_instance_create_error() {
    let mut mock_rds = MockRdsImpl::default();

    mock_rds
        .expect_create_db_cluster()
        .withf(|id, params, engine, version, username, password| {
            assert_eq!(id, "RustSDKCodeExamplesDBCluster");
            assert_eq!(params, "RustSDKCodeExamplesDBParameterGroup");
            assert_eq!(engine, "aurora-mysql");
            assert_eq!(version, "aurora-mysql8.0");
            assert_eq!(username, "test username");
            assert_eq!(password.expose_secret(), "test password");
            true
        })
        .return_once(|id, _, _, _, _, _| {
            Ok(CreateDbClusterOutput::builder()

 .db_cluster(DbCluster::builder().db_cluster_identifier(id).build())
```

```
                .build())
        });

    mock_rds
        .expect_create_db_instance()
        .return_once(|_, _, _, _| {
            Err(SdkError::service_error(
                CreateDBInstanceError::unhandled(Box::new(Error::new(
                    ErrorKind::Other,
                    "create db instance error",
                ))),
                Response::new(StatusCode::try_from(400).unwrap(),
 SdkBody::empty()),
            ))
        });

    let mut scenario = AuroraScenario::new(mock_rds);
    scenario.engine_version = Some("aurora-mysql8.0".into());
    scenario.instance_class = Some("m5.large".into());
    scenario.username = Some("test username".into());
    scenario.password = Some(SecretString::new("test password".into()));

    let create = scenario.start_cluster_and_instance().await;
    assert_matches!(create, Err(ScenarioError { message, context: _ }) if message
 == "Failed to create Instance in DB Cluster")
}

#[tokio::test]
async fn test_start_cluster_and_instance_wait_hiccup() {
    let mut mock_rds = MockRdsImpl::default();

    mock_rds
        .expect_create_db_cluster()
        .withf(|id, params, engine, version, username, password| {
            assert_eq!(id, "RustSDKCodeExamplesDBCluster");
            assert_eq!(params, "RustSDKCodeExamplesDBParameterGroup");
            assert_eq!(engine, "aurora-mysql");
            assert_eq!(version, "aurora-mysql8.0");
            assert_eq!(username, "test username");
            assert_eq!(password.expose_secret(), "test password");
            true
        })
        .return_once(|id, _, _, _, _, _| {
            Ok(CreateDbClusterOutput::builder()
```

```
.db_cluster(DbCluster::builder().db_cluster_identifier(id).build())
                .build())
        });

    mock_rds
        .expect_create_db_instance()
        .withf(|cluster, name, class, engine| {
            assert_eq!(cluster, "RustSDKCodeExamplesDBCluster");
            assert_eq!(name, "RustSDKCodeExamplesDBInstance");
            assert_eq!(class, "m5.large");
            assert_eq!(engine, "aurora-mysql");
            true
        })
        .return_once(|cluster, name, class, _| {
            Ok(CreateDbInstanceOutput::builder()
                .db_instance(
                    DbInstance::builder()
                        .db_cluster_identifier(cluster)
                        .db_instance_identifier(name)
                        .db_instance_class(class)
                        .build(),
                )
                .build())
        });

    mock_rds
        .expect_describe_db_clusters()
        .with(eq("RustSDKCodeExamplesDBCluster"))
        .times(1)
        .returning(|_| {
            Err(SdkError::service_error(
                DescribeDBClustersError::unhandled(Box::new(Error::new(
                    ErrorKind::Other,
                    "describe cluster error",
                ))),
                Response::new(StatusCode::try_from(400).unwrap(),
SdkBody::empty()),
            ))
        })
        .with(eq("RustSDKCodeExamplesDBCluster"))
        .times(1)
        .returning(|id| {
            Ok(DescribeDbClustersOutput::builder()
```

```
 .db_clusters(DbCluster::builder().db_cluster_identifier(id).build())
                .build())
        });

    mock_rds.expect_describe_db_instance().return_once(|name| {
        Ok(DescribeDbInstancesOutput::builder()
            .db_instances(
                DbInstance::builder()
                    .db_instance_identifier(name)
                    .db_instance_status("Available")
                    .build(),
            )
            .build())
    });

    mock_rds
        .expect_describe_db_cluster_endpoints()
        .return_once(|_| {
            Ok(DescribeDbClusterEndpointsOutput::builder()

 .db_cluster_endpoints(DbClusterEndpoint::builder().status("available").build())
                .build())
        });

    let mut scenario = AuroraScenario::new(mock_rds);
    scenario.engine_version = Some("aurora-mysql8.0".into());
    scenario.instance_class = Some("m5.large".into());
    scenario.username = Some("test username".into());
    scenario.password = Some(SecretString::new("test password".into()));

    tokio::time::pause();
    let assertions = tokio::spawn(async move {
        let create = scenario.start_cluster_and_instance().await;
        assert!(create.is_ok());
    });

    tokio::time::advance(Duration::from_secs(1)).await;
    tokio::time::advance(Duration::from_secs(1)).await;
    tokio::time::resume();
    let _ = assertions.await;
}
```

- Per i dettagli sull'API, consulta DBInstanceCreate in AWSSDK for Rust API reference.

Per un elenco completo delle guide per sviluppatori AWS SDK e degli esempi di codice, consulta. Utilizzo di questo servizio con un AWS SDK Questo argomento include anche informazioni su come iniziare e dettagli sulle versioni precedenti dell'SDK.

Utilizzo **`DeleteDBCluster`** con un AWS SDK

Gli esempi di codice seguenti mostrano come utilizzare `DeleteDBCluster`.

Gli esempi di operazioni sono estratti di codice da programmi più grandi e devono essere eseguiti nel contesto. È possibile visualizzare questa operazione nel contesto nel seguente esempio di codice:

- Informazioni di base

.NET

SDK per .NET(v4)

> **Note**
>
> C'è altro da fare. GitHub Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice AWS.

```
   /// <summary>
   /// Delete a particular DB cluster.
   /// </summary>
   /// <param name="dbClusterIdentifier">DB cluster identifier.</param>
   /// <returns>DB cluster object.</returns>
   public async Task<DBCluster> DeleteDBClusterByIdentifierAsync(string
dbClusterIdentifier)
   {
       var response = await _amazonRDS.DeleteDBClusterAsync(
           new DeleteDBClusterRequest()
           {
               DBClusterIdentifier = dbClusterIdentifier,
               SkipFinalSnapshot = true
           });
```

```
        return response.DBCluster;
    }
```

- Per i dettagli sull'API, consulta Delete DBCluster in AWS SDK per .NETAPI Reference.

C++

SDK per C++

> **Note**
>
> C'è altro su GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice AWS.

```
        Aws::Client::ClientConfiguration clientConfig;
        // Optional: Set to the AWS Region (overrides config file).
        // clientConfig.region = "us-east-1";

    Aws::RDS::RDSClient client(clientConfig);

            Aws::RDS::Model::DeleteDBClusterRequest request;
            request.SetDBClusterIdentifier(dbClusterIdentifier);
            request.SetSkipFinalSnapshot(true);

            Aws::RDS::Model::DeleteDBClusterOutcome outcome =
                    client.DeleteDBCluster(request);

            if (outcome.IsSuccess()) {
                std::cout << "DB cluster deletion has started."
                          << std::endl;
                clusterDeleting = true;
                std::cout
                        << "Waiting for DB cluster to delete before deleting the
parameter group."
                        << std::endl;
                std::cout << "This may take a while." << std::endl;
            }
            else {
                std::cerr << "Error with Aurora::DeleteDBCluster. "
                          << outcome.GetError().GetMessage()
```

```
                        << std::endl;
                result = false;
            }
```

- Per i dettagli sull'API, consulta Delete DBCluster in AWS SDK per C++API Reference.

Go

SDK per Go V2

> **Note**
>
> C'è altro su GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice AWS.

```
import (
 "context"
 "errors"
 "log"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/service/rds"
 "github.com/aws/aws-sdk-go-v2/service/rds/types"
)

type DbClusters struct {
 AuroraClient *rds.Client
}



// DeleteDbCluster deletes a DB cluster without keeping a final snapshot.
func (clusters *DbClusters) DeleteDbCluster(ctx context.Context, clusterName
 string) error {
 _, err := clusters.AuroraClient.DeleteDBCluster(ctx, &rds.DeleteDBClusterInput{
  DBClusterIdentifier: aws.String(clusterName),
  SkipFinalSnapshot:   aws.Bool(true),
 })
 if err != nil {
```

```
  log.Printf("Couldn't delete DB cluster %v: %v\n", clusterName, err)
  return err
 } else {
  return nil
 }
}
```

- Per i dettagli sull'API, consulta Delete DBCluster in AWS SDK per GoAPI Reference.

Java

SDK per Java 2.x

> **Note**
>
> C'è altro su GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice AWS.

```
   public static void deleteCluster(RdsClient rdsClient, String
dbInstanceClusterIdentifier) {
       try {
           DeleteDbClusterRequest deleteDbClusterRequest =
DeleteDbClusterRequest.builder()
                   .dbClusterIdentifier(dbInstanceClusterIdentifier)
                   .skipFinalSnapshot(true)
                   .build();

           rdsClient.deleteDBCluster(deleteDbClusterRequest);
           System.out.println(dbInstanceClusterIdentifier + " was deleted!");

       } catch (RdsException e) {
           System.out.println(e.getLocalizedMessage());
           System.exit(1);
       }
   }
```

- Per i dettagli sull'API, consulta Delete DBCluster in AWS SDK for Java 2.xAPI Reference.

Kotlin

SDK per Kotlin

> **Note**
>
> C'è altro su GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice AWS.

```
suspend fun deleteCluster(dbInstanceClusterIdentifier: String) {
    val deleteDbClusterRequest =
        DeleteDbClusterRequest {
            dbClusterIdentifier = dbInstanceClusterIdentifier
            skipFinalSnapshot = true
        }

    RdsClient.fromEnvironment { region = "us-west-2" }.use { rdsClient ->
        rdsClient.deleteDbCluster(deleteDbClusterRequest)
        println("$dbInstanceClusterIdentifier was deleted!")
    }
}
```

- Per i dettagli sull'API, consulta Delete DBCluster in AWSSDK for Kotlin API reference.

Python

SDK per Python (Boto3)

> **Note**
>
> C'è di più su. GitHub Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice AWS.

```
class AuroraWrapper:
    """Encapsulates Aurora DB cluster actions."""

    def __init__(self, rds_client):
```

```
        """
        :param rds_client: A Boto3 Amazon Relational Database Service (Amazon
RDS) client.
        """
        self.rds_client = rds_client

    @classmethod
    def from_client(cls):
        """
        Instantiates this class from a Boto3 client.
        """
        rds_client = boto3.client("rds")
        return cls(rds_client)


    def delete_db_cluster(self, cluster_name):
        """
        Deletes a DB cluster.

        :param cluster_name: The name of the DB cluster to delete.
        """
        try:
            self.rds_client.delete_db_cluster(
                DBClusterIdentifier=cluster_name, SkipFinalSnapshot=True
            )
            logger.info("Deleted DB cluster %s.", cluster_name)
        except ClientError:
            logger.exception("Couldn't delete DB cluster %s.", cluster_name)
            raise
```

- Per i dettagli sull'API, consulta Delete DBCluster in AWSSDK for Python (Boto3) API Reference.

Rust

SDK per Rust

> **Note**
>
> C'è altro da fare. GitHub Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice AWS.

```
    pub async fn clean_up(self) -> Result<(), Vec<ScenarioError>> {
        let mut clean_up_errors: Vec<ScenarioError> = vec![];

        // Delete the instance. rds.DeleteDbInstance.
        let delete_db_instance = self
            .rds
            .delete_db_instance(
                self.db_instance_identifier
                    .as_deref()
                    .expect("instance identifier"),
            )
            .await;
        if let Err(err) = delete_db_instance {
            let identifier = self
                .db_instance_identifier
                .as_deref()
                .unwrap_or("Missing Instance Identifier");
            let message = format!("failed to delete db instance {identifier}");
            clean_up_errors.push(ScenarioError::new(message, &err));
        } else {
            // Wait for the instance to delete
            let waiter = Waiter::default();
            while waiter.sleep().await.is_ok() {
                let describe_db_instances =
self.rds.describe_db_instances().await;
                if let Err(err) = describe_db_instances {
                    clean_up_errors.push(ScenarioError::new(
                        "Failed to check instance state during deletion",
                        &err,
                    ));
                    break;
                }
```

```
                let db_instances = describe_db_instances
                    .unwrap()
                    .db_instances()
                    .iter()
                    .filter(|instance| instance.db_cluster_identifier ==
self.db_cluster_identifier)
                    .cloned()
                    .collect::<Vec<DbInstance>>();

                if db_instances.is_empty() {
                    trace!("Delete Instance waited and no instances were found");
                    break;
                }
                match db_instances.first().unwrap().db_instance_status() {
                    Some("Deleting") => continue,
                    Some(status) => {
                        info!("Attempting to delete but instances is in
{status}");
                        continue;
                    }
                    None => {
                        warn!("No status for DB instance");
                        break;
                    }
                }
            }
        }

        // Delete the DB cluster. rds.DeleteDbCluster.
        let delete_db_cluster = self
            .rds
            .delete_db_cluster(
                self.db_cluster_identifier
                    .as_deref()
                    .expect("cluster identifier"),
            )
            .await;

        if let Err(err) = delete_db_cluster {
            let identifier = self
                .db_cluster_identifier
                .as_deref()
                .unwrap_or("Missing DB Cluster Identifier");
            let message = format!("failed to delete db cluster {identifier}");
```

```
            clean_up_errors.push(ScenarioError::new(message, &err));
        } else {
            // Wait for the instance and cluster to fully delete.
rds.DescribeDbInstances and rds.DescribeDbClusters until both are not found.
            let waiter = Waiter::default();
            while waiter.sleep().await.is_ok() {
                let describe_db_clusters = self
                    .rds
                    .describe_db_clusters(
                        self.db_cluster_identifier
                            .as_deref()
                            .expect("cluster identifier"),
                    )
                    .await;
                if let Err(err) = describe_db_clusters {
                    clean_up_errors.push(ScenarioError::new(
                        "Failed to check cluster state during deletion",
                        &err,
                    ));
                    break;
                }
                let describe_db_clusters = describe_db_clusters.unwrap();
                let db_clusters = describe_db_clusters.db_clusters();
                if db_clusters.is_empty() {
                    trace!("Delete cluster waited and no clusters were found");
                    break;
                }
                match db_clusters.first().unwrap().status() {
                    Some("Deleting") => continue,
                    Some(status) => {
                        info!("Attempting to delete but clusters is in
{status}");
                        continue;
                    }
                    None => {
                        warn!("No status for DB cluster");
                        break;
                    }
                }
            }
        }

        // Delete the DB cluster parameter group.
rds.DeleteDbClusterParameterGroup.
```

```
        let delete_db_cluster_parameter_group = self
            .rds
            .delete_db_cluster_parameter_group(
                self.db_cluster_parameter_group
                    .map(|g| {
                        g.db_cluster_parameter_group_name
                            .unwrap_or_else(||
 DB_CLUSTER_PARAMETER_GROUP_NAME.to_string())
                    })
                    .as_deref()
                    .expect("cluster parameter group name"),
            )
            .await;
        if let Err(error) = delete_db_cluster_parameter_group {
            clean_up_errors.push(ScenarioError::new(
                "Failed to delete the db cluster parameter group",
                &error,
            ))
        }

        if clean_up_errors.is_empty() {
            Ok(())
        } else {
            Err(clean_up_errors)
        }
    }

    pub async fn delete_db_cluster(
        &self,
        cluster_identifier: &str,
    ) -> Result<DeleteDbClusterOutput, SdkError<DeleteDBClusterError>> {
        self.inner
            .delete_db_cluster()
            .db_cluster_identifier(cluster_identifier)
            .skip_final_snapshot(true)
            .send()
            .await
    }

#[tokio::test]
async fn test_scenario_clean_up() {
    let mut mock_rds = MockRdsImpl::default();

    mock_rds
```

```
        .expect_delete_db_instance()
        .with(eq("MockInstance"))
        .return_once(|_| Ok(DeleteDbInstanceOutput::builder().build()));

    mock_rds
        .expect_describe_db_instances()
        .with()
        .times(1)
        .returning(|| {
            Ok(DescribeDbInstancesOutput::builder()
                .db_instances(
                    DbInstance::builder()
                        .db_cluster_identifier("MockCluster")
                        .db_instance_status("Deleting")
                        .build(),
                )
                .build())
        })
        .with()
        .times(1)
        .returning(|| Ok(DescribeDbInstancesOutput::builder().build()));

    mock_rds
        .expect_delete_db_cluster()
        .with(eq("MockCluster"))
        .return_once(|_| Ok(DeleteDbClusterOutput::builder().build()));

    mock_rds
        .expect_describe_db_clusters()
        .with(eq("MockCluster"))
        .times(1)
        .returning(|id| {
            Ok(DescribeDbClustersOutput::builder()
                .db_clusters(
                    DbCluster::builder()
                        .db_cluster_identifier(id)
                        .status("Deleting")
                        .build(),
                )
                .build())
        })
        .with(eq("MockCluster"))
        .times(1)
        .returning(|_| Ok(DescribeDbClustersOutput::builder().build()));
```

```
    mock_rds
        .expect_delete_db_cluster_parameter_group()
        .with(eq("MockParamGroup"))
        .return_once(|_|
 Ok(DeleteDbClusterParameterGroupOutput::builder().build()));

    let mut scenario = AuroraScenario::new(mock_rds);
    scenario.db_cluster_identifier = Some(String::from("MockCluster"));
    scenario.db_instance_identifier = Some(String::from("MockInstance"));
    scenario.db_cluster_parameter_group = Some(
        DbClusterParameterGroup::builder()
            .db_cluster_parameter_group_name("MockParamGroup")
            .build(),
    );

    tokio::time::pause();
    let assertions = tokio::spawn(async move {
        let clean_up = scenario.clean_up().await;
        assert!(clean_up.is_ok());
    });

    tokio::time::advance(Duration::from_secs(1)).await; // Wait for first
 Describe Instances
    tokio::time::advance(Duration::from_secs(1)).await; // Wait for second
 Describe Instances
    tokio::time::advance(Duration::from_secs(1)).await; // Wait for first
 Describe Cluster
    tokio::time::advance(Duration::from_secs(1)).await; // Wait for second
 Describe Cluster
    tokio::time::resume();
    let _ = assertions.await;
}

#[tokio::test]
async fn test_scenario_clean_up_errors() {
    let mut mock_rds = MockRdsImpl::default();

    mock_rds
        .expect_delete_db_instance()
        .with(eq("MockInstance"))
        .return_once(|_| Ok(DeleteDbInstanceOutput::builder().build()));

    mock_rds
```

```
        .expect_describe_db_instances()
        .with()
        .times(1)
        .returning(|| {
            Ok(DescribeDbInstancesOutput::builder()
                .db_instances(
                    DbInstance::builder()
                        .db_cluster_identifier("MockCluster")
                        .db_instance_status("Deleting")
                        .build(),
                )
                .build())
        })
        .with()
        .times(1)
        .returning(|| {
            Err(SdkError::service_error(
                DescribeDBInstancesError::unhandled(Box::new(Error::new(
                    ErrorKind::Other,
                    "describe db instances error",
                ))),
                Response::new(StatusCode::try_from(400).unwrap(),
SdkBody::empty()),
            ))
        });

    mock_rds
        .expect_delete_db_cluster()
        .with(eq("MockCluster"))
        .return_once(|_| Ok(DeleteDbClusterOutput::builder().build()));

    mock_rds
        .expect_describe_db_clusters()
        .with(eq("MockCluster"))
        .times(1)
        .returning(|id| {
            Ok(DescribeDbClustersOutput::builder()
                .db_clusters(
                    DbCluster::builder()
                        .db_cluster_identifier(id)
                        .status("Deleting")
                        .build(),
                )
                .build())
```

```
        })
        .with(eq("MockCluster"))
        .times(1)
        .returning(|_| {
            Err(SdkError::service_error(
                DescribeDBClustersError::unhandled(Box::new(Error::new(
                    ErrorKind::Other,
                    "describe db clusters error",
                ))),
                Response::new(StatusCode::try_from(400).unwrap(),
SdkBody::empty()),
            ))
        });

    mock_rds
        .expect_delete_db_cluster_parameter_group()
        .with(eq("MockParamGroup"))
        .return_once(|_|
Ok(DeleteDbClusterParameterGroupOutput::builder().build()));

    let mut scenario = AuroraScenario::new(mock_rds);
    scenario.db_cluster_identifier = Some(String::from("MockCluster"));
    scenario.db_instance_identifier = Some(String::from("MockInstance"));
    scenario.db_cluster_parameter_group = Some(
        DbClusterParameterGroup::builder()
            .db_cluster_parameter_group_name("MockParamGroup")
            .build(),
    );

    tokio::time::pause();
    let assertions = tokio::spawn(async move {
        let clean_up = scenario.clean_up().await;
        assert!(clean_up.is_err());
        let errs = clean_up.unwrap_err();
        assert_eq!(errs.len(), 2);
        assert_matches!(errs.first(), Some(ScenarioError {message, context: _})
if message == "Failed to check instance state during deletion");
        assert_matches!(errs.get(1), Some(ScenarioError {message, context: _}) if
message == "Failed to check cluster state during deletion");
    });

    tokio::time::advance(Duration::from_secs(1)).await; // Wait for first
Describe Instances
```

```
    tokio::time::advance(Duration::from_secs(1)).await; // Wait for second
 Describe Instances
    tokio::time::advance(Duration::from_secs(1)).await; // Wait for first
 Describe Cluster
    tokio::time::advance(Duration::from_secs(1)).await; // Wait for second
 Describe Cluster
    tokio::time::resume();
    let _ = assertions.await;
}
```

- Per i dettagli sull'API, consulta Delete DBCluster in AWSSDK for Rust API reference.

Per un elenco completo delle guide per sviluppatori AWS SDK e degli esempi di codice, consulta. Utilizzo di questo servizio con un AWS SDK Questo argomento include anche informazioni su come iniziare e dettagli sulle versioni precedenti dell'SDK.

Utilizzo **`DeleteDBClusterParameterGroup`** con un AWS SDK

Gli esempi di codice seguenti mostrano come utilizzare `DeleteDBClusterParameterGroup`.

Gli esempi di operazioni sono estratti di codice da programmi più grandi e devono essere eseguiti nel contesto. È possibile visualizzare questa operazione nel contesto nel seguente esempio di codice:

- Informazioni di base

.NET

SDK per .NET(v4)

> **Note**
>
> C'è altro da fare. GitHub Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice AWS.

```
    /// <summary>
    /// Delete a particular parameter group by name.
    /// </summary>
    /// <param name="groupName">The name of the parameter group.</param>
```

```
    /// <returns>True if successful.</returns>
    public async Task<bool> DeleteClusterParameterGroupByNameAsync(string
groupName)
    {
        var request = new DeleteDBClusterParameterGroupRequest
        {
            DBClusterParameterGroupName = groupName,
        };

        var response = await
_amazonRDS.DeleteDBClusterParameterGroupAsync(request);
        return response.HttpStatusCode == System.Net.HttpStatusCode.OK;
    }
```

- Per i dettagli sull'API, consulta Delete DBCluster ParameterGroup in AWS SDK per .NETAPI Reference.

C++

SDK per C++

> **Note**
>
> C'è altro su GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice AWS.

```
        Aws::Client::ClientConfiguration clientConfig;
        // Optional: Set to the AWS Region (overrides config file).
        // clientConfig.region = "us-east-1";

    Aws::RDS::RDSClient client(clientConfig);

        Aws::RDS::Model::DeleteDBClusterParameterGroupRequest request;
        request.SetDBClusterParameterGroupName(parameterGroupName);

        Aws::RDS::Model::DeleteDBClusterParameterGroupOutcome outcome =
                client.DeleteDBClusterParameterGroup(request);

        if (outcome.IsSuccess()) {
```

```
            std::cout << "The DB parameter group was successfully deleted."
                      << std::endl;
        }
        else {
            std::cerr << "Error with Aurora::DeleteDBClusterParameterGroup. "
                      << outcome.GetError().GetMessage()
                      << std::endl;
            result = false;
        }
```

- Per i dettagli sull'API, consulta Delete DBCluster ParameterGroup in AWS SDK per C++API Reference.

Go

SDK per Go V2

> **Note**
>
> C'è altro su GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice AWS.

```
import (
 "context"
 "errors"
 "log"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/service/rds"
 "github.com/aws/aws-sdk-go-v2/service/rds/types"
)

type DbClusters struct {
 AuroraClient *rds.Client
}
```

```
// DeleteParameterGroup deletes the named DB cluster parameter group.
func (clusters *DbClusters) DeleteParameterGroup(ctx context.Context,
 parameterGroupName string) error {
 _, err := clusters.AuroraClient.DeleteDBClusterParameterGroup(ctx,
  &rds.DeleteDBClusterParameterGroupInput{
   DBClusterParameterGroupName: aws.String(parameterGroupName),
  })
 if err != nil {
  log.Printf("Couldn't delete parameter group %v: %v\n", parameterGroupName, err)
  return err
 } else {
  return nil
 }
}
```

- Per i dettagli sull'API, consulta Delete DBCluster ParameterGroup in AWS SDK per GoAPI Reference.

Java

SDK per Java 2.x

> **Note**
>
> C'è altro su GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice AWS.

```
   public static void deleteDBClusterGroup(RdsClient rdsClient, String
dbClusterGroupName, String clusterDBARN)
           throws InterruptedException {
       try {
           boolean isDataDel = false;
           boolean didFind;
           String instanceARN;

           // Make sure that the database has been deleted.
           while (!isDataDel) {
```

```
                DescribeDbInstancesResponse response =
rdsClient.describeDBInstances();
                List<DBInstance> instanceList = response.dbInstances();
                int listSize = instanceList.size();
                didFind = false;
                int index = 1;
                for (DBInstance instance : instanceList) {
                    instanceARN = instance.dbInstanceArn();
                    if (instanceARN.compareTo(clusterDBARN) == 0) {
                        System.out.println(clusterDBARN + " still exists");
                        didFind = true;
                    }
                    if ((index == listSize) && (!didFind)) {
                        // Went through the entire list and did not find the
database ARN.
                        isDataDel = true;
                    }
                    Thread.sleep(sleepTime * 1000);
                    index++;
                }
            }

            DeleteDbClusterParameterGroupRequest clusterParameterGroupRequest =
DeleteDbClusterParameterGroupRequest
                    .builder()
                    .dbClusterParameterGroupName(dbClusterGroupName)
                    .build();


rdsClient.deleteDBClusterParameterGroup(clusterParameterGroupRequest);
            System.out.println(dbClusterGroupName + " was deleted.");

        } catch (RdsException e) {
            System.out.println(e.getLocalizedMessage());
            System.exit(1);
        }
    }
```

- Per i dettagli sull'API, consulta Delete DBCluster ParameterGroup in AWS SDK for Java 2.xAPI Reference.

Kotlin

SDK per Kotlin

> **Note**
>
> C'è altro su GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice AWS.

```
@Throws(InterruptedException::class)
suspend fun deleteDBClusterGroup(
    dbClusterGroupName: String,
    clusterDBARN: String,
) {
    var isDataDel = false
    var didFind: Boolean
    var instanceARN: String

    RdsClient.fromEnvironment { region = "us-west-2" }.use { rdsClient ->
        // Make sure that the database has been deleted.
        while (!isDataDel) {
            val response = rdsClient.describeDbInstances()
            val instanceList = response.dbInstances
            val listSize = instanceList?.size
            isDataDel = false
            didFind = false
            var index = 1
            if (instanceList != null) {
                for (instance in instanceList) {
                    instanceARN = instance.dbInstanceArn.toString()
                    if (instanceARN.compareTo(clusterDBARN) == 0) {
                        println("$clusterDBARN still exists")
                        didFind = true
                    }
                    if (index == listSize && !didFind) {
                        // Went through the entire list and did not find the
 database ARN.
                        isDataDel = true
                    }
                    delay(slTime * 1000)
                    index++
```

```
                }
            }
        }
        val clusterParameterGroupRequest =
            DeleteDbClusterParameterGroupRequest {
                dbClusterParameterGroupName = dbClusterGroupName
            }

        rdsClient.deleteDbClusterParameterGroup(clusterParameterGroupRequest)
        println("$dbClusterGroupName was deleted.")
    }
}
```

- Per i dettagli sull'API, consulta Delete DBCluster ParameterGroup in AWSSDK for Kotlin API reference.

Python

SDK per Python (Boto3)

> **Note**
>
> C'è di più su. GitHub Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice AWS.

```
class AuroraWrapper:
    """Encapsulates Aurora DB cluster actions."""

    def __init__(self, rds_client):
        """
        :param rds_client: A Boto3 Amazon Relational Database Service (Amazon
 RDS) client.
        """
        self.rds_client = rds_client

    @classmethod
    def from_client(cls):
        """
        Instantiates this class from a Boto3 client.
```

```
        """
        rds_client = boto3.client("rds")
        return cls(rds_client)


    def delete_parameter_group(self, parameter_group_name):
        """
        Deletes a DB cluster parameter group.

        :param parameter_group_name: The name of the parameter group to delete.
        :return: Data about the parameter group.
        """
        try:
            response = self.rds_client.delete_db_cluster_parameter_group(
                DBClusterParameterGroupName=parameter_group_name
            )
        except ClientError as err:
            logger.error(
                "Couldn't delete parameter group %s. Here's why: %s: %s",
                parameter_group_name,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
        else:
            return response
```

- Per i dettagli sull'API, consulta Delete DBCluster ParameterGroup in AWSSDK for Python (Boto3) API Reference.

Rust

SDK per Rust

> **Note**
>
> C'è altro da fare. GitHub Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice AWS.

```
    pub async fn clean_up(self) -> Result<(), Vec<ScenarioError>> {
        let mut clean_up_errors: Vec<ScenarioError> = vec![];

        // Delete the instance. rds.DeleteDbInstance.
        let delete_db_instance = self
            .rds
            .delete_db_instance(
                self.db_instance_identifier
                    .as_deref()
                    .expect("instance identifier"),
            )
            .await;
        if let Err(err) = delete_db_instance {
            let identifier = self
                .db_instance_identifier
                .as_deref()
                .unwrap_or("Missing Instance Identifier");
            let message = format!("failed to delete db instance {identifier}");
            clean_up_errors.push(ScenarioError::new(message, &err));
        } else {
            // Wait for the instance to delete
            let waiter = Waiter::default();
            while waiter.sleep().await.is_ok() {
                let describe_db_instances =
self.rds.describe_db_instances().await;
                if let Err(err) = describe_db_instances {
                    clean_up_errors.push(ScenarioError::new(
                        "Failed to check instance state during deletion",
                        &err,
                    ));
                    break;
                }
                let db_instances = describe_db_instances
                    .unwrap()
                    .db_instances()
                    .iter()
                    .filter(|instance| instance.db_cluster_identifier ==
self.db_cluster_identifier)
                    .cloned()
                    .collect::<Vec<DbInstance>>();

                if db_instances.is_empty() {
                    trace!("Delete Instance waited and no instances were found");
```

```
                    break;
                }
                match db_instances.first().unwrap().db_instance_status() {
                    Some("Deleting") => continue,
                    Some(status) => {
                        info!("Attempting to delete but instances is in
{status}");
                        continue;
                    }
                    None => {
                        warn!("No status for DB instance");
                        break;
                    }
                }
            }
        }

        // Delete the DB cluster. rds.DeleteDbCluster.
        let delete_db_cluster = self
            .rds
            .delete_db_cluster(
                self.db_cluster_identifier
                    .as_deref()
                    .expect("cluster identifier"),
            )
            .await;

        if let Err(err) = delete_db_cluster {
            let identifier = self
                .db_cluster_identifier
                .as_deref()
                .unwrap_or("Missing DB Cluster Identifier");
            let message = format!("failed to delete db cluster {identifier}");
            clean_up_errors.push(ScenarioError::new(message, &err));
        } else {
            // Wait for the instance and cluster to fully delete.
rds.DescribeDbInstances and rds.DescribeDbClusters until both are not found.
            let waiter = Waiter::default();
            while waiter.sleep().await.is_ok() {
                let describe_db_clusters = self
                    .rds
                    .describe_db_clusters(
                        self.db_cluster_identifier
                            .as_deref()
```

```
                            .expect("cluster identifier"),
                    )
                    .await;
                if let Err(err) = describe_db_clusters {
                    clean_up_errors.push(ScenarioError::new(
                        "Failed to check cluster state during deletion",
                        &err,
                    ));
                    break;
                }
                let describe_db_clusters = describe_db_clusters.unwrap();
                let db_clusters = describe_db_clusters.db_clusters();
                if db_clusters.is_empty() {
                    trace!("Delete cluster waited and no clusters were found");
                    break;
                }
                match db_clusters.first().unwrap().status() {
                    Some("Deleting") => continue,
                    Some(status) => {
                        info!("Attempting to delete but clusters is in
{status}");
                        continue;
                    }
                    None => {
                        warn!("No status for DB cluster");
                        break;
                    }
                }
            }
        }

        // Delete the DB cluster parameter group.
rds.DeleteDbClusterParameterGroup.
        let delete_db_cluster_parameter_group = self
            .rds
            .delete_db_cluster_parameter_group(
                self.db_cluster_parameter_group
                    .map(|g| {
                        g.db_cluster_parameter_group_name
                            .unwrap_or_else(||
DB_CLUSTER_PARAMETER_GROUP_NAME.to_string())
                    })
                    .as_deref()
                    .expect("cluster parameter group name"),
```

```
            )
            .await;
        if let Err(error) = delete_db_cluster_parameter_group {
            clean_up_errors.push(ScenarioError::new(
                "Failed to delete the db cluster parameter group",
                &error,
            ))
        }

        if clean_up_errors.is_empty() {
            Ok(())
        } else {
            Err(clean_up_errors)
        }
    }

    pub async fn delete_db_cluster_parameter_group(
        &self,
        name: &str,
    ) -> Result<DeleteDbClusterParameterGroupOutput,
 SdkError<DeleteDBClusterParameterGroupError>>
    {
        self.inner
            .delete_db_cluster_parameter_group()
            .db_cluster_parameter_group_name(name)
            .send()
            .await
    }

#[tokio::test]
async fn test_scenario_clean_up() {
    let mut mock_rds = MockRdsImpl::default();

    mock_rds
        .expect_delete_db_instance()
        .with(eq("MockInstance"))
        .return_once(|_| Ok(DeleteDbInstanceOutput::builder().build()));

    mock_rds
        .expect_describe_db_instances()
        .with()
        .times(1)
        .returning(|| {
            Ok(DescribeDbInstancesOutput::builder()
```

```
                .db_instances(
                    DbInstance::builder()
                        .db_cluster_identifier("MockCluster")
                        .db_instance_status("Deleting")
                        .build(),
                )
                .build())
        })
        .with()
        .times(1)
        .returning(|| Ok(DescribeDbInstancesOutput::builder().build()));

    mock_rds
        .expect_delete_db_cluster()
        .with(eq("MockCluster"))
        .return_once(|_| Ok(DeleteDbClusterOutput::builder().build()));

    mock_rds
        .expect_describe_db_clusters()
        .with(eq("MockCluster"))
        .times(1)
        .returning(|id| {
            Ok(DescribeDbClustersOutput::builder()
                .db_clusters(
                    DbCluster::builder()
                        .db_cluster_identifier(id)
                        .status("Deleting")
                        .build(),
                )
                .build())
        })
        .with(eq("MockCluster"))
        .times(1)
        .returning(|_| Ok(DescribeDbClustersOutput::builder().build()));

    mock_rds
        .expect_delete_db_cluster_parameter_group()
        .with(eq("MockParamGroup"))
        .return_once(|_|
Ok(DeleteDbClusterParameterGroupOutput::builder().build()));

    let mut scenario = AuroraScenario::new(mock_rds);
    scenario.db_cluster_identifier = Some(String::from("MockCluster"));
    scenario.db_instance_identifier = Some(String::from("MockInstance"));
```

```
    scenario.db_cluster_parameter_group = Some(
        DbClusterParameterGroup::builder()
            .db_cluster_parameter_group_name("MockParamGroup")
            .build(),
    );

    tokio::time::pause();
    let assertions = tokio::spawn(async move {
        let clean_up = scenario.clean_up().await;
        assert!(clean_up.is_ok());
    });

    tokio::time::advance(Duration::from_secs(1)).await; // Wait for first
 Describe Instances
    tokio::time::advance(Duration::from_secs(1)).await; // Wait for second
 Describe Instances
    tokio::time::advance(Duration::from_secs(1)).await; // Wait for first
 Describe Cluster
    tokio::time::advance(Duration::from_secs(1)).await; // Wait for second
 Describe Cluster
    tokio::time::resume();
    let _ = assertions.await;
}

#[tokio::test]
async fn test_scenario_clean_up_errors() {
    let mut mock_rds = MockRdsImpl::default();

    mock_rds
        .expect_delete_db_instance()
        .with(eq("MockInstance"))
        .return_once(|_| Ok(DeleteDbInstanceOutput::builder().build()));

    mock_rds
        .expect_describe_db_instances()
        .with()
        .times(1)
        .returning(|| {
            Ok(DescribeDbInstancesOutput::builder()
                .db_instances(
                    DbInstance::builder()
                        .db_cluster_identifier("MockCluster")
                        .db_instance_status("Deleting")
                        .build(),
```

```
                )
                .build())
        })
        .with()
        .times(1)
        .returning(|| {
            Err(SdkError::service_error(
                DescribeDBInstancesError::unhandled(Box::new(Error::new(
                    ErrorKind::Other,
                    "describe db instances error",
                ))),
                Response::new(StatusCode::try_from(400).unwrap(),
SdkBody::empty()),
            ))
        });

    mock_rds
        .expect_delete_db_cluster()
        .with(eq("MockCluster"))
        .return_once(|_| Ok(DeleteDbClusterOutput::builder().build()));

    mock_rds
        .expect_describe_db_clusters()
        .with(eq("MockCluster"))
        .times(1)
        .returning(|id| {
            Ok(DescribeDbClustersOutput::builder()
                .db_clusters(
                    DbCluster::builder()
                        .db_cluster_identifier(id)
                        .status("Deleting")
                        .build(),
                )
                .build())
        })
        .with(eq("MockCluster"))
        .times(1)
        .returning(|_| {
            Err(SdkError::service_error(
                DescribeDBClustersError::unhandled(Box::new(Error::new(
                    ErrorKind::Other,
                    "describe db clusters error",
                ))),
```

```
                Response::new(StatusCode::try_from(400).unwrap(),
 SdkBody::empty()),
            ))
        });

    mock_rds
        .expect_delete_db_cluster_parameter_group()
        .with(eq("MockParamGroup"))
        .return_once(|_|
 Ok(DeleteDbClusterParameterGroupOutput::builder().build()));

    let mut scenario = AuroraScenario::new(mock_rds);
    scenario.db_cluster_identifier = Some(String::from("MockCluster"));
    scenario.db_instance_identifier = Some(String::from("MockInstance"));
    scenario.db_cluster_parameter_group = Some(
        DbClusterParameterGroup::builder()
            .db_cluster_parameter_group_name("MockParamGroup")
            .build(),
    );

    tokio::time::pause();
    let assertions = tokio::spawn(async move {
        let clean_up = scenario.clean_up().await;
        assert!(clean_up.is_err());
        let errs = clean_up.unwrap_err();
        assert_eq!(errs.len(), 2);
        assert_matches!(errs.first(), Some(ScenarioError {message, context: _})
 if message == "Failed to check instance state during deletion");
        assert_matches!(errs.get(1), Some(ScenarioError {message, context: _}) if
 message == "Failed to check cluster state during deletion");
    });

    tokio::time::advance(Duration::from_secs(1)).await; // Wait for first
 Describe Instances
    tokio::time::advance(Duration::from_secs(1)).await; // Wait for second
 Describe Instances
    tokio::time::advance(Duration::from_secs(1)).await; // Wait for first
 Describe Cluster
    tokio::time::advance(Duration::from_secs(1)).await; // Wait for second
 Describe Cluster
    tokio::time::resume();
    let _ = assertions.await;
}
```

- Per i dettagli sull'API, consulta Delete DBCluster ParameterGroup in AWSSDK for Rust API reference.

Per un elenco completo delle guide per sviluppatori AWS SDK e degli esempi di codice, consulta. Utilizzo di questo servizio con un AWS SDK Questo argomento include anche informazioni su come iniziare e dettagli sulle versioni precedenti dell'SDK.

Utilizzo **`DeleteDBInstance`** con un AWS SDK

Gli esempi di codice seguenti mostrano come utilizzare `DeleteDBInstance`.

Gli esempi di operazioni sono estratti di codice da programmi più grandi e devono essere eseguiti nel contesto. È possibile visualizzare questa operazione nel contesto nel seguente esempio di codice:

- Informazioni di base

.NET

SDK per .NET(v4)

> **Note**
>
> C'è altro da fare. GitHub Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice AWS.

```
   /// <summary>
   /// Delete a particular DB instance.
   /// </summary>
   /// <param name="dbInstanceIdentifier">DB instance identifier.</param>
   /// <returns>DB instance object.</returns>
   public async Task<DBInstance> DeleteDBInstanceByIdentifierAsync(string
dbInstanceIdentifier)
   {
       var response = await _amazonRDS.DeleteDBInstanceAsync(
           new DeleteDBInstanceRequest()
           {
```

```
                DBInstanceIdentifier = dbInstanceIdentifier,
                SkipFinalSnapshot = true,
                DeleteAutomatedBackups = true
            });

        return response.DBInstance;
    }
```

- Per i dettagli sull'API, consulta Delete DBInstance in AWS SDK per .NETAPI Reference.

C++

SDK per C++

> **Note**
>
> C'è altro su GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice AWS.

```
        Aws::Client::ClientConfiguration clientConfig;
        // Optional: Set to the AWS Region (overrides config file).
        // clientConfig.region = "us-east-1";

    Aws::RDS::RDSClient client(clientConfig);

            Aws::RDS::Model::DeleteDBInstanceRequest request;
            request.SetDBInstanceIdentifier(dbInstanceIdentifier);
            request.SetSkipFinalSnapshot(true);
            request.SetDeleteAutomatedBackups(true);

            Aws::RDS::Model::DeleteDBInstanceOutcome outcome =
                    client.DeleteDBInstance(request);

            if (outcome.IsSuccess()) {
                std::cout << "DB instance deletion has started."
                          << std::endl;
                instanceDeleting = true;
                std::cout
                        << "Waiting for DB instance to delete before deleting the
parameter group."
```

```
                        << std::endl;
            }
            else {
                std::cerr << "Error with Aurora::DeleteDBInstance. "
                          << outcome.GetError().GetMessage()
                          << std::endl;
                result = false;
            }
```

- Per i dettagli sull'API, consulta Delete DBInstance in AWS SDK per C++API Reference.

Go

SDK per Go V2

> **Note**
>
> C'è altro su GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice AWS.

```
import (
 "context"
 "errors"
 "log"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/service/rds"
 "github.com/aws/aws-sdk-go-v2/service/rds/types"
)

type DbClusters struct {
 AuroraClient *rds.Client
}



// DeleteInstance deletes a DB instance.
func (clusters *DbClusters) DeleteInstance(ctx context.Context, instanceName
 string) error {
```

```
 _, err := clusters.AuroraClient.DeleteDBInstance(ctx,
 &rds.DeleteDBInstanceInput{
  DBInstanceIdentifier:   aws.String(instanceName),
  SkipFinalSnapshot:      aws.Bool(true),
  DeleteAutomatedBackups: aws.Bool(true),
 })
 if err != nil {
  log.Printf("Couldn't delete instance %v: %v\n", instanceName, err)
  return err
 } else {
  return nil
 }
}
```

- Per i dettagli sull'API, consulta Delete DBInstance in AWS SDK per GoAPI Reference.

Java

SDK per Java 2.x

> **Note**
>
> C'è altro su GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice AWS.

```
   public static void deleteDatabaseInstance(RdsClient rdsClient, String
dbInstanceIdentifier) {
       try {
           DeleteDbInstanceRequest deleteDbInstanceRequest =
DeleteDbInstanceRequest.builder()
                   .dbInstanceIdentifier(dbInstanceIdentifier)
                   .deleteAutomatedBackups(true)
                   .skipFinalSnapshot(true)
                   .build();

           DeleteDbInstanceResponse response =
rdsClient.deleteDBInstance(deleteDbInstanceRequest);
           System.out.println("The status of the database is " +
response.dbInstance().dbInstanceStatus());
```

```
        } catch (RdsException e) {
            System.out.println(e.getLocalizedMessage());
            System.exit(1);
        }
    }
```

- Per i dettagli sull'API, consulta Delete DBInstance in AWS SDK for Java 2.xAPI Reference.

Kotlin

SDK per Kotlin

> **Note**
>
> C'è altro su GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice AWS.

```
suspend fun deleteDBInstance(dbInstanceIdentifierVal: String) {
    val deleteDbInstanceRequest =
        DeleteDbInstanceRequest {
            dbInstanceIdentifier = dbInstanceIdentifierVal
            deleteAutomatedBackups = true
            skipFinalSnapshot = true
        }

    RdsClient.fromEnvironment { region = "us-west-2" }.use { rdsClient ->
        val response = rdsClient.deleteDbInstance(deleteDbInstanceRequest)
        print("The status of the database is
 ${response.dbInstance?.dbInstanceStatus}")
    }
}
```

- Per i dettagli sull'API, consulta Delete DBInstance in AWSSDK for Kotlin API reference.

Python

SDK per Python (Boto3)

> **Note**
>
> C'è di più su. GitHub Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice AWS.

```
class AuroraWrapper:
    """Encapsulates Aurora DB cluster actions."""

    def __init__(self, rds_client):
        """
        :param rds_client: A Boto3 Amazon Relational Database Service (Amazon
 RDS) client.
        """
        self.rds_client = rds_client

    @classmethod
    def from_client(cls):
        """
        Instantiates this class from a Boto3 client.
        """
        rds_client = boto3.client("rds")
        return cls(rds_client)


    def delete_db_instance(self, instance_id):
        """
        Deletes a DB instance.

        :param instance_id: The ID of the DB instance to delete.
        :return: Data about the deleted DB instance.
        """
        try:
            response = self.rds_client.delete_db_instance(
                DBInstanceIdentifier=instance_id,
                SkipFinalSnapshot=True,
                DeleteAutomatedBackups=True,
            )
```

```
            db_inst = response["DBInstance"]
        except ClientError as err:
            logger.error(
                "Couldn't delete DB instance %s. Here's why: %s: %s",
                instance_id,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
        else:
            return db_inst
```

- Per i dettagli sull'API, consulta Delete DBInstance in AWSSDK for Python (Boto3) API Reference.

Rust

SDK per Rust

> **Note**
>
> C'è altro da fare. GitHub Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice AWS.

```
    pub async fn clean_up(self) -> Result<(), Vec<ScenarioError>> {
        let mut clean_up_errors: Vec<ScenarioError> = vec![];

        // Delete the instance. rds.DeleteDbInstance.
        let delete_db_instance = self
            .rds
            .delete_db_instance(
                self.db_instance_identifier
                    .as_deref()
                    .expect("instance identifier"),
            )
            .await;
        if let Err(err) = delete_db_instance {
            let identifier = self
```

```
                .db_instance_identifier
                .as_deref()
                .unwrap_or("Missing Instance Identifier");
            let message = format!("failed to delete db instance {identifier}");
            clean_up_errors.push(ScenarioError::new(message, &err));
        } else {
            // Wait for the instance to delete
            let waiter = Waiter::default();
            while waiter.sleep().await.is_ok() {
                let describe_db_instances =
self.rds.describe_db_instances().await;
                if let Err(err) = describe_db_instances {
                    clean_up_errors.push(ScenarioError::new(
                        "Failed to check instance state during deletion",
                        &err,
                    ));
                    break;
                }
                let db_instances = describe_db_instances
                    .unwrap()
                    .db_instances()
                    .iter()
                    .filter(|instance| instance.db_cluster_identifier ==
self.db_cluster_identifier)
                    .cloned()
                    .collect::<Vec<DbInstance>>();

                if db_instances.is_empty() {
                    trace!("Delete Instance waited and no instances were found");
                    break;
                }
                match db_instances.first().unwrap().db_instance_status() {
                    Some("Deleting") => continue,
                    Some(status) => {
                        info!("Attempting to delete but instances is in
{status}");
                        continue;
                    }
                    None => {
                        warn!("No status for DB instance");
                        break;
                    }
                }
            }
```

```
        }

        // Delete the DB cluster. rds.DeleteDbCluster.
        let delete_db_cluster = self
            .rds
            .delete_db_cluster(
                self.db_cluster_identifier
                    .as_deref()
                    .expect("cluster identifier"),
            )
            .await;

        if let Err(err) = delete_db_cluster {
            let identifier = self
                .db_cluster_identifier
                .as_deref()
                .unwrap_or("Missing DB Cluster Identifier");
            let message = format!("failed to delete db cluster {identifier}");
            clean_up_errors.push(ScenarioError::new(message, &err));
        } else {
            // Wait for the instance and cluster to fully delete.
rds.DescribeDbInstances and rds.DescribeDbClusters until both are not found.
            let waiter = Waiter::default();
            while waiter.sleep().await.is_ok() {
                let describe_db_clusters = self
                    .rds
                    .describe_db_clusters(
                        self.db_cluster_identifier
                            .as_deref()
                            .expect("cluster identifier"),
                    )
                    .await;
                if let Err(err) = describe_db_clusters {
                    clean_up_errors.push(ScenarioError::new(
                        "Failed to check cluster state during deletion",
                        &err,
                    ));
                    break;
                }
                let describe_db_clusters = describe_db_clusters.unwrap();
                let db_clusters = describe_db_clusters.db_clusters();
                if db_clusters.is_empty() {
                    trace!("Delete cluster waited and no clusters were found");
                    break;
```

```
                }
                match db_clusters.first().unwrap().status() {
                    Some("Deleting") => continue,
                    Some(status) => {
                        info!("Attempting to delete but clusters is in
{status}");
                        continue;
                    }
                    None => {
                        warn!("No status for DB cluster");
                        break;
                    }
                }
            }
        }

        // Delete the DB cluster parameter group.
rds.DeleteDbClusterParameterGroup.
        let delete_db_cluster_parameter_group = self
            .rds
            .delete_db_cluster_parameter_group(
                self.db_cluster_parameter_group
                    .map(|g| {
                        g.db_cluster_parameter_group_name
                            .unwrap_or_else(||
DB_CLUSTER_PARAMETER_GROUP_NAME.to_string())
                    })
                    .as_deref()
                    .expect("cluster parameter group name"),
            )
            .await;
        if let Err(error) = delete_db_cluster_parameter_group {
            clean_up_errors.push(ScenarioError::new(
                "Failed to delete the db cluster parameter group",
                &error,
            ))
        }

        if clean_up_errors.is_empty() {
            Ok(())
        } else {
            Err(clean_up_errors)
        }
    }
```

```
    pub async fn delete_db_instance(
        &self,
        instance_identifier: &str,
    ) -> Result<DeleteDbInstanceOutput, SdkError<DeleteDBInstanceError>> {
        self.inner
            .delete_db_instance()
            .db_instance_identifier(instance_identifier)
            .skip_final_snapshot(true)
            .send()
            .await
    }

#[tokio::test]
async fn test_scenario_clean_up() {
    let mut mock_rds = MockRdsImpl::default();

    mock_rds
        .expect_delete_db_instance()
        .with(eq("MockInstance"))
        .return_once(|_| Ok(DeleteDbInstanceOutput::builder().build()));

    mock_rds
        .expect_describe_db_instances()
        .with()
        .times(1)
        .returning(|| {
            Ok(DescribeDbInstancesOutput::builder()
                .db_instances(
                    DbInstance::builder()
                        .db_cluster_identifier("MockCluster")
                        .db_instance_status("Deleting")
                        .build(),
                )
                .build())
        })
        .with()
        .times(1)
        .returning(|| Ok(DescribeDbInstancesOutput::builder().build()));

    mock_rds
        .expect_delete_db_cluster()
        .with(eq("MockCluster"))
        .return_once(|_| Ok(DeleteDbClusterOutput::builder().build()));
```

```
    mock_rds
        .expect_describe_db_clusters()
        .with(eq("MockCluster"))
        .times(1)
        .returning(|id| {
            Ok(DescribeDbClustersOutput::builder()
                .db_clusters(
                    DbCluster::builder()
                        .db_cluster_identifier(id)
                        .status("Deleting")
                        .build(),
                )
                .build())
        })
        .with(eq("MockCluster"))
        .times(1)
        .returning(|_| Ok(DescribeDbClustersOutput::builder().build()));

    mock_rds
        .expect_delete_db_cluster_parameter_group()
        .with(eq("MockParamGroup"))
        .return_once(|_|
Ok(DeleteDbClusterParameterGroupOutput::builder().build()));

    let mut scenario = AuroraScenario::new(mock_rds);
    scenario.db_cluster_identifier = Some(String::from("MockCluster"));
    scenario.db_instance_identifier = Some(String::from("MockInstance"));
    scenario.db_cluster_parameter_group = Some(
        DbClusterParameterGroup::builder()
            .db_cluster_parameter_group_name("MockParamGroup")
            .build(),
    );

    tokio::time::pause();
    let assertions = tokio::spawn(async move {
        let clean_up = scenario.clean_up().await;
        assert!(clean_up.is_ok());
    });

    tokio::time::advance(Duration::from_secs(1)).await; // Wait for first
Describe Instances
    tokio::time::advance(Duration::from_secs(1)).await; // Wait for second
Describe Instances
```

```
    tokio::time::advance(Duration::from_secs(1)).await; // Wait for first
 Describe Cluster
    tokio::time::advance(Duration::from_secs(1)).await; // Wait for second
 Describe Cluster
    tokio::time::resume();
    let _ = assertions.await;
}

#[tokio::test]
async fn test_scenario_clean_up_errors() {
    let mut mock_rds = MockRdsImpl::default();

    mock_rds
        .expect_delete_db_instance()
        .with(eq("MockInstance"))
        .return_once(|_| Ok(DeleteDbInstanceOutput::builder().build()));

    mock_rds
        .expect_describe_db_instances()
        .with()
        .times(1)
        .returning(|| {
            Ok(DescribeDbInstancesOutput::builder()
                .db_instances(
                    DbInstance::builder()
                        .db_cluster_identifier("MockCluster")
                        .db_instance_status("Deleting")
                        .build(),
                )
                .build())
        })
        .with()
        .times(1)
        .returning(|| {
            Err(SdkError::service_error(
                DescribeDBInstancesError::unhandled(Box::new(Error::new(
                    ErrorKind::Other,
                    "describe db instances error",
                ))),
                Response::new(StatusCode::try_from(400).unwrap(),
 SdkBody::empty()),
            ))
        });
```

```
    mock_rds
        .expect_delete_db_cluster()
        .with(eq("MockCluster"))
        .return_once(|_| Ok(DeleteDbClusterOutput::builder().build()));

    mock_rds
        .expect_describe_db_clusters()
        .with(eq("MockCluster"))
        .times(1)
        .returning(|id| {
            Ok(DescribeDbClustersOutput::builder()
                .db_clusters(
                    DbCluster::builder()
                        .db_cluster_identifier(id)
                        .status("Deleting")
                        .build(),
                )
                .build())
        })
        .with(eq("MockCluster"))
        .times(1)
        .returning(|_| {
            Err(SdkError::service_error(
                DescribeDBClustersError::unhandled(Box::new(Error::new(
                    ErrorKind::Other,
                    "describe db clusters error",
                ))),
                Response::new(StatusCode::try_from(400).unwrap(),
SdkBody::empty()),
            ))
        });

    mock_rds
        .expect_delete_db_cluster_parameter_group()
        .with(eq("MockParamGroup"))
        .return_once(|_|
Ok(DeleteDbClusterParameterGroupOutput::builder().build()));

    let mut scenario = AuroraScenario::new(mock_rds);
    scenario.db_cluster_identifier = Some(String::from("MockCluster"));
    scenario.db_instance_identifier = Some(String::from("MockInstance"));
    scenario.db_cluster_parameter_group = Some(
        DbClusterParameterGroup::builder()
            .db_cluster_parameter_group_name("MockParamGroup")
```

```
            .build(),
    );

    tokio::time::pause();
    let assertions = tokio::spawn(async move {
        let clean_up = scenario.clean_up().await;
        assert!(clean_up.is_err());
        let errs = clean_up.unwrap_err();
        assert_eq!(errs.len(), 2);
        assert_matches!(errs.first(), Some(ScenarioError {message, context: _})
 if message == "Failed to check instance state during deletion");
        assert_matches!(errs.get(1), Some(ScenarioError {message, context: _}) if
 message == "Failed to check cluster state during deletion");
    });

    tokio::time::advance(Duration::from_secs(1)).await; // Wait for first
 Describe Instances
    tokio::time::advance(Duration::from_secs(1)).await; // Wait for second
 Describe Instances
    tokio::time::advance(Duration::from_secs(1)).await; // Wait for first
 Describe Cluster
    tokio::time::advance(Duration::from_secs(1)).await; // Wait for second
 Describe Cluster
    tokio::time::resume();
    let _ = assertions.await;
}
```

- Per i dettagli sull'API, consulta Delete DBInstance in AWSSDK for Rust API reference.

Per un elenco completo delle guide per sviluppatori AWS SDK e degli esempi di codice, consulta. Utilizzo di questo servizio con un AWS SDK Questo argomento include anche informazioni su come iniziare e dettagli sulle versioni precedenti dell'SDK.

Utilizzo **`DescribeDBClusterParameterGroups`** con un AWS SDK

Gli esempi di codice seguenti mostrano come utilizzare `DescribeDBClusterParameterGroups`.

Gli esempi di operazioni sono estratti di codice da programmi più grandi e devono essere eseguiti nel contesto. È possibile visualizzare questa operazione nel contesto nel seguente esempio di codice:

- Informazioni di base

.NET

SDK per .NET(v4)

> **Note**
>
> C'è altro da fare. GitHub Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice AWS.

```
    /// <summary>
    /// Get the description of a DB cluster parameter group by name.
    /// </summary>
    /// <param name="name">The name of the DB parameter group to describe.</
param>
    /// <returns>The parameter group description.</returns>
    public async Task<DBClusterParameterGroup?>
 DescribeCustomDBClusterParameterGroupAsync(string name)
    {
        var response = await _amazonRDS.DescribeDBClusterParameterGroupsAsync(
            new DescribeDBClusterParameterGroupsRequest()
            {
                DBClusterParameterGroupName = name
            });
        return response.DBClusterParameterGroups.FirstOrDefault();
    }
```

- Per i dettagli sull'API, consulta Descrivi DBCluster ParameterGroups in AWS SDK per .NETAPI Reference.

C++

SDK per C++

> **Note**
>
> C'è altro su GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice AWS.

```
        Aws::Client::ClientConfiguration clientConfig;
        // Optional: Set to the AWS Region (overrides config file).
        // clientConfig.region = "us-east-1";

    Aws::RDS::RDSClient client(clientConfig);

        Aws::RDS::Model::DescribeDBClusterParameterGroupsRequest request;
        request.SetDBClusterParameterGroupName(CLUSTER_PARAMETER_GROUP_NAME);

        Aws::RDS::Model::DescribeDBClusterParameterGroupsOutcome outcome =
                client.DescribeDBClusterParameterGroups(request);

        if (outcome.IsSuccess()) {
            std::cout << "DB cluster parameter group named '" <<
                      CLUSTER_PARAMETER_GROUP_NAME << "' already exists." <<
 std::endl;
            dbParameterGroupFamily =
 outcome.GetResult().GetDBClusterParameterGroups()
[0].GetDBParameterGroupFamily();
        }

        else {
            std::cerr << "Error with Aurora::DescribeDBClusterParameterGroups. "
                      << outcome.GetError().GetMessage()
                      << std::endl;
            return false;
        }
```

- Per i dettagli sull'API, consulta Descrivi DBCluster ParameterGroups in AWS SDK per C++ +API Reference.

Go

SDK per Go V2

> **Note**
>
> C'è altro su GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice AWS.

```
import (
 "context"
 "errors"
 "log"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/service/rds"
 "github.com/aws/aws-sdk-go-v2/service/rds/types"
)

type DbClusters struct {
 AuroraClient *rds.Client
}



// GetParameterGroup gets a DB cluster parameter group by name.
func (clusters *DbClusters) GetParameterGroup(ctx context.Context,
 parameterGroupName string) (
 *types.DBClusterParameterGroup, error) {
 output, err := clusters.AuroraClient.DescribeDBClusterParameterGroups(
  ctx, &rds.DescribeDBClusterParameterGroupsInput{
   DBClusterParameterGroupName: aws.String(parameterGroupName),
  })
 if err != nil {
  var notFoundError *types.DBParameterGroupNotFoundFault
  if errors.As(err, &notFoundError) {
   log.Printf("Parameter group %v does not exist.\n", parameterGroupName)
   err = nil
  } else {
   log.Printf("Error getting parameter group %v: %v\n", parameterGroupName, err)
  }
  return nil, err
 } else {
  return &output.DBClusterParameterGroups[0], err
 }
}
```

- Per i dettagli sull'API, consulta Descrivi DBCluster ParameterGroups in AWS SDK per GoAPI Reference.

Java

SDK per Java 2.x

> **Note**
>
> C'è altro su GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice AWS.

```
   public static void describeDbClusterParameterGroups(RdsClient rdsClient,
String dbClusterGroupName) {
       try {
           DescribeDbClusterParameterGroupsRequest groupsRequest =
DescribeDbClusterParameterGroupsRequest.builder()
                   .dbClusterParameterGroupName(dbClusterGroupName)
                   .maxRecords(20)
                   .build();

           List<DBClusterParameterGroup> groups =
rdsClient.describeDBClusterParameterGroups(groupsRequest)
                   .dbClusterParameterGroups();
           for (DBClusterParameterGroup group : groups) {
               System.out.println("The group name is " +
group.dbClusterParameterGroupName());
               System.out.println("The group ARN is " +
group.dbClusterParameterGroupArn());
           }

       } catch (RdsException e) {
           System.out.println(e.getLocalizedMessage());
           System.exit(1);
       }
   }
```

- Per i dettagli sull'API, consulta Descrivi DBCluster ParameterGroups in AWS SDK for Java 2.xAPI Reference.

Kotlin

SDK per Kotlin

> **Note**
>
> C'è altro su GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice AWS.

```
suspend fun describeDbClusterParameterGroups(dbClusterGroupName: String?) {
    val groupsRequest =
        DescribeDbClusterParameterGroupsRequest {
            dbClusterParameterGroupName = dbClusterGroupName
            maxRecords = 20
        }

    RdsClient.fromEnvironment { region = "us-west-2" }.use { rdsClient ->
        val response = rdsClient.describeDbClusterParameterGroups(groupsRequest)
        response.dbClusterParameterGroups?.forEach { group ->
            println("The group name is ${group.dbClusterParameterGroupName}")
            println("The group ARN is ${group.dbClusterParameterGroupArn}")
        }
    }
}
```

- Per i dettagli sull'API, consulta Descrivi DBCluster ParameterGroups in AWSSDK per il riferimento all'API Kotlin.

Python

SDK per Python (Boto3)

> **Note**
>
> C'è altro su. GitHub Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice AWS.

```
class AuroraWrapper:
    """Encapsulates Aurora DB cluster actions."""

    def __init__(self, rds_client):
        """
        :param rds_client: A Boto3 Amazon Relational Database Service (Amazon
 RDS) client.
        """
        self.rds_client = rds_client

    @classmethod
    def from_client(cls):
        """
        Instantiates this class from a Boto3 client.
        """
        rds_client = boto3.client("rds")
        return cls(rds_client)


    def get_parameter_group(self, parameter_group_name):
        """
        Gets a DB cluster parameter group.

        :param parameter_group_name: The name of the parameter group to retrieve.
        :return: The requested parameter group.
        """
        try:
            response = self.rds_client.describe_db_cluster_parameter_groups(
                DBClusterParameterGroupName=parameter_group_name
            )
            parameter_group = response["DBClusterParameterGroups"][0]
        except ClientError as err:
            if err.response["Error"]["Code"] == "DBParameterGroupNotFound":
                logger.info("Parameter group %s does not exist.",
 parameter_group_name)
            else:
                logger.error(
                    "Couldn't get parameter group %s. Here's why: %s: %s",
                    parameter_group_name,
                    err.response["Error"]["Code"],
                    err.response["Error"]["Message"],
                )
                raise
```

```
        else:
            return parameter_group
```

- Per i dettagli sull'API, consulta Descrivi DBCluster ParameterGroups in AWSSDK for Python (Boto3) API Reference.

Per un elenco completo delle guide per sviluppatori AWS SDK e degli esempi di codice, consulta. Utilizzo di questo servizio con un AWS SDK Questo argomento include anche informazioni su come iniziare e dettagli sulle versioni precedenti dell'SDK.

Utilizzo **`DescribeDBClusterParameters`** con un AWS SDK

Gli esempi di codice seguenti mostrano come utilizzare `DescribeDBClusterParameters`.

Gli esempi di operazioni sono estratti di codice da programmi più grandi e devono essere eseguiti nel contesto. È possibile visualizzare questa operazione nel contesto nel seguente esempio di codice:

- Informazioni di base

.NET

SDK per .NET(v4)

> **Note**
>
> C'è altro da fare. GitHub Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice AWS.

```
    /// <summary>
    /// Describe the cluster parameters in a parameter group.
    /// </summary>
    /// <param name="groupName">The name of the parameter group.</param>
    /// <param name="source">The optional name of the source filter.</param>
    /// <returns>The collection of parameters.</returns>
    public async Task<List<Parameter>>
DescribeDBClusterParametersInGroupAsync(string groupName, string? source = null)
    {
```

```
        var paramList = new List<Parameter>();

        DescribeDBClusterParametersResponse response;
        var request = new DescribeDBClusterParametersRequest
        {
            DBClusterParameterGroupName = groupName,
            Source = source,
        };

        // Get the full list if there are multiple pages.
        do
        {
            response = await
_amazonRDS.DescribeDBClusterParametersAsync(request);
            paramList.AddRange(response.Parameters);

            request.Marker = response.Marker;
        }
        while (response.Marker is not null);

        return paramList;
    }
```

- Per i dettagli sull'API, consulta DBClusterDescrivi i parametri in AWS SDK per .NETAPI Reference.

C++

SDK per C++

> **Note**
>
> C'è altro su GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice AWS.

```
        Aws::Client::ClientConfiguration clientConfig;
        // Optional: Set to the AWS Region (overrides config file).
        // clientConfig.region = "us-east-1";
```

```
    Aws::RDS::RDSClient client(clientConfig);


//! Routine which gets DB parameters using the 'DescribeDBClusterParameters' api.
/*!
 \sa getDBCLusterParameters()
 \param parameterGroupName: The name of the cluster parameter group.
 \param namePrefix: Prefix string to filter results by parameter name.
 \param source: A source such as 'user', ignored if empty.
 \param parametersResult: Vector of 'Parameter' objects returned by the routine.
 \param client: 'RDSClient' instance.
 \return bool: Successful completion.
 */
bool AwsDoc::Aurora::getDBCLusterParameters(const Aws::String
 &parameterGroupName,
                                             const Aws::String &namePrefix,
                                             const Aws::String &source,

 Aws::Vector<Aws::RDS::Model::Parameter> &parametersResult,
                                             const Aws::RDS::RDSClient &client) {
    Aws::String marker; // The marker is used for pagination.
    do {
        Aws::RDS::Model::DescribeDBClusterParametersRequest request;
        request.SetDBClusterParameterGroupName(CLUSTER_PARAMETER_GROUP_NAME);
        if (!marker.empty()) {
            request.SetMarker(marker);
        }
        if (!source.empty()) {
            request.SetSource(source);
        }

        Aws::RDS::Model::DescribeDBClusterParametersOutcome outcome =
                client.DescribeDBClusterParameters(request);

        if (outcome.IsSuccess()) {
            const Aws::Vector<Aws::RDS::Model::Parameter> &parameters =
                    outcome.GetResult().GetParameters();
            for (const Aws::RDS::Model::Parameter &parameter: parameters) {
                if (!namePrefix.empty()) {
                    if (parameter.GetParameterName().find(namePrefix) == 0) {
                        parametersResult.push_back(parameter);
                    }
                }
                else {
```

```
                    parametersResult.push_back(parameter);
                }
            }

            marker = outcome.GetResult().GetMarker();
        }
        else {
            std::cerr << "Error with Aurora::DescribeDBClusterParameters. "
                      << outcome.GetError().GetMessage()
                      << std::endl;
            return false;
        }
    } while (!marker.empty());

    return true;
}
```

- Per i dettagli sull'API, consulta DBClusterDescrivi i parametri in AWS SDK per C++API Reference.

Go

SDK per Go V2

> **Note**
>
> C'è altro su GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice AWS.

```
import (
 "context"
 "errors"
 "log"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/service/rds"
 "github.com/aws/aws-sdk-go-v2/service/rds/types"
)
```

```
type DbClusters struct {
 AuroraClient *rds.Client
}



// GetParameters gets the parameters that are contained in a DB cluster parameter
 group.
func (clusters *DbClusters) GetParameters(ctx context.Context, parameterGroupName
 string, source string) (
 []types.Parameter, error) {

 var output *rds.DescribeDBClusterParametersOutput
 var params []types.Parameter
 var err error
 parameterPaginator :=
 rds.NewDescribeDBClusterParametersPaginator(clusters.AuroraClient,
  &rds.DescribeDBClusterParametersInput{
   DBClusterParameterGroupName: aws.String(parameterGroupName),
   Source:                      aws.String(source),
  })
 for parameterPaginator.HasMorePages() {
  output, err = parameterPaginator.NextPage(ctx)
  if err != nil {
   log.Printf("Couldn't get paramaeters for %v: %v\n", parameterGroupName, err)
   break
  } else {
   params = append(params, output.Parameters...)
  }
 }
 return params, err
}
```

- Per i dettagli sull'API, consulta DBClusterDescrivi i parametri in AWS SDK per GoAPI Reference.

Java

SDK per Java 2.x

> **Note**
>
> C'è altro su GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice AWS.

```
   public static void describeDbClusterParameters(RdsClient rdsClient, String
dbCLusterGroupName, int flag) {
       try {
           DescribeDbClusterParametersRequest dbParameterGroupsRequest;
           if (flag == 0) {
               dbParameterGroupsRequest =
DescribeDbClusterParametersRequest.builder()
                       .dbClusterParameterGroupName(dbCLusterGroupName)
                       .build();
           } else {
               dbParameterGroupsRequest =
DescribeDbClusterParametersRequest.builder()
                       .dbClusterParameterGroupName(dbCLusterGroupName)
                       .source("user")
                       .build();
           }

           DescribeDbClusterParametersResponse response = rdsClient
                   .describeDBClusterParameters(dbParameterGroupsRequest);
           List<Parameter> dbParameters = response.parameters();
           String paraName;
           for (Parameter para : dbParameters) {
               // Only print out information about either auto_increment_offset
or
               // auto_increment_increment.
               paraName = para.parameterName();
               if ((paraName.compareTo("auto_increment_offset") == 0)
                       || (paraName.compareTo("auto_increment_increment ") ==
0)) {
                   System.out.println("*** The parameter name is  " + paraName);
                   System.out.println("*** The parameter value is  " +
para.parameterValue());
```

```
                    System.out.println("*** The parameter data type is " +
 para.dataType());
                    System.out.println("*** The parameter description is " +
 para.description());
                    System.out.println("*** The parameter allowed values  is " +
 para.allowedValues());
                }
            }

        } catch (RdsException e) {
            System.out.println(e.getLocalizedMessage());
            System.exit(1);
        }
    }
```

- Per i dettagli sull'API, consulta DBClusterDescrivi i parametri in AWS SDK for Java 2.xAPI Reference.

Kotlin

SDK per Kotlin

> **Note**
>
> C'è altro su GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice AWS.

```
suspend fun describeDbClusterParameters(
    dbCLusterGroupName: String?,
    flag: Int,
) {
    val dbParameterGroupsRequest: DescribeDbClusterParametersRequest
    dbParameterGroupsRequest =
        if (flag == 0) {
            DescribeDbClusterParametersRequest {
                dbClusterParameterGroupName = dbCLusterGroupName
            }
        } else {
            DescribeDbClusterParametersRequest {
```

```
                dbClusterParameterGroupName = dbCLusterGroupName
                source = "user"
            }
        }

    RdsClient.fromEnvironment { region = "us-west-2" }.use { rdsClient ->
        val response =
rdsClient.describeDbClusterParameters(dbParameterGroupsRequest)
        response.parameters?.forEach { para ->
            // Only print out information about either auto_increment_offset or
auto_increment_increment.
            val paraName = para.parameterName
            if (paraName != null) {
                if (paraName.compareTo("auto_increment_offset") == 0 ||
paraName.compareTo("auto_increment_increment ") == 0) {
                    println("*** The parameter name is  $paraName")
                    println("*** The parameter value is  ${para.parameterValue}")
                    println("*** The parameter data type is ${para.dataType}")
                    println("*** The parameter description is
${para.description}")
                    println("*** The parameter allowed values  is
${para.allowedValues}")
                }
            }
        }
    }
}
```

- Per i dettagli sull'API, consulta DBClusterDescrivi i parametri nell'AWSSDK per il riferimento all'API Kotlin.

Python

SDK per Python (Boto3)

> **Note**
>
> C'è di più su. GitHub Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice AWS.

```
class AuroraWrapper:
    """Encapsulates Aurora DB cluster actions."""

    def __init__(self, rds_client):
        """
        :param rds_client: A Boto3 Amazon Relational Database Service (Amazon
 RDS) client.
        """
        self.rds_client = rds_client

    @classmethod
    def from_client(cls):
        """
        Instantiates this class from a Boto3 client.
        """
        rds_client = boto3.client("rds")
        return cls(rds_client)


    def get_parameters(self, parameter_group_name, name_prefix="", source=None):
        """
        Gets the parameters that are contained in a DB cluster parameter group.

        :param parameter_group_name: The name of the parameter group to query.
        :param name_prefix: When specified, the retrieved list of parameters is
 filtered
                            to contain only parameters that start with this
 prefix.
        :param source: When specified, only parameters from this source are
 retrieved.
                       For example, a source of 'user' retrieves only parameters
 that
                       were set by a user.
        :return: The list of requested parameters.
        """
        try:
            kwargs = {"DBClusterParameterGroupName": parameter_group_name}
            if source is not None:
                kwargs["Source"] = source
            parameters = []
            paginator =
 self.rds_client.get_paginator("describe_db_cluster_parameters")
            for page in paginator.paginate(**kwargs):
```

```
                parameters += [
                    p
                    for p in page["Parameters"]
                    if p["ParameterName"].startswith(name_prefix)
                ]
        except ClientError as err:
            logger.error(
                "Couldn't get parameters for %s. Here's why: %s: %s",
                parameter_group_name,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
        else:
            return parameters
```

- Per i dettagli sull'API, consulta DBClusterDescrivi i parametri nella AWSguida di riferimento all'API SDK for Python (Boto3).

Rust

SDK per Rust

> **Note**
>
> C'è altro da fare. GitHub Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice AWS.

```
   // Get the parameter group. rds.DescribeDbClusterParameterGroups
   // Get parameters in the group. This is a long list so you will have to
paginate. Find the auto_increment_offset and auto_increment_increment parameters
(by ParameterName). rds.DescribeDbClusterParameters
   // Parse the ParameterName, Description, and AllowedValues values and display
them.
   pub async fn cluster_parameters(&self) ->
Result<Vec<AuroraScenarioParameter>, ScenarioError> {
       let parameters_output = self
           .rds
```

```
            .describe_db_cluster_parameters(DB_CLUSTER_PARAMETER_GROUP_NAME)
            .await;

        if let Err(err) = parameters_output {
            return Err(ScenarioError::new(
                format!("Failed to retrieve parameters for
 {DB_CLUSTER_PARAMETER_GROUP_NAME}"),
                &err,
            ));
        }

        let parameters = parameters_output
            .unwrap()
            .into_iter()
            .flat_map(|p| p.parameters.unwrap_or_default().into_iter())
            .filter(|p|
 FILTER_PARAMETER_NAMES.contains(p.parameter_name().unwrap_or_default()))
            .map(AuroraScenarioParameter::from)
            .collect::<Vec<_>>();

        Ok(parameters)
    }

    pub async fn describe_db_cluster_parameters(
        &self,
        name: &str,
    ) -> Result<Vec<DescribeDbClusterParametersOutput>,
 SdkError<DescribeDBClusterParametersError>>
    {
        self.inner
            .describe_db_cluster_parameters()
            .db_cluster_parameter_group_name(name)
            .into_paginator()
            .send()
            .try_collect()
            .await
    }

#[tokio::test]
async fn test_scenario_cluster_parameters() {
    let mut mock_rds = MockRdsImpl::default();

    mock_rds
        .expect_describe_db_cluster_parameters()
```

```
        .with(eq("RustSDKCodeExamplesDBParameterGroup"))
        .return_once(|_| {
            Ok(vec![DescribeDbClusterParametersOutput::builder()
                .parameters(Parameter::builder().parameter_name("a").build())
                .parameters(Parameter::builder().parameter_name("b").build())
                .parameters(
                    Parameter::builder()
                        .parameter_name("auto_increment_offset")
                        .build(),
                )
                .parameters(Parameter::builder().parameter_name("c").build())
                .parameters(
                    Parameter::builder()
                        .parameter_name("auto_increment_increment")
                        .build(),
                )
                .parameters(Parameter::builder().parameter_name("d").build())
                .build()])
        });

    let mut scenario = AuroraScenario::new(mock_rds);
    scenario.db_cluster_identifier = Some("RustSDKCodeExamplesDBCluster".into());

    let params = scenario.cluster_parameters().await.expect("cluster params");
    let names: Vec<String> = params.into_iter().map(|p| p.name).collect();
    assert_eq!(
        names,
        vec!["auto_increment_offset", "auto_increment_increment"]
    );
}

#[tokio::test]
async fn test_scenario_cluster_parameters_error() {
    let mut mock_rds = MockRdsImpl::default();

    mock_rds
        .expect_describe_db_cluster_parameters()
        .with(eq("RustSDKCodeExamplesDBParameterGroup"))
        .return_once(|_| {
            Err(SdkError::service_error(
                DescribeDBClusterParametersError::unhandled(Box::new(Error::new(
                    ErrorKind::Other,
                    "describe_db_cluster_parameters_error",
                ))),
```

```
                Response::new(StatusCode::try_from(400).unwrap(),
 SdkBody::empty()),
            ))
        });

    let mut scenario = AuroraScenario::new(mock_rds);
    scenario.db_cluster_identifier = Some("RustSDKCodeExamplesDBCluster".into());
    let params = scenario.cluster_parameters().await;
    assert_matches!(params, Err(ScenarioError { message, context: _ }) if message
 == "Failed to retrieve parameters for RustSDKCodeExamplesDBParameterGroup");
}
```

- Per i dettagli sull'API, consulta DBClusterDescrivi i parametri nell'AWSSDK for Rust API reference.

Per un elenco completo delle guide per sviluppatori AWS SDK e degli esempi di codice, consulta. Utilizzo di questo servizio con un AWS SDK Questo argomento include anche informazioni su come iniziare e dettagli sulle versioni precedenti dell'SDK.

Utilizzo **`DescribeDBClusterSnapshots`** con un AWS SDK

Gli esempi di codice seguenti mostrano come utilizzare `DescribeDBClusterSnapshots`.

Gli esempi di operazioni sono estratti di codice da programmi più grandi e devono essere eseguiti nel contesto. È possibile visualizzare questa operazione nel contesto nel seguente esempio di codice:

- Informazioni di base

.NET

SDK per .NET(v4)

> **Note**
>
> C'è altro da fare. GitHub Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice AWS.

```
    /// <summary>
```

```
    /// Return a list of DB snapshots for a particular DB cluster.
    /// </summary>
    /// <param name="dbClusterIdentifier">DB cluster identifier.</param>
    /// <returns>List of DB snapshots.</returns>
    public async Task<List<DBClusterSnapshot>>
DescribeDBClusterSnapshotsByIdentifierAsync(string dbClusterIdentifier)
    {
        var results = new List<DBClusterSnapshot>();

        DescribeDBClusterSnapshotsResponse response;
        DescribeDBClusterSnapshotsRequest request = new
DescribeDBClusterSnapshotsRequest
        {
            DBClusterIdentifier = dbClusterIdentifier
        };
        // Get the full list if there are multiple pages.
        do
        {
            response = await _amazonRDS.DescribeDBClusterSnapshotsAsync(request);
            results.AddRange(response.DBClusterSnapshots);
            request.Marker = response.Marker;
        }
        while (response.Marker is not null);
        return results;
    }
```

- Per i dettagli sull'API, consulta Descrivi le DBCluster istantanee in AWS SDK per .NETAPI Reference.

C++

SDK per C++

> **Note**
>
> C'è altro su. GitHub Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice AWS.

```
        Aws::Client::ClientConfiguration clientConfig;
```

```
        // Optional: Set to the AWS Region (overrides config file).
        // clientConfig.region = "us-east-1";

    Aws::RDS::RDSClient client(clientConfig);

            Aws::RDS::Model::DescribeDBClusterSnapshotsRequest request;
            request.SetDBClusterSnapshotIdentifier(snapshotID);

            Aws::RDS::Model::DescribeDBClusterSnapshotsOutcome outcome =
                    client.DescribeDBClusterSnapshots(request);

            if (outcome.IsSuccess()) {
                snapshot = outcome.GetResult().GetDBClusterSnapshots()[0];
            }
            else {
                std::cerr << "Error with Aurora::DescribeDBClusterSnapshots. "
                          << outcome.GetError().GetMessage()
                          << std::endl;
                cleanUpResources(CLUSTER_PARAMETER_GROUP_NAME,
                                 DB_CLUSTER_IDENTIFIER, DB_INSTANCE_IDENTIFIER,
client);
                return false;
            }
```

- Per i dettagli sull'API, consulta Descrivi le DBCluster istantanee in AWS SDK per C++API Reference.

Go

SDK per Go V2

> **Note**
>
> C'è altro su. GitHub Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice AWS.

```
import (
 "context"
```

```
 "errors"
 "log"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/service/rds"
 "github.com/aws/aws-sdk-go-v2/service/rds/types"
)

type DbClusters struct {
 AuroraClient *rds.Client
}



// GetClusterSnapshot gets a DB cluster snapshot.
func (clusters *DbClusters) GetClusterSnapshot(ctx context.Context, snapshotName
 string) (*types.DBClusterSnapshot, error) {
 output, err := clusters.AuroraClient.DescribeDBClusterSnapshots(ctx,
  &rds.DescribeDBClusterSnapshotsInput{
   DBClusterSnapshotIdentifier: aws.String(snapshotName),
  })
 if err != nil {
  log.Printf("Couldn't get snapshot %v: %v\n", snapshotName, err)
  return nil, err
 } else {
  return &output.DBClusterSnapshots[0], nil
 }
}
```

- Per i dettagli sull'API, consulta Descrivi le DBCluster istantanee in AWS SDK per GoAPI Reference.

Java

SDK per Java 2.x

> **Note**
>
> C'è altro su. GitHub Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice AWS.

```
   public static void waitForSnapshotReady(RdsClient rdsClient, String
dbSnapshotIdentifier,
           String dbInstanceClusterIdentifier) {
       try {
           boolean snapshotReady = false;
           String snapshotReadyStr;
           System.out.println("Waiting for the snapshot to become available.");

           DescribeDbClusterSnapshotsRequest snapshotsRequest =
DescribeDbClusterSnapshotsRequest.builder()
                   .dbClusterSnapshotIdentifier(dbSnapshotIdentifier)
                   .dbClusterIdentifier(dbInstanceClusterIdentifier)
                   .build();

           while (!snapshotReady) {
               DescribeDbClusterSnapshotsResponse response =
rdsClient.describeDBClusterSnapshots(snapshotsRequest);
               List<DBClusterSnapshot> snapshotList =
response.dbClusterSnapshots();
               for (DBClusterSnapshot snapshot : snapshotList) {
                   snapshotReadyStr = snapshot.status();
                   if (snapshotReadyStr.contains("available")) {
                       snapshotReady = true;
                   } else {
                       System.out.println(".");
                       Thread.sleep(sleepTime * 5000);
                   }
               }
           }

           System.out.println("The Snapshot is available!");
```

```
        } catch (RdsException | InterruptedException e) {
            System.out.println(e.getLocalizedMessage());
            System.exit(1);
        }
    }
```

- Per i dettagli sull'API, consulta Descrivi le DBCluster istantanee in AWS SDK for Java 2.xAPI Reference.

Kotlin

SDK per Kotlin

> **Note**
>
> C'è altro su. GitHub Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice AWS.

```
suspend fun waitSnapshotReady(
    dbSnapshotIdentifier: String?,
    dbInstanceClusterIdentifier: String?,
) {
    var snapshotReady = false
    var snapshotReadyStr: String
    println("Waiting for the snapshot to become available.")

    val snapshotsRequest =
        DescribeDbClusterSnapshotsRequest {
            dbClusterSnapshotIdentifier = dbSnapshotIdentifier
            dbClusterIdentifier = dbInstanceClusterIdentifier
        }

    RdsClient.fromEnvironment { region = "us-west-2" }.use { rdsClient ->
        while (!snapshotReady) {
            val response = rdsClient.describeDbClusterSnapshots(snapshotsRequest)
            val snapshotList = response.dbClusterSnapshots
            if (snapshotList != null) {
                for (snapshot in snapshotList) {
                    snapshotReadyStr = snapshot.status.toString()
```

```
                    if (snapshotReadyStr.contains("available")) {
                        snapshotReady = true
                    } else {
                        println(".")
                        delay(slTime * 5000)
                    }
                }
            }
        }
    }
    println("The Snapshot is available!")
}
```

- Per i dettagli sull'API, consulta Descrivi le DBCluster istantanee nell'AWSSDK per il riferimento all'API Kotlin.

Python

SDK per Python (Boto3)

> **Note**
>
> C'è di più su. GitHub Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice AWS.

```
class AuroraWrapper:
    """Encapsulates Aurora DB cluster actions."""

    def __init__(self, rds_client):
        """
        :param rds_client: A Boto3 Amazon Relational Database Service (Amazon
 RDS) client.
        """
        self.rds_client = rds_client

    @classmethod
    def from_client(cls):
        """
        Instantiates this class from a Boto3 client.
```

```
        """
        rds_client = boto3.client("rds")
        return cls(rds_client)


    def get_cluster_snapshot(self, snapshot_id):
        """
        Gets a DB cluster snapshot.

        :param snapshot_id: The ID of the snapshot to retrieve.
        :return: The retrieved snapshot.
        """
        try:
            response = self.rds_client.describe_db_cluster_snapshots(
                DBClusterSnapshotIdentifier=snapshot_id
            )
            snapshot = response["DBClusterSnapshots"][0]
        except ClientError as err:
            logger.error(
                "Couldn't get DB cluster snapshot %s. Here's why: %s: %s",
                snapshot_id,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
        else:
            return snapshot
```

- Per i dettagli sull'API, consulta Descrivi le DBCluster istantanee nella guida di riferimento all'API AWSSDK for Python (Boto3).

Per un elenco completo delle guide per sviluppatori AWS SDK e degli esempi di codice, consulta. Utilizzo di questo servizio con un AWS SDK Questo argomento include anche informazioni su come iniziare e dettagli sulle versioni precedenti dell'SDK.

Utilizzo **`DescribeDBClusters`** con un AWS SDK

Gli esempi di codice seguenti mostrano come utilizzare `DescribeDBClusters`.

Gli esempi di operazioni sono estratti di codice da programmi più grandi e devono essere eseguiti nel contesto. È possibile visualizzare questa operazione nel contesto nel seguente esempio di codice:

- Informazioni di base

.NET

SDK per .NET(v4)

> **Note**
>
> C'è altro da fare. GitHub Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice AWS.

```
   /// <summary>
   /// Returns a list of DB clusters.
   /// </summary>
   /// <param name="dbInstanceIdentifier">Optional name of a specific DB
cluster.</param>
   /// <returns>List of DB clusters.</returns>
   public async Task<List<DBCluster>> DescribeDBClustersPagedAsync(string?
dbClusterIdentifier = null)
   {
       var results = new List<DBCluster>();

       DescribeDBClustersResponse response;
       DescribeDBClustersRequest request = new DescribeDBClustersRequest
       {
           DBClusterIdentifier = dbClusterIdentifier
       };
       // Get the full list if there are multiple pages.
       do
       {
           response = await _amazonRDS.DescribeDBClustersAsync(request);
           if (response.DBClusters != null)
           {
               results.AddRange(response.DBClusters);
           }
           request.Marker = response.Marker;
       }
```

```
        while (response.Marker is not null);
        return results;
    }
```

- Per i dettagli sull'API, consulta Descrivi DBClusters in AWS SDK per .NETAPI Reference.

C++

SDK per C++

> **Note**
>
> C'è altro su GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice AWS.

```
        Aws::Client::ClientConfiguration clientConfig;
        // Optional: Set to the AWS Region (overrides config file).
        // clientConfig.region = "us-east-1";

    Aws::RDS::RDSClient client(clientConfig);

//! Routine which gets a DB cluster description.
/*!
 \sa describeDBCluster()
 \param dbClusterIdentifier: A DB cluster identifier.
 \param clusterResult: The 'DBCluster' object containing the description.
 \param client: 'RDSClient' instance.
 \return bool: Successful completion.
 */
bool AwsDoc::Aurora::describeDBCluster(const Aws::String &dbClusterIdentifier,
                                       Aws::RDS::Model::DBCluster &clusterResult,
                                       const Aws::RDS::RDSClient &client) {
    Aws::RDS::Model::DescribeDBClustersRequest request;
    request.SetDBClusterIdentifier(dbClusterIdentifier);

    Aws::RDS::Model::DescribeDBClustersOutcome outcome =
            client.DescribeDBClusters(request);

    bool result = true;
    if (outcome.IsSuccess()) {
```

```
        clusterResult = outcome.GetResult().GetDBClusters()[0];
    }
    else if (outcome.GetError().GetErrorType() !=
             Aws::RDS::RDSErrors::D_B_CLUSTER_NOT_FOUND_FAULT) {
        result = false;
        std::cerr << "Error with Aurora::GDescribeDBClusters. "
                  << outcome.GetError().GetMessage()
                  << std::endl;
    }
        // This example does not log an error if the DB cluster does not exist.
        // Instead, clusterResult is set to empty.
    else {
        clusterResult = Aws::RDS::Model::DBCluster();
    }

    return result;

}
```

- Per i dettagli sull'API, consulta Descrivi DBClusters in AWS SDK per C++API Reference.

Go

SDK per Go V2

> **Note**
>
> C'è altro su GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice AWS.

```
import (
 "context"
 "errors"
 "log"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/service/rds"
 "github.com/aws/aws-sdk-go-v2/service/rds/types"
)
```

```
type DbClusters struct {
 AuroraClient *rds.Client
}



// GetDbCluster gets data about an Aurora DB cluster.
func (clusters *DbClusters) GetDbCluster(ctx context.Context, clusterName string)
 (*types.DBCluster, error) {
 output, err := clusters.AuroraClient.DescribeDBClusters(ctx,
  &rds.DescribeDBClustersInput{
   DBClusterIdentifier: aws.String(clusterName),
  })
 if err != nil {
  var notFoundError *types.DBClusterNotFoundFault
  if errors.As(err, &notFoundError) {
   log.Printf("DB cluster %v does not exist.\n", clusterName)
   err = nil
  } else {
   log.Printf("Couldn't get DB cluster %v: %v\n", clusterName, err)
  }
  return nil, err
 } else {
  return &output.DBClusters[0], err
 }
}
```

- Per i dettagli sull'API, consulta Descrivi DBClusters in AWS SDK per GoAPI Reference.

Java

SDK per Java 2.x

> **Note**
>
> C'è altro su GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice AWS.

```
   public static void describeDbClusterParameters(RdsClient rdsClient, String
dbCLusterGroupName, int flag) {
       try {
           DescribeDbClusterParametersRequest dbParameterGroupsRequest;
           if (flag == 0) {
               dbParameterGroupsRequest =
DescribeDbClusterParametersRequest.builder()
                       .dbClusterParameterGroupName(dbCLusterGroupName)
                       .build();
           } else {
               dbParameterGroupsRequest =
DescribeDbClusterParametersRequest.builder()
                       .dbClusterParameterGroupName(dbCLusterGroupName)
                       .source("user")
                       .build();
           }

           DescribeDbClusterParametersResponse response = rdsClient
                   .describeDBClusterParameters(dbParameterGroupsRequest);
           List<Parameter> dbParameters = response.parameters();
           String paraName;
           for (Parameter para : dbParameters) {
               // Only print out information about either auto_increment_offset
or
               // auto_increment_increment.
               paraName = para.parameterName();
               if ((paraName.compareTo("auto_increment_offset") == 0)
                       || (paraName.compareTo("auto_increment_increment ") ==
0)) {
                   System.out.println("*** The parameter name is  " + paraName);
                   System.out.println("*** The parameter value is  " +
para.parameterValue());
                   System.out.println("*** The parameter data type is " +
para.dataType());
                   System.out.println("*** The parameter description is " +
para.description());
                   System.out.println("*** The parameter allowed values  is " +
para.allowedValues());
               }
           }

       } catch (RdsException e) {
           System.out.println(e.getLocalizedMessage());
```

```
            System.exit(1);
        }
    }
```

- Per i dettagli sull'API, consulta Descrivi DBClusters in AWS SDK for Java 2.xAPI Reference.

Kotlin

SDK per Kotlin

> **Note**
>
> C'è altro su GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice AWS.

```
suspend fun describeDbClusterParameters(
    dbCLusterGroupName: String?,
    flag: Int,
) {
    val dbParameterGroupsRequest: DescribeDbClusterParametersRequest
    dbParameterGroupsRequest =
        if (flag == 0) {
            DescribeDbClusterParametersRequest {
                dbClusterParameterGroupName = dbCLusterGroupName
            }
        } else {
            DescribeDbClusterParametersRequest {
                dbClusterParameterGroupName = dbCLusterGroupName
                source = "user"
            }
        }

    RdsClient.fromEnvironment { region = "us-west-2" }.use { rdsClient ->
        val response =
 rdsClient.describeDbClusterParameters(dbParameterGroupsRequest)
        response.parameters?.forEach { para ->
            // Only print out information about either auto_increment_offset or
 auto_increment_increment.
            val paraName = para.parameterName
            if (paraName != null) {
```

```
                if (paraName.compareTo("auto_increment_offset") == 0 ||
 paraName.compareTo("auto_increment_increment ") == 0) {
                    println("*** The parameter name is  $paraName")
                    println("*** The parameter value is  ${para.parameterValue}")
                    println("*** The parameter data type is ${para.dataType}")
                    println("*** The parameter description is
 ${para.description}")
                    println("*** The parameter allowed values  is
 ${para.allowedValues}")
                }
            }
        }
    }
}
```

- Per i dettagli sull'API, consulta Descrivi DBClusters in AWSSDK per il riferimento all'API Kotlin.

Python

SDK per Python (Boto3)

> **Note**
>
> C'è altro su. GitHub Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice AWS.

```
class AuroraWrapper:
    """Encapsulates Aurora DB cluster actions."""

    def __init__(self, rds_client):
        """
        :param rds_client: A Boto3 Amazon Relational Database Service (Amazon
 RDS) client.
        """
        self.rds_client = rds_client

    @classmethod
    def from_client(cls):
```

```
        """
        Instantiates this class from a Boto3 client.
        """
        rds_client = boto3.client("rds")
        return cls(rds_client)


    def get_db_cluster(self, cluster_name):
        """
        Gets data about an Aurora DB cluster.

        :param cluster_name: The name of the DB cluster to retrieve.
        :return: The retrieved DB cluster.
        """
        try:
            response = self.rds_client.describe_db_clusters(
                DBClusterIdentifier=cluster_name
            )
            cluster = response["DBClusters"][0]
        except ClientError as err:
            if err.response["Error"]["Code"] == "DBClusterNotFoundFault":
                logger.info("Cluster %s does not exist.", cluster_name)
            else:
                logger.error(
                    "Couldn't verify the existence of DB cluster %s. Here's why:
%s: %s",
                    cluster_name,
                    err.response["Error"]["Code"],
                    err.response["Error"]["Message"],
                )
                raise
        else:
            return cluster
```

- Per i dettagli sull'API, consulta Descrivi DBClusters in AWSSDK for Python (Boto3) API Reference.

Rust

SDK per Rust

> **Note**
>
> C'è altro da fare. GitHub Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice AWS.

```
   // Get a list of allowed engine versions.
rds.DescribeDbEngineVersions(Engine='aurora-mysql', DBParameterGroupFamily=<the
family used to create your parameter group in step 2>)
   // Create an Aurora DB cluster database cluster that contains a MySql
database and uses the parameter group you created.
   // Wait for DB cluster to be ready. Call rds.DescribeDBClusters and check for
Status == 'available'.
   // Get a list of instance classes available for the selected engine
and engine version. rds.DescribeOrderableDbInstanceOptions(Engine='mysql',
EngineVersion=).

   // Create a database instance in the cluster.
   // Wait for DB instance to be ready. Call rds.DescribeDbInstances and check
for DBInstanceStatus == 'available'.
   pub async fn start_cluster_and_instance(&mut self) -> Result<(),
ScenarioError> {
       if self.password.is_none() {
           return Err(ScenarioError::with(
               "Must set Secret Password before starting a cluster",
           ));
       }
       let create_db_cluster = self
           .rds
           .create_db_cluster(
               DB_CLUSTER_IDENTIFIER,
               DB_CLUSTER_PARAMETER_GROUP_NAME,
               DB_ENGINE,
               self.engine_version.as_deref().expect("engine version"),
               self.username.as_deref().expect("username"),
               self.password
                   .replace(SecretString::new("".to_string()))
                   .expect("password"),
```

```
            )
            .await;
        if let Err(err) = create_db_cluster {
            return Err(ScenarioError::new(
                "Failed to create DB Cluster with cluster group",
                &err,
            ));
        }

        self.db_cluster_identifier = create_db_cluster
            .unwrap()
            .db_cluster
            .and_then(|c| c.db_cluster_identifier);

        if self.db_cluster_identifier.is_none() {
            return Err(ScenarioError::with("Created DB Cluster missing
Identifier"));
        }

        info!(
            "Started a db cluster: {}",
            self.db_cluster_identifier
                .as_deref()
                .unwrap_or("Missing ARN")
        );

        let create_db_instance = self
            .rds
            .create_db_instance(
                self.db_cluster_identifier.as_deref().expect("cluster name"),
                DB_INSTANCE_IDENTIFIER,
                self.instance_class.as_deref().expect("instance class"),
                DB_ENGINE,
            )
            .await;
        if let Err(err) = create_db_instance {
            return Err(ScenarioError::new(
                "Failed to create Instance in DB Cluster",
                &err,
            ));
        }

        self.db_instance_identifier = create_db_instance
            .unwrap()
```

```
            .db_instance
            .and_then(|i| i.db_instance_identifier);

        // Cluster creation can take up to 20 minutes to become available
        let cluster_max_wait = Duration::from_secs(20 * 60);
        let waiter = Waiter::builder().max(cluster_max_wait).build();
        while waiter.sleep().await.is_ok() {
            let cluster = self
                .rds
                .describe_db_clusters(
                    self.db_cluster_identifier
                        .as_deref()
                        .expect("cluster identifier"),
                )
                .await;

            if let Err(err) = cluster {
                warn!(?err, "Failed to describe cluster while waiting for
ready");
                continue;
            }

            let instance = self
                .rds
                .describe_db_instance(
                    self.db_instance_identifier
                        .as_deref()
                        .expect("instance identifier"),
                )
                .await;
            if let Err(err) = instance {
                return Err(ScenarioError::new(
                    "Failed to find instance for cluster",
                    &err,
                ));
            }

            let instances_available = instance
                .unwrap()
                .db_instances()
                .iter()
                .all(|instance| instance.db_instance_status() ==
Some("Available"));
```

```
            let endpoints = self
                .rds
                .describe_db_cluster_endpoints(
                    self.db_cluster_identifier
                        .as_deref()
                        .expect("cluster identifier"),
                )
                .await;

            if let Err(err) = endpoints {
                return Err(ScenarioError::new(
                    "Failed to find endpoint for cluster",
                    &err,
                ));
            }

            let endpoints_available = endpoints
                .unwrap()
                .db_cluster_endpoints()
                .iter()
                .all(|endpoint| endpoint.status() == Some("available"));

            if instances_available && endpoints_available {
                return Ok(());
            }
        }

        Err(ScenarioError::with("timed out waiting for cluster"))
    }

    pub async fn describe_db_clusters(
        &self,
        id: &str,
    ) -> Result<DescribeDbClustersOutput, SdkError<DescribeDBClustersError>> {
        self.inner
            .describe_db_clusters()
            .db_cluster_identifier(id)
            .send()
            .await
    }

#[tokio::test]
async fn test_start_cluster_and_instance() {
    let mut mock_rds = MockRdsImpl::default();
```

```
    mock_rds
        .expect_create_db_cluster()
        .withf(|id, params, engine, version, username, password| {
            assert_eq!(id, "RustSDKCodeExamplesDBCluster");
            assert_eq!(params, "RustSDKCodeExamplesDBParameterGroup");
            assert_eq!(engine, "aurora-mysql");
            assert_eq!(version, "aurora-mysql8.0");
            assert_eq!(username, "test username");
            assert_eq!(password.expose_secret(), "test password");
            true
        })
        .return_once(|id, _, _, _, _, _| {
            Ok(CreateDbClusterOutput::builder()

.db_cluster(DbCluster::builder().db_cluster_identifier(id).build())
                .build())
        });

    mock_rds
        .expect_create_db_instance()
        .withf(|cluster, name, class, engine| {
            assert_eq!(cluster, "RustSDKCodeExamplesDBCluster");
            assert_eq!(name, "RustSDKCodeExamplesDBInstance");
            assert_eq!(class, "m5.large");
            assert_eq!(engine, "aurora-mysql");
            true
        })
        .return_once(|cluster, name, class, _| {
            Ok(CreateDbInstanceOutput::builder()
                .db_instance(
                    DbInstance::builder()
                        .db_cluster_identifier(cluster)
                        .db_instance_identifier(name)
                        .db_instance_class(class)
                        .build(),
                )
                .build())
        });

    mock_rds
        .expect_describe_db_clusters()
        .with(eq("RustSDKCodeExamplesDBCluster"))
        .return_once(|id| {
```

```
            Ok(DescribeDbClustersOutput::builder()

.db_clusters(DbCluster::builder().db_cluster_identifier(id).build())
                .build())
        });

    mock_rds
        .expect_describe_db_instance()
        .with(eq("RustSDKCodeExamplesDBInstance"))
        .return_once(|name| {
            Ok(DescribeDbInstancesOutput::builder()
                .db_instances(
                    DbInstance::builder()
                        .db_instance_identifier(name)
                        .db_instance_status("Available")
                        .build(),
                )
                .build())
        });

    mock_rds
        .expect_describe_db_cluster_endpoints()
        .with(eq("RustSDKCodeExamplesDBCluster"))
        .return_once(|_| {
            Ok(DescribeDbClusterEndpointsOutput::builder()

.db_cluster_endpoints(DbClusterEndpoint::builder().status("available").build())
                .build())
        });

    let mut scenario = AuroraScenario::new(mock_rds);
    scenario.engine_version = Some("aurora-mysql8.0".into());
    scenario.instance_class = Some("m5.large".into());
    scenario.username = Some("test username".into());
    scenario.password = Some(SecretString::new("test password".into()));

    tokio::time::pause();
    let assertions = tokio::spawn(async move {
        let create = scenario.start_cluster_and_instance().await;
        assert!(create.is_ok());
        assert!(scenario
            .password
            .replace(SecretString::new("BAD SECRET".into()))
            .unwrap()
```

```
            .expose_secret()
            .is_empty());
        assert_eq!(
            scenario.db_cluster_identifier,
            Some("RustSDKCodeExamplesDBCluster".into())
        );
    });
    tokio::time::advance(Duration::from_secs(1)).await;
    tokio::time::resume();
    let _ = assertions.await;
}

#[tokio::test]
async fn test_start_cluster_and_instance_cluster_create_error() {
    let mut mock_rds = MockRdsImpl::default();

    mock_rds
        .expect_create_db_cluster()
        .return_once(|_, _, _, _, _, _| {
            Err(SdkError::service_error(
                CreateDBClusterError::unhandled(Box::new(Error::new(
                    ErrorKind::Other,
                    "create db cluster error",
                ))),
                Response::new(StatusCode::try_from(400).unwrap(),
 SdkBody::empty()),
            ))
        });

    let mut scenario = AuroraScenario::new(mock_rds);
    scenario.engine_version = Some("aurora-mysql8.0".into());
    scenario.instance_class = Some("m5.large".into());
    scenario.username = Some("test username".into());
    scenario.password = Some(SecretString::new("test password".into()));

    let create = scenario.start_cluster_and_instance().await;
    assert_matches!(create, Err(ScenarioError { message, context: _}) if message
 == "Failed to create DB Cluster with cluster group")
}

#[tokio::test]
async fn test_start_cluster_and_instance_cluster_create_missing_id() {
    let mut mock_rds = MockRdsImpl::default();
```

```
    mock_rds
        .expect_create_db_cluster()
        .return_once(|_, _, _, _, _, _| {
            Ok(CreateDbClusterOutput::builder()
                .db_cluster(DbCluster::builder().build())
                .build())
        });

    let mut scenario = AuroraScenario::new(mock_rds);
    scenario.engine_version = Some("aurora-mysql8.0".into());
    scenario.instance_class = Some("m5.large".into());
    scenario.username = Some("test username".into());
    scenario.password = Some(SecretString::new("test password".into()));

    let create = scenario.start_cluster_and_instance().await;
    assert_matches!(create, Err(ScenarioError { message, context:_ }) if message
 == "Created DB Cluster missing Identifier");
}

#[tokio::test]
async fn test_start_cluster_and_instance_instance_create_error() {
    let mut mock_rds = MockRdsImpl::default();

    mock_rds
        .expect_create_db_cluster()
        .withf(|id, params, engine, version, username, password| {
            assert_eq!(id, "RustSDKCodeExamplesDBCluster");
            assert_eq!(params, "RustSDKCodeExamplesDBParameterGroup");
            assert_eq!(engine, "aurora-mysql");
            assert_eq!(version, "aurora-mysql8.0");
            assert_eq!(username, "test username");
            assert_eq!(password.expose_secret(), "test password");
            true
        })
        .return_once(|id, _, _, _, _, _| {
            Ok(CreateDbClusterOutput::builder()

 .db_cluster(DbCluster::builder().db_cluster_identifier(id).build())
                .build())
        });

    mock_rds
        .expect_create_db_instance()
        .return_once(|_, _, _, _| {
```

```
            Err(SdkError::service_error(
                CreateDBInstanceError::unhandled(Box::new(Error::new(
                    ErrorKind::Other,
                    "create db instance error",
                ))),
                Response::new(StatusCode::try_from(400).unwrap(),
 SdkBody::empty()),
            ))
        });

    let mut scenario = AuroraScenario::new(mock_rds);
    scenario.engine_version = Some("aurora-mysql8.0".into());
    scenario.instance_class = Some("m5.large".into());
    scenario.username = Some("test username".into());
    scenario.password = Some(SecretString::new("test password".into()));

    let create = scenario.start_cluster_and_instance().await;
    assert_matches!(create, Err(ScenarioError { message, context: _ }) if message
 == "Failed to create Instance in DB Cluster")
}

#[tokio::test]
async fn test_start_cluster_and_instance_wait_hiccup() {
    let mut mock_rds = MockRdsImpl::default();

    mock_rds
        .expect_create_db_cluster()
        .withf(|id, params, engine, version, username, password| {
            assert_eq!(id, "RustSDKCodeExamplesDBCluster");
            assert_eq!(params, "RustSDKCodeExamplesDBParameterGroup");
            assert_eq!(engine, "aurora-mysql");
            assert_eq!(version, "aurora-mysql8.0");
            assert_eq!(username, "test username");
            assert_eq!(password.expose_secret(), "test password");
            true
        })
        .return_once(|id, _, _, _, _, _| {
            Ok(CreateDbClusterOutput::builder()

 .db_cluster(DbCluster::builder().db_cluster_identifier(id).build())
                .build())
        });

    mock_rds
```

```
        .expect_create_db_instance()
        .withf(|cluster, name, class, engine| {
            assert_eq!(cluster, "RustSDKCodeExamplesDBCluster");
            assert_eq!(name, "RustSDKCodeExamplesDBInstance");
            assert_eq!(class, "m5.large");
            assert_eq!(engine, "aurora-mysql");
            true
        })
        .return_once(|cluster, name, class, _| {
            Ok(CreateDbInstanceOutput::builder()
                .db_instance(
                    DbInstance::builder()
                        .db_cluster_identifier(cluster)
                        .db_instance_identifier(name)
                        .db_instance_class(class)
                        .build(),
                )
                .build())
        });

    mock_rds
        .expect_describe_db_clusters()
        .with(eq("RustSDKCodeExamplesDBCluster"))
        .times(1)
        .returning(|_| {
            Err(SdkError::service_error(
                DescribeDBClustersError::unhandled(Box::new(Error::new(
                    ErrorKind::Other,
                    "describe cluster error",
                ))),
                Response::new(StatusCode::try_from(400).unwrap(),
SdkBody::empty()),
            ))
        })
        .with(eq("RustSDKCodeExamplesDBCluster"))
        .times(1)
        .returning(|id| {
            Ok(DescribeDbClustersOutput::builder()

.db_clusters(DbCluster::builder().db_cluster_identifier(id).build())
                .build())
        });

    mock_rds.expect_describe_db_instance().return_once(|name| {
```

```
        Ok(DescribeDbInstancesOutput::builder()
            .db_instances(
                DbInstance::builder()
                    .db_instance_identifier(name)
                    .db_instance_status("Available")
                    .build(),
            )
            .build())
    });

    mock_rds
        .expect_describe_db_cluster_endpoints()
        .return_once(|_| {
            Ok(DescribeDbClusterEndpointsOutput::builder()

 .db_cluster_endpoints(DbClusterEndpoint::builder().status("available").build())
                .build())
        });

    let mut scenario = AuroraScenario::new(mock_rds);
    scenario.engine_version = Some("aurora-mysql8.0".into());
    scenario.instance_class = Some("m5.large".into());
    scenario.username = Some("test username".into());
    scenario.password = Some(SecretString::new("test password".into()));

    tokio::time::pause();
    let assertions = tokio::spawn(async move {
        let create = scenario.start_cluster_and_instance().await;
        assert!(create.is_ok());
    });

    tokio::time::advance(Duration::from_secs(1)).await;
    tokio::time::advance(Duration::from_secs(1)).await;
    tokio::time::resume();
    let _ = assertions.await;
}
```

- Per i dettagli sull'API, consulta Descrivi DBClusters nella AWSguida di riferimento all'API SDK for Rust.

Per un elenco completo delle guide per sviluppatori AWS SDK e degli esempi di codice, consulta. Utilizzo di questo servizio con un AWS SDK Questo argomento include anche informazioni su come iniziare e dettagli sulle versioni precedenti dell'SDK.

Utilizzo **`DescribeDBEngineVersions`** con un AWS SDK

Gli esempi di codice seguenti mostrano come utilizzare `DescribeDBEngineVersions`.

Gli esempi di operazioni sono estratti di codice da programmi più grandi e devono essere eseguiti nel contesto. È possibile visualizzare questa operazione nel contesto nel seguente esempio di codice:

- Informazioni di base

.NET

SDK per .NET(v4)

> **Note**
>
> C'è altro da fare. GitHub Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice AWS.

```
   /// <summary>
   /// Get a list of DB engine versions for a particular DB engine.
   /// </summary>
   /// <param name="engine">The name of the engine.</param>
   /// <param name="parameterGroupFamily">Optional parameter group family
name.</param>
   /// <returns>A list of DBEngineVersions.</returns>
   public async Task<List<DBEngineVersion>>
DescribeDBEngineVersionsForEngineAsync(string engine,
       string? parameterGroupFamily = null)
   {
       var response = await _amazonRDS.DescribeDBEngineVersionsAsync(
           new DescribeDBEngineVersionsRequest()
           {
               Engine = engine,
               DBParameterGroupFamily = parameterGroupFamily
           });
       return response.DBEngineVersions;
```

```
    }
```

- Per i dettagli sull'API, consulta Descrivi DBEngine le versioni in AWS SDK per .NETAPI Reference.

C++

SDK per C++

> **Note**
>
> C'è di più su GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice AWS.

```
        Aws::Client::ClientConfiguration clientConfig;
        // Optional: Set to the AWS Region (overrides config file).
        // clientConfig.region = "us-east-1";

    Aws::RDS::RDSClient client(clientConfig);


//! Routine which gets available DB engine versions for an engine name and
//! an optional parameter group family.
/*!
 \sa getDBEngineVersions()
 \param engineName: A DB engine name.
 \param parameterGroupFamily: A parameter group family name, ignored if empty.
 \param engineVersionsResult: Vector of 'DBEngineVersion' objects returned by the
 routine.
 \param client: 'RDSClient' instance.
 \return bool: Successful completion.
 */
bool AwsDoc::Aurora::getDBEngineVersions(const Aws::String &engineName,
                                         const Aws::String &parameterGroupFamily,

 Aws::Vector<Aws::RDS::Model::DBEngineVersion> &engineVersionsResult,
                                         const Aws::RDS::RDSClient &client) {
    Aws::RDS::Model::DescribeDBEngineVersionsRequest request;
    request.SetEngine(engineName);
```

```
    if (!parameterGroupFamily.empty()) {
        request.SetDBParameterGroupFamily(parameterGroupFamily);
    }

    engineVersionsResult.clear();
    Aws::String marker; // The marker is used for pagination.
    do {
        if (!marker.empty()) {
            request.SetMarker(marker);
        }

        Aws::RDS::Model::DescribeDBEngineVersionsOutcome outcome =
                client.DescribeDBEngineVersions(request);

        if (outcome.IsSuccess()) {
            const Aws::Vector<Aws::RDS::Model::DBEngineVersion> &engineVersions =
                    outcome.GetResult().GetDBEngineVersions();

            engineVersionsResult.insert(engineVersionsResult.end(),
                                        engineVersions.begin(),
 engineVersions.end());
            marker  = outcome.GetResult().GetMarker();
        }
        else {
            std::cerr << "Error with Aurora::DescribeDBEngineVersionsRequest. "
                      << outcome.GetError().GetMessage()
                      << std::endl;
        }
    } while (!marker.empty());

    return true;
}
```

- Per i dettagli sull'API, consulta Descrivi DBEngine le versioni in AWS SDK per C++API Reference.

Go

SDK per Go V2

> **Note**
>
> C'è di più su GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice AWS.

```
import (
 "context"
 "errors"
 "log"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/service/rds"
 "github.com/aws/aws-sdk-go-v2/service/rds/types"
)

type DbClusters struct {
 AuroraClient *rds.Client
}



// GetEngineVersions gets database engine versions that are available for the
 specified engine
// and parameter group family.
func (clusters *DbClusters) GetEngineVersions(ctx context.Context, engine string,
 parameterGroupFamily string) (
 []types.DBEngineVersion, error) {
 output, err := clusters.AuroraClient.DescribeDBEngineVersions(ctx,
  &rds.DescribeDBEngineVersionsInput{
   Engine:                 aws.String(engine),
   DBParameterGroupFamily: aws.String(parameterGroupFamily),
  })
 if err != nil {
  log.Printf("Couldn't get engine versions for %v: %v\n", engine, err)
  return nil, err
 } else {
```

```
  return output.DBEngineVersions, nil
 }
}
```

- Per i dettagli sull'API, consulta Descrivi DBEngine le versioni in AWS SDK per GoAPI Reference.

Java

SDK per Java 2.x

> **Note**
>
> C'è di più su GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice AWS.

```
    public static void describeDBEngines(RdsClient rdsClient) {
        try {
            DescribeDbEngineVersionsRequest engineVersionsRequest =
DescribeDbEngineVersionsRequest.builder()
                    .engine("aurora-mysql")
                    .defaultOnly(true)
                    .maxRecords(20)
                    .build();

            DescribeDbEngineVersionsResponse response =
rdsClient.describeDBEngineVersions(engineVersionsRequest);
            List<DBEngineVersion> engines = response.dbEngineVersions();

            // Get all DBEngineVersion objects.
            for (DBEngineVersion engineOb : engines) {
                System.out.println("The name of the DB parameter group family for
the database engine is "
                        + engineOb.dbParameterGroupFamily());
                System.out.println("The name of the database engine " +
engineOb.engine());
                System.out.println("The version number of the database engine " +
engineOb.engineVersion());
```

```
            }

        } catch (RdsException e) {
            System.out.println(e.getLocalizedMessage());
            System.exit(1);
        }
    }
```

- Per i dettagli sull'API, consulta Descrivi DBEngine le versioni in AWS SDK for Java 2.xAPI Reference.

Kotlin

SDK per Kotlin

> **Note**
>
> C'è di più su GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice AWS.

```
// Get a list of allowed engine versions.
suspend fun getAllowedClusterEngines(dbParameterGroupFamilyVal: String?) {
    val versionsRequest =
        DescribeDbEngineVersionsRequest {
            dbParameterGroupFamily = dbParameterGroupFamilyVal
            engine = "aurora-mysql"
        }

    RdsClient.fromEnvironment { region = "us-west-2" }.use { rdsClient ->
        val response = rdsClient.describeDbEngineVersions(versionsRequest)
        response.dbEngineVersions?.forEach { dbEngine ->
            println("The engine version is ${dbEngine.engineVersion}")
            println("The engine description is ${dbEngine.dbEngineDescription}")
        }
    }
}
```

- Per i dettagli sull'API, consulta Descrivi DBEngine le versioni nell'AWSSDK per il riferimento all'API Kotlin.

Python

SDK per Python (Boto3)

> **Note**
>
> C'è di più su. GitHub Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice AWS.

```
class AuroraWrapper:
    """Encapsulates Aurora DB cluster actions."""

    def __init__(self, rds_client):
        """
        :param rds_client: A Boto3 Amazon Relational Database Service (Amazon
 RDS) client.
        """
        self.rds_client = rds_client

    @classmethod
    def from_client(cls):
        """
        Instantiates this class from a Boto3 client.
        """
        rds_client = boto3.client("rds")
        return cls(rds_client)


    def get_engine_versions(self, engine, parameter_group_family=None):
        """
        Gets database engine versions that are available for the specified engine
        and parameter group family.

        :param engine: The database engine to look up.
        :param parameter_group_family: When specified, restricts the returned
 list of
```

```
                                        engine versions to those that are
compatible with
                                        this parameter group family.
        :return: The list of database engine versions.
        """
        try:
            kwargs = {"Engine": engine}
            if parameter_group_family is not None:
                kwargs["DBParameterGroupFamily"] = parameter_group_family
            response = self.rds_client.describe_db_engine_versions(**kwargs)
            versions = response["DBEngineVersions"]
        except ClientError as err:
            logger.error(
                "Couldn't get engine versions for %s. Here's why: %s: %s",
                engine,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
        else:
            return versions
```

- Per i dettagli sull'API, consulta Descrivi DBEngine le versioni in AWSSDK for Python (Boto3) API Reference.

Rust

SDK per Rust

> **Note**
>
> C'è altro da fare. GitHub Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice AWS.

```
   // Get available engine families for Aurora MySql.
rds.DescribeDbEngineVersions(Engine='aurora-mysql') and build a set of the
'DBParameterGroupFamily' field values. I get {aurora-mysql8.0, aurora-mysql5.7}.
```

```
   pub async fn get_engines(&self) -> Result<HashMap<String, Vec<String>>,
ScenarioError> {
       let describe_db_engine_versions =
self.rds.describe_db_engine_versions(DB_ENGINE).await;
       trace!(versions=?describe_db_engine_versions, "full list of versions");

       if let Err(err) = describe_db_engine_versions {
           return Err(ScenarioError::new(
               "Failed to retrieve DB Engine Versions",
               &err,
           ));
       };

       let version_count = describe_db_engine_versions
           .as_ref()
           .map(|o| o.db_engine_versions().len())
           .unwrap_or_default();
       info!(version_count, "got list of versions");

       // Create a map of engine families to their available versions.
       let mut versions = HashMap::<String, Vec<String>>::new();
       describe_db_engine_versions
           .unwrap()
           .db_engine_versions()
           .iter()
           .filter_map(
               |v| match (&v.db_parameter_group_family, &v.engine_version) {
                   (Some(family), Some(version)) => Some((family.clone(),
version.clone())),
                   _ => None,
               },
           )
           .for_each(|(family, version)|
versions.entry(family).or_default().push(version));

       Ok(versions)
   }

   pub async fn describe_db_engine_versions(
       &self,
       engine: &str,
   ) -> Result<DescribeDbEngineVersionsOutput,
SdkError<DescribeDBEngineVersionsError>> {
       self.inner
```

```
            .describe_db_engine_versions()
            .engine(engine)
            .send()
            .await
    }

#[tokio::test]
async fn test_scenario_get_engines() {
    let mut mock_rds = MockRdsImpl::default();

    mock_rds
        .expect_describe_db_engine_versions()
        .with(eq("aurora-mysql"))
        .return_once(|_| {
            Ok(DescribeDbEngineVersionsOutput::builder()
                .db_engine_versions(
                    DbEngineVersion::builder()
                        .db_parameter_group_family("f1")
                        .engine_version("f1a")
                        .build(),
                )
                .db_engine_versions(
                    DbEngineVersion::builder()
                        .db_parameter_group_family("f1")
                        .engine_version("f1b")
                        .build(),
                )
                .db_engine_versions(
                    DbEngineVersion::builder()
                        .db_parameter_group_family("f2")
                        .engine_version("f2a")
                        .build(),
                )
                .db_engine_versions(DbEngineVersion::builder().build())
                .build())
        });

    let scenario = AuroraScenario::new(mock_rds);

    let versions_map = scenario.get_engines().await;

    assert_eq!(
        versions_map,
        Ok(HashMap::from([
```

```
            ("f1".into(), vec!["f1a".into(), "f1b".into()]),
            ("f2".into(), vec!["f2a".into()])
        ]))
    );
}

#[tokio::test]
async fn test_scenario_get_engines_failed() {
    let mut mock_rds = MockRdsImpl::default();

    mock_rds
        .expect_describe_db_engine_versions()
        .with(eq("aurora-mysql"))
        .return_once(|_| {
            Err(SdkError::service_error(
                DescribeDBEngineVersionsError::unhandled(Box::new(Error::new(
                    ErrorKind::Other,
                    "describe_db_engine_versions error",
                ))),
                Response::new(StatusCode::try_from(400).unwrap(),
 SdkBody::empty()),
            ))
        });

    let scenario = AuroraScenario::new(mock_rds);

    let versions_map = scenario.get_engines().await;
    assert_matches!(
        versions_map,
        Err(ScenarioError { message, context: _ }) if message == "Failed to
 retrieve DB Engine Versions"
    );
}
```

- Per i dettagli sull'API, consulta Descrivi DBEngine le versioni nell'AWSSDK per il riferimento all'API Rust.

Per un elenco completo delle guide per sviluppatori AWS SDK e degli esempi di codice, consulta. Utilizzo di questo servizio con un AWS SDK Questo argomento include anche informazioni su come iniziare e dettagli sulle versioni precedenti dell'SDK.

Utilizzo **`DescribeDBInstances`** con un AWS SDK

Gli esempi di codice seguenti mostrano come utilizzare `DescribeDBInstances`.

Gli esempi di operazioni sono estratti di codice da programmi più grandi e devono essere eseguiti nel contesto. È possibile visualizzare questa operazione nel contesto nel seguente esempio di codice:

- Informazioni di base

.NET

SDK per .NET(v4)

> **Note**
>
> C'è altro da fare. GitHub Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice AWS.

```
   /// <summary>
   /// Returns a list of DB instances.
   /// </summary>
   /// <param name="dbInstanceIdentifier">Optional name of a specific DB
instance.</param>
   /// <returns>List of DB instances.</returns>
   public async Task<List<DBInstance>> DescribeDBInstancesPagedAsync(string?
dbInstanceIdentifier = null)
   {
       var results = new List<DBInstance>();
       var instancesPaginator = _amazonRDS.Paginators.DescribeDBInstances(
           new DescribeDBInstancesRequest
           {
               DBInstanceIdentifier = dbInstanceIdentifier
           });
       // Get the entire list using the paginator.
       await foreach (var instances in instancesPaginator.DBInstances)
       {
           results.Add(instances);
       }
       return results;
   }
```

- Per i dettagli sull'API, consulta Descrivi DBInstances in AWS SDK per .NETAPI Reference.

C++

SDK per C++

> **Note**
>
> C'è altro su GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice AWS.

```
        Aws::Client::ClientConfiguration clientConfig;
        // Optional: Set to the AWS Region (overrides config file).
        // clientConfig.region = "us-east-1";

    Aws::RDS::RDSClient client(clientConfig);


//! Routine which gets a DB instance description.
/*!
 \sa describeDBCluster()
 \param dbInstanceIdentifier: A DB instance identifier.
 \param instanceResult: The 'DBInstance' object containing the description.
 \param client: 'RDSClient' instance.
 \return bool: Successful completion.
 */
bool AwsDoc::Aurora::describeDBInstance(const Aws::String &dbInstanceIdentifier,
                                        Aws::RDS::Model::DBInstance
 &instanceResult,
                                        const Aws::RDS::RDSClient &client) {
    Aws::RDS::Model::DescribeDBInstancesRequest request;
    request.SetDBInstanceIdentifier(dbInstanceIdentifier);

    Aws::RDS::Model::DescribeDBInstancesOutcome outcome =
            client.DescribeDBInstances(request);

    bool result = true;
    if (outcome.IsSuccess()) {
        instanceResult = outcome.GetResult().GetDBInstances()[0];
```

```
    }
    else if (outcome.GetError().GetErrorType() !=
             Aws::RDS::RDSErrors::D_B_INSTANCE_NOT_FOUND_FAULT) {
        result = false;
        std::cerr << "Error with Aurora::DescribeDBInstances. "
                  << outcome.GetError().GetMessage()
                  << std::endl;
    }
        // This example does not log an error if the DB instance does not exist.
        // Instead, instanceResult is set to empty.
    else {
        instanceResult = Aws::RDS::Model::DBInstance();
    }

    return result;
}
```

- Per i dettagli sull'API, consulta Descrivi DBInstances in AWS SDK per C++API Reference.

Go

SDK per Go V2

> **Note**
>
> C'è altro su GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice AWS.

```
import (
 "context"
 "errors"
 "log"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/service/rds"
 "github.com/aws/aws-sdk-go-v2/service/rds/types"
)

type DbClusters struct {
```

```
 AuroraClient *rds.Client
}



// GetInstance gets data about a DB instance.
func (clusters *DbClusters) GetInstance(ctx context.Context, instanceName string)
 (
 *types.DBInstance, error) {
 output, err := clusters.AuroraClient.DescribeDBInstances(ctx,
  &rds.DescribeDBInstancesInput{
   DBInstanceIdentifier: aws.String(instanceName),
  })
 if err != nil {
  var notFoundError *types.DBInstanceNotFoundFault
  if errors.As(err, &notFoundError) {
   log.Printf("DB instance %v does not exist.\n", instanceName)
   err = nil
  } else {
   log.Printf("Couldn't get instance %v: %v\n", instanceName, err)
  }
  return nil, err
 } else {
  return &output.DBInstances[0], nil
 }
}
```

- Per i dettagli sull'API, consulta Descrivi DBInstances in AWS SDK per GoAPI Reference.

Java

SDK per Java 2.x

> **Note**
>
> C'è altro su GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice AWS.

```
    // Waits until the database instance is available.
```

```
   public static void waitForInstanceReady(RdsClient rdsClient, String
dbClusterIdentifier) {
       boolean instanceReady = false;
       String instanceReadyStr;
       System.out.println("Waiting for instance to become available.");
       try {
           DescribeDbClustersRequest instanceRequest =
DescribeDbClustersRequest.builder()
                   .dbClusterIdentifier(dbClusterIdentifier)
                   .build();

           while (!instanceReady) {
               DescribeDbClustersResponse response =
rdsClient.describeDBClusters(instanceRequest);
               List<DBCluster> clusterList = response.dbClusters();
               for (DBCluster cluster : clusterList) {
                   instanceReadyStr = cluster.status();
                   if (instanceReadyStr.contains("available")) {
                       instanceReady = true;
                   } else {
                       System.out.print(".");
                       Thread.sleep(sleepTime * 1000);
                   }
               }
           }
           System.out.println("Database cluster is available!");

       } catch (RdsException | InterruptedException e) {
           System.err.println(e.getMessage());
           System.exit(1);
       }
   }
```

- Per i dettagli sull'API, consulta Descrivi DBInstances in AWS SDK for Java 2.xAPI Reference.

Kotlin

SDK per Kotlin

> **Note**
>
> C'è altro su GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice AWS.

```
suspend fun waitDBAuroraInstanceReady(dbInstanceIdentifierVal: String?) {
    var instanceReady = false
    var instanceReadyStr: String
    println("Waiting for instance to become available.")
    val instanceRequest =
        DescribeDbInstancesRequest {
            dbInstanceIdentifier = dbInstanceIdentifierVal
        }

    var endpoint = ""
    RdsClient.fromEnvironment { region = "us-west-2" }.use { rdsClient ->
        while (!instanceReady) {
            val response = rdsClient.describeDbInstances(instanceRequest)
            response.dbInstances?.forEach { instance ->
                instanceReadyStr = instance.dbInstanceStatus.toString()
                if (instanceReadyStr.contains("available")) {
                    endpoint = instance.endpoint?.address.toString()
                    instanceReady = true
                } else {
                    print(".")
                    delay(sleepTime * 1000)
                }
            }
        }
    }
    println("Database instance is available! The connection endpoint is
 $endpoint")
}
```

- Per i dettagli sull'API, consulta Descrivi DBInstances in AWSSDK per il riferimento all'API Kotlin.

Python

SDK per Python (Boto3)

> **Note**
>
> C'è altro su. GitHub Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice AWS.

```
class AuroraWrapper:
    """Encapsulates Aurora DB cluster actions."""

    def __init__(self, rds_client):
        """
        :param rds_client: A Boto3 Amazon Relational Database Service (Amazon
 RDS) client.
        """
        self.rds_client = rds_client

    @classmethod
    def from_client(cls):
        """
        Instantiates this class from a Boto3 client.
        """
        rds_client = boto3.client("rds")
        return cls(rds_client)


    def get_db_instance(self, instance_id):
        """
        Gets data about a DB instance.

        :param instance_id: The ID of the DB instance to retrieve.
        :return: The retrieved DB instance.
        """
        try:
            response = self.rds_client.describe_db_instances(
                DBInstanceIdentifier=instance_id
            )
            db_inst = response["DBInstances"][0]
        except ClientError as err:
```

```
            if err.response["Error"]["Code"] == "DBInstanceNotFound":
                logger.info("Instance %s does not exist.", instance_id)
            else:
                logger.error(
                    "Couldn't get DB instance %s. Here's why: %s: %s",
                    instance_id,
                    err.response["Error"]["Code"],
                    err.response["Error"]["Message"],
                )
                raise
        else:
            return db_inst
```

- Per i dettagli sull'API, consulta Descrivi DBInstances in AWSSDK for Python (Boto3) API Reference.

Rust

SDK per Rust

> **Note**
>
> C'è altro da fare. GitHub Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice AWS.

```
    pub async fn clean_up(self) -> Result<(), Vec<ScenarioError>> {
        let mut clean_up_errors: Vec<ScenarioError> = vec![];

        // Delete the instance. rds.DeleteDbInstance.
        let delete_db_instance = self
            .rds
            .delete_db_instance(
                self.db_instance_identifier
                    .as_deref()
                    .expect("instance identifier"),
            )
            .await;
        if let Err(err) = delete_db_instance {
```

```
            let identifier = self
                .db_instance_identifier
                .as_deref()
                .unwrap_or("Missing Instance Identifier");
            let message = format!("failed to delete db instance {identifier}");
            clean_up_errors.push(ScenarioError::new(message, &err));
        } else {
            // Wait for the instance to delete
            let waiter = Waiter::default();
            while waiter.sleep().await.is_ok() {
                let describe_db_instances =
self.rds.describe_db_instances().await;
                if let Err(err) = describe_db_instances {
                    clean_up_errors.push(ScenarioError::new(
                        "Failed to check instance state during deletion",
                        &err,
                    ));
                    break;
                }
                let db_instances = describe_db_instances
                    .unwrap()
                    .db_instances()
                    .iter()
                    .filter(|instance| instance.db_cluster_identifier ==
self.db_cluster_identifier)
                    .cloned()
                    .collect::<Vec<DbInstance>>();

                if db_instances.is_empty() {
                    trace!("Delete Instance waited and no instances were found");
                    break;
                }
                match db_instances.first().unwrap().db_instance_status() {
                    Some("Deleting") => continue,
                    Some(status) => {
                        info!("Attempting to delete but instances is in
{status}");
                        continue;
                    }
                    None => {
                        warn!("No status for DB instance");
                        break;
                    }
                }
```

```
            }
        }

        // Delete the DB cluster. rds.DeleteDbCluster.
        let delete_db_cluster = self
            .rds
            .delete_db_cluster(
                self.db_cluster_identifier
                    .as_deref()
                    .expect("cluster identifier"),
            )
            .await;

        if let Err(err) = delete_db_cluster {
            let identifier = self
                .db_cluster_identifier
                .as_deref()
                .unwrap_or("Missing DB Cluster Identifier");
            let message = format!("failed to delete db cluster {identifier}");
            clean_up_errors.push(ScenarioError::new(message, &err));
        } else {
            // Wait for the instance and cluster to fully delete.
rds.DescribeDbInstances and rds.DescribeDbClusters until both are not found.
            let waiter = Waiter::default();
            while waiter.sleep().await.is_ok() {
                let describe_db_clusters = self
                    .rds
                    .describe_db_clusters(
                        self.db_cluster_identifier
                            .as_deref()
                            .expect("cluster identifier"),
                    )
                    .await;
                if let Err(err) = describe_db_clusters {
                    clean_up_errors.push(ScenarioError::new(
                        "Failed to check cluster state during deletion",
                        &err,
                    ));
                    break;
                }
                let describe_db_clusters = describe_db_clusters.unwrap();
                let db_clusters = describe_db_clusters.db_clusters();
                if db_clusters.is_empty() {
                    trace!("Delete cluster waited and no clusters were found");
```

```
                    break;
                }
                match db_clusters.first().unwrap().status() {
                    Some("Deleting") => continue,
                    Some(status) => {
                        info!("Attempting to delete but clusters is in
{status}");
                        continue;
                    }
                    None => {
                        warn!("No status for DB cluster");
                        break;
                    }
                }
            }
        }

        // Delete the DB cluster parameter group.
rds.DeleteDbClusterParameterGroup.
        let delete_db_cluster_parameter_group = self
            .rds
            .delete_db_cluster_parameter_group(
                self.db_cluster_parameter_group
                    .map(|g| {
                        g.db_cluster_parameter_group_name
                            .unwrap_or_else(||
DB_CLUSTER_PARAMETER_GROUP_NAME.to_string())
                    })
                    .as_deref()
                    .expect("cluster parameter group name"),
            )
            .await;
        if let Err(error) = delete_db_cluster_parameter_group {
            clean_up_errors.push(ScenarioError::new(
                "Failed to delete the db cluster parameter group",
                &error,
            ))
        }

        if clean_up_errors.is_empty() {
            Ok(())
        } else {
            Err(clean_up_errors)
        }
```

```
    }

    pub async fn describe_db_instances(
        &self,
    ) -> Result<DescribeDbInstancesOutput, SdkError<DescribeDBInstancesError>> {
        self.inner.describe_db_instances().send().await
    }

#[tokio::test]
async fn test_scenario_clean_up() {
    let mut mock_rds = MockRdsImpl::default();

    mock_rds
        .expect_delete_db_instance()
        .with(eq("MockInstance"))
        .return_once(|_| Ok(DeleteDbInstanceOutput::builder().build()));

    mock_rds
        .expect_describe_db_instances()
        .with()
        .times(1)
        .returning(|| {
            Ok(DescribeDbInstancesOutput::builder()
                .db_instances(
                    DbInstance::builder()
                        .db_cluster_identifier("MockCluster")
                        .db_instance_status("Deleting")
                        .build(),
                )
                .build())
        })
        .with()
        .times(1)
        .returning(|| Ok(DescribeDbInstancesOutput::builder().build()));

    mock_rds
        .expect_delete_db_cluster()
        .with(eq("MockCluster"))
        .return_once(|_| Ok(DeleteDbClusterOutput::builder().build()));

    mock_rds
        .expect_describe_db_clusters()
        .with(eq("MockCluster"))
        .times(1)
```

```
        .returning(|id| {
            Ok(DescribeDbClustersOutput::builder()
                .db_clusters(
                    DbCluster::builder()
                        .db_cluster_identifier(id)
                        .status("Deleting")
                        .build(),
                )
                .build())
        })
        .with(eq("MockCluster"))
        .times(1)
        .returning(|_| Ok(DescribeDbClustersOutput::builder().build()));

    mock_rds
        .expect_delete_db_cluster_parameter_group()
        .with(eq("MockParamGroup"))
        .return_once(|_|
Ok(DeleteDbClusterParameterGroupOutput::builder().build()));

    let mut scenario = AuroraScenario::new(mock_rds);
    scenario.db_cluster_identifier = Some(String::from("MockCluster"));
    scenario.db_instance_identifier = Some(String::from("MockInstance"));
    scenario.db_cluster_parameter_group = Some(
        DbClusterParameterGroup::builder()
            .db_cluster_parameter_group_name("MockParamGroup")
            .build(),
    );

    tokio::time::pause();
    let assertions = tokio::spawn(async move {
        let clean_up = scenario.clean_up().await;
        assert!(clean_up.is_ok());
    });

    tokio::time::advance(Duration::from_secs(1)).await; // Wait for first
Describe Instances
    tokio::time::advance(Duration::from_secs(1)).await; // Wait for second
Describe Instances
    tokio::time::advance(Duration::from_secs(1)).await; // Wait for first
Describe Cluster
    tokio::time::advance(Duration::from_secs(1)).await; // Wait for second
Describe Cluster
    tokio::time::resume();
```

```
    let _ = assertions.await;
}

#[tokio::test]
async fn test_scenario_clean_up_errors() {
    let mut mock_rds = MockRdsImpl::default();

    mock_rds
        .expect_delete_db_instance()
        .with(eq("MockInstance"))
        .return_once(|_| Ok(DeleteDbInstanceOutput::builder().build()));

    mock_rds
        .expect_describe_db_instances()
        .with()
        .times(1)
        .returning(|| {
            Ok(DescribeDbInstancesOutput::builder()
                .db_instances(
                    DbInstance::builder()
                        .db_cluster_identifier("MockCluster")
                        .db_instance_status("Deleting")
                        .build(),
                )
                .build())
        })
        .with()
        .times(1)
        .returning(|| {
            Err(SdkError::service_error(
                DescribeDBInstancesError::unhandled(Box::new(Error::new(
                    ErrorKind::Other,
                    "describe db instances error",
                ))),
                Response::new(StatusCode::try_from(400).unwrap(),
 SdkBody::empty()),
            ))
        });

    mock_rds
        .expect_delete_db_cluster()
        .with(eq("MockCluster"))
        .return_once(|_| Ok(DeleteDbClusterOutput::builder().build()));
```

```
    mock_rds
        .expect_describe_db_clusters()
        .with(eq("MockCluster"))
        .times(1)
        .returning(|id| {
            Ok(DescribeDbClustersOutput::builder()
                .db_clusters(
                    DbCluster::builder()
                        .db_cluster_identifier(id)
                        .status("Deleting")
                        .build(),
                )
                .build())
        })
        .with(eq("MockCluster"))
        .times(1)
        .returning(|_| {
            Err(SdkError::service_error(
                DescribeDBClustersError::unhandled(Box::new(Error::new(
                    ErrorKind::Other,
                    "describe db clusters error",
                ))),
                Response::new(StatusCode::try_from(400).unwrap(),
SdkBody::empty()),
            ))
        });

    mock_rds
        .expect_delete_db_cluster_parameter_group()
        .with(eq("MockParamGroup"))
        .return_once(|_|
Ok(DeleteDbClusterParameterGroupOutput::builder().build()));

    let mut scenario = AuroraScenario::new(mock_rds);
    scenario.db_cluster_identifier = Some(String::from("MockCluster"));
    scenario.db_instance_identifier = Some(String::from("MockInstance"));
    scenario.db_cluster_parameter_group = Some(
        DbClusterParameterGroup::builder()
            .db_cluster_parameter_group_name("MockParamGroup")
            .build(),
    );

    tokio::time::pause();
    let assertions = tokio::spawn(async move {
```

```
        let clean_up = scenario.clean_up().await;
        assert!(clean_up.is_err());
        let errs = clean_up.unwrap_err();
        assert_eq!(errs.len(), 2);
        assert_matches!(errs.first(), Some(ScenarioError {message, context: _})
 if message == "Failed to check instance state during deletion");
        assert_matches!(errs.get(1), Some(ScenarioError {message, context: _}) if
 message == "Failed to check cluster state during deletion");
    });

    tokio::time::advance(Duration::from_secs(1)).await; // Wait for first
 Describe Instances
    tokio::time::advance(Duration::from_secs(1)).await; // Wait for second
 Describe Instances
    tokio::time::advance(Duration::from_secs(1)).await; // Wait for first
 Describe Cluster
    tokio::time::advance(Duration::from_secs(1)).await; // Wait for second
 Describe Cluster
    tokio::time::resume();
    let _ = assertions.await;
}
```

- Per i dettagli sull'API, consulta Descrivi DBInstances nella AWSguida di riferimento all'API SDK for Rust.

Per un elenco completo delle guide per sviluppatori AWS SDK e degli esempi di codice, consulta. Utilizzo di questo servizio con un AWS SDK Questo argomento include anche informazioni su come iniziare e dettagli sulle versioni precedenti dell'SDK.

Utilizzo **`DescribeOrderableDBInstanceOptions`** con un AWS SDK o una CLI

Gli esempi di codice seguenti mostrano come utilizzare `DescribeOrderableDBInstanceOptions`.

Gli esempi di operazioni sono estratti di codice da programmi più grandi e devono essere eseguiti nel contesto. È possibile visualizzare questa operazione nel contesto nel seguente esempio di codice:

- Informazioni di base

.NET

SDK per .NET(v4)

> **Note**
>
> C'è altro da fare. GitHub Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice AWS.

```
   /// <summary>
   /// Get a list of orderable DB instance options for a specific
   /// engine and engine version.
   /// </summary>
   /// <param name="engine">Name of the engine.</param>
   /// <param name="engineVersion">Version of the engine.</param>
   /// <returns>List of OrderableDBInstanceOptions.</returns>
   public async Task<List<OrderableDBInstanceOption>>
DescribeOrderableDBInstanceOptionsPagedAsync(string engine, string
engineVersion)
   {
       // Use a paginator to get a list of DB instance options.
       var results = new List<OrderableDBInstanceOption>();
       var paginateInstanceOptions =
_amazonRDS.Paginators.DescribeOrderableDBInstanceOptions(
           new DescribeOrderableDBInstanceOptionsRequest()
           {
               Engine = engine,
               EngineVersion = engineVersion,
           });
       // Get the entire list using the paginator.
       await foreach (var instanceOptions in
paginateInstanceOptions.OrderableDBInstanceOptions)
       {
           results.Add(instanceOptions);
       }
       return results;
   }
```

- Per i dettagli sull'API, consulta DescribeOrderableDBInstanceOpzioni in AWS SDK per .NETAPI Reference.

C++

SDK per C++

> **Note**
>
> C'è altro su GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice AWS.

```
        Aws::Client::ClientConfiguration clientConfig;
        // Optional: Set to the AWS Region (overrides config file).
        // clientConfig.region = "us-east-1";

    Aws::RDS::RDSClient client(clientConfig);


//! Routine which gets available DB instance classes, displays the list
//! to the user, and returns the user selection.
/*!
 \sa chooseDBInstanceClass()
 \param engineName: The DB engine name.
 \param engineVersion: The DB engine version.
 \param dbInstanceClass: String for DB instance class chosen by the user.
 \param client: 'RDSClient' instance.
 \return bool: Successful completion.
 */
bool AwsDoc::Aurora::chooseDBInstanceClass(const Aws::String &engine,
                                           const Aws::String &engineVersion,
                                           Aws::String &dbInstanceClass,
                                           const Aws::RDS::RDSClient &client) {
    std::vector<Aws::String> instanceClasses;
    Aws::String marker; // The marker is used for pagination.
    do {
        Aws::RDS::Model::DescribeOrderableDBInstanceOptionsRequest request;
        request.SetEngine(engine);
        request.SetEngineVersion(engineVersion);
        if (!marker.empty()) {
            request.SetMarker(marker);
        }

        Aws::RDS::Model::DescribeOrderableDBInstanceOptionsOutcome outcome =
```

```
                client.DescribeOrderableDBInstanceOptions(request);

        if (outcome.IsSuccess()) {
            const Aws::Vector<Aws::RDS::Model::OrderableDBInstanceOption>
 &options =
                    outcome.GetResult().GetOrderableDBInstanceOptions();
            for (const Aws::RDS::Model::OrderableDBInstanceOption &option:
 options) {
                const Aws::String &instanceClass = option.GetDBInstanceClass();
                if (std::find(instanceClasses.begin(), instanceClasses.end(),
                              instanceClass) == instanceClasses.end()) {
                    instanceClasses.push_back(instanceClass);
                }
            }
            marker = outcome.GetResult().GetMarker();
        }
        else {
            std::cerr << "Error with Aurora::DescribeOrderableDBInstanceOptions.
 "
                      << outcome.GetError().GetMessage()
                      << std::endl;
            return false;
        }
    } while (!marker.empty());

    std::cout << "The available DB instance classes for your database engine
 are:"
              << std::endl;
    for (int i = 0; i < instanceClasses.size(); ++i) {
        std::cout << "   " << i + 1 << ": " << instanceClasses[i] << std::endl;
    }

    int choice = askQuestionForIntRange(
            "Which DB instance class do you want to use? ",
            1, static_cast<int>(instanceClasses.size()));
    dbInstanceClass = instanceClasses[choice - 1];
    return true;
}
```

- Per i dettagli sull'API, consulta DescribeOrderableDBInstanceOpzioni in AWS SDK per C++API Reference.

Go

SDK per Go V2

> **Note**
>
> C'è altro su GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice AWS.

```
import (
 "context"
 "errors"
 "log"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/service/rds"
 "github.com/aws/aws-sdk-go-v2/service/rds/types"
)

type DbClusters struct {
 AuroraClient *rds.Client
}



// GetOrderableInstances uses a paginator to get DB instance options that can be
 used to create DB instances that are
// compatible with a set of specifications.
func (clusters *DbClusters) GetOrderableInstances(ctx context.Context, engine
 string, engineVersion string) (
 []types.OrderableDBInstanceOption, error) {

 var output *rds.DescribeOrderableDBInstanceOptionsOutput
 var instances []types.OrderableDBInstanceOption
 var err error
 orderablePaginator :=
 rds.NewDescribeOrderableDBInstanceOptionsPaginator(clusters.AuroraClient,
  &rds.DescribeOrderableDBInstanceOptionsInput{
   Engine:        aws.String(engine),
   EngineVersion: aws.String(engineVersion),
```

```
 })
 for orderablePaginator.HasMorePages() {
  output, err = orderablePaginator.NextPage(ctx)
  if err != nil {
   log.Printf("Couldn't get orderable DB instances: %v\n", err)
   break
  } else {
   instances = append(instances, output.OrderableDBInstanceOptions...)
  }
 }
 return instances, err
}
```

- Per i dettagli sull'API, consulta DescribeOrderableDBInstanceOpzioni in AWS SDK per GoAPI Reference.

Java

SDK per Java 2.x

> **Note**
>
> C'è altro su GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice AWS.

```
    public static void describeDBEngines(RdsClient rdsClient) {
        try {
            DescribeDbEngineVersionsRequest engineVersionsRequest =
DescribeDbEngineVersionsRequest.builder()
                    .engine("aurora-mysql")
                    .defaultOnly(true)
                    .maxRecords(20)
                    .build();

            DescribeDbEngineVersionsResponse response =
rdsClient.describeDBEngineVersions(engineVersionsRequest);
            List<DBEngineVersion> engines = response.dbEngineVersions();
```

```
            // Get all DBEngineVersion objects.
            for (DBEngineVersion engineOb : engines) {
                System.out.println("The name of the DB parameter group family for
 the database engine is "
                        + engineOb.dbParameterGroupFamily());
                System.out.println("The name of the database engine " +
 engineOb.engine());
                System.out.println("The version number of the database engine " +
 engineOb.engineVersion());
            }

        } catch (RdsException e) {
            System.out.println(e.getLocalizedMessage());
            System.exit(1);
        }
    }
```

- Per i dettagli sull'API, consulta DescribeOrderableDBInstanceOpzioni in AWS SDK for Java 2.xAPI Reference.

PowerShell

Strumenti per PowerShell V4

Esempio 1: questo esempio elenca le versioni del motore DB che supportano una classe di istanza database specifica in una Regione AWS.

```
$params = @{
  Engine = 'aurora-postgresql'
  DBInstanceClass = 'db.r5.large'
  Region = 'us-east-1'
}
Get-RDSOrderableDBInstanceOption @params
```

Esempio 2: questo esempio elenca le classi di istanza database che supportano una versione specifica del motore DB in una Regione AWS.

```
$params = @{
  Engine = 'aurora-postgresql'
  EngineVersion = '13.6'
```

```
  Region = 'us-east-1'
}
Get-RDSOrderableDBInstanceOption @params
```

- Per i dettagli sull'API, vedere DescribeOrderableDBInstanceOpzioni in AWS Strumenti per PowerShellCmdlet Reference (V4).

Strumenti per V5 PowerShell

Esempio 1: questo esempio elenca le versioni del motore DB che supportano una classe di istanza database specifica in una Regione AWS.

```
$params = @{
  Engine = 'aurora-postgresql'
  DBInstanceClass = 'db.r5.large'
  Region = 'us-east-1'
}
Get-RDSOrderableDBInstanceOption @params
```

Esempio 2: questo esempio elenca le classi di istanza database che supportano una versione specifica del motore DB in una Regione AWS.

```
$params = @{
  Engine = 'aurora-postgresql'
  EngineVersion = '13.6'
  Region = 'us-east-1'
}
Get-RDSOrderableDBInstanceOption @params
```

- Per i dettagli sull'API, vedere DescribeOrderableDBInstanceOpzioni in AWS Strumenti per PowerShellCmdlet Reference (V5).

Python

SDK per Python (Boto3)

> **Note**
>
> C'è di più su. GitHub Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice AWS.

```
class AuroraWrapper:
    """Encapsulates Aurora DB cluster actions."""

    def __init__(self, rds_client):
        """
        :param rds_client: A Boto3 Amazon Relational Database Service (Amazon
 RDS) client.
        """
        self.rds_client = rds_client

    @classmethod
    def from_client(cls):
        """
        Instantiates this class from a Boto3 client.
        """
        rds_client = boto3.client("rds")
        return cls(rds_client)


    def get_orderable_instances(self, db_engine, db_engine_version):
        """
        Gets DB instance options that can be used to create DB instances that are
        compatible with a set of specifications.

        :param db_engine: The database engine that must be supported by the DB
 instance.
        :param db_engine_version: The engine version that must be supported by
 the DB instance.
        :return: The list of DB instance options that can be used to create a
 compatible DB instance.
        """
        try:
            inst_opts = []
            paginator = self.rds_client.get_paginator(
                "describe_orderable_db_instance_options"
            )
            for page in paginator.paginate(
                Engine=db_engine, EngineVersion=db_engine_version
            ):
                inst_opts += page["OrderableDBInstanceOptions"]
        except ClientError as err:
            logger.error(
                "Couldn't get orderable DB instances. Here's why: %s: %s",
```

```
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
        else:
            return inst_opts
```

- Per i dettagli sull'API, consulta DescribeOrderableDBInstanceOptions in AWSSDK for Python (Boto3) API Reference.

Rust

SDK per Rust

> **Note**
>
> C'è altro da fare. GitHub Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice AWS.

```
   pub async fn get_instance_classes(&self) -> Result<Vec<String>,
ScenarioError> {
        let describe_orderable_db_instance_options_items = self
            .rds
            .describe_orderable_db_instance_options(
                DB_ENGINE,
                self.engine_version
                    .as_ref()
                    .expect("engine version for db instance options")
                    .as_str(),
            )
            .await;

        describe_orderable_db_instance_options_items
            .map(|options| {
                options
                    .iter()
                    .filter(|o| o.storage_type() == Some("aurora"))
```

```
                    .map(|o|
 o.db_instance_class().unwrap_or_default().to_string())
                    .collect::<Vec<String>>()
            })
            .map_err(|err| ScenarioError::new("Could not get available instance
 classes", &err))
    }

    pub async fn describe_orderable_db_instance_options(
        &self,
        engine: &str,
        engine_version: &str,
    ) -> Result<Vec<OrderableDbInstanceOption>,
 SdkError<DescribeOrderableDBInstanceOptionsError>>
    {
        self.inner
            .describe_orderable_db_instance_options()
            .engine(engine)
            .engine_version(engine_version)
            .into_paginator()
            .items()
            .send()
            .try_collect()
            .await
    }

#[tokio::test]
async fn test_scenario_get_instance_classes() {
    let mut mock_rds = MockRdsImpl::default();

    mock_rds
        .expect_create_db_cluster_parameter_group()
        .return_once(|_, _, _| {
            Ok(CreateDbClusterParameterGroupOutput::builder()

 .db_cluster_parameter_group(DbClusterParameterGroup::builder().build())
                .build())
        });

    mock_rds
        .expect_describe_orderable_db_instance_options()
        .with(eq("aurora-mysql"), eq("aurora-mysql8.0"))
        .return_once(|_, _| {
            Ok(vec![
```

```
                OrderableDbInstanceOption::builder()
                    .db_instance_class("t1")
                    .storage_type("aurora")
                    .build(),
                OrderableDbInstanceOption::builder()
                    .db_instance_class("t1")
                    .storage_type("aurora-iopt1")
                    .build(),
                OrderableDbInstanceOption::builder()
                    .db_instance_class("t2")
                    .storage_type("aurora")
                    .build(),
                OrderableDbInstanceOption::builder()
                    .db_instance_class("t3")
                    .storage_type("aurora")
                    .build(),
            ])
        });

    let mut scenario = AuroraScenario::new(mock_rds);
    scenario
        .set_engine("aurora-mysql", "aurora-mysql8.0")
        .await
        .expect("set engine");

    let instance_classes = scenario.get_instance_classes().await;

    assert_eq!(
        instance_classes,
        Ok(vec!["t1".into(), "t2".into(), "t3".into()])
    );
}

#[tokio::test]
async fn test_scenario_get_instance_classes_error() {
    let mut mock_rds = MockRdsImpl::default();

    mock_rds
        .expect_describe_orderable_db_instance_options()
        .with(eq("aurora-mysql"), eq("aurora-mysql8.0"))
        .return_once(|_, _| {
            Err(SdkError::service_error(

 DescribeOrderableDBInstanceOptionsError::unhandled(Box::new(Error::new(
```

```
                    ErrorKind::Other,
                    "describe_orderable_db_instance_options_error",
                ))),
                Response::new(StatusCode::try_from(400).unwrap(),
 SdkBody::empty()),
            ))
        });

    let mut scenario = AuroraScenario::new(mock_rds);
    scenario.engine_family = Some("aurora-mysql".into());
    scenario.engine_version = Some("aurora-mysql8.0".into());

    let instance_classes = scenario.get_instance_classes().await;

    assert_matches!(
        instance_classes,
        Err(ScenarioError {message, context: _}) if message == "Could not get
 available instance classes"
    );
}
```

- Per i dettagli sull'API, consulta DescribeOrderableDBInstanceOptions in AWSSDK for Rust API reference.

Per un elenco completo delle guide per sviluppatori AWS SDK e degli esempi di codice, consulta. Utilizzo di questo servizio con un AWS SDK Questo argomento include anche informazioni su come iniziare e dettagli sulle versioni precedenti dell'SDK.

Utilizzo **`ModifyDBClusterParameterGroup`** con un AWS SDK

Gli esempi di codice seguenti mostrano come utilizzare `ModifyDBClusterParameterGroup`.

Gli esempi di operazioni sono estratti di codice da programmi più grandi e devono essere eseguiti nel contesto. È possibile visualizzare questa operazione nel contesto nel seguente esempio di codice:

- Informazioni di base

.NET

SDK per .NET(v4)

> **Note**
>
> C'è altro da fare. GitHub Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice AWS.

```
    /// <summary>
    /// Modify the specified integer parameters with new values from user input.
    /// </summary>
    /// <param name="groupName">The group name for the parameters.</param>
    /// <param name="parameters">The list of integer parameters to modify.</
param>
    /// <param name="newValue">Optional int value to set for parameters.</param>
    /// <returns>The name of the group that was modified.</returns>
    public async Task<string> ModifyIntegerParametersInGroupAsync(string
 groupName, List<Parameter> parameters, int newValue = 0)
    {
        foreach (var p in parameters)
        {
            if (p.IsModifiable.GetValueOrDefault() && p.DataType == "integer")
            {
                while (newValue == 0)
                {
                    Console.WriteLine(
                        $"Enter a new value for {p.ParameterName} from the
 allowed values {p.AllowedValues} ");

                    var choice = Console.ReadLine();
                    int.TryParse(choice, out newValue);
                }

                p.ParameterValue = newValue.ToString();
            }
        }

        var request = new ModifyDBClusterParameterGroupRequest
        {
            Parameters = parameters,
```

```
            DBClusterParameterGroupName = groupName,
        };

        var result = await
_amazonRDS.ModifyDBClusterParameterGroupAsync(request);
        return result.DBClusterParameterGroupName;
    }
```

- Per i dettagli sull'API, consulta Modify DBCluster ParameterGroup in AWS SDK per .NETAPI Reference.

C++

SDK per C++

> **Note**
>
> C'è altro su GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice AWS.

```
        Aws::Client::ClientConfiguration clientConfig;
        // Optional: Set to the AWS Region (overrides config file).
        // clientConfig.region = "us-east-1";

    Aws::RDS::RDSClient client(clientConfig);

        Aws::RDS::Model::ModifyDBClusterParameterGroupRequest request;
        request.SetDBClusterParameterGroupName(CLUSTER_PARAMETER_GROUP_NAME);
        request.SetParameters(updateParameters);

        Aws::RDS::Model::ModifyDBClusterParameterGroupOutcome outcome =
                client.ModifyDBClusterParameterGroup(request);

        if (outcome.IsSuccess()) {
            std::cout << "The DB cluster parameter group was successfully
modified."
                      << std::endl;
        }
        else {
```

```
            std::cerr << "Error with Aurora::ModifyDBClusterParameterGroup. "
                      << outcome.GetError().GetMessage()
                      << std::endl;
        }
```

- Per i dettagli sull'API, consulta Modify DBCluster ParameterGroup in AWS SDK per C++API Reference.

Go

SDK per Go V2

> **Note**
>
> C'è altro su GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice AWS.

```
import (
 "context"
 "errors"
 "log"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/service/rds"
 "github.com/aws/aws-sdk-go-v2/service/rds/types"
)

type DbClusters struct {
 AuroraClient *rds.Client
}



// UpdateParameters updates parameters in a named DB cluster parameter group.
func (clusters *DbClusters) UpdateParameters(ctx context.Context,
 parameterGroupName string, params []types.Parameter) error {
 _, err := clusters.AuroraClient.ModifyDBClusterParameterGroup(ctx,
  &rds.ModifyDBClusterParameterGroupInput{
```

```
    DBClusterParameterGroupName: aws.String(parameterGroupName),
    Parameters:                  params,
   })
  if err != nil {
   log.Printf("Couldn't update parameters in %v: %v\n", parameterGroupName, err)
   return err
  } else {
   return nil
  }
}
```

- Per i dettagli sull'API, consulta Modify DBCluster ParameterGroup in AWS SDK per GoAPI Reference.

Java

SDK per Java 2.x

> **Note**
>
> C'è altro su GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice AWS.

```
    public static void describeDbClusterParameterGroups(RdsClient rdsClient,
String dbClusterGroupName) {
        try {
            DescribeDbClusterParameterGroupsRequest groupsRequest =
DescribeDbClusterParameterGroupsRequest.builder()
                    .dbClusterParameterGroupName(dbClusterGroupName)
                    .maxRecords(20)
                    .build();

            List<DBClusterParameterGroup> groups =
rdsClient.describeDBClusterParameterGroups(groupsRequest)
                    .dbClusterParameterGroups();
            for (DBClusterParameterGroup group : groups) {
                System.out.println("The group name is " +
group.dbClusterParameterGroupName());
```

```
                System.out.println("The group ARN is " +
 group.dbClusterParameterGroupArn());
            }

        } catch (RdsException e) {
            System.out.println(e.getLocalizedMessage());
            System.exit(1);
        }
    }
```

- Per i dettagli sull'API, consulta Modify DBCluster ParameterGroup in AWS SDK for Java 2.xAPI Reference.

Kotlin

SDK per Kotlin

> **Note**
>
> C'è altro su GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice AWS.

```
// Modify the auto_increment_offset parameter.
suspend fun modifyDBClusterParas(dClusterGroupName: String?) {
    val parameter1 =
        Parameter {
            parameterName = "auto_increment_offset"
            applyMethod = ApplyMethod.fromValue("immediate")
            parameterValue = "5"
        }

    val paraList = ArrayList<Parameter>()
    paraList.add(parameter1)
    val groupRequest =
        ModifyDbClusterParameterGroupRequest {
            dbClusterParameterGroupName = dClusterGroupName
            parameters = paraList
        }
```

```
    RdsClient.fromEnvironment { region = "us-west-2" }.use { rdsClient ->
        val response = rdsClient.modifyDbClusterParameterGroup(groupRequest)
        println("The parameter group ${response.dbClusterParameterGroupName} was
 successfully modified")
    }
}
```

- Per i dettagli sull'API, consulta Modify DBCluster ParameterGroup in AWSSDK for Kotlin API reference.

Python

SDK per Python (Boto3)

> **Note**
>
> C'è di più su. GitHub Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice AWS.

```
class AuroraWrapper:
    """Encapsulates Aurora DB cluster actions."""

    def __init__(self, rds_client):
        """
        :param rds_client: A Boto3 Amazon Relational Database Service (Amazon
 RDS) client.
        """
        self.rds_client = rds_client

    @classmethod
    def from_client(cls):
        """
        Instantiates this class from a Boto3 client.
        """
        rds_client = boto3.client("rds")
        return cls(rds_client)


    def update_parameters(self, parameter_group_name, update_parameters):
```

```
        """
        Updates parameters in a custom DB cluster parameter group.

        :param parameter_group_name: The name of the parameter group to update.
        :param update_parameters: The parameters to update in the group.
        :return: Data about the modified parameter group.
        """
        try:
            response = self.rds_client.modify_db_cluster_parameter_group(
                DBClusterParameterGroupName=parameter_group_name,
                Parameters=update_parameters,
            )
        except ClientError as err:
            logger.error(
                "Couldn't update parameters in %s. Here's why: %s: %s",
                parameter_group_name,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
        else:
            return response
```

- Per i dettagli sull'API, consulta Modify DBCluster ParameterGroup in AWSSDK for Python (Boto3) API Reference.

Rust

SDK per Rust

> **Note**
>
> C'è altro da fare. GitHub Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice AWS.

```
   // Modify both the auto_increment_offset and auto_increment_increment
parameters in one call in the custom parameter group. Set their ParameterValue
fields to a new allowable value. rds.ModifyDbClusterParameterGroup.
```

```
    pub async fn update_auto_increment(
        &self,
        offset: u8,
        increment: u8,
    ) -> Result<(), ScenarioError> {
        let modify_db_cluster_parameter_group = self
            .rds
            .modify_db_cluster_parameter_group(
                DB_CLUSTER_PARAMETER_GROUP_NAME,
                vec![
                    Parameter::builder()
                        .parameter_name("auto_increment_offset")
                        .parameter_value(format!("{offset}"))
                        .apply_method(aws_sdk_rds::types::ApplyMethod::Immediate)
                        .build(),
                    Parameter::builder()
                        .parameter_name("auto_increment_increment")
                        .parameter_value(format!("{increment}"))
                        .apply_method(aws_sdk_rds::types::ApplyMethod::Immediate)
                        .build(),
                ],
            )
            .await;

        if let Err(error) = modify_db_cluster_parameter_group {
            return Err(ScenarioError::new(
                "Failed to modify cluster parameter group",
                &error,
            ));
        }

        Ok(())
    }

    pub async fn modify_db_cluster_parameter_group(
        &self,
        name: &str,
        parameters: Vec<Parameter>,
    ) -> Result<ModifyDbClusterParameterGroupOutput,
SdkError<ModifyDBClusterParameterGroupError>>
    {
        self.inner
            .modify_db_cluster_parameter_group()
            .db_cluster_parameter_group_name(name)
```

```
            .set_parameters(Some(parameters))
            .send()
            .await
    }

#[tokio::test]
async fn test_scenario_update_auto_increment() {
    let mut mock_rds = MockRdsImpl::default();

    mock_rds
        .expect_modify_db_cluster_parameter_group()
        .withf(|name, params| {
            assert_eq!(name, "RustSDKCodeExamplesDBParameterGroup");
            assert_eq!(
                params,
                &vec![
                    Parameter::builder()
                        .parameter_name("auto_increment_offset")
                        .parameter_value("10")
                        .apply_method(aws_sdk_rds::types::ApplyMethod::Immediate)
                        .build(),
                    Parameter::builder()
                        .parameter_name("auto_increment_increment")
                        .parameter_value("20")
                        .apply_method(aws_sdk_rds::types::ApplyMethod::Immediate)
                        .build(),
                ]
            );
            true
        })
        .return_once(|_, _|
 Ok(ModifyDbClusterParameterGroupOutput::builder().build()));

    let scenario = AuroraScenario::new(mock_rds);

    scenario
        .update_auto_increment(10, 20)
        .await
        .expect("update auto increment");
}

#[tokio::test]
async fn test_scenario_update_auto_increment_error() {
    let mut mock_rds = MockRdsImpl::default();
```

```
    mock_rds
        .expect_modify_db_cluster_parameter_group()
        .return_once(|_, _| {
            Err(SdkError::service_error(

 ModifyDBClusterParameterGroupError::unhandled(Box::new(Error::new(
                    ErrorKind::Other,
                    "modify_db_cluster_parameter_group_error",
                ))),
                Response::new(StatusCode::try_from(400).unwrap(),
 SdkBody::empty()),
            ))
        });

    let scenario = AuroraScenario::new(mock_rds);

    let update = scenario.update_auto_increment(10, 20).await;
    assert_matches!(update, Err(ScenarioError { message, context: _}) if message
 == "Failed to modify cluster parameter group");
}
```

- Per i dettagli sull'API, consulta Modify DBCluster ParameterGroup in AWSSDK for Rust API reference.

Per un elenco completo delle guide per sviluppatori AWS SDK e degli esempi di codice, consulta. Utilizzo di questo servizio con un AWS SDK Questo argomento include anche informazioni su come iniziare e dettagli sulle versioni precedenti dell'SDK.

Scenari per l'utilizzo di Aurora AWS SDKs

I seguenti esempi di codice mostrano come implementare scenari comuni in Aurora con. AWS SDKs Questi scenari illustrano come eseguire attività specifiche chiamando più funzioni all'interno di Aurora o in combinazione con altri Servizi AWS. Ogni scenario include un collegamento al codice sorgente completo, dove è possibile trovare le istruzioni su come configurare ed eseguire il codice.

Gli scenari sono relativi a un livello intermedio di esperienza per aiutarti a comprendere le azioni di servizio nel contesto.

Esempi

- Creazione di una REST API per la libreria di prestiti
- Creazione di un tracciatore di elementi di lavoro di Aurora Serverless

Creazione di una REST API per la libreria di prestiti

L'esempio di codice seguente mostra come creare una libreria di prestiti in cui gli utenti possono prendere in prestito e restituire libri tramite una REST API supportata da un database Amazon Aurora.

Python

SDK per Python (Boto3)

Mostra come utilizzare l'AWS SDK per Python (Boto3)API Amazon Relational Database Service (Amazon RDS) e AWS Chalice per creare un'API REST supportata da un database Amazon Aurora. Il servizio Web è completamente serverless e rappresenta una semplice libreria di prestiti in cui gli utenti possono prendere in prestito e restituire libri. Scopri come:

- Creare e gestire un cluster di database Aurora serverless.
- Utilizzalo per gestire le credenziali Gestione dei segreti AWS del database.
- Implementare un livello di archiviazione di dati che utilizza Amazon RDS per spostare i dati dentro e fuori dal database.
- Usa AWS Chalice per distribuire un'API REST serverless su Amazon API Gateway e. AWS Lambda
- Utilizza il pacchetto Richieste per inviare le richieste al servizio Web.

Per il codice sorgente completo e le istruzioni su come configurarlo ed eseguirlo, consulta l'esempio completo su. GitHub

Servizi utilizzati in questo esempio

- Gateway API
- Aurora
- Lambda
- Secrets Manager

Per un elenco completo delle guide per sviluppatori AWS SDK e degli esempi di codice, consultaUtilizzo di questo servizio con un AWS SDK. Questo argomento include anche informazioni su come iniziare e dettagli sulle versioni precedenti dell'SDK.

Creazione di un tracciatore di elementi di lavoro di Aurora Serverless

I seguenti esempi di codice mostrano come creare un'applicazione web che traccia gli elementi di lavoro in database Amazon Aurora serverless e utilizza Amazon Simple Email Service (Amazon SES) per inviare report.

.NET

SDK per .NET

Mostra come utilizzare per AWS SDK per .NET creare un'applicazione Web che tenga traccia degli elementi di lavoro in un database Amazon Aurora e invii report tramite e-mail utilizzando Amazon Simple Email Service (Amazon SES). Questo esempio utilizza un front-end creato con React.js per interagire con un RESTful backend.NET.

- Integra un'applicazione web React con AWS i servizi.
- Elenco, aggiunta e aggiornamento di elementi in una tabella Aurora.
- Invia un report per e-mail degli articoli di lavoro filtrati tramite Amazon SES.
- Distribuisci e gestisci risorse di esempio con lo AWS CloudFormation script incluso.

Per il codice sorgente completo e le istruzioni su come configurarlo ed eseguirlo, vedi l'esempio completo su GitHub.

Servizi utilizzati in questo esempio

- Aurora
- Amazon RDS
- Servizi di dati di Amazon RDS
- Amazon SES

C++

SDK per C++

Mostra come creare un'applicazione web che traccia gli elementi di lavoro archiviati in un database Amazon Aurora serverless, con i relativi report.

Per il codice sorgente completo e le istruzioni su come configurare un'API REST C++ che interroga dati Amazon Aurora Serverless e per l'utilizzo da parte di un'applicazione React, consulta l'esempio completo su. GitHub

Servizi utilizzati in questo esempio

- Aurora
- Amazon RDS
- Servizi di dati di Amazon RDS
- Amazon SES

Java

SDK per Java 2.x

Mostra come creare un'applicazione Web che traccia e segnala gli elementi di lavoro archiviati in un database Amazon RDS.

Per il codice sorgente completo e le istruzioni su come configurare un'API Spring REST che interroga i dati Serverless di Amazon Aurora e per l'utilizzo da parte di un'applicazione React, consulta l'esempio completo su. GitHub

Per il codice sorgente completo e le istruzioni su come configurare ed eseguire un esempio che utilizza l'API JDBC, consulta l'esempio completo su. GitHub

Servizi utilizzati in questo esempio

- Aurora
- Amazon RDS
- Servizi di dati di Amazon RDS
- Amazon SES

JavaScript

SDK per (v3 JavaScript)

Mostra come utilizzare AWS SDK per JavaScript (v3) per creare un'applicazione Web che tenga traccia degli elementi di lavoro in un database Amazon Aurora e invii report tramite e-mail utilizzando Amazon Simple Email Service (Amazon SES). Questo esempio utilizza un front-end creato con React.js per interagire con un backend Express Node.js.

- Integra un'applicazione web React.js con. Servizi AWS
- Elenca, aggiungi e aggiorna elementi in una tabella Aurora.
- Invia un report per e-mail degli elementi di lavoro filtrati tramite Amazon SES.
- Distribuisci e gestisci risorse di esempio con lo AWS CloudFormation script incluso.

Per il codice sorgente completo e le istruzioni su come configurarlo ed eseguirlo, vedi l'esempio completo su GitHub.

Servizi utilizzati in questo esempio

- Aurora
- Amazon RDS
- Servizi di dati di Amazon RDS
- Amazon SES

Kotlin

SDK per Kotlin

Mostra come creare un'applicazione web che traccia e segnala gli elementi di lavoro archiviati in un database Amazon RDS.

Per il codice sorgente completo e le istruzioni su come configurare un'API Spring REST che interroga i dati Serverless di Amazon Aurora e per l'utilizzo da parte di un'applicazione React, consulta l'esempio completo su. GitHub

Servizi utilizzati in questo esempio

- Aurora
- Amazon RDS
- Servizi di dati di Amazon RDS
- Amazon SES

PHP

SDK per PHP

Mostra come utilizzare per AWS SDK per PHP creare un'applicazione Web che tenga traccia degli elementi di lavoro in un database Amazon RDS e invii report tramite e-mail utilizzando

Amazon Simple Email Service (Amazon SES). Questo esempio utilizza un front-end creato con React.js per interagire con un backend RESTful PHP.

- Integra un'applicazione web React.js con AWS i servizi.
- Elenca, aggiungi, aggiorna ed elimina gli elementi in una tabella Amazon RDS.
- Invia un report per e-mail degli articoli di lavoro filtrati tramite Amazon SES.
- Distribuisci e gestisci risorse di esempio con lo AWS CloudFormation script incluso.

Per il codice sorgente completo e le istruzioni su come configurarlo ed eseguirlo, vedi l'esempio completo su GitHub.

Servizi utilizzati in questo esempio

- Aurora
- Amazon RDS
- Servizi di dati di Amazon RDS
- Amazon SES

Python

SDK per Python (Boto3)

Mostra come utilizzare per AWS SDK per Python (Boto3) creare un servizio REST che tenga traccia degli elementi di lavoro in un database Amazon Aurora Serverless e invii report tramite e-mail utilizzando Amazon Simple Email Service (Amazon SES). Questo esempio utilizza il framework Web Flask per gestire il routing HTTP e si integra con una pagina web React per presentare un'applicazione web completamente funzionale.

- Crea un servizio Flask REST che si integri con. Servizi AWS
- Lettura, scrittura e aggiornamento degli elementi di lavoro archiviati in un database Aurora Serverless.
- Crea un Gestione dei segreti AWS segreto che contenga le credenziali del database e usalo per autenticare le chiamate al database.
- Utilizzo di Amazon SES per inviare report via e-mail sugli elementi di lavoro.

Per il codice sorgente completo e le istruzioni su come configurarlo ed eseguirlo, guarda l'esempio completo su. GitHub

Servizi utilizzati in questo esempio

- Aurora
- Amazon RDS
- Servizi di dati di Amazon RDS
- Amazon SES

Per un elenco completo delle guide per sviluppatori AWS SDK e degli esempi di codice, consultaUtilizzo di questo servizio con un AWS SDK. Questo argomento include anche informazioni su come iniziare e dettagli sulle versioni precedenti dell'SDK.

Best practice con Amazon Aurora

Questo argomento include informazioni sulle best practice e sulle opzioni generali per l'utilizzo o la migrazione di dati a un cluster database Amazon Aurora.

Alcune delle best practice per Amazon Aurora sono specifiche di un motore di database particolare. Per ulteriori informazioni sulle best practice Aurora specifiche per un motore di database, consulta quanto segue:

Motore di database	Best practice
Amazon Aurora MySQL	Per informazioni, consulta Best practice con Amazon Aurora MySQL
Amazon Aurora PostgreSQL	Per informazioni, consulta Best practice con Amazon Aurora PostgreSQL

> **Note**
>
> Per suggerimenti comuni per Aurora, consulta Suggerimenti di Amazon Aurora.

Argomenti

- Linee guida operative di base per Amazon Aurora
- Suggerimenti relativi alla RAM per un'istanza di database
- Driver di database AWS
- Monitoraggio di Amazon Aurora
- Uso di gruppi di parametri database e gruppi di parametri cluster database
- Video sulle best practice Amazon Aurora

Linee guida operative di base per Amazon Aurora

Di seguito sono illustrate le linee guida operative di base e le best practice che tutti dovrebbero seguire durante l'utilizzo di Amazon Aurora. Il contratto sul livello di servizio di Amazon RDS richiede che tu segua queste linee guida.

- Monitora l'utilizzo della memoria, della CPU e dello storage. Amazon CloudWatch può essere configurato per avvisarti quando i modelli di utilizzo cambiano o quando stai per raggiungere la capacità massima della distribuzione. In questo modo, è possibile mantenere le prestazioni e la disponibilità del sistema.
- Se l'applicazione client memorizza nella cache i dati DNS (Domain Name Service) delle tue istanze di database, imposta un valore time-to-live (TTL) inferiore a 30 secondi. L'indirizzo IP sottostante di un'istanza DB può cambiare dopo un failover. Pertanto, la memorizzazione nella cache dei dati DNS per un periodo di tempo prolungato può causare errori di connessione se l'applicazione tenta di connettersi a un indirizzo IP che non è più in servizio. Nei cluster di database Aurora con più repliche di lettura si possono verificare errori di connessione quando le connessioni utilizzano l'endpoint di lettura e una delle istanze di replica di lettura è in modalità di manutenzione o è stata eliminata.
- Prova il failover per il cluster per capire quanto tempo impiega il processo per il tuo caso d'uso. Il test di failover consente di garantire che l'applicazione che accede al cluster DB possa connettersi automaticamente al nuovo cluster DB dopo il failover.

Suggerimenti relativi alla RAM per un'istanza di database

Per ottimizzare le prestazioni, alloca RAM sufficiente in modo che il working set risieda quasi interamente nella memoria. Per verificare se il working set è quasi completamente in memoria, esamina i parametri seguenti in Amazon CloudWatch:

- `VolumeReadIOPS` – Misura il numero medio di operazioni I/O di lettura da un volume di cluster, indicato a intervalli di 5 minuti. Il valore di `VolumeReadIOPS` dovrebbe essere di piccola entità e stabile. In alcuni casi, potresti scoprire che l'I/O di lettura è in aumento o è più alto del solito. In tal caso, esaminare le istanze DB nel cluster DB per vedere quali istanze DB stanno causando l'aumento dell'I/O.

> **Tip**
>
> Se il tuo cluster Aurora MySQL utilizza una query parallela, potresti vedere un aumento dei valori di `VolumeReadIOPS`. Le query parallele non utilizzano il pool di buffer. Pertanto, sebbene le query siano veloci, questa elaborazione ottimizzata può comportare un aumento delle operazioni di lettura e degli addebiti associati.

- `BufferCacheHitRatio` – Questo parametro misura la percentuale di richieste gestite dalla cache del buffer di un'istanza database nel cluster di database. Questo parametro offre visibilità sulla quantità di dati serviti dalla memoria.

  Una percentuale di riscontri elevata indica che l'istanza database dispone di memoria sufficiente. Una percentuale di riscontri bassa indica che le query su questa istanza database vengono spesso trasferite su disco. In questo caso esamina il carico di lavoro per stabilire quali query causano questo comportamento.

Se dall'esame del carico di lavoro emerge la necessità di aumentare la memoria, il dimensionamento della classe di istanza database con passaggio a una classe con più RAM potrebbe risultare vantaggioso. In seguito, potrai esaminare i parametri riportati sopra e incrementare ulteriormente in base alle esigenze. Se il cluster Aurora è più grande di 40 TB, non utilizzare le classi di istanza db.t2, db.t3 or db.t4g.

Per ulteriori informazioni, consulta CloudWatch Parametri Amazon per Amazon Aurora.

Driver di database AWS

Consigliamo la suite di driver AWS per la connettività delle applicazioni. I driver sono stati progettati per fornire supporto per tempi di switchover e failover più rapidi e per l'autenticazione con Gestione dei segreti AWS, AWS Identity and Access Management (IAM) e l'identità federata. I driver AWS si basano sul monitoraggio dello stato del cluster di database e conoscono la topologia del cluster per determinare la nuova istanza di scrittura. Questo approccio riduce i tempi di switchover e failover a meno di dieci secondi, rispetto alle decine di secondi necessari per i driver open source.

Con l'introduzione di nuove funzionalità di servizio, l'obiettivo della suite di driver AWS è quello di disporre di un supporto integrato per le stesse.

Per ulteriori informazioni, consulta Connessione ai cluster di database Aurora con i driver AWS.

Monitoraggio di Amazon Aurora

Amazon Aurora fornisce varie metriche e approfondimenti che è possibile monitorare per determinare lo stato e le prestazioni del cluster database Aurora. Per visualizzare i parametri Aurora, puoi utilizzare diversi strumenti, come la Console di gestione AWS, AWS CLI e l'API CloudWatch. È possibile visualizzare le metriche combinate di Performance Insights e CloudWatch nel pannello di controllo di Performance Insights e monitorare l'istanza database. Per utilizzare

questa visualizzazione di monitoraggio, Performance Insights deve essere attivato per l'istanza database specifica. Per ulteriori informazioni su questa visualizzazione di monitoraggio, consulta Visualizzazione delle metriche combinate con la dashboard di Approfondimenti sulle prestazioni.

È possibile creare un report di analisi delle prestazioni per un periodo di tempo specifico e visualizzare le informazioni dettagliate identificate e i suggerimenti per risolvere i problemi. Per ulteriori informazioni, consulta Creazione di un report di analisi delle prestazioni in Approfondimenti sulle prestazioni.

Uso di gruppi di parametri database e gruppi di parametri cluster database

È consigliabile provare le modifiche al gruppo di parametri database e al gruppo di parametri del cluster di database su un cluster di database di test prima di applicarle al cluster di database di produzione. Un'impostazione non corretta dei parametri del motore di database può avere ripercussioni negative non previste, tra cui prestazioni scadenti e instabilità del sistema.

Fai sempre attenzione quando modifichi i parametri del motore di database ed effettui il backup di un cluster di database prima di modificare un gruppo di parametri database. Per ulteriori informazioni sul backup di un cluster di database, consulta Backup e ripristino di un cluster DB Amazon Aurora.

Video sulle best practice Amazon Aurora

Nel canale AWS Online Tech Talks su YouTube è disponibile un video di presentazione sulle best practice per la creazione e la configurazione di un cluster database Amazon Aurora sicuro e a disponibilità elevata. Visualizzare il video sulle best practice per l'elevata disponibilità di Amazon Aurora.

Esecuzione di un proof of concept con Amazon Aurora

Di seguito, troverai una spiegazione di come impostare ed eseguire un proof of concept per Aurora. Un proof of concept è una verifica che esegui per capire se Aurora è la scelta giusta per la tua applicazione. Il proof of concept permette di comprendere le funzionalità di Aurora nel contesto delle tue applicazioni di database e di confrontare Aurora con il tuo attuale ambiente di database. Inoltre, mostra il livello di impegno necessario per spostare i dati, trasferire il codice SQL, ottimizzare le prestazioni e adattare le procedure di gestione correnti.

In questo argomento, troverai una panoramica e un riepilogo dettagliato delle procedure e delle decisioni generali associate all'esecuzione di un proof of concept, elencate di seguito. Per istruzioni dettagliate, puoi utilizzare i link alla documentazione completa per argomenti specifici.

Panoramica di un proof of concept di Aurora

L'esecuzione di un proof of concept per Amazon Aurora ti permette di capire cosa devi fare per trasferire i dati e le applicazioni SQL esistenti in Aurora. Puoi mettere in pratica gli aspetti importanti di Aurora su scala, utilizzando un volume di dati e attività rappresentativo del tuo ambiente di produzione. L'obiettivo è accertarti che i punti di forza di Aurora rispondano efficacemente alle sfide che ti hanno portato ad abbandonare la tua precedente infrastruttura di database. Al termine di un proof of concept, disporrai di un solido piano per il benchmark delle prestazioni su più vasta scala e il test dell'applicazione. A quel punto, sarai in grado di comprendere i maggiori elementi di lavoro per una distribuzione in produzione.

I seguenti consigli sulle best practice ti aiutano a evitare errori comuni che causano problemi durante il benchmark. Tuttavia, questo argomento non illustra il processo dettagliato per eseguire i benchmark e l'ottimizzazione delle prestazioni. Tali procedure variano a seconda del carico di lavoro e delle funzionalità di Aurora che utilizzi. Per informazioni dettagliate, consulta la documentazione relativa alle prestazioni, ad esempio Gestione delle prestazioni e del dimensionamento dei cluster DB Aurora, Miglioramenti alle prestazioni di Amazon Aurora MySQL, Prestazioni e dimensionamento per Amazon Aurora PostgreSQL e Monitoraggio del carico DB con Performance Insights su Amazon Aurora.

Le informazioni in questo argomento si riferiscono principalmente alle applicazioni in cui la tua organizzazione scrive il codice e progetta lo schema e che supportano i motori di database open source MySQL e PostgreSQL. Se stai testando un'applicazione commerciale o un codice generato dal framework di un'applicazione, potresti non avere la flessibilità necessaria per applicare tutte le

linee guida. In tal caso, rivolgiti al tuo rappresentante di AWS per sapere se esistono best practice o casi di studio di Aurora per il tuo tipo di applicazione.

1. Individuare gli obiettivi

Quando valuti Aurora nell'ambito di un proof of concept, puoi scegliere quali misurazioni eseguire e come valutare il successo della prova.

Devi accertarti che tutte le funzionalità dell'applicazione siano compatibili con Aurora. Poiché le versioni principali di Aurora sono compatibili con le corrispondenti versioni principali di MySQL e PostgreSQL, anche la maggior parte delle applicazioni sviluppate per tali motori è compatibile con Aurora. Tuttavia, è sempre necessario convalidare la compatibilità per ciascuna applicazione.

Ad esempio, alcune scelte relative alla configurazione fatte durante l'impostazione di un cluster Aurora influenzano la possibilità o la necessità di utilizzare specifiche funzionalità del database. Potresti iniziare con il tipo di cluster Aurora destinato a scopi più generici, detto assegnato. Successivamente, puoi stabilire se una configurazione specifica, ad esempio serverless o con query in parallelo, offre vantaggi per il tuo carico di lavoro.

Le domande seguenti aiutano a individuare e quantificare gli obiettivi:

- Aurora supporta tutti i casi d'uso funzionali del tuo carico di lavoro?
- Qual è la dimensione del set di dati o il livello di carico che desideri? Sei in grado di raggiungere tale livello?
- Quali sono i requisiti specifici in termini di throughput delle query o latenza? Sei in grado di soddisfarli?
- Qual è la quantità minima accettabile di tempo di inattività pianificato e non pianificato per il carico di lavoro? Sei in grado di raggiungerla?
- Quali sono i parametri necessari per l'efficienza operativa? Sei in grado di monitorarli in modo accurato?
- Aurora supporta obiettivi di business specifici, come la riduzione dei costi e l'aumento della velocità di distribuzione o provisioning? Hai un modo per quantificare questi obiettivi?
- Puoi soddisfare tutti i requisiti di sicurezza e conformità per il carico di lavoro?

Ti invitiamo ad acquisire conoscenza sui motori di database e sulle funzionalità della piattaforma di Aurora e a consultare la documentazione sui servizi. Prendi nota di tutte le funzionalità che possono

aiutarti a raggiungere i risultati desiderati. Una di queste potrebbe essere il consolidamento dei carichi di lavoro, descritto nel post di AWS Database Blog Come pianificare e ottimizzare la compatibilità di Amazon Aurora con MySQL per carichi di lavoro consolidati. Un'altra potrebbe essere la scalabilità in base alla domanda, descritta in Dimensionamento automatico di Amazon Aurora con le repliche Aurora nella Guida per l'utente di Amazon Aurora. Altre ancora potrebbero essere l'aumento delle prestazioni o la semplificazione delle operazioni di database.

2. Comprendere le caratteristiche del carico di lavoro

Valuta Aurora nel contesto del caso d'uso previsto. Aurora è una valida scelta per i carichi di lavoro di elaborazione di transazioni online (OLTP). Inoltre, è possibile eseguire report sul cluster che ospita i dati OLTP in tempo reale senza il provisioning di un cluster di data warehouse separato. Per capire se il tuo caso d'uso rientra in queste categorie, verifica le caratteristiche seguenti:

- Elevata simultaneità, con decine, centinaia o migliaia di client simultanei.
- Ampio volume di query a bassa latenza (da millisecondi a secondi).
- Transazioni brevi e in tempo reale.
- Modelli di query altamente selettivi, con ricerche basate su indici.
- Per l'HTAP, query analitiche che possono sfruttare la funzionalità di query in parallelo di Aurora.

Uno dei principali fattori che influenzano le scelte relative al database è la velocità dei dati. Velocità elevata significa che i dati vengono inseriti e aggiornati molto frequentemente. Un sistema di questo tipo potrebbe avere migliaia di connessioni e centinaia di migliaia di query simultanee con lettura e scrittura da e verso un database. Di solito, le query in sistemi a velocità elevata interessano un numero relativamente ridotto di righe e in genere accedono a più colonne nella stessa riga.

Aurora è concepito per gestire dati a velocità elevata. A seconda del carico di lavoro, un cluster Aurora con un'unica istanza di database r4.16xlarge può elaborare più di 600.000 istruzioni `SELECT` al secondo. Anche in questo caso, a seconda del carico di lavoro, tale cluster può elaborare 200.000 `INSERT`, `UPDATE` e `DELETE` istruzioni al secondo Aurora è un database con archiviazione su righe, ideale per carichi di lavoro OLTP con elevati livelli di volume, throughput e parallelismo.

Aurora può inoltre eseguire query di report nello stesso cluster che gestisce il carico di lavoro OLTP. Aurora supporta fino a 15 repliche, ciascuna delle quali è in media a 10-20 millisecondi dall'istanza primaria. Gli analisti possono eseguire query su dati OLTP in tempo reale senza copiare i dati in un cluster di data warehouse separato. Con i cluster Aurora che utilizzano la funzionalità di query in

parallelo, puoi effettuare l'offload di gran parte del lavoro di elaborazione, filtraggio e aggregazione al sottosistema di storage Aurora a elevata distribuzione.

Utilizza questa fase di pianificazione per acquisire familiarità con le funzionalità di Aurora, altri servizi AWS, la Console di gestione AWS e la AWS CLI. Inoltre, verifica il modo in cui queste funzionalità interagiscono con gli strumenti che prevedi di utilizzare nel proof of concept.

3. Fare pratica con la Console di gestione AWS o la AWS CLI

La fase successiva è fare pratica con la Console di gestione AWS o la AWS CLI, per prendere dimestichezza con questi strumenti e con Aurora.

Pratica con la Console di gestione AWS

Le seguenti attività iniziali con i cluster database Aurora consentono principalmente di familiarizzare con l'ambiente Console di gestione AWSe fare pratica con la configurazione e la modifica di cluster Aurora. Se utilizzi motori di database compatibili con MySQL e PostgreSQL con Amazon RDS, puoi sfruttare le stesse conoscenze per usare Aurora.

Sfruttando il modello di storage condiviso di Aurora e funzionalità come la replica e le snapshot, puoi trattare interi cluster database come un altro tipo di oggetto che gestisci liberamente. Durante il proof of concept, puoi frequentemente impostare, eliminare e modificare la capacità dei cluster Aurora. Non sei vincolato alle scelte iniziali relative a capacità, impostazioni del database e layout fisico dei dati.

Per iniziare, imposta un cluster Aurora vuoto. Scegli il tipo di capacità assegnato e l'ubicazione regionale per gli esperimenti iniziali.

Collegati al cluster utilizzando un programma client come un'applicazione a riga di comando SQL. Inizialmente, ti colleghi tramite l'endpoint del cluster. Ti collegi a tale endpoint per eseguire qualsiasi operazione di scrittura, come le istruzioni DDL (Data Definition Language) e i processi di estrazione, trasformazione e caricamento (ETL). Nelle fasi successive del proof of concept, colleghi sessioni con un numero elevato di query utilizzando l'endpoint di lettura, che distribuisce il carico di lavoro di query su più istanze database nel cluster.

Dimensiona il cluster aggiungendo altre repliche Aurora. Per queste procedure, consulta Replica con Amazon Aurora. Aumenta e riduci le istanze database modificando la classe di istanza AWS. Osserva come Aurora semplifica questi tipi di operazioni in modo tale che, se le stime iniziali della capacità di sistema non sono accurate, puoi regolarle in un secondo momento senza dover ricominciare da capo.

Crea una snapshot e ripristinala in un cluster diverso.

Esamina i parametri del cluster per visualizzare l'attività nel tempo e il modo in cui i parametri si applicano alle istanze database nel cluster.

È utile capire bene fin dall'inizio come svolgere queste operazioni mediante la Console di gestione AWS. Dopo aver capito cosa è possibile fare con Aurora, puoi procedere con l'automazione di tali operazioni tramite la AWS CLI. Le sezioni seguenti forniscono ulteriori dettagli sulle procedure e le best practice per queste attività durante il periodo del proof of concept.

Pratica con la AWS CLI

Consigliamo di automatizzare le procedure di distribuzione e gestione, anche nell'ambito di un proof of concept. Per farlo, acquisisci familiarità con la AWS CLI se non la conosci già. Se utilizzi motori di database compatibili con MySQL e PostgreSQL con Amazon RDS, puoi sfruttare le stesse conoscenze per usare Aurora.

Aurora include generalmente gruppi di istanze database disposte in cluster. Pertanto, molte operazioni prevedono di stabilire quali istanze database sono associate a un cluster ed eseguire poi le operazioni amministrative in un loop per tutte le istanze.

Ad esempio, potresti automatizzare passaggi come la creazione di cluster Aurora e il successivo dimensionamento con classi di istanze più grandi o il dimensionamento orizzontale con ulteriori istanze database. Questo aiuta a ripetere qualsiasi fase nel proof of concept e a esplorare scenari ipotetici con diversi tipi di configurazioni di cluster Aurora.

Scopri le funzionalità e le limitazioni degli strumenti di distribuzione dell'infrastruttura come AWS CloudFormation. Potresti constatare che le attività che esegui nel contesto di un proof of concept non sono indicate per l'utilizzo in produzione. Ad esempio, il comportamento di CloudFormation per la modifica è creare una nuova istanza ed eliminare quella corrente, compresi i relativi dati. Per maggiori dettagli su questo comportamento, consulta Aggiornamento dei comportamenti delle risorse stack nella Guida per l'utente di AWS CloudFormation.

4. Creare il cluster Aurora

Con Aurora, puoi esplorare scenari ipotetici aggiungendo istanze database al cluster e dimensionando le istanze database a classi di istanze più potenti. Puoi anche creare cluster con diverse impostazioni di configurazione per eseguire lo stesso carico di lavoro contemporaneamente. Con Aurora, hai molta flessibilità per impostare, eliminare e riconfigurare i cluster database. Pertanto,

è utile fare pratica con queste tecniche nelle prime fasi del processo di proof of concept. Per le procedure generali per la creazione di cluster Aurora, consulta Creazione di un cluster database Amazon Aurora.

Se utile, inizia con un cluster con le seguenti impostazioni. Salta questo passaggio solo se prevedi casi d'uso specifici. Ad esempio, potresti saltare questo passaggio se il caso d'uso richiede un tipo speciale di cluster Aurora oppure se necessiti di una particolare combinazione di versione e motore di database.

- Disattiva Creazione rapida. Per il proof of concept, ti consigliamo di conoscere tutte le impostazioni scelte in modo da poter creare cluster identici o leggermente diversi in un secondo momento.
- Usa una versione del motore di database recente. Questa combinazione di versione e motore del database ha un'ampia compatibilità con altre funzionalità di Aurora e un notevole utilizzo da parte dei clienti per le applicazioni di produzione.
  - Aurora MySQL versione 3 (compatibile con MySQL 8.0)
  - Aurora PostgreSQL versione 15.x o 16x
- Scegli il modello Dev/Test (Sviluppo/Test). Questa scelta non è rilevante per le attività del proof of concept.
- Per DB instance class (Classe dell'istanza database), scegli Memory optimized classes (Classi ottimizzate per la memoria) e una delle classi di istanze xlarge. Puoi modificare la classe di istanza in un secondo momento.
- Sotto Multi-AZ Deployment (Implementazione Multi-AZ), scegli Create an Aurora Replica or Reader node in a different AZ (Crea una replica Aurora o nodo di lettura in un AZ differente). Molti degli aspetti più utili di Aurora riguardano i cluster e le diverse istanze database. È utile iniziare sempre con almeno due istanze database in qualsiasi nuovo cluster. L'utilizzo di una diversa zona di disponibilità per la seconda istanza database aiuta a testare diversi scenari di disponibilità elevata.
- Quando scegli i nomi per le istanze database, utilizza una convenzione di denominazione generica. Non fare riferimento alle istanze database del cluster come "di scrittura", in quanto diverse istanze database assumono tali ruoli in base alle esigenze. Consigliamo di utilizzare una denominazione di tipo `clustername-az-serialnumber`, ad esempio `myprodappdb-a-01`. Tali denominazioni identificano l'istanza database e il suo posizionamento.
- Imposta una conservazione elevata del backup per il cluster Aurora. Con un lungo periodo di conservazione, puoi eseguire il ripristino point-in-time (PITR) per un periodo che arriva fino a 35 giorni. Puoi ripristinare il database a uno stato noto dopo l'esecuzione dei test che includono istruzioni DDL e DML (Data Manipulation Language). Inoltre, puoi eseguire il ripristino in caso di eliminazione o modifica involontaria dei dati.

- Abilita ulteriori funzionalità di ripristino, registrazione e monitoraggio delle funzioni al momento della creazione del cluster. Attiva tutte le scelte in Backtrack, Approfondimenti sulle prestazioni, Monitoraggio e Esportazioni log. L'attivazione di queste funzionalità consente di testare l'idoneità di funzionalità come backtracking, Enhanced Monitoring (Monitoraggio avanzato) o Performance Insights per il carico di lavoro in uso. Inoltre, puoi eseguire facilmente l'analisi delle prestazioni e la risoluzione dei problemi durante il proof of concept.

5. Configurare lo schema

Nel cluster Aurora, configura i database, le tabelle, gli indici, le chiavi esterne e gli altri oggetti dello schema per l'applicazione. Se effettui il passaggio da un altro sistema di database compatibile con MySQL o con PostgreSQL, questa fase sarà semplice e diretta. Per il motore di database, utilizzi le stesse sintassi e riga di comando di SQL o altre applicazioni client che già conosci.

Per inviare istruzioni SQL al cluster, individua l'endpoint del cluster e fornisci quel valore come parametro di connessione all'applicazione client. Puoi trovare l'endpoint del cluster nella scheda Connectivity (Connettività) della pagina dei dettagli del cluster. L'endpoint del cluster è quello con l'etichetta Writer (di scrittura). L'altro endpoint, con etichetta Reader (di lettura), rappresenta una connessione di sola lettura che puoi fornire agli utenti finali che eseguono report o altre query di sola lettura. Per eventuali problemi di connessione al cluster, consulta Connessione a un cluster database Amazon Aurora.

Se trasferisci lo schema e i dati da un sistema di database diverso, probabilmente a questo punto dovrai apportare modifiche allo schema. Queste modifiche allo schema devono essere in linea con la sintassi e le funzionalità SQL disponibili in Aurora. A questo punto, potresti tralasciare alcuni vincoli, colonne e trigger o altri oggetti dello schema. Questo può essere utile soprattutto se tali oggetti richiedono la rielaborazione per la compatibilità Aurora e non sono importanti per gli obiettivi del proof of concept.

Se esegui la migrazione da un sistema di database con un motore sottostante diverso da quello di Aurora, valuta l'utilizzo di AWS Schema Conversion Tool (AWS SCT) per semplificare il processo. Per informazioni dettagliate, consulta la Guida per l'utente di AWS Schema Conversion Tool. Per informazioni generiche sulle attività di migrazione e trasferimento, consulta il whitepaper AWS Migrazione dei database su Amazon Aurora .

Durante questa fase, puoi valutare se vi sono inefficienze nell'impostazione dello schema, ad esempio nella strategia di indicizzazione o altre strutture a tabella come le tabelle partizionate. Tali inefficienze possono essere amplificate quando implementi l'applicazione su un cluster con

più istanze database e un carico di lavoro pesante. Valuta se è possibile ottimizzare questi aspetti prestazionali ora o durante attività successive come un test di benchmark completo.

6. Importare i dati

Durante il proof of concept, trasferisci i dati o un campione rappresentativo degli stessi dal precedente sistema di database. Se utile, imposta almeno alcuni dati in ciascuna delle tue tabelle. Ciò aiuta a testare la compatibilità di tutti i tipi di dati e le funzionalità dello schema. Dopo aver fatto pratica con le funzionalità base di Aurora, aumenta la quantità dei dati. Al termine del proof of concept, dovresti testare gli strumenti ETL, le query e il carico di lavoro complessivo con un set di dati abbastanza grande da consentire di trarre conclusioni accurate.

Puoi utilizzare diverse tecniche per importare dati di backup fisici o logici in Aurora. Per i dettagli, consulta Migrazione di dati a un cluster di database Amazon Aurora MySQL o Migrazione di dati su Amazon Aurora con compatibilità PostgreSQL a seconda del motore di database che utilizzi nel proof of concept.

Esegui gli esperimenti con gli strumenti e le tecnologie ETL che stai valutando. Individua quelli che soddisfano meglio le tue esigenze. Prendi in considerazione sia il throughput che la flessibilità. Ad esempio, alcuni strumenti ETL eseguono un trasferimento una tantum e altri implicano la replica continua dal sistema precedente ad Aurora.

Se effettui la migrazione da un sistema compatibile con MySQL ad Aurora MySQL, puoi utilizzare gli strumenti di trasferimento dati nativi. Lo stesso vale se effettui la migrazione da un sistema compatibile con PostgreSQL ad Aurora PostgreSQL. Se effettui la migrazione da un sistema di database che utilizza un motore sottostante diverso da quello di Aurora, puoi continuare a sperimentare con AWS Database Migration Service (AWS DMS). Per ulteriori dettagli su AWS DMS, consulta la Guida per l'utente di AWS Database Migration Service.

Per informazioni sulle attività di migrazione e trasferimento, consulta il whitepaper AWS Manuale di migrazione ad Aurora.

7. Trasferire il codice SQL

Provare SQL e le applicazioni correlate richiede diversi livelli di impegno a seconda dei casi. In particolare, il livello di impegno dipende dal fatto che il passaggio avvenga da un sistema compatibile con MySQL o con PostgreSQL oppure da un altro tipo di sistema.

- Se effettui il passaggio da RDS for MySQL o RDS for PostgreSQL, le modifiche di SQL sono abbastanza ridotte da consentire di provare il codice SQL originale con Aurora e incorporare manualmente le modifiche necessarie.
- Analogamente, se il passaggio avviene da un database locale compatibile con MySQL o PostgreSQL, puoi provare il codice SQL originale e incorporare le modifiche manualmente.
- Se provieni da un database commerciale diverso, le modifiche SQL richieste sono più ampie. In questo caso, valuta l'utilizzo di AWS SCT.

Durante questa fase, puoi valutare se vi sono inefficienze nell'impostazione dello schema, ad esempio nella strategia di indicizzazione o altre strutture a tabella come le tabelle partizionate. Valuta se è possibile ottimizzare questi aspetti prestazionali ora o durante attività successive come un test di benchmark completo.

Puoi verificare la logica di connessione del database nella tua applicazione. Per sfruttare l'elaborazione distribuita di Aurora, potresti dover utilizzare connessioni separate per le operazioni di lettura e scrittura e utilizzare sessioni relativamente brevi per le operazioni di query. Per informazioni sulle connessioni, consulta 9. Connettere ad Aurora.

Prendi in considerazione l'eventualità di dover scendere a compromessi o fare rinunce per risolvere problemi nel database di produzione. Nella pianificazione del proof of concept prevedi del tempo da dedicare al miglioramento della progettazione dello schema e alle query. Per capire se puoi facilmente ottenere vantaggi in termini di prestazioni, costi operativi e scalabilità, prova l'applicazione originale e quella modificata contemporaneamente su cluster Aurora diversi.

Per informazioni sulle attività di migrazione e trasferimento, consulta il whitepaper AWS Manuale di migrazione ad Aurora.

8. Specificare le impostazioni di configurazione

Nell'ambito del proof of concept di Aurora, puoi anche rivedere i parametri di configurazione del database. Nell'ambiente corrente potresti già avere impostazioni di configurazione di MySQL o PostgreSQL ottimizzate per le prestazioni e la scalabilità. Il sottosistema di storage Aurora è adattato e ottimizzato per un ambiente basato sul cloud e distribuito con un sottosistema di storage ad alta velocità. Di conseguenza, molte impostazioni del motore di database precedente non sono valide. Consigliamo di condurre gli esperimenti iniziali con le impostazioni di configurazione di Aurora predefinite. Riapplica le impostazioni dell'ambiente corrente solo in caso di colli di bottiglia di

prestazioni e scalabilità. Ulteriori informazioni su questo tema sono disponibili nel post Introduzione al motore di archiviazione Aurora su AWS Database Blog.

Aurora agevola il riutilizzo delle impostazioni di configurazione ottimali per una particolare applicazione o caso d'uso. Anziché modificare un file di configurazione separato per ciascuna istanza database, gestisci set di parametri che assegni a interi cluster o a specifiche istanze database. Ad esempio, l'impostazione del fuso orario si applica a tutte le istanze database nel cluster e puoi regolare l'impostazione della dimensione della cache della pagina per ciascuna istanza database.

Inizi con uno dei set di parametri predefiniti e applichi le modifiche solo ai parametri che devi ottimizzare. Per informazioni sull'utilizzo dei gruppi di parametri, consulta Parametri dell'istanza database e del cluster database di Amazon Aurora. Per le impostazioni di configurazione che sono o non sono applicabili ai cluster Aurora, consulta Parametri di configurazione Aurora MySQL o Amazon Aurora PostgreSQL parametri a seconda del motore di database in uso.

9. Connettere ad Aurora

Proprio come per l'impostazione dello schema e dei dati iniziali e l'esecuzione di query di esempio, puoi connetterti a diversi endpoint in un cluster Aurora. L'endpoint da utilizzare dipende dal fatto che l'operazione sia una lettura come l'istruzione `SELECT` oppure una scrittura come l'istruzione `CREATE` o `INSERT`. Quando incrementi il carico di lavoro su un cluster Aurora e sperimenti le funzionalità di Aurora, è importante che l'applicazione assegni ciascuna operazione all'endpoint appropriato.

Utilizzando l'endpoint del cluster per operazioni di scrittura, ti connetti sempre a un'istanza database nel cluster che ha funzionalità di lettura e scrittura. Per impostazione predefinita, solo un'istanza database in un cluster Aurora ha funzionalità di lettura e scrittura. Questa istanza database è detta istanza primaria. Se l'istanza primaria originale diventa non disponibile, Aurora attiva un meccanismo di failover e un'istanza database diversa diventa quella primaria.

Analogamente, indirizzando istruzioni `SELECT` all'endpoint di lettura, distribuisci il lavoro di elaborazione delle query tra le istanze database nel cluster. Ciascuna connessione viene assegnata a un'istanza database diversa tramite risoluzione DNS round robin. L'esecuzione della maggior parte del lavoro di query sulle repliche database Aurora di sola lettura riduce il carico sull'istanza primaria, liberandola per la gestione delle istruzioni DDL e DML.

L'utilizzo di questi endpoint riduce la dipendenza a nomi host hardcoded e aiuta l'applicazione a eseguire un ripristino più rapido in caso di errori delle istanze database.

> **Note**
>
> Inoltre Aurora consente di creare endpoint personalizzati. Questi endpoint non sono generalmente necessari durante un proof of concept.

Le repliche Aurora sono soggette a un ritardo di replica, anche se tale ritardo è di solito da 10 a 20 millisecondi. Puoi monitorare il ritardo di replica e stabilire se rientra nell'intervallo dei requisiti di coerenza dei tuoi dati. In alcuni casi, le query di lettura potrebbero richiedere un'elevata coerenza di lettura (coerenza lettura dopo scrittura). In questi casi, puoi continuare a utilizzare l'endpoint del cluster anziché l'endpoint di lettura.

Per sfruttare appieno le funzionalità di Aurora per l'esecuzione in parallelo distribuita, potresti dover modificare la logica della connessione. L'obiettivo è evitare di inviare tutte le richieste di lettura all'istanza primaria. Le repliche Aurora di sola lettura sono in stand by, con tutti gli stessi dati, pronte per gestire le istruzioni `SELECT`. Codifica la logica dell'applicazione per utilizzare l'endpoint appropriato per ciascun tipo di operazione. Segui queste linee guida generali:

- Evita di utilizzare un'unica stringa di connessione hardcoded per tutte le sessioni di database.
- Se utile, includi operazioni di scrittura come le istruzioni DDL e DML nelle funzioni nel codice dell'applicazione client. In questo modo, diversi tipi di applicazioni possono utilizzare connessioni specifiche.
- Creare funzioni separate per le operazioni di query. Aurora assegna ciascuna nuova connessione all'endpoint di lettura a una replica Aurora diversa per bilanciare il carico per applicazioni con attività di lettura intensiva.
- Per le operazioni che implicano set di query, chiudi e riapri la connessione all'endpoint di lettura quando ciascun set di query correlate è terminato. Utilizza il pooling della connessione se tale funzionalità è disponibile nello stack del software. L'indirizzamento di query a diverse connessioni aiuta Aurora a distribuire il carico di lavoro di lettura tra le istanze database nel cluster.

Per informazioni generali sulla gestione della connessione e sugli endpoint per Aurora, consulta Connessione a un cluster database Amazon Aurora. Per un approfondimento di questo argomento, consulta l'argomento relativo alla gestione delle connessioni in Aurora MySQL Database Administrator's Handbook –.

10. Eseguire il carico di lavoro

Dopo aver eseguito le impostazioni di schema, dati e configurazione, puoi iniziare a esercitarti con il cluster eseguendo il carico di lavoro. Nel proof of concept utilizza un carico di lavoro che rispecchi i numerosi aspetti del tuo carico di lavoro di produzione. Consigliamo di prendere sempre le decisioni relative alle prestazioni utilizzando test e carichi di lavoro realistici anziché benchmark sintetici come sysbench o TPC-C. Nel caso, raccogli le misurazioni in base allo schema, ai modelli di query e al volume di utilizzo.

Nella misura in cui risulta utile, replica le effettive condizioni in cui verrà eseguita l'applicazione. Ad esempio, generalmente esegui il codice dell'applicazione su istanze Amazon EC2 nella stessa regione AWS e nello stesso Virtual Private Cloud (VPC) del cluster Aurora. Se l'applicazione di produzione viene eseguita su più istanze EC2 disposte in più zone di disponibilità, imposta l'ambiente del proof of concept nello stesso modo. Per ulteriori informazioni sulle regioni AWS, consulta Regioni e zone di disponibilità nella Guida per l'utente di Amazon RDS. Per ulteriori informazioni sul servizio Amazon VPC, consulta Che cos'è Amazon VPC? nella Guida per l'utente di Amazon VPC.

Dopo aver verificato il funzionamento delle funzionalità di base dell'applicazione e la possibilità di accedere ai dati tramite Aurora, puoi fare pratica con gli aspetti del cluster Aurora. Alcune funzionalità che potresti voler provare sono connessioni simultanee con il bilanciamento del carico, transazioni simultanee e replica automatica.

A questo punto, i meccanismi di trasferimento dei dati dovrebbero essere familiari e puoi quindi eseguire i test con una maggiore quantità di dati campione.

Questa è la fase in cui puoi vedere gli effetti della modifica delle impostazioni di configurazione come i limiti di memoria e di connessione. Rivedi le procedure che hai esplorato in 8. Specificare le impostazioni di configurazione.

Puoi inoltre sperimentare meccanismi come la creazione e il ripristino di snapshot. Ad esempio, puoi creare cluster con diverse classi di istanze AWS, numeri di repliche AWS e così via. In ogni cluster, puoi quindi ripristinare la stessa snapshot che contiene lo schema e tutti i dati. Per i dettagli su questo ciclo, consulta Creazione di uno snapshot del cluster database e Ripristino da uno snapshot cluster database.

11. Misurare le prestazioni

Le best practice in questa area sono state concepite per assicurare l'impostazione di tutti gli strumenti e i processi giusti per isolare velocemente comportamenti anomali durante le operazioni del carico di

lavoro. Servono inoltre a constatare la possibilità di identificare in modo attendibile eventuali cause applicabili.

Puoi sempre visualizzare lo stato corrente del cluster o esaminare le tendenze nel tempo, analizzando la scheda Monitoring (Monitoraggio). Questa scheda è disponibile nella pagina dei dettagli della console per ciascun cluster Aurora o istanza database. Mostra i parametri del servizio di monitoraggio di Amazon CloudWatch sotto forma di grafici. Puoi filtrare i parametri per nome, per istanza database e per periodo di tempo.

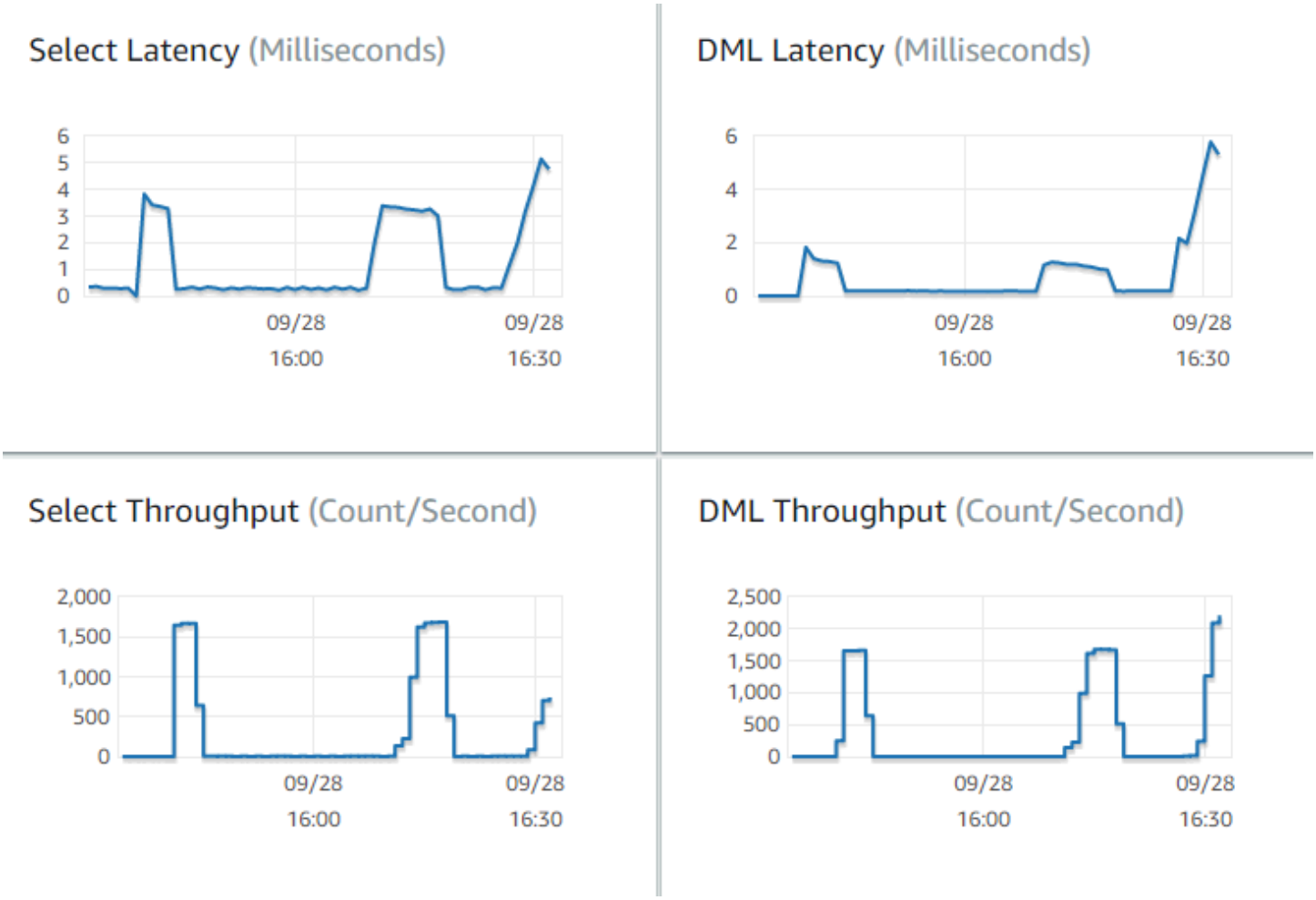
Per avere più scelte nella scheda Monitoring (Monitoraggio), abilita Enhanced Monitoring (Monitoraggio avanzato) e Performance Insights nelle impostazioni del cluster. Inoltre, se non abiliti queste scelte al momento della configurazione del cluster, puoi farlo in un secondo momento.

Per misurare le prestazioni, puoi utilizzare i grafici che mostrano l'attività dell'intero cluster Aurora. Puoi verificare se le repliche Aurora hanno simili tempi di caricamento e risposta. Puoi anche visualizzare la suddivisione del lavoro tra l'istanza primaria di lettura e scrittura e le repliche Aurora di sola lettura. Se le istanze database non sono bilanciate o nel caso in cui un problema interessi solo un'istanza database, puoi esaminare la scheda Monitoring (Monitoraggio) di quella specifica istanza.

Dopo l'impostazione dell'ambiente e del carico di lavoro effettivo per emulare l'applicazione di produzione, puoi misurare le prestazioni di Aurora. Le domande più importanti da porsi sono le seguenti:

- Quante query al secondo elabora Aurora? Puoi esaminare i parametri Throughput per visualizzare i dati per diversi tipi di operazioni.
- Quanto tempo impiega in media Aurora per elaborare una determinata query? Puoi esaminare le metriche Latency (Latenza) per visualizzare i dati per diversi tipi di operazioni.

Per visualizzare le metriche di throughput e latenza, consulta la scheda Monitoraggio per un determinato cluster Aurora nella console Amazon RDS. Lo screenshot seguente mostra un esempio delle metriche Select Latency (Seleziona latenza), DML Latency (Latenza DML), Select Throughput (Seleziona throughput) e DML Throughput (Throughput DML) nella scheda Monitoraggio.



Se possibile, stabilisci valori di base per questi parametri nell'ambiente corrente. Se ciò non è pratico, crea una baseline nel cluster Aurora eseguendo un carico di lavoro equivalente all'applicazione di produzione. Ad esempio, esegui il carico di lavoro Aurora con un numero simile di utenti e query simultanei. Osserva quindi in che modo i valori cambiano durante l'esperimento con diverse classi di istanze, dimensioni del cluster, impostazioni di configurazione e così via.

Se i valori del throughput sono inferiori a quanto previsto, analizza ulteriormente i fattori che influenzano le prestazioni del database per il carico di lavoro. Analogamente, se i valori della latenza sono superiori rispetto a quanto previsto, effettua ulteriori analisi. Per farlo, monitora i parametri secondari per il server database (CPU, memoria e così via). Puoi vedere se le istanze database sono vicine ai loro limiti. Inoltre, puoi vedere quanta capacità aggiuntiva hanno le istanze database per gestire più query simultanee, query su tabelle di grandi dimensioni e così via.

> **Tip**
>
> Per rilevare i valori dei parametri che non rientrano negli intervalli previsti, imposta gli allarmi di CloudWatch .

Quando valuti le dimensioni e la capacità ideali del cluster Aurora, puoi trovare la configurazione che raggiunge le massime prestazioni dell'applicazione senza l'overprovisioning delle risorse. Un fattore importante è trovare le dimensioni giuste per le istanze database nel cluster Aurora. Inizia selezionando le dimensioni di un'istanza che ha CPU e capacità di memoria simili al tuo attuale ambiente di produzione. Raccogli i valori di throughput e latenza per il carico di lavoro con queste dimensioni dell'istanza. Dopo di che, aumenta l'istanza alla dimensione successiva più grande. Verifica se i valori di throughput e latenza migliorano. Inoltre, riduci le dimensioni dell'istanza per vedere se i valori di latenza e throughput rimangono invariati. L'obiettivo è ottenere il massimo throughput e la minima latenza sull'istanza più piccola possibile.

> **Tip**
>
> Dimensiona i cluster Aurora e le istanze database correlate con abbastanza capacità esistente per gestire picchi di traffico improvvisi e imprevedibili. Per i database mission critical, lascia almeno il 20% di CPU e capacità di memoria di scorta.

Esegui test delle prestazioni abbastanza lunghi da misurare le prestazioni del database in uno stato attivo e costante. Potresti dover eseguire il carico di lavoro per molti minuti o anche qualche ora prima di raggiungere questo stato costante. All'inizio di un'esecuzione è normale avere delle variazioni. Queste variazioni avvengono perché ciascuna replica Aurora attiva le proprie cache in base alle query `SELECT` che gestisce.

Aurora offre massime prestazioni con carichi di lavoro transazionali che implicano più utenti e query simultanei. Per accertarti che il carico sia sufficiente per prestazioni ottimali, esegui benchmark che utilizzano multithreading o esegui più istanze dei test prestazionali simultaneamente. Misura le prestazioni con centinaia o anche migliaia di thread client simultanei. Simula il numero di thread simultanei che prevedi nel tuo ambiente di produzione. Potresti anche eseguire ulteriori stress test con più thread per misurare la scalabilità di Aurora.

12. Fare pratica con la disponibilità elevata di Aurora

Molte delle principali funzionalità di Aurora implicano disponibilità elevata. Queste funzionalità includono la replica automatica, il failover automatico, i backup automatici con ripristino point-in-time e la possibilità di aggiungere istanze database al cluster. La sicurezza e l'affidabilità derivanti da funzionalità come queste sono importanti per le applicazioni mission critical.

La valutazione di tali funzionalità richiede un determinato approccio. Nelle attività precedenti, come la misurazione delle prestazioni, puoi osservare le prestazioni del sistema quando tutto funziona correttamente. Il test della disponibilità elevata richiede di ipotizzare il peggiore comportamento possibile. Occorre considerare vari tipi di errore, anche se tali condizioni sono rare. Potresti creare intenzionalmente dei problemi per accertarti che il sistema effettui il ripristino in modo corretto e veloce.

> **Tip**
>
> Per un proof of concept, imposta tutte le istanze database in un cluster Aurora con la stessa classe di istanza AWS. In questo modo è possibile provare la funzionalità di disponibilità di Aurora senza importanti modifiche alle prestazioni e alla scalabilità quando porti le istanze database offline per simulare gli errori.

Consigliamo di utilizzare almeno due istanze in ciascun cluster Aurora. Le istanze database in un cluster Aurora possono coprire fino a tre zone di disponibilità. Posiziona ciascuna delle prime due o tre istanze database in una zona di disponibilità diversa. Quando inizi a utilizzare cluster più grandi, distribuisci le istanze database in tutte le zone di disponibilità della tua regione AWS. In questo modo, aumenta la tolleranza agli errori. Anche se un problema colpisce un'intera zona di disponibilità, Aurora esegue il failover su un'istanza database in un'altra zona di disponibilità. Se esegui un cluster con più di tre istanze, distribuisci le istanze database il più equamente possibile in tutte e tre le zone di disponibilità.

> **Tip**
>
> Lo storage di un cluster Aurora è indipendente dalle istanze database. Lo storage di ciascun cluster Aurora copre sempre tre zone di disponibilità.

> Quando testi le funzionalità di disponibilità elevata, utilizza sempre istanze database con capacità identica nel cluster del test. Ciò permette di evitare cambiamenti imprevisti di prestazioni, latenza e così via quando un'istanza database prende il posto di un'altra.

Per capire come simulare condizioni di errore per testare le funzionalità di disponibilità elevata, consulta Test di Amazon Aurora MySQL mediante query Fault Injection.

Nell'ambito del proof of concept, uno degli obiettivi è trovare il numero ideale di istanze database e la classe di istanza ottimale per tali istanze. Ciò richiede il bilanciamento dei requisiti di disponibilità elevata e delle prestazioni.

Per Aurora, più istanze database sono presenti in un cluster, maggiori sono i vantaggi per la disponibilità elevata. Inoltre, avere più istanze database migliora la scalabilità delle applicazioni a lettura intensiva. Aurora può distribuire più connessioni per query `SELECT` tra le repliche Aurora di sola lettura.

Dall'altro lato, limitare il numero di istanze database riduce il traffico di replica dal nodo primario. Il traffico di replica consuma larghezza di banda di rete, altro aspetto delle prestazioni complessive e della scalabilità. Pertanto, per le applicazioni OLTP con scrittura intensiva, è preferibile un numero ridotto di grandi istanze database anziché numerose istanze database di piccole dimensioni.

In un cluster Aurora tipico, un'unica istanza database (istanza primaria) gestisce tutte le istruzioni DDL e DML. Le altre istanze database (repliche Aurora) gestiscono solo le istruzioni `SELECT`. Sebbene le istanze database non svolgano esattamente la stessa quantità di lavoro, consigliamo di utilizzare la stessa classe di istanza per tutte le istanze database nel cluster. In questo modo, se si verifica un errore e Aurora promuove una delle istanze database di sola lettura a nuova istanza primaria, l'istanza primaria ha la stessa capacità di prima.

Se devi utilizzare istanze database con capacità diverse nello stesso cluster, imposta i tier di failover per le istanze database. Questi tier determinano l'ordine in cui le repliche Aurora vengono promosse dal meccanismo di failover. Le istanze database che sono molto più grandi o più piccole delle altre vanno collocate in un tier di failover inferiore. Ciò assicura che verranno scelte per ultime per la promozione.

Fai pratica con le funzionalità di ripristino dei dati di Aurora, come il ripristino point-in-time automatico, le snapshot e il ripristino manuali e il backtrack del cluster. Se appropriato, copia le snapshot in altre regioni AWS ed esegui il ripristino in altre regioni AWS per simulare scenari di DR.

Scopri quali sono i requisiti della tua organizzazione in termini di RTO (Restore Time Objective), RPO (Restore Point Objective) e ridondanza geografica. La maggior parte delle organizzazioni raggruppa questi elementi nell'ampia categoria del disaster recovery. Valuta le funzionalità di disponibilità elevata di Aurora descritte in questa sezione nel contesto del tuo processo di disaster recovery per accertarti che i tuoi requisiti di RTO e RPO siano soddisfatti.

13. Cosa fare in seguito

Il buon esito del processo di proof of concept conferma che Aurora è la soluzione giusta per il tuo carico di lavoro previsto. Durante il processo precedente, hai verificato il funzionamento di Aurora in un ambiente operativo realistico e lo hai valutato a fronte dei tuoi criteri di successo.

Dopo aver impostato e avviato l'ambiente di database con Aurora, puoi procedere a una valutazione più dettagliata, che porta alla migrazione finale e alla distribuzione della produzione. A seconda della situazione, questi ulteriori passaggi potrebbero essere inclusi o meno nel processo del proof of concept. Per informazioni sulle attività di migrazione e trasferimento, consulta il whitepaper AWS Manuale di migrazione ad Aurora.

In un altro passaggio successivo, prendi in considerazione le configurazioni di sicurezza pertinenti per il tuo carico di lavoro e concepite per rispondere ai requisiti di sicurezza in un ambiente di produzione. Pianifica i controlli da predisporre per proteggere l'accesso alle credenziali degli utenti master del cluster Aurora. Definisci i ruoli e le responsabilità degli utenti del database per controllare l'accesso ai dati archiviati nel cluster Aurora. Prendi in considerazione i requisiti di accesso al database per le applicazioni, gli script e gli strumenti o i servizi di terze parti. Esplora i servizi e le funzionalità di AWS, come l'autenticazione Gestione dei segreti AWS e AWS Identity and Access Management (IAM).

A questo punto, le procedure e le best practice per l'esecuzione di test di benchmark con Aurora dovrebbero essere chiare. Potresti dover effettuare un'ulteriore ottimizzazione delle prestazioni. Per i dettagli, consulta Gestione delle prestazioni e del dimensionamento dei cluster DB Aurora, Miglioramenti alle prestazioni di Amazon Aurora MySQL, Prestazioni e dimensionamento per Amazon Aurora PostgreSQL e Monitoraggio del carico DB con Performance Insights su Amazon Aurora. Se effettui un'ulteriore ottimizzazione, assicurati di conoscere bene i parametri che hai raccolto durante il proof of concept. Come passaggio successivo, potresti creare nuovi cluster con diverse scelte per le impostazioni di configurazione, il motore di database e la versione del database. In alternativa, potresti creare tipi di cluster Aurora per rispondere ai requisiti di specifici casi d'uso.

Ad esempio, puoi esplorare i cluster di query in parallelo di Aurora per le applicazioni HTAP (Hybrid Transaction/Analytical Processing). Se un'ampia distribuzione geografica è fondamentale per il

disaster recovery o per ridurre al minimo la latenza, puoi esaminare Aurora Global Database. Se il tuo carico di lavoro è intermittente o se utilizzi Aurora in uno scenario di sviluppo/test, puoi esaminare i cluster Aurora Serverless.

I cluster di produzione potrebbero inoltre dover gestire elevati volumi di connessioni in entrata. Per conoscere queste tecniche, consulta il whitepaper AWSManuale per l'amministratore del database Aurora MySQL: gestione delle connessioni.

Se, dopo il proof of concept, stabilisci che il tuo caso d'uso non è adatto per Aurora, valuta questi altri sevizi AWS:

- Per casi di utilizzo puramente analitici, i carichi di lavoro beneficiano di un formato di archiviazione colonnare e di altre funzionalità più adatte ai carichi di lavoro OLAP. I servizi AWS che affrontano tali casi d'uso includono quanto segue:
  - Amazon Redshift
  - Amazon EMR
  - Amazon Athena
- Molti carichi di lavoro traggono vantaggio da una combinazione di Aurora con uno o più di questi servizi. Puoi spostare i dati tra questi servizi utilizzando quanto segue:
  - AWS Glue
  - AWS DMS
  - Importazione da Amazon S3, come descritto nella Guida per l'utente di Amazon Aurora
  - Esportazione verso Amazon S3, come descritto nella Guida per l'utente di Amazon Aurora
  - Molti altri strumenti ETL diffusi

Sicurezza in Amazon Aurora

La sicurezza del cloud AWS è la massima priorità. In qualità di AWS cliente, puoi beneficiare di un data center e di un'architettura di rete progettati per soddisfare i requisiti delle organizzazioni più sensibili alla sicurezza.

La sicurezza è una responsabilità condivisa tra AWS te e te. Il modello di responsabilità condivisa descrive questo come sicurezza del cloud e sicurezza nel cloud:

- Sicurezza del cloud: AWS è responsabile della protezione dell'infrastruttura che gestisce AWS i servizi nel AWS cloud. AWS ti fornisce anche servizi che puoi utilizzare in modo sicuro. I revisori di terze parti testano e verificano regolarmente l'efficacia della sicurezza come parte dei programmi di conformitàAWS. Per ulteriori informazioni sui programmi di conformità che si applicano ad Amazon Aurora (Aurora), consulta Servizi AWS coperti dal programma di compliance.
- Sicurezza nel cloud: la tua responsabilità è determinata dal AWS servizio che utilizzi. L'utente è anche responsabile per altri fattori, tra cui la riservatezza dei dati, i requisiti dell'azienda, nonché le leggi e le normative applicabili.

La presente documentazione aiuta a comprendere come applicare il modello di responsabilità condivisa quando si utilizza Amazon Aurora. I seguenti argomenti illustrano come configurare Amazon Aurora per soddisfare gli obiettivi di sicurezza e conformità. Scopri anche come utilizzare altri AWS servizi che ti aiutano a monitorare e proteggere le tue risorse Amazon Aurora.

È possibile gestire l’accesso alle risorse Amazon Aurora e ai database in un cluster di database. Il metodo utilizzato per gestire l’accesso dipende dal tipo di attività che l’utente deve eseguire con Amazon Aurora:

- Esegui il cluster di database in un cloud privato virtuale (VPC) basato sul servizio Amazon VPC per il massimo controllo possibile dell’accesso alla rete. Per ulteriori informazioni sulla creazione di un cluster di database in un VPC, consulta VPC Amazon e Amazon Aurora .
- Utilizza le policy AWS Identity and Access Management (IAM) per assegnare autorizzazioni che determinano chi è autorizzato a gestire le risorse Amazon . Ad esempio, è possibile utilizzare IAM per determinare chi è autorizzato a creare, descrivere, modificare ed eliminare cluster di database, applicare tag alle risorse oppure modificare i gruppi di sicurezza.

  Per esempi di policy IAM, consulta Esempi di policy di Amazon Aurora basate su identità.

- Usa i gruppi di sicurezza per controllare quali indirizzi IP o EC2 istanze Amazon possono connettersi ai tuoi database su un cluster di DB. Quando si crea per la prima volta un cluster di database, il firewall impedisce qualsiasi accesso al database tranne che attraverso le regole specificate da un gruppo di sicurezza associato.
- Utilizzare le connessioni Secure Socket Layer (SSL) o Transport Layer Security (TLS) con cluster di database che eseguono Aurora MySQL o Aurora PostgreSQL. Per ulteriori informazioni sull'utilizzo SSL/TLS con un cluster DB, vedereUtilizzo SSL/TLS per crittografare una connessione a un'.
- Utilizzare la crittografia Amazon Aurora per proteggere i cluster di database e gli snapshot a riposo. La crittografia Amazon Aurora utilizza l'algoritmo di crittografia AES-256 standard del settore per crittografare i dati sul server che ospita il cluster di database. Per ulteriori informazioni, consulta Crittografia delle risorse Amazon Aurora.
- Utilizzare le funzionalità di sicurezza del motore di database per controllare chi può accedere ai database su un cluster di database. Queste funzionalità funzionano come se il database si trovasse sulla rete locale.

  Per informazioni sulla sicurezza con Aurora MySQL, consulta Sicurezza con Amazon Aurora MySQL. Per informazioni sulla sicurezza con Aurora PostgreSQL, consulta Sicurezza con Amazon Aurora PostgreSQL.

Aurora fa parte del servizio di database gestito Amazon Relational Database Service (Amazon RDS). Amazon RDS è un servizio Web che semplifica la configurazione, l'uso e il dimensionamento dei database relazionali nel cloud. Se è la prima volta che utilizzi Amazon RDS, consulta la Guida per l'utente di Amazon RDS.

Aurora include un sottosistema di storage ad alte prestazioni. I relativi motori di database compatibili con MySQL e PostgreSQL sono personalizzati per beneficiare dello storage distribuito e rapido. Aurora inoltre automatizza ed esegue la standardizzazione del clustering e della replica di database, ovvero due aspetti in genere tra i più impegnativi nell'ambito della configurazione e dell'amministrazione dei database.

Sia per Amazon RDS che per Aurora, puoi accedere all'API RDS a livello di codice e puoi utilizzarla per accedere all'API RDS in modo AWS CLI interattivo. Alcune operazioni API RDS e comandi AWS CLI si applicano sia a Amazon RDS che ad Aurora, mentre altri si applicano a uno o all'altro. Per informazioni sulle operazioni API RDS, consulta Documentazione di riferimento dell'API Amazon RDS. Per ulteriori informazioni su AWS CLI, consulta il AWS Command Line Interface riferimento per Amazon RDS.

> **Note**
>
> Devi configurare la sicurezza solo per i tuoi casi d'uso. Non devi configurare l'accesso di sicurezza per i processi gestiti da Amazon Aurora. Sono inclusi la creazione di backup, il failover automatico e altri processi.

Per ulteriori informazioni sulla gestione dell'accesso alle risorse Amazon Aurora e ai database su un cluster di database, consulta i seguenti argomenti.

Argomenti

- Autenticazione del database con Amazon Aurora
- Gestione delle password con Amazon Aurora e Gestione dei segreti AWS
- Protezione dei dati in Amazon RDS
- Gestione accessi e identità per Amazon Aurora
- Registrazione e monitoraggio in Amazon Aurora
- Convalida della conformità per Amazon Aurora
- Resilienza in Amazon Aurora
- Sicurezza dell'infrastruttura in Amazon Aurora
- API Amazon RDS ed endpoint VPC dell'interfaccia (AWS PrivateLink)
- Best practice relative alla sicurezza di Amazon Aurora
- Controllo dell'accesso con i gruppi di sicurezza
- Privilegi dell'account utente master
- Utilizzo di ruoli collegati ai servizi per Amazon Aurora
- VPC Amazon e Amazon Aurora

Autenticazione del database con Amazon Aurora

Amazon Aurora supporta diversi modi per autenticare gli utenti del database.

L'autenticazione con password è disponibile per impostazione predefinita per tutti i cluster database. Per Aurora MySQL e Aurora PostgreSQL, puoi aggiungere sia l'autenticazione del database IAM che l'autenticazione Kerberos per lo stesso cluster di database.

L'autenticazione con password, Kerberos e del database IAM utilizzano diversi metodi di autenticazione nel database. Pertanto, un utente specifico può accedere a un database utilizzando un solo metodo di autenticazione.

Per PostgreSQL, utilizza solo una delle seguenti impostazioni di ruolo per un utente di un database specifico:

- Per utilizzare l'autenticazione del database IAM, assegna all'utente il ruolo `rds_iam`.
- Per utilizzare l'autenticazione Kerberos, assegna all'utente il ruolo `rds_ad`.
- Per utilizzare l'autenticazione con password, non assegnare i ruoli `rds_iam` o `rds_ad`.

Non assegnare entrambi i ruoli `rds_iam` e `rds_ad` a un utente di un database PostgreSQL direttamente o indirettamente mediante l'accesso di concessione nidificato. Se all'utente master, viene aggiunto il ruolo `rds_iam`, l'autenticazione IAM ha la precedenza sull'autenticazione con password, quindi l'utente master dovrà accedere come utente IAM.

> **Important**
>
> Si consiglia di non utilizzare l'utente master direttamente nelle applicazioni. Rispetta piuttosto la best practice di utilizzare un utente del database creato con i privilegi minimi richiesti per l'applicazione.

Argomenti

- Autenticazione password
- Autenticazione del database IAM
- Autenticazione Kerberos

Autenticazione password

Con autenticazione con password il database esegue tutta l'amministrazione degli account utente. È possibile creare utenti con istruzioni SQL come `CREATE USER`, con la clausola appropriata richiesta dal motore DB per specificare le password. Ad esempio, in MySQL l'istruzione è `CREATE USER` *`nome`* `IDENTIFIED BY` *`password`*, mentre in PostgreSQL, l'istruzione è `CREATE USER` *`nome`* `WITH PASSWORD` *`password`*.

Con l'autenticazione con password, il database controlla e autentica gli account utente. Se un motore DB dispone di potenti funzionalità di gestione delle password, può migliorare la sicurezza. L'autenticazione del database potrebbe essere più semplice da amministrare utilizzando l'autenticazione con password quando si dispone di comunità di utenti di piccole dimensioni. Poiché in questo caso vengono generate password di testo non crittografato, l'integrazione con Gestione dei segreti AWS può migliorare la sicurezza.

Per informazioni sull'utilizzo di Secrets Manager con Amazon Aurora, consulta Creazione di un segreto di base e Rotazione dei segreti per database Amazon RDS supportati nella Guida per l'utente di Gestione dei segreti AWS. Per informazioni sul recupero a livello di programmazione dei segreti nelle applicazioni personalizzate, consulta Recupero del valore segreto nella Guida per l'utente di Gestione dei segreti AWS.

Autenticazione del database IAM

Puoi eseguire l'autenticazione nel cluster di database tramite l'autenticazione del database AWS Identity and Access Management (IAM). Con questo metodo di autenticazione, non devi utilizzare una password quando esegui la connessione al cluster di database. Utilizzi invece un token di autenticazione.

Per ulteriori informazioni sull'autenticazione del database IAM, incluse le informazioni sulla disponibilità per specifici motori di database, consulta Autenticazione del database IAM .

Autenticazione Kerberos

Amazon Aurora supporta l'autenticazione esterna degli utenti dei database con Kerberos e Microsoft Active Directory. Kerberos è un protocollo di autenticazione di rete che utilizza ticket e crittografia a chiave simmetrica eliminando la necessità di scambiare password sulla rete. È stato integrato in Microsoft Active Directory ed è progettato per autenticare gli utenti su risorse di rete, ad esempio i database.

Il supporto Amazon Aurora per Kerberos e Active Directory offre i vantaggi del Single Sign-On e dell'autenticazione centralizzata degli utenti dei database. Puoi mantenere le tue credenziali utente in Active Directory. Active Directory fornisce una posizione centralizzata per archiviare e gestire le credenziali per più cluster di database.

Per utilizzare le credenziali di Active Directory autogestito, devi configurare una relazione di trust verso il Directory Service per Microsoft Active Directory a cui è collegato il cluster di database.

Aurora PostgreSQL e Aurora MySQL supportano relazioni di trust tra foreste unidirezionali e bidirezionali con autenticazione a livello di foresta o autenticazione selettiva.

In alcuni scenari, è possibile configurare l'autenticazione Kerberos tramite una relazione di trust esterna. Ciò richiede che Active Directory autogestito disponga di impostazioni aggiuntive. Questo include, ad esempio, Kerberos Forest Search Order.

Aurora supporta l'autenticazione Kerberos per i cluster di database Aurora MySQL e Aurora PostgreSQL. Per ulteriori informazioni, consulta Utilizzo dell'autenticazione Kerberos per Aurora MySQL e Utilizzo di Autenticazione Kerberos con Aurora PostgreSQL.

Gestione delle password con Amazon Aurora e Gestione dei segreti AWS

Amazon Aurora si integra con Secrets Manager per gestire le password degli utenti master per .

Argomenti

- Disponibilità di regioni e versioni
- Limitazioni per l'integrazione di Secrets Manager con Amazon Aurora
- Panoramica della gestione delle password degli utenti principali con Gestione dei segreti AWS
- Vantaggi della gestione delle password degli utenti master con Secrets Manager
- Autorizzazioni necessarie per l'integrazione di Secrets Manager
- Applicazione della gestione della password dell'utente principale in Gestione dei segreti AWS
- Gestione della password dell'utente master per un cluster database con Secrets Manager
- Rotazione del segreto della password dell'utente master per un cluster database
- Visualizzazione dei dettagli di un segreto per un cluster database

Disponibilità di regioni e versioni

Il supporto varia a seconda delle versioni specifiche di ciascun motore di database e a seconda delle Regioni AWS. Per ulteriori informazioni sulla disponibilità di versioni e regioni con l'integrazione di Secrets Manager con Amazon Aurora, consulta Regioni e motori di database Aurora supportati per l'integrazione di Secrets Manager.

Limitazioni per l'integrazione di Secrets Manager con Amazon Aurora

La gestione delle password degli utenti master con Secrets Manager non è supportata per le seguenti funzionalità:

- Implementazioni Amazon RDS Blue/Green
- Cluster database che fanno parte di un database globale Aurora
- Cluster di database Aurora Serverless v1
- Repliche di lettura tra regioni diverse
- Replica esterna di log binari

Panoramica della gestione delle password degli utenti principali con Gestione dei segreti AWS

ConGestione dei segreti AWS, puoi sostituire le credenziali codificate nel codice, incluse le password del database, con una chiamata API a Secrets Manager per recuperare il segreto a livello di codice. Per ulteriori informazioni su Secrets Manager, consultare la Guida per l'utente di Gestione dei segreti AWS.

Quando memorizzi i segreti del database in Secrets Manager, ti vengono Account AWS addebitati dei costi. Per informazioni sui prezzi, consulta Gestione dei segreti AWSPricing.

Puoi specificare che Aurora gestisca la password dell'utente master in Secrets Manager per un database Amazon Aurora quando esegui una delle seguenti operazioni:

- Creazione di un cluster di database
- Modificare un cluster di database
- Ripristino di un cluster di database da Amazon S3 (solo Aurora MySQL)

Quando specifichi che Aurora gestisce la password dell'utente master in Secrets Manager, Aurora genera la password e la memorizza in Secrets Manager. Puoi interagire direttamente con il segreto per recuperare le credenziali dell'utente master. Puoi anche specificare una chiave gestita dal cliente per crittografare il segreto o utilizzare la chiave KMS fornita da Secrets Manager.

Aurora gestisce le impostazioni del segreto e lo ruota ogni sette giorni per impostazione predefinita. È possibile modificare alcune impostazioni, ad esempio il programma di rotazione. Se si elimina un cluster database che gestisce un segreto in Secrets Manager, vengono eliminati anche il segreto e i metadati associati.

Per connetterti a con le credenziali in un segreto, puoi recuperare il segreto da Secrets Manager. Per ulteriori informazioni, consulta Recupera segreti da Gestione dei segreti AWS e Connettiti a un database SQL con credenziali in un Gestione dei segreti AWS segreto nella Guida per l'Gestione dei segreti AWSutente.

Vantaggi della gestione delle password degli utenti master con Secrets Manager

La gestione delle password degli utenti master Aurora con Secrets Manager offre i seguenti vantaggi:

- Aurora genera automaticamente le credenziali del database.
- Aurora archivia e gestisce automaticamente le credenziali del database in. Gestione dei segreti AWS
- Aurora ruota regolarmente le credenziali del database, senza richiedere modifiche all'applicazione.
- Secrets Manager protegge le credenziali del database dall'accesso umano e dalla visualizzazione in testo normale.
- Secrets Manager consente il recupero delle credenziali del database nei segreti per le connessioni al database.
- Secrets Manager consente un controllo dettagliato dell'accesso alle credenziali del database nei segreti utilizzando IAM.
- Facoltativamente, puoi separare la crittografia del database dalla crittografia delle credenziali con chiavi KMS diverse.
- Puoi eliminare la gestione manuale e la rotazione delle credenziali del database.
- Puoi monitorare facilmente le credenziali del database con AWS CloudTrail Amazon CloudWatch.

Per ulteriori informazioni sui vantaggi di Secrets Manager, consulta la Guida per l'utente di Gestione dei segreti AWS.

Autorizzazioni necessarie per l'integrazione di Secrets Manager

Gli utenti devono disporre delle autorizzazioni necessarie per eseguire le operazioni relative all'integrazione di Secrets Manager. Puoi creare le policy IAM che concedono l'autorizzazione per eseguire operazioni API specifiche sulle risorse indicate necessarie. Puoi quindi collegare tali policy ai ruoli o ai set di autorizzazioni IAM che richiedono le autorizzazioni. Per ulteriori informazioni, consulta Gestione accessi e identità per Amazon Aurora.

Per le operazioni di creazione, modifica o ripristino, l'utente che specifica che Aurora gestisce la password dell'utente master in Secrets Manager deve disporre delle autorizzazioni per eseguire le seguenti operazioni:

- `kms:DescribeKey`
- `secretsmanager:CreateSecret`
- `secretsmanager:TagResource`

L’autorizzazione `kms:DescribeKey` è necessaria per accedere alla chiave gestita dal cliente per `MasterUserSecretKmsKeyId` e per descrivere `aws/secretsmanager`.

Per le operazioni di creazione, modifica o ripristino, l'utente che specifica la chiave gestita dal cliente per crittografare il segreto in Secrets Manager deve disporre delle autorizzazioni per eseguire le seguenti operazioni:

- `kms:Decrypt`
- `kms:GenerateDataKey`
- `kms:CreateGrant`

Per le operazioni di modifica, l'utente che ruota la password dell'utente master in Secrets Manager deve disporre delle autorizzazioni per eseguire la seguente operazione:

- `secretsmanager:RotateSecret`

Applicazione della gestione della password dell'utente principale in Gestione dei segreti AWS

È possibile utilizzare le chiavi di condizione IAM per implementare la gestione da parte di Aurora della password dell'utente master in Gestione dei segreti AWS. La seguente policy non consente agli utenti di creare o ripristinare istanze database a meno che la password dell’utente master non sia gestita da Aurora in Secrets Manager.

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Deny",
            "Action": ["rds:CreateDBInstance", "rds:CreateDBCluster",
 "rds:RestoreDBInstanceFromS3", "rds:RestoreDBClusterFromS3"],
            "Resource": "*",
            "Condition": {
                "Bool": {
                    "rds:ManageMasterUserPassword": false
                }
```

```
            }
        }
    ]
}
```

> **Note**
>
> Questa politica applica la gestione delle password al momento della creazione. Gestione dei segreti AWS Tuttavia, puoi comunque disabilitare l'integrazione di Secrets Manager e impostare manualmente una password master modificando il cluster.
> Per evitare questa procedura, includi `rds:ModifyDBInstance`, `rds:ModifyDBCluster` nel blocco operazione della policy. Tieni presente che in tal modo impedisci all'utente di applicare ulteriori modifiche ai cluster esistenti in cui non è abilitata l'integrazione di Secrets Manager.

Per ulteriori informazioni sull'utilizzo delle chiavi di condizione nelle policy IAM, consulta Chiavi di condizione delle policy per Aurora e Policy di esempio: Utilizzo di chiavi di condizione.

Gestione della password dell'utente master per un cluster database con Secrets Manager

È possibile configurare la gestione Aurora della password dell'utente master in Secrets Manager eseguendo le seguenti operazioni:

- Creazione di un cluster database Amazon Aurora
- Modifica di un cluster database Amazon Aurora
- La migrazione di dati da un database MySQL esterno a un cluster database Amazon Aurora MySQL

È possibile utilizzare la console RDSAWS CLI, il o l'API RDS per eseguire queste azioni.

Console

Segui le istruzioni per creare o modificare un cluster database con la console RDS:

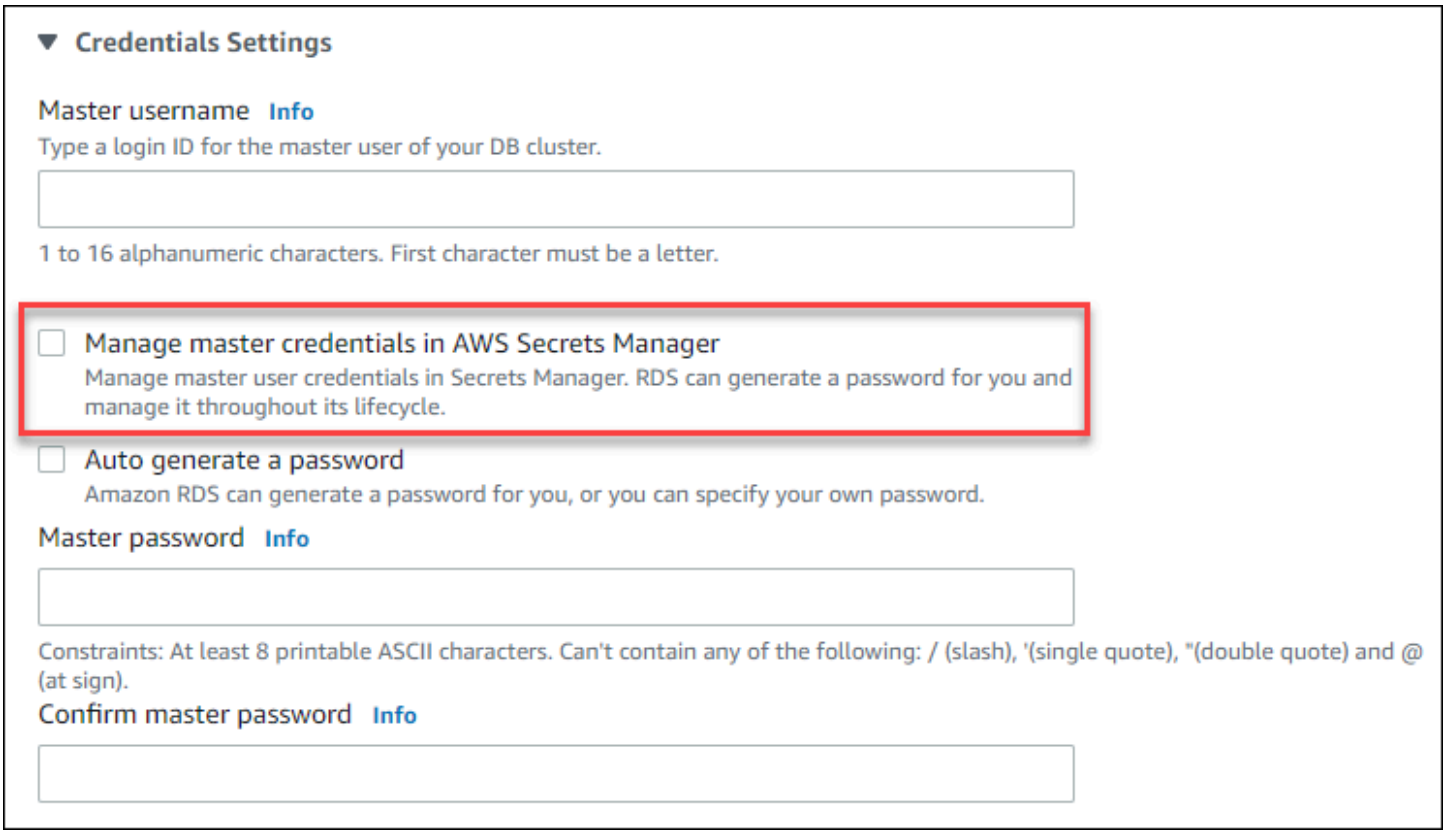
- Creazione di un cluster di database

- Modifica di un'istanza database in un cluster database

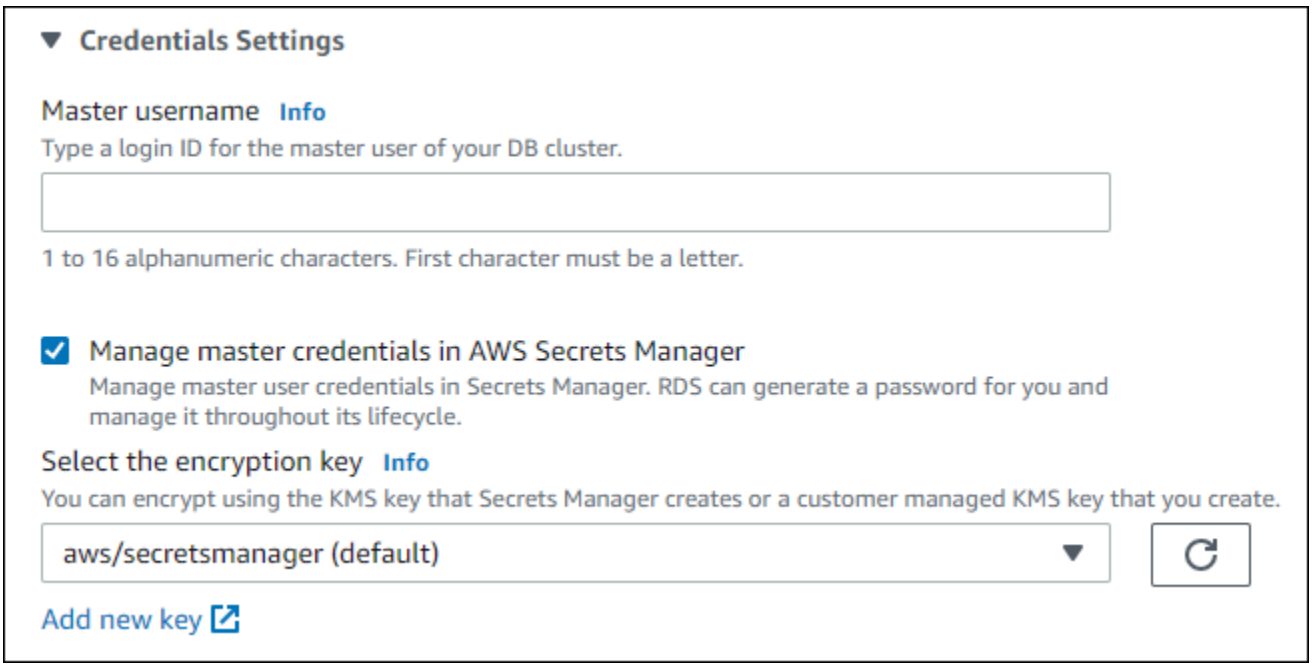
  Nella console RDS puoi modificare qualsiasi istanza database per specificare le impostazioni di gestione della password dell'utente master per l'intero cluster database.
- Ripristino di un cluster di database Amazon Aurora MySQL da un bucket Amazon S3

Quando usi la console RDS per eseguire una di queste operazioni, è possibile specificare che la password dell'utente master sia gestita da Aurora in Secrets Manager. A tale scopo, durante la creazione o il ripristino di un cluster di database, seleziona Gestisci le credenziali master in Gestione dei segreti AWS in Impostazioni credenziali. Quando modifichi un cluster database, seleziona Manage master credentials in Gestione dei segreti AWS (Gestione credenziali master in Gestione dei segreti AWS) in Settings (Impostazioni).

L'immagine seguente è un esempio di impostazione Manage master credentials in Gestione dei segreti AWS (Gestione credenziali master in Gestione dei segreti AWS) durante la creazione o il ripristino di un cluster database.



Quando selezioni questa opzione, Aurora genera la password dell'utente master e la gestisce per tutto il suo ciclo di vita in Secrets Manager.



Puoi scegliere di crittografare il segreto con una chiave KMS fornita da Secrets Manager o con una chiave gestita dal cliente creata da te. Dopo che Aurora gestisce le credenziali del database per un cluster database, non puoi modificare la chiave KMS utilizzata per crittografare il segreto.

Puoi scegliere altre impostazioni per soddisfare le tue esigenze.

Per ulteriori informazioni sulle impostazioni disponibili per la creazione di un cluster database, consulta Impostazioni per cluster di database Aurora. Per ulteriori informazioni sulle impostazioni disponibili per la modifica di un cluster database, consulta Impostazioni per Amazon Aurora.

AWS CLI

Per specificare che Aurora gestisce la password dell'utente master in Secrets Manager, imposta l'opzione `--manage-master-user-password` in uno dei seguenti comandi:

- create-db-cluster
- modify-db-cluster
- restore-db-cluster-from-s3

Quando si specifica l'opzione `--manage-master-user-password` in questi comandi, Aurora genera la password dell'utente master e la gestisce per tutto il suo ciclo di vita in Secrets Manager.

Per crittografare il segreto, è possibile specificare una chiave gestita dal cliente o utilizzare la chiave KMS predefinita fornita da Secrets Manager. Per specificare la chiave gestita dal cliente usa l'opzione

`--master-user-secret-kms-key-id`. L'identificatore della chiave AWS KMS è l'ARN della chiave, l'ID chiave, l'alias ARN o il nome alias per la chiave KMS. Per utilizzare una chiave KMS in un'altra chiaveAccount AWS, specifica la chiave ARN o l'alias ARN. Dopo che Aurora gestisce le credenziali del database per un cluster database, non puoi modificare la chiave KMS utilizzata per crittografare il segreto.

Puoi scegliere altre impostazioni per soddisfare le tue esigenze.

Per ulteriori informazioni sulle impostazioni disponibili per la creazione di un cluster database, consulta Impostazioni per cluster di database Aurora. Per ulteriori informazioni sulle impostazioni disponibili per la modifica di un cluster database, consulta Impostazioni per Amazon Aurora.

Questo esempio crea un cluster database e specifica che Aurora gestisce la password in Secrets Manager. Il segreto viene crittografato utilizzando la chiave KMS fornita da Secrets Manager.

Example

Per Linux, macOS o Unix:

```
aws rds create-db-cluster \
     --db-cluster-identifier sample-cluster \
     --engine aurora-mysql \
     --engine-version 8.0 \
     --master-username admin \
     --manage-master-user-password
```

Per Windows:

```
aws rds create-db-cluster ^
     --db-cluster-identifier sample-cluster ^
     --engine aurora-mysql ^
     --engine-version 8.0 ^
     --master-username admin ^
     --manage-master-user-password
```

API RDS

Per specificare che Aurora gestisce la password dell'utente master in Secrets Manager, imposta il parametro `ManageMasterUserPassword` su `true` in una delle seguenti operazioni:

- CreaDBCluster

- ModificaDBCluster
- Ripristina da S3 DBCluster

Quando imposti il parametro `ManageMasterUserPassword` su `true` in una di queste operazioni, Aurora genera la password dell'utente master e la gestisce per tutto il suo ciclo di vita in Secrets Manager.

Per crittografare il segreto, è possibile specificare una chiave gestita dal cliente o utilizzare la chiave KMS predefinita fornita da Secrets Manager. Per specificare la chiave gestita dal cliente usa il parametro `MasterUserSecretKmsKeyId`. L'identificatore della chiave AWS KMS è l'ARN della chiave, l'ID chiave, l'alias ARN o il nome alias per la chiave KMS. Per usate una chiave KMS in un Account AWS diverso, specifica l'ARN della chiave o dell'alias. Dopo che Aurora gestisce le credenziali del database per un cluster database, non puoi modificare la chiave KMS utilizzata per crittografare il segreto.

Rotazione del segreto della password dell'utente master per un cluster database

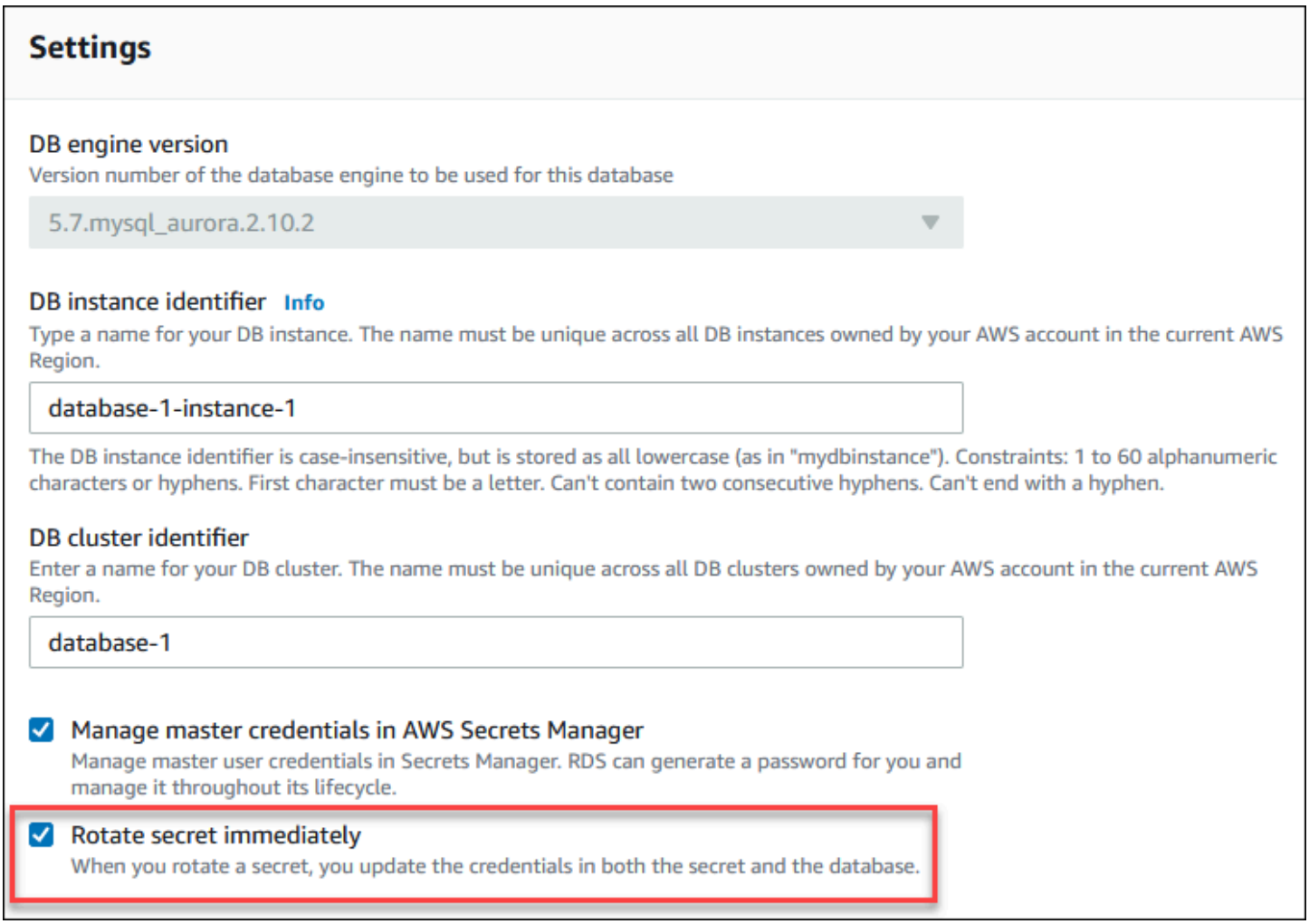
Quando Aurora ruota il segreto della password di un utente master, Secrets Manager genera una nuova versione del segreto esistente. La nuova versione del segreto contiene la nuova password dell'utente master. Amazon Aurora modifica la password dell'utente master per il cluster database in modo che corrisponda alla password per la nuova versione del segreto.

Puoi ruotare un segreto immediatamente invece di aspettare la rotazione programmata. Per ruotare il segreto della password dell'utente master in Secrets Manager, modifica il cluster database . Per informazioni sulla modifica di un cluster database, consulta Modifica di un cluster database Amazon Aurora.

È possibile ruotare immediatamente una password utente principale segreta con la console RDSAWS CLI, l'o l'API RDS. La nuova password è sempre lunga 28 caratteri e contiene almeno un carattere maiuscolo e minuscolo, un numero e un carattere di punteggiatura.

Console

Per ruotare il segreto della password dell'utente master utilizzando la console RDS, modifica il cluster database e seleziona Rotate secret immediately (Ruota il segreto immediatamente) in Settings (Impostazioni).



Per modificare un cluster database con la console RDS segui le istruzioni presenti in Modifica del cluster di database tramite la console, la CLI e l'API. È necessario scegliere Apply immediately (Applica immediatamente) nella pagina di conferma.

AWS CLI

Per ruotare la password segreta di un utente principale utilizzando ilAWS CLI, usa il modify-db-clustercomando e specifica l'opzione. `--rotate-master-user-password` È necessario specificare l'opzione `--apply-immediately` quando si ruota la password master.

Questo esempio ruota il segreto della password dell'utente master.

Example

Per Linux, macOS o Unix:

```
aws rds modify-db-cluster \
```

```
    --db-cluster-identifier mydbcluster \
    --rotate-master-user-password \
    --apply-immediately
```

Per Windows:

```
aws rds modify-db-cluster ^
    --db-cluster-identifier mydbcluster ^
    --rotate-master-user-password ^
    --apply-immediately
```

API RDS

È possibile ruotare una password utente principale segreta utilizzando l'DBClusteroperazione Modifica e impostando il `RotateMasterUserPassword` parametro su. `true` È necessario impostare il parametro `ApplyImmediately` su `true` quando si ruota la password master.

Visualizzazione dei dettagli di un segreto per un cluster database

Puoi recuperare i tuoi segreti usando la console (https://console.aws.amazon.com/secretsmanager/) o il comando AWS CLI (get-secret-valueSecrets Manager).

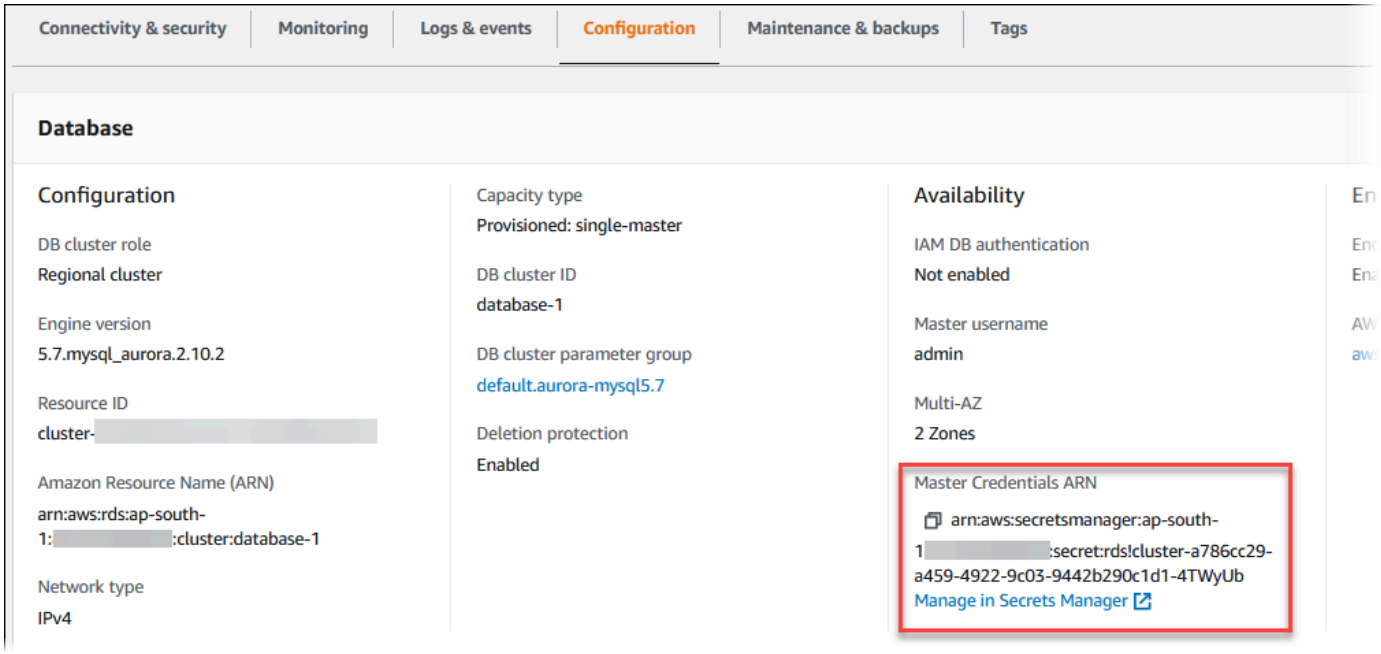
Puoi trovare l'Amazon Resource Name (ARN) di un segreto gestito da Aurora in Secrets Manager con la console RDSAWS CLI, l'o l'API RDS.

Console

Per visualizzare i dettagli di un segreto gestito da Aurora in Secrets Manager

1. Accedi a Console di gestione AWS e apri la console Amazon RDS all'indirizzo https://console.aws.amazon.com/rds/.
2. Nel pannello di navigazione, seleziona Database.
3. Scegli il nome del cluster database per visualizzarne i dettagli.
4. Scegli la scheda Configurazione.

   In Master Credentials ARN (ARN credenziali master), puoi visualizzare l'ARN del segreto.



Puoi selezionare il collegamento Manage in Secrets Manager (Gestisci in Secrets Manager) per visualizzare e gestire il segreto nella console di Secrets Manager.

AWS CLI

È possibile utilizzare il AWS CLI [describe-db-clusters](#)comando RDS per trovare le seguenti informazioni su un segreto gestito da in Secrets Manager:

- `SecretArn`: l'ARN del segreto
- `SecretStatus`: lo stato del segreto

  I valori possibili per lo stato sono:

  - `creating`: il segreto è in fase di creazione.
  - `active`: il segreto è disponibile per l'uso normale e la rotazione.
  - `rotating`: il segreto è in fase di rotazione.
  - `impaired`: il segreto può essere utilizzato per accedere alle credenziali del database, ma non può essere ruotato. Un segreto può avere questo stato se, ad esempio, le autorizzazioni vengono modificate in modo che RDS non può più accedere al segreto o alla chiave KMS del segreto.

    Quando un segreto ha questo stato, puoi correggere la condizione che lo ha causato. Se correggi la condizione che ha causato lo stato, lo stato rimane `impaired` fino alla rotazione

successiva. In alternativa, è possibile modificare il cluster database per disattivare la gestione automatica delle credenziali del database e quindi modificare nuovamente il cluster database per attivare la gestione automatica delle credenziali del database. Per modificare il cluster DB, utilizzare l'`--manage-master-user-password`opzione nel comando. modify-db-cluster

- `KmsKeyId`: l'ARN della chiave KMS utilizzata per crittografare il segreto

Specifica l'opzione `--db-cluster-identifier` per mostrare l'output per un cluster database specifico. Questo esempio mostra l'output di un segreto utilizzato da un cluster database.

Example

```
aws rds describe-db-clusters --db-cluster-identifier mydbcluster
```

L'esempio seguente mostra l'output di un segreto:

```
"MasterUserSecret": {
                "SecretArn": "arn:aws:secretsmanager:eu-west-1:123456789012:secret:rds!
cluster-033d7456-2c96-450d-9d48-f5de3025e51c-xmJRDx",
                "SecretStatus": "active",
                "KmsKeyId": "arn:aws:kms:eu-
west-1:123456789012:key/0987dcba-09fe-87dc-65ba-ab0987654321"
            }
```

Quando si dispone dell'ARN segreto, è possibile visualizzare i dettagli sul segreto utilizzando il comando get-secret-valueSecrets Manager CLI.

Questo esempio mostra i dettagli del segreto nell'output di esempio precedente.

Example

Per Linux, macOS o Unix:

```
aws secretsmanager get-secret-value \
    --secret-id 'arn:aws:secretsmanager:eu-west-1:123456789012:secret:rds!
cluster-033d7456-2c96-450d-9d48-f5de3025e51c-xmJRDx'
```

Per Windows:

```
aws secretsmanager get-secret-value ^
```

```
    --secret-id 'arn:aws:secretsmanager:eu-west-1:123456789012:secret:rds!
cluster-033d7456-2c96-450d-9d48-f5de3025e51c-xmJRDx'
```

API RDS

È possibile visualizzare l'ARN, lo stato e la chiave KMS per un segreto gestito da RDS in Secrets Manager utilizzando l'operazione DBClusters Descrivi RDS e `DBClusterIdentifier` impostando il parametro su un identificatore di cluster DB. I dettagli del segreto sono inclusi nell'output.

Quando si dispone dell'ARN segreto, è possibile visualizzare i dettagli sul segreto utilizzando l'operazione GetSecretValueSecrets Manager.

Protezione dei dati in Amazon RDS

Il modello di responsabilità condivisa di AWS si applica alla protezione dei dati in Amazon Relational Database Service. Come descritto in questo modello, AWS è responsabile della protezione dell'infrastruttura globale che esegue tutto l'Cloud AWS. L'utente è responsabile del controllo dei contenuti ospitati su questa infrastruttura. L'utente è inoltre responsabile della configurazione della protezione e delle attività di gestione per i Servizi AWS utilizzati. Per ulteriori informazioni sulla privacy dei dati, vedi le Domande frequenti sulla privacy dei dati. Per informazioni sulla protezione dei dati in Europa, consulta il post del blog relativo al Modello di responsabilità condivisa AWSe GDPR nel Blog sulla sicurezza AWS.

Per garantire la protezione dei dati, ti suggeriamo di proteggere le credenziali Account AWSe di configurare i singoli utenti con AWS IAM Identity Centero AWS Identity and Access Management(IAM). In tal modo, a ogni utente verranno assegnate solo le autorizzazioni necessarie per svolgere i suoi compiti. Ti suggeriamo, inoltre, di proteggere i dati nei seguenti modi:

- Utilizza l'autenticazione a più fattori (MFA) con ogni account.
- Utilizza SSL/TLS per comunicare con le risorse AWS. È richiesto TLS 1.2 ed è consigliato TLS 1.3.
- Configura l'API e la registrazione delle attività degli utenti con AWS CloudTrail. Per informazioni sull'utilizzo dei percorsi CloudTrail per acquisire le attività AWS, consulta Utilizzo dei percorsi CloudTrail nella Guida per l'utente di AWS CloudTrail.
- Utilizza le soluzioni di crittografia AWS, insieme a tutti i controlli di sicurezza di default all'interno dei Servizi AWS.
- Utilizza i servizi di sicurezza gestiti avanzati, come Amazon Macie, che aiutano a individuare e proteggere i dati sensibili archiviati in Amazon S3.

- Se necessiti di moduli crittografici convalidati FIPS 140-3 quando accedi ad AWS attraverso un'interfaccia a riga di comando o un'API, utilizza un endpoint FIPS. Per ulteriori informazioni sugli endpoint FIPS disponibili, consulta il Federal Information Processing Standard (FIPS) 140-3.

Ti consigliamo di non inserire mai informazioni riservate o sensibili, ad esempio gli indirizzi e-mail dei clienti, nei tag o nei campi di testo in formato libero, ad esempio nel campo Nome. Questo suggerimento è relativo all'utilizzo di Amazon RDS o altri Servizi AWS tramite la console, l'API, AWS CLI o gli SDK AWS. I dati inseriti nei tag o nei campi di testo in formato libero utilizzati per i nomi possono essere utilizzati per i la fatturazione o i log di diagnostica. Quando fornisci un URL a un server esterno, ti suggeriamo vivamente di non includere informazioni sulle credenziali nell'URL per convalidare la tua richiesta al server.

Argomenti

- Protezione dei dati tramite crittografia
- Riservatezza del traffico Internet

Protezione dei dati tramite crittografia

Puoi abilitare la crittografia per le risorse di database. Puoi anche crittografare le connessioni ai cluster di database.

Argomenti

- Crittografia delle risorse Amazon Aurora
- Gestione di AWS KMS key
- Utilizzo SSL/TLS per crittografare una connessione a un'
- Rotazione del certificato SSL/TLS

Crittografia delle risorse Amazon Aurora

Amazon Aurora può crittografare i cluster database di Amazon Aurora. I dati che vengono crittografati quando sono inattivi includono lo storage sottostante per le cluster database, i backup automatici, le repliche di lettura e gli snapshot.

I istanze database Amazon Aurora crittografate utilizzano l'algoritmo di crittografia AES-256 standard del settore per crittografare i dati sul server che ospita i istanze database Amazon Aurora. Una volta crittografati i dati, Amazon Aurora gestisce l'autenticazione dell'accesso e la decrittografia dei dati

in modo trasparente con un impatto minimo sulle prestazioni. Non è quindi necessario modificare le applicazioni client di database per utilizzare la crittografia.

> **Note**
>
> Per cluster di database crittografati e non crittografati, i dati in transito tra l'origine e le repliche di lettura vengono crittografati, anche durante la replica tra regioni AWS.

Argomenti

- Panoramica della crittografia delle risorse Amazon Aurora
- Creazione di un cluster di database Amazon Aurora
- Determinare se la crittografia è attivata per un cluster database
- Disponibilità della crittografia Amazon Aurora
- Crittografia in transito
- Limiti relativi a istanze database crittografate Amazon Aurora

Panoramica della crittografia delle risorse Amazon Aurora

I cluster database Amazon Aurora crittografati offrono un livello aggiuntivo di sicurezza dei dati proteggendoli dagli accessi non autorizzati nello storage sottostante. Puoi utilizzare la crittografia Amazon Aurora per aumentare la protezione dei dati delle applicazioni che vengono distribuite nel cloud e per soddisfare i requisiti di conformità per la crittografia dei dati inattivi. Per un cluster database Amazon Aurora crittografato, vengono crittografati tutti i log, i backup e le snapshot. Per ulteriori informazioni sulla disponibilità e sui limiti della crittografia, consulta Disponibilità della crittografia Amazon Aurora e Limiti relativi a istanze database crittografate Amazon Aurora.

Amazon Aurora utilizza una chiave AWS Key Management Service per crittografare queste risorse. AWS KMS combina hardware e software sicuri e a disponibilità elevata per offrire un sistema di gestione delle chiavi a misura di cloud. Puoi utilizzare una Chiave gestita da AWS oppure creare le chiavi gestite dal cliente.

Quando crei un cluster di database crittografato, puoi scegliere una chiave gestita dal cliente o la Chiave gestita da AWS per Amazon Aurora per la crittografia del cluster di database. Se non specifichi l'identificatore della chiave per una chiave gestita dal cliente, Amazon Aurora utilizza la Chiave gestita da AWS per il nuovo cluster di database. Amazon Aurora crea una Chiave gestita da

AWS per Amazon Aurora per il tuo account AWS. Il tuo account AWS dispone di una Chiave gestita da AWS diversa per Amazon Aurora per ogni Regione AWS.

Per gestire le chiavi gestite dal cliente utilizzate per crittografare e decrittografare le risorse di Amazon Aurora, utilizzare AWS Key Management Service (AWS KMS).

AWS KMS consente di creare chiavi gestite dal cliente e definire le policy per controllare l'utilizzo di queste chiavi gestite dal cliente. AWS KMS supporta CloudTrail, in modo da poter controllare l'uso della chiave KMS per verificare che le chiavi gestite dal cliente vengano utilizzate in modo appropriato. Puoi utilizzare le chiavi gestite dal cliente con Amazon Aurora e i servizi AWS supportati come Simple Storage Service (Amazon S3), Amazon EBS e Amazon Redshift. Per un elenco dei servizi integrati con AWS KMS, consulta Integrazione dei servizi AWS. Alcune considerazioni sull'utilizzo delle chiavi KMS:

- Una volta creata un'istanza database crittografata, non potrai più modificare la chiave KMS utilizzata da quell'istanza database. Pertanto, assicurati di determinare i requisiti della chiave KMS prima di creare la tua istanza database crittografata.

  Se è necessario modificare la chiave di crittografia per il cluster di database, creare uno snapshot manuale del cluster e abilitare la crittografia durante la copia dello snapshot. Per ulteriori informazioni, consulta l'articolo re:Post Knowledge.
- Se copi uno snapshot crittografata, puoi utilizzare una chiave KMS diversa per crittografare la snapshot di destinazione rispetto a quella utilizzata per crittografare la snapshot di origine.
- Non puoi condividere uno snapshot che è stata crittografata con la Chiave gestita da AWS dell'account AWS che ha condiviso la snapshot.
- Ogni istanza database nel cluster database viene crittografata utilizzando la stessa chiave KMS del cluster di database.
- Puoi anche crittografare una replica di lettura di un cluster crittografato con Amazon Aurora.

> **Important**
>
> In alcuni casi, Amazon Aurora può perdere l'accesso alla chiave KMS per un cluster di database quando si disabilita la chiave KMS. In questi casi, il cluster di database crittografato entra nello stato `inaccessible-encryption-credentials-recoverable`. Il cluster di database rimane in questo stato per sette giorni, durante i quali l'istanza viene arrestata. Le chiamate API effettuate al cluster di database durante questo periodo potrebbero non avere esito positivo. Per ripristinare il cluster di database, abilitare la chiave KMS e riavviare questo

cluster di database. Abilitare la chiave KMS dalla Console di gestione AWS, da AWS CLI o dall'API RDS. Riavvia il cluster di database tramite il comando AWS CLI start-db-cluster o la Console di gestione AWS.
Lo stato `inaccessible-encryption-credentials-recoverable` si applica solo ai cluster di database che possono essere arrestati. Questo stato ripristinabile non è applicabile alle istanze che non possono essere arrestate, come i cluster con repliche di lettura tra Regioni. Per ulteriori informazioni, consulta the section called "Restrizioni".
Se il cluster di database non viene ripristinato entro sette giorni, passa allo stato terminale `inaccessible-encryption-credentials`. In questo stato, il cluster di database non è più utilizzabile e sarà possibile ripristinarlo solo da un backup. È consigliabile attivare sempre i backup per i cluster di database crittografati per evitare la perdita di dati crittografati nei database.
Durante la creazione di un cluster di database, Aurora verifica se il principale chiamante ha accesso alla chiave KMS e genera una concessione dalla chiave KMS che utilizza per l'intera durata del cluster di database. La revoca dell'accesso del principale chiamante alla chiave KMS non influisce su un database in esecuzione. Quando si utilizzano le chiavi KMS in scenari che coinvolgono più account, ad esempio per copiare uno snapshot in un altro account, la chiave KMS deve essere condivisa con l'altro account. Se si crea un cluster di database dallo snapshot senza specificare una chiave KMS diversa, il nuovo cluster utilizza la chiave KMS dell'account di origine. La revoca dell'accesso alla chiave dopo la creazione del cluster di database non influisce sul cluster. Tuttavia, la disabilitazione della chiave influisce su tutti i cluster di database crittografati con tale chiave. Per evitare che ciò accada, specificare una chiave diversa durante l'operazione di copia dello snapshot.

Per ulteriori informazioni sulle chiavi KMS, consulta AWS KMS keys nella Guida per sviluppatori di AWS Key Management Service e Gestione di AWS KMS key.

Creazione di un cluster di database Amazon Aurora

Per crittografare un nuovo cluster di database, scegliere Enable encryption (Abilita crittografia) nella console. Per ulteriori informazioni sulla creazione di un cluster database, consulta Creazione di un cluster database Amazon Aurora.

Se utilizzi il comando create-db-cluster di AWS CLI per creare un cluster database crittografato, imposta il parametro `--storage-encrypted`. Se utilizzi l'operazione API CreateDBCluster, imposta il parametro `StorageEncrypted` su true.

Una volta creato un cluster di database crittografato, non potrai più modificare la chiave KMS utilizzata da quel cluster di database. Assicurati quindi di determinare i requisiti della chiave KMS prima di creare il cluster di database crittografato.

Se utilizzi il comando AWS CLI `create-db-cluster` per creare un cluster database crittografato con una chiave gestita dal cliente, imposta il parametro `--kms-key-id` su qualsiasi identificatore di chiave per la chiave KMS. Se utilizzi la funzionalità `CreateDBInstance` dell'API Amazon RDS, imposta il parametro `KmsKeyId` su un qualsiasi identificatore chiave per la chiave KMS. Per utilizzare una chiave gestita dal cliente in un diverso account AWS, specifica l'ARN della chiave o dell'alias.

Determinare se la crittografia è attivata per un cluster database

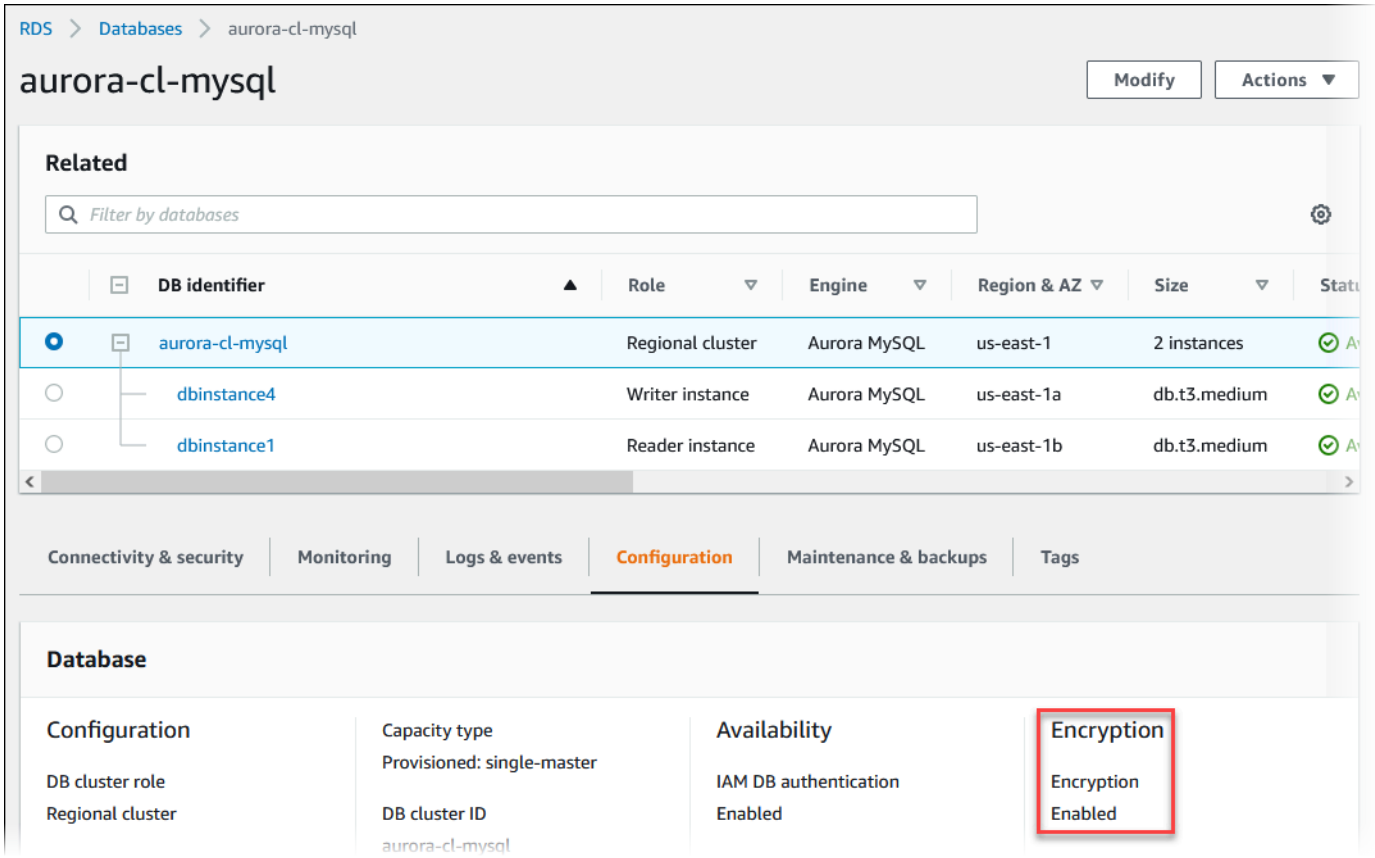
Puoi utilizzare il plugin Console di gestione AWS, AWS CLI o API RDS per determinare se la crittografia a riposo è attivata per un cluster database.

Console

Per determinare se la crittografia a riposo è attivata per un cluster database

1. Accedi alla Console di gestione AWS e apri la console Amazon RDS all'indirizzo https://console.aws.amazon.com/rds/.
2. Nel pannello di navigazione, scegliere Databases (Database).
3. Scegliere il nome del cluster database da controllare per visualizzarne i dettagli.
4. Selezionare la casella Configurazione e controllare il valore Crittografia.

   Mostra Enabled (Abilitato) o Non abilitato.



AWS CLI

Per determinare se la crittografia a riposo è attivata per un cluster database utilizzando il AWS CLI, richiamare il comando describe-db-clusters con la seguente opzione:

- `--db-cluster-identifier`: il nome del cluster di database.

Nell'esempio seguente viene utilizzata una query per restituire `TRUE` o `FALSE` per quanto riguarda la crittografia inattiva per il cluster database `mydb`.

Example

```
aws rds describe-db-clusters --db-cluster-identifier mydb --query "*[].
{StorageEncrypted:StorageEncrypted}" --output text
```

API RDS

Per determinare se è attiva la crittografia dei dati inattivi per un cluster database utilizzando l'API Amazon RDS, richiamare l'operazione DescribeDBClusters con il parametro seguente:

- `DBClusterIdentifier`: il nome del cluster di database.

Disponibilità della crittografia Amazon Aurora

La crittografia Amazon Aurora è attualmente disponibile per tutti i motori di database e i tipi di storage.

> **Note**
>
> La crittografia Amazon Aurora non è disponibile per la classe di istanza database db.t2.micro.

Crittografia in transito

Crittografia a livello fisico

Tutti i dati che scorrono attraverso le Regioni AWS sulla rete globale AWS vengono automaticamente crittografati a livello fisico prima che escano dalle strutture protette di AWS. Tutto il traffico tra le AZ è crittografato. Ulteriori livelli di crittografia, inclusi quelli elencati in questa sezione, possono fornire ulteriore protezione.

Crittografia fornita da peering di Amazon VPC e peering fra regioni Transit Gateway

Tutto il traffico tra regioni che utilizza il peering Amazon VPC e Transit Gateway viene automaticamente crittografato in massa quando esce da una regione. Un ulteriore livello di crittografia viene fornito automaticamente al livello fisico per tutto il traffico prima che esca dalle strutture AWS protette.

Crittografia tra istanze

AWS fornisce una connettività privata e sicura tra le istanze database di tutti i tipi. Inoltre, alcuni tipi di istanza utilizzano le funzionalità di offload dell'hardware Nitro System sottostante per crittografare automaticamente il traffico in transito tra le istanze. Questa crittografia utilizza algoritmi AEAD (Authenticated Encryption with Associated Data), con crittografia a 256 bit. Non vi è alcun impatto sulle prestazioni della rete. Per supportare questa crittografia aggiuntiva del traffico in transito tra istanze, è necessario soddisfare i seguenti requisiti:

- Le istanze utilizzano i seguenti tipi di istanza:
  - Uso generico: M6i, M6id, M6in, M6idn, M7g
  - Ottimizzate per la memoria: R6i, R6id, R6in, R6idn, R7g, X2idn, X2iedn, X2iezn

- Le istanze si trovano nella stessa Regione AWS.
- Le istanze si trovano nello stesso VPC o VPC con peering e il traffico non passa attraverso un dispositivo di rete virtuale, ad esempio un load balancer (load balancer) o un Transit Gateway.

Limiti relativi a istanze database crittografate Amazon Aurora

Esistono le seguenti limitazioni per le istanze database crittografate Amazon Aurora:

- Non puoi disattivare la crittografia di un cluster database crittografato.
- Non puoi creare uno snapshot crittografata per un cluster database non crittografato.
- Una snapshot di un cluster database crittografato deve essere crittografata utilizzando la stessa chiave KMS del cluster database.
- Non è possibile convertire un cluster database non crittografato in uno crittografato. Tuttavia, puoi ripristinare uno snapshot di un cluster database non crittografato in un cluster database Aurora crittografato. Per eseguire questa operazione, specifica una chiave KMS quando ripristini dalla snapshot non crittografata.
- Non è possibile creare una replica Aurora crittografata da un cluster database Aurora non crittografato. Non è possibile creare una replica Aurora non crittografata da un cluster database Aurora crittografato.
- Per copiare uno snapshot crittografata da una regione AWS a un'altra, devi specificare la chiave KMS nella Regione AWS di destinazione. Questo perché le chiave KMS sono specifiche della Regione AWS in cui vengono create.

  La snapshot di origine resta crittografata nel processo di copia. Amazon Aurora utilizza la crittografia envelope per proteggere i dati durante il processo di copia. Per ulteriori informazioni sulla crittografia envelope, consulta Crittografia envelope nella Guida per sviluppatori di AWS Key Management Service.
- Non è possibile decrittografare un cluster database crittografato. Tuttavia, puoi esportare i dati da un cluster database crittografato e importarli in un cluster database non crittografato.

Gestione di AWS KMS key

Amazon Aurora si integra automaticamente con AWS Key Management Service (AWS KMS) per la gestione delle chiavi. Amazon Aurora utilizza la crittografia a busta. Per ulteriori informazioni sulla crittografia envelope, consulta Crittografia envelope nella Guida per sviluppatori di AWS Key Management Service.

È possibile utilizzare due tipi di chiavi AWS KMS per crittografare i cluster di database.

- Per avere il pieno controllo su una chiave KMS, devi creare una chiave gestita dal cliente. Per ulteriori informazioni sulle chiavi gestite dal cliente, consulta Chiavi gestite dal cliente nella Guida per gli sviluppatori di AWS Key Management Service.
- Le Chiavi gestite da AWS sono chiavi KMS nel tuo account create, gestite e utilizzate a tuo nome da un servizio AWS che si integra con AWS KMS. Per impostazione predefinita, la Chiave gestita da AWS RDS (`aws/rds`) viene utilizzata per la crittografia. Non è possibile gestire, ruotare o eliminare la Chiave gestita da AWS RDS. Per ulteriori informazioni su Chiavi gestite da AWS, consulta Chiavi gestite da AWS in AWS Key Management Service Guida per gli sviluppatori.

Per gestire le chiavi KMS utilizzate per le i cluster di database crittografati di Amazon Aurora, utilizzare AWS Key Management Service (AWS KMS) nella console AWS KMS, la AWS CLI o l'API AWS KMS. Puoi visualizzare i log di controllo di ogni operazione eseguita con una chiave gestita da AWS o dal cliente utilizzando AWS CloudTrail. Per ulteriori informazioni sulla rotazione delle chiavi, consulta Rotazione delle chiavi AWS KMS.

Autorizzazione dell'uso di una chiave gestita dal cliente

Quando Aurora utilizza una chiave gestita dal cliente in operazioni che coinvolgono la crittografia, funziona per conto dell'utente che crea o modifica la risorsa Aurora.

Per creare una risorsa Aurora utilizzando una chiave gestita dal cliente, un utente deve disporre delle autorizzazioni per chiamare le seguenti operazioni su tale chiave:

- `kms:CreateGrant`
- `kms:DescribeKey`

Puoi specificare queste autorizzazioni necessarie in una policy chiave o in una policy IAM se la policy chiave lo consente.

> **Important**
>
> Quando si utilizzano istruzioni di rifiuto esplicite per tutte le risorse (*) nelle policy della chiave AWS KMS con servizi gestiti come Amazon RDS, è necessario specificare una condizione per consentire l'account proprietario della risorsa. L'operazione potrebbe non riuscire senza questa condizione, anche se la regola di rifiuto include eccezioni per l'utente IAM.

> **Tip**
>
> Per seguire il principio del privilegio minimo, non consentire l'accesso completo a `kms:CreateGrant`. Utilizzare, invece, la chiave di condizione kms:ViaService per consentire all'utente di creare concessioni sulla chiave KMS solo quando la concessione viene creata per conto dell'utente da un servizio AWS.

Esistono diversi modi per rendere la policy IAM più efficace. Ad esempio, se si desidera consentire l'uso della chiave gestita dal cliente solo per le richieste che provengono da Aurora, utilizzare la chiave di condizione kms:ViaService con il valore `rds.<region>.amazonaws.com`. Puoi inoltre usare le chiavi o i valori nel Contesto di crittografia di Amazon RDS come condizione per utilizzare la chiave gestita dal cliente per la crittografia.

Per ulteriori informazioni, consulta Autorizzazione per gli utenti in altri account di utilizzare una chiave KMS nella Guida per gli sviluppatori di AWS Key Management Service e Policy delle chiavi in AWS KMS.

Contesto di crittografia di Amazon RDS

Quando Aurora utilizza la chiave KMS o quando Amazon EBS utilizza la chiave KMS per conto di Aurora, il servizio specifica un contesto di crittografia. Il contesto di crittografia rappresenta dati autenticati supplementari (AAD) utilizzati da AWS KMS per garantire l'integrità dei dati. Quando viene specificato un contesto di crittografia per un'operazione di crittografia, il servizio deve specificare lo stesso contesto di crittografia per l'operazione di decrittografia. In caso contrario, la decrittografia ha esito negativo. Il contesto di crittografia viene scritto nei log AWS CloudTrail per aiutarti a comprendere perché è stata utilizzata una determinata chiave KMS. I log CloudTrail potrebbero contenere molte voci che descrivono l'utilizzo di una chiave KMS, ma il contesto di crittografia in ciascuna voce di log può aiutarti a determinare il motivo per quel particolare uso.

Come minimo, Aurora utilizza sempre l'ID del cluster di database per il contesto di crittografia, come nel seguente esempio in formato JSON:

```
{ "aws:rds:dbc-id": "cluster-CQYSMDPBRZ7BPMH7Y3RTDG5QY" }
```

Questo contesto di crittografia consente di identificare il cluster di database per cui è stata utilizzata la chiave KMS.

Quando la chiave KMS viene utilizzata per un cluster di database specifico e un determinato volume Amazon EBS, sia l'ID del cluster di database e l'ID del volume Amazon EBS vengono utilizzati per il contesto di crittografia, come nel seguente esempio in formato JSON:

```
{
  "aws:rds:dbc-id": "cluster-BRG7VYS3SVIFQW7234EJQOM5RQ",
  "aws:ebs:id": "vol-ad8c6542"
}
```

Utilizzo SSL/TLS per crittografare una connessione a un'

Puoi utilizzare Secure Socket Layer (SSL) o Transport Layer Security (TLS) dall'applicazione per crittografare una connessione a un cluster di database che esegue Aurora MySQL o Aurora PostgreSQL.

Le connessioni SSL/TLS forniscono un livello di sicurezza tramite la crittografia dei dati che si spostano tra il client e un cluster. Facoltativamente, la SSL/TLS connessione può eseguire la verifica dell'identità del server convalidando il certificato del server installato nel database. Per richiedere la verifica dell'identità del server, esegui questa procedura generale:

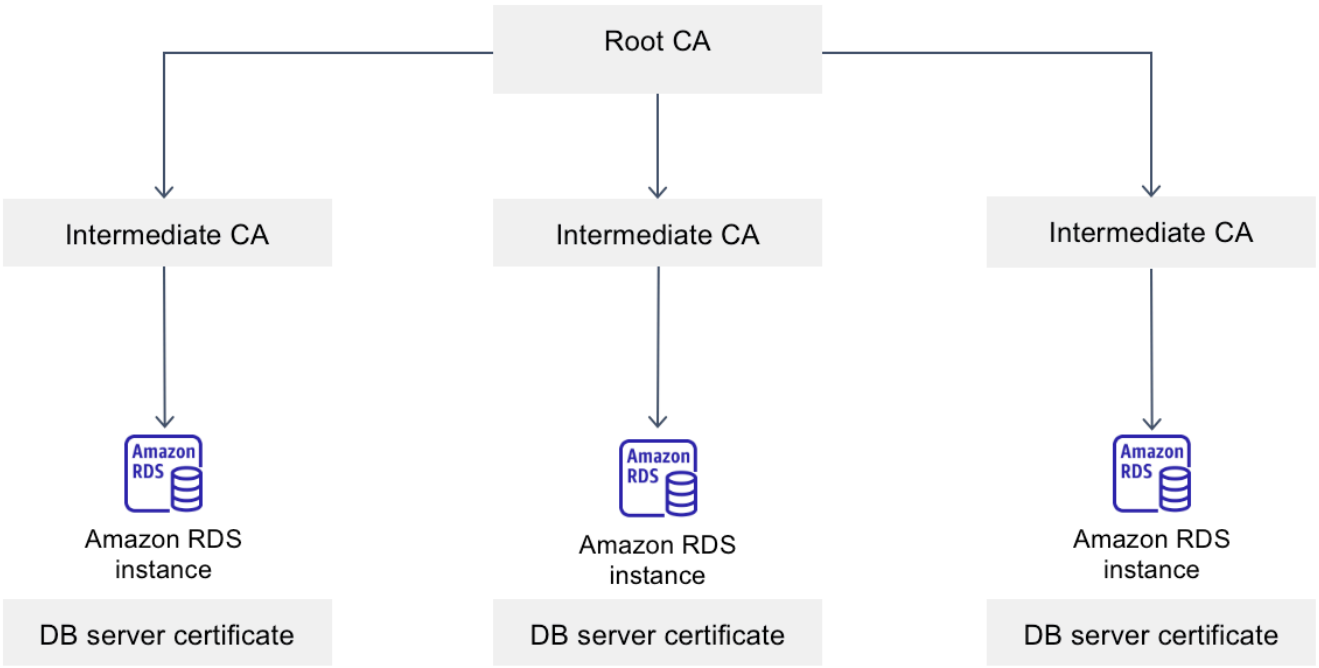
1. Scegli l'autorità di certificazione (CA) che firma il certificato del server di database per il database. Per ulteriori informazioni sulle autorità di certificazione, consulta Autorità di certificazione.
2. Scarica un bundle di certificati da utilizzare quando ti connetti al database. Per scaricare un bundle di certificati, consulta Pacchetti di certificati di Regione AWS.

   > **Note**
   >
   > Tutti i certificati sono disponibili solo per il download tramite connessioni SSL/TLS.

3. Connettiti al database utilizzando il processo del tuo motore DB per l'implementazione delle SSL/TLS connessioni. Ogni motore DB ha il proprio processo di implementazione SSL/TLS. To learn how to implement SSL/TLS per il tuo database, segui il link che corrisponde al tuo motore DB:
   - Sicurezza con Amazon Aurora MySQL
   - Sicurezza con Amazon Aurora PostgreSQL

Autorità di certificazione

L'autorità di certificazione (CA) è il certificato che identifica la CA root della catena di certificati. La CA firma il certificato del server di database, che è il certificato del server installato su ogni istanza database. Il certificato del server di database identifica l'istanza database come server attendibile.



Amazon RDS fornisce quanto segue CAs per firmare il certificato del server DB per un database.

Autorità di certificazione (CA)	Description	Common name (CN) (Nome comune)
rds-ca-rsa2048-g1	Utilizza un'autorità di certificazione con algoritmo a chiave privata RSA 2048 e algoritmo di firma nella maggior parte dei casi. SHA256 Regioni AWS Nel AWS GovCloud (US) Regions, questa CA utilizza un'autorità di certificazione con algoritmo a chiave privata RSA 2048 e algoritmo di firma. SHA384	Amazon RDS *region-id entifier* root CA RSA2 048 G1

Autorità di certificazione (CA)	Description	Common name (CN) (Nome comune)
	supporta la rotazione automatica dei certificati del server.	
rds-ca-rsa4096-g1	Utilizza un'autorità di certificazione con algoritmo a chiave privata RSA 4096 e algoritmo di firma. SHA384 supporta la rotazione automatica dei certificati del server.	Amazon RDS *region-id entifier* root CA RSA4 096 G1
rds-ca-ecc384 g1	Utilizza un'autorità di certificazione con algoritmo a chiave privata ECC 384 e algoritmo di firma. SHA384 supporta la rotazione automatica dei certificati del server.	Amazon RDS *region-id entifier* Root CA ECC384 G1

> **Note**
>
> Se utilizzi il AWS CLI, puoi vedere le validità delle autorità di certificazione sopra elencate utilizzando describe-certificates.

Questi certificati CA sono inclusi nel bundle di certificati regionali e globali. Quando si utilizza la CA rds-ca-rsa 2048-g1, rds-ca-rsa 4096-g1 o rds-ca-ecc 384-g1 con un database, RDS gestisce il certificato del server DB sul database. RDS esegue automaticamente la rotazione del certificato del server di database prima della scadenza.

Impostazione della CA per il database

Puoi impostare la CA per un database quando esegui le seguenti attività:

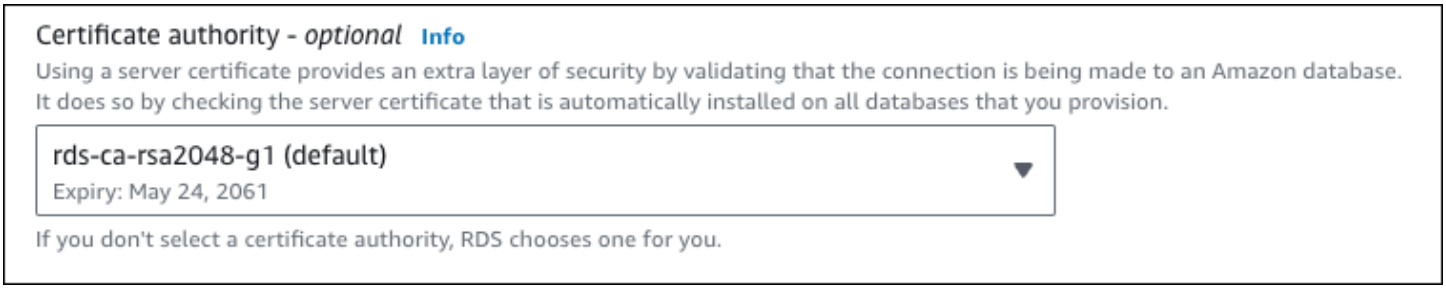
- Crea un cluster Aurora DB: puoi impostare la CA per un'istanza DB in un cluster Aurora quando crei la prima istanza DB nel cluster DB utilizzando l' AWS CLI API o RDS. Attualmente, non puoi impostare la CA per istanze database in un cluster database durante la creazione del cluster

database usando la console RDS. Per le istruzioni, consulta Creazione di un cluster database Amazon Aurora.

- Modifica di un'istanza database: puoi impostare la CA per un'istanza database in un cluster database modificandola. Per le istruzioni, consulta Modifica di un'istanza database in un cluster database.

> **Note**
>
> La CA predefinita è impostata su 2048-g1. rds-ca-rsa È possibile sovrascrivere la CA predefinita per il proprio Account AWS utilizzando il comando modify-certificates.

Le opzioni disponibili CAs dipendono dal motore DB e dalla versione del motore DB. Quando si utilizza il Console di gestione AWS, è possibile scegliere la CA utilizzando l'impostazione dell'autorità di certificazione, come mostrato nell'immagine seguente.



La console mostra solo CAs le versioni disponibili per il motore DB e la versione del motore DB. Se si utilizza il AWS CLI, è possibile impostare la CA per un'istanza DB utilizzando il modify-db-instancecomando create-db-instanceor.

Se utilizzi il AWS CLI, puoi vedere quello disponibile CAs per il tuo account utilizzando il comando describe-certificates. Questo comando mostra nell'output anche la data di scadenza per ogni CA in `ValidTill`. Puoi trovare CAs quelli disponibili per uno specifico motore DB e una versione del motore DB utilizzando il comando. describe-db-engine-versions

L'esempio seguente mostra la CAs versione del motore DB RDS per PostgreSQL disponibile per la versione predefinita.

```
aws rds describe-db-engine-versions --default-only --engine postgres
```

L’output è simile a quello riportato di seguito. I disponibili sono elencati in. CAs `SupportedCACertificateIdentifiers` L'output mostra anche se la versione del motore di database supporta la rotazione del certificato senza riavvio in `SupportsCertificateRotationWithoutRestart`.

```
{
    "DBEngineVersions": [
        {
            "Engine": "postgres",
            "MajorEngineVersion": "13",
            "EngineVersion": "13.4",
            "DBParameterGroupFamily": "postgres13",
            "DBEngineDescription": "PostgreSQL",
            "DBEngineVersionDescription": "PostgreSQL 13.4-R1",
            "ValidUpgradeTarget": [],
            "SupportsLogExportsToCloudwatchLogs": false,
            "SupportsReadReplica": true,
            "SupportedFeatureNames": [
                "Lambda"
            ],
            "Status": "available",
            "SupportsParallelQuery": false,
            "SupportsGlobalDatabases": false,
            "SupportsBabelfish": false,
            "SupportsCertificateRotationWithoutRestart": true,
            "SupportedCACertificateIdentifiers": [
                "rds-ca-rsa2048-g1",
                "rds-ca-ecc384-g1",
                "rds-ca-rsa4096-g1"
            ]
        }
    ]
}
```

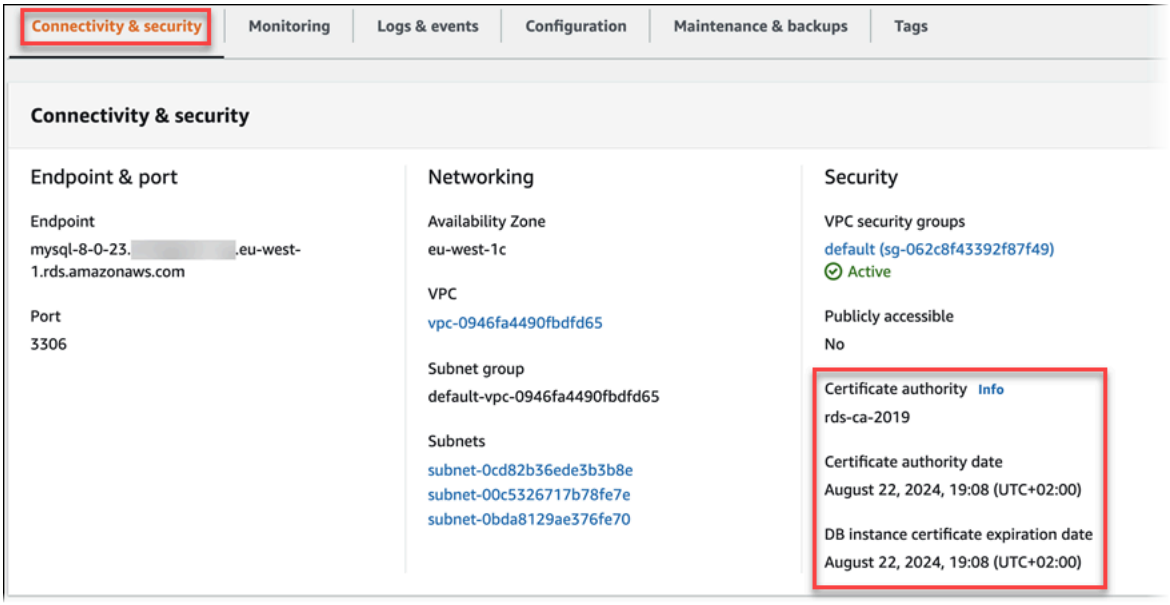
Validità dei certificati del server di database

La validità del certificato del server di database dipende dal motore di database e dalla versione del motore di database. Se la versione del motore di database supporta la rotazione del certificato senza riavvio, la validità del certificato del server di database è di 1 anno. In caso contrario, la validità è di 3 anni.

Per ulteriori informazioni sulla rotazione dei certificati del server di database, consulta Rotazione automatica dei certificati del server.

Visualizzazione della CA per l'istanza database

È possibile vedere i dettagli sulla CA per un database visualizzando la scheda Connettività e sicurezza nella console, come nell'immagine seguente.



Se si utilizza il AWS CLI, è possibile visualizzare i dettagli sulla CA per un'istanza DB utilizzando il describe-db-instancescomando.

Download di bundle di certificati per Aurora

Quando ci si connette al database con SSL o TLS, l'istanza del database richiede un certificato di attendibilità di Amazon RDS. Seleziona il link appropriato nella tabella seguente per scaricare il bundle corrispondente alla Regione AWS in cui si ospita il database.

Pacchetti di certificati di Regione AWS

I pacchetti di certificati per tutte le regioni Regioni AWS e GovCloud (Stati Uniti) contengono i seguenti certificati CA root:

- `rds-ca-rsa2048-g1`
- `rds-ca-rsa4096-g1`
- `rds-ca-ecc384-g1`

I certificati `rds-ca-rsa4096-g1` e `rds-ca-ecc384-g1` non sono disponibili nelle seguenti Regioni:

- Asia Pacifico (Mumbai)
- Asia Pacifico (Melbourne)
- Canada occidentale (Calgary)
- Europa (Zurigo)
- Europa (Spagna)
- Israele (Tel Aviv)

L'application trust store deve registrare solo il certificato CA principale. Non registrate i certificati CA intermedi nel vostro trust store poiché ciò potrebbe causare problemi di connessione quando RDS ruota automaticamente il certificato del server DB.

> **Note**
>
> Il proxy e Aurora Serverless v1 l'uso di Amazon RDS i certificati di AWS Certificate Manager (ACM). Se si utilizza Server proxy per RDS, non è necessario scaricare certificati Amazon RDS o aggiornare applicazioni che utilizzano connessioni Server proxy per RDS. Per ulteriori informazioni, consulta Utilizzo TLS/SSL con RDS Proxy.
> Se si utilizza Aurora Serverless v1, il download dei certificati Amazon RDS non è richiesto. Per ulteriori informazioni, consulta Utilizzo di TLS/SSL con Aurora Serverless v1.

Per scaricare un pacchetto di certificati per un Regione AWS, seleziona il link Regione AWS che ospita il database nella tabella seguente.

AWS Region	Bundle di certificati (PEM)	Pacchetto di certificati () PKCS7
Qualsiasi pubblicità Regione AWS	global-bundle.pem	global-bundle.p7b
Stati Uniti orientali (Virginia settentrionale)	us-east-1-bundle.pem	us-east-1-bundle.p7b

AWS Region	Bundle di certificati (PEM)	Pacchetto di certificati () PKCS7
US East (Ohio)	us-east-2-bundle.pem	us-east-2-bundle.p7b
US West (N. California)	us-west-1-bundle.pem	us-west-1-bundle.p7b
US West (Oregon)	us-west-2-bundle.pem	us-west-2-bundle.p7b
Africa (Cape Town)	af-south-1-bundle.pem	af-sud-1-bundle.p7b
Asia Pacific (Hong Kong)	ap-east-1-bundle.pem	ap-east-1-bundle.p7b
Asia Pacific (Hyderabad)	ap-south-2-bundle.pem	ap-south-2-bundle.p7b
Asia Pacifico (Giacarta)	ap-southeast-3-bundle.pem	ap-southeast-3-bundle.p7b
Asia Pacifico (Malesia)	ap-southeast-5-bundle.pem	ap-southeast-5-bundle.p7b
Asia Pacifico (Melbourne)	ap-southeast-4-bundle.pem	ap-southeast-4-bundle.p7b
Asia Pacifico (Mumbai)	ap-south-1-bundle.pem	ap-south-1-bundle.p7b
Asia Pacific (Osaka)	ap-northeast-3-bundle.pem	ap-northeast-3-bundle.p7b
Asia Pacifico (Thailandia)	ap-southeast-7-bundle.pem	ap-southeast-7-bundle.p7b
Asia Pacifico (Tokyo)	ap-northeast-1-bundle.pem	ap-northeast-1-bundle.p7b
Asia Pacific (Seoul)	ap-northeast-2-bundle.pem	ap-northeast-2-bundle.p7b
Asia Pacific (Singapore)	ap-southeast-1-bundle.pem	ap-southeast-1-bundle.p7b
Asia Pacific (Sydney)	ap-southeast-2-bundle.pem	ap-southeast-2-bundle.p7b
Canada (Central)	ca-central-1-bundle.pem	ca-central-1-bundle.p7b
Canada occidentale (Calgary)	ca-west-1-bundle.pem	ca-west-1-bundle.p7b
Europa (Francoforte)	eu-central-1-bundle.pem	eu-central-1-bundle.p7b
Europe (Ireland)	eu-west-1-bundle.pem	eu-west-1-bundle.p7b

AWS Region	Bundle di certificati (PEM)	Pacchetto di certificati () PKCS7
Europe (London)	eu-west-2-bundle.pem	eu-west-2-bundle.p7b
Europe (Milan)	eu-south-1-bundle.pem	eu-sud-1-bundle.p7b
Europe (Paris)	eu-west-3-bundle.pem	eu-west-3-bundle.p7b
Europa (Spagna)	eu-south-2-bundle.pem	eu-south-2-bundle.p7b
Europa (Stoccolma)	eu-nord-1-bundle.pem	eu-nord-1-bundle.p7b
Europa (Zurigo)	eu-central-2-bundle.pem	eu-central-2-bundle.p7b
Israele (Tel Aviv)	il-central-1-bundle.pem	il-central-1-bundle.p7b
Messico (centrale)	mx-central-1-bundle.pem	mx-central-1-bundle.p7b
Middle East (Bahrain)	me-sud-1-bundle.pem	me-sud-1-bundle.p7b
Medio Oriente (Emirati Arabi Uniti)	me-central-1-bundle.pem	me-central-1-bundle.p7b
Sud America (San Paolo)	sa-east-1-bundle.pem	sa-east-1-bundle.p7b
Qualsiasi AWS GovCloud (US) Region s	global-bundle.pem	global-bundle.p7b
AWS GovCloud (Stati Uniti orientali)	us-gov-east-1 bundle.pem	us-gov-east-1-pacchetto.p7b
AWS GovCloud (Stati Uniti occidentali)	us-gov-west-1-bundle.pem	us-gov-west-1-pacchetto.p7b

Visualizzazione del contenuto del certificato CA

Per verificare il contenuto del bundle di certificati CA, utilizza il comando seguente:

```
keytool -printcert -v -file global-bundle.pem
```

Rotazione del certificato SSL/TLS

I certificati dell'autorità di certificazione Amazon RDS rds-ca-2019 scadono ad agosto 2024. Se utilizzi o prevedi di utilizzare Secure Sockets Layer (SSL) o Transport Layer Security (TLS) con verifica del certificato per connetterti alle istanze DB RDS o ai 384-g1. rds-ca-rsa rds-ca-rsa rds-ca-ecc Se attualmente non lo utilizzi SSL/TLS con la verifica dei certificati, potresti avere ancora un certificato CA scaduto e devi aggiornarlo con un nuovo certificato CA se prevedi di utilizzarlo SSL/TLS con la verifica dei certificati per connetterti ai tuoi database RDS.

Amazon RDS fornisce nuovi certificati CA come best practice di AWS sicurezza. Per informazioni sui nuovi certificati e sulle AWS regioni supportate, consultaUtilizzo SSL/TLS per crittografare una connessione a un'.

Per aggiornare il certificato CA per il database, utilizza i seguenti metodi:

- Aggiornamento del certificato CA modificando l'istanza database
- Aggiornamento del certificato CA mediante l'applicazione di manutenzione

Prima di aggiornare le istanze database per utilizzare il nuovo certificato CA, assicurati di aggiornare i client o le applicazioni che si collegano ai database RDS.

Considerazioni sulla rotazione dei certificati

Considera le seguenti situazioni prima di ruotare il certificato:

- Il proxy e Aurora Serverless v1 l'uso di Amazon RDS i certificati di AWS Certificate Manager (ACM). Se utilizzi RDS Proxy, quando ruoti il SSL/TLS certificato non devi aggiornare le applicazioni che utilizzano connessioni proxy RDS. Per ulteriori informazioni, consulta Utilizzo TLS/SSL con RDS Proxy.
- Se si utilizza Aurora Serverless v1, il download dei certificati Amazon RDS non è richiesto. Per ulteriori informazioni, consulta Utilizzo di TLS/SSL con Aurora Serverless v1.
- Se si utilizza un'applicazione Go versione 1.15 con un'istanza database creata o aggiornata al certificato rds-ca-2019 prima del 28 luglio 2020, è necessario aggiornare nuovamente il certificato.

  Utilizza il comando `modify-db-instance` , utilizzando il nuovo identificatore del certificato CA. Puoi trovare quelli disponibili per uno specifico motore DB e una versione del motore DB utilizzando CAs il comando. `describe-db-engine-versions`

Se hai creato il database o aggiornato il relativo certificato dopo il 28 luglio 2020, non è richiesta alcuna azione. Per ulteriori informazioni, vedete Go GitHub issue #39568.

Aggiornamento del certificato CA modificando l'istanza database

L'esempio seguente aggiorna il certificato CA da rds-ca-2019 a 2048-g1. rds-ca-rsa Puoi scegliere un certificato diverso. Per ulteriori informazioni, consultaAutorità di certificazione.

Aggiorna l'archivio di trust delle applicazioni per ridurre il tempo di inattività associato all'aggiornamento del certificato CA. Per ulteriori informazioni sui riavvii associati alla rotazione dei certificati CA, consulta Rotazione automatica dei certificati del server.

Per aggiornare il certificato CA modificando l'istanza database

1. Scarica il nuovo SSL/TLS certificato come descritto inUtilizzo SSL/TLS per crittografare una connessione a un'.
2. Aggiornare le applicazioni per utilizzare il nuovo certificato SSL/TLS.

   I metodi per aggiornare le richieste di nuovi SSL/TLS certificati dipendono dalle applicazioni specifiche. Collabora con gli sviluppatori delle tue applicazioni per aggiornare i SSL/TLS certificati delle tue applicazioni.

   Per informazioni sul controllo delle SSL/TLS connessioni e sull'aggiornamento delle applicazioni per ogni motore di database, consultate i seguenti argomenti:

   - Aggiornamento delle applicazioni per la connessione ai cluster database Aurora MySQL utilizzando nuovi certificati TLS
   - Aggiornamento delle applicazioni per la connessione ai cluster DB Aurora PostgreSQL utilizzando nuovi certificati SSL/TLS

   Per uno script di esempio che aggiorna un trust store per un sistema operativo Linux, consultaScript di esempio per l'importazione di certificati nel tuo archivio di trust.

   > **Note**
   >
   > Il bundle di certificati contiene certificati per la vecchia e la nuova CA, pertanto puoi aggiornare l'applicazione in modo sicuro e mantenere la connettività durante il periodo di transizione. Se si utilizza il AWS Database Migration Service per migrare un database

verso un', si consiglia di utilizzare il pacchetto di certificati per garantire la connettività durante la migrazione.

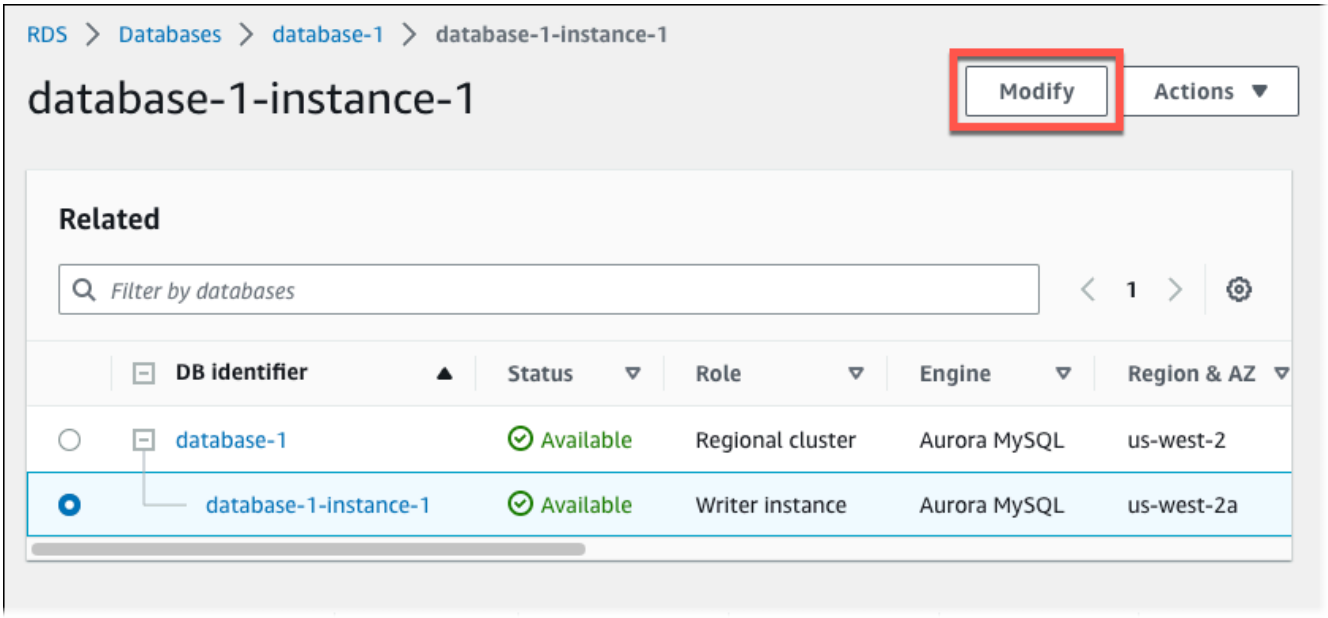
3. Modifica l'istanza DB per cambiare la CA da rds-ca-2019 a 2048-g1. rds-ca-rsa Per verificare se il database richiede un riavvio per aggiornare i certificati CA, utilizza il comando e seleziona il flag. describe-db-engine-versions`SupportsCertificateRotationWithoutRestart`

> **Note**
>
> Riavvia il cluster Babelfish dopo averlo modificato per aggiornare il certificato CA.

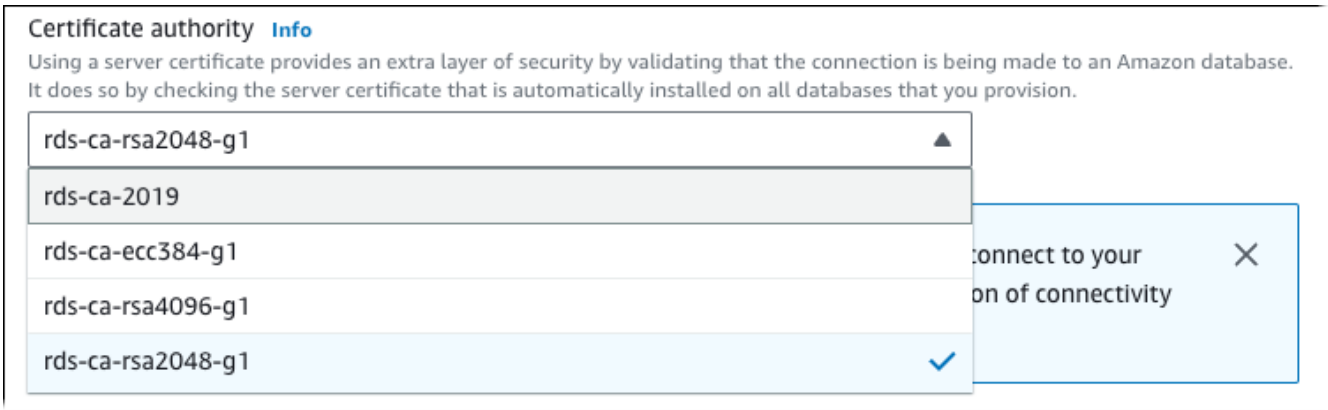
> **Important**
>
> Se si verificano problemi di connettività dopo la scadenza del certificato, utilizzare l'opzione Applica immediatamente specificando Apply immediately (Applica immediatamente) nella console o specificando l'opzione `--apply-immediately` mediante AWS CLI. Per impostazione predefinita, questa operazione è pianificata per l'esecuzione durante la prossima finestra di manutenzione.
> Per impostare una sostituzione della CA per il cluster diversa dalla CA RDS predefinita, utilizza il comando CLI modify-certificates.

Console

1. Accedi a Console di gestione AWS e apri la console Amazon RDS all'indirizzo https://console.aws.amazon.com/rds/.
2. Nel pannello di navigazione, scegli Database, quindi scegli l’istanza database che vuoi modificare.
3. Scegli Modifica.



4. Nella sezione Connettività, scegli rds-ca-rsa2048-g1.



5. Scegliere Continue (Continua) e controllare il riepilogo delle modifiche.
6. Per applicare immediatamente le modifiche, scegliere Apply immediately (Applica immediatamente).
7. Nella pagina di conferma esaminare le modifiche. Se sono corrette, scegliere Modifica istanza DB per salvare le modifiche.

> **Important**
>
> Quando si pianifica questa operazione, accertarsi di aver aggiornato in anticipo l'archivio di trust lato client.

Oppure scegliere Back (Indietro) per cambiare le modifiche o Cancel (Annulla) per annullare le modifiche.

AWS CLI

Specificare l'identificatore dell'istanza database e l'opzione `--ca-certificate-identifier`.

Utilizzare il parametro `--apply-immediately` per applicare immediatamente l'aggiornamento. Per impostazione predefinita, questa operazione è pianificata per l'esecuzione durante la prossima finestra di manutenzione.

> **Important**
>
> Quando si pianifica questa operazione, accertarsi di aver aggiornato in anticipo l'archivio di trust lato client.

Example

L'esempio seguente modifica `mydbinstance` impostando il certificato CA su `rds-ca-rsa2048-g1`.

Per Linux, macOS o Unix:

```
aws rds modify-db-instance \
    --db-instance-identifier mydbinstance \
    --ca-certificate-identifier rds-ca-rsa2048-g1
```

Per Windows:

```
aws rds modify-db-instance ^
    --db-instance-identifier mydbinstance ^
    --ca-certificate-identifier rds-ca-rsa2048-g1
```

> **Note**
>
> Se l'istanza richiede il riavvio, puoi utilizzare il comando modify-db-instanceCLI e specificare `--no-certificate-rotation-restart` l'opzione.

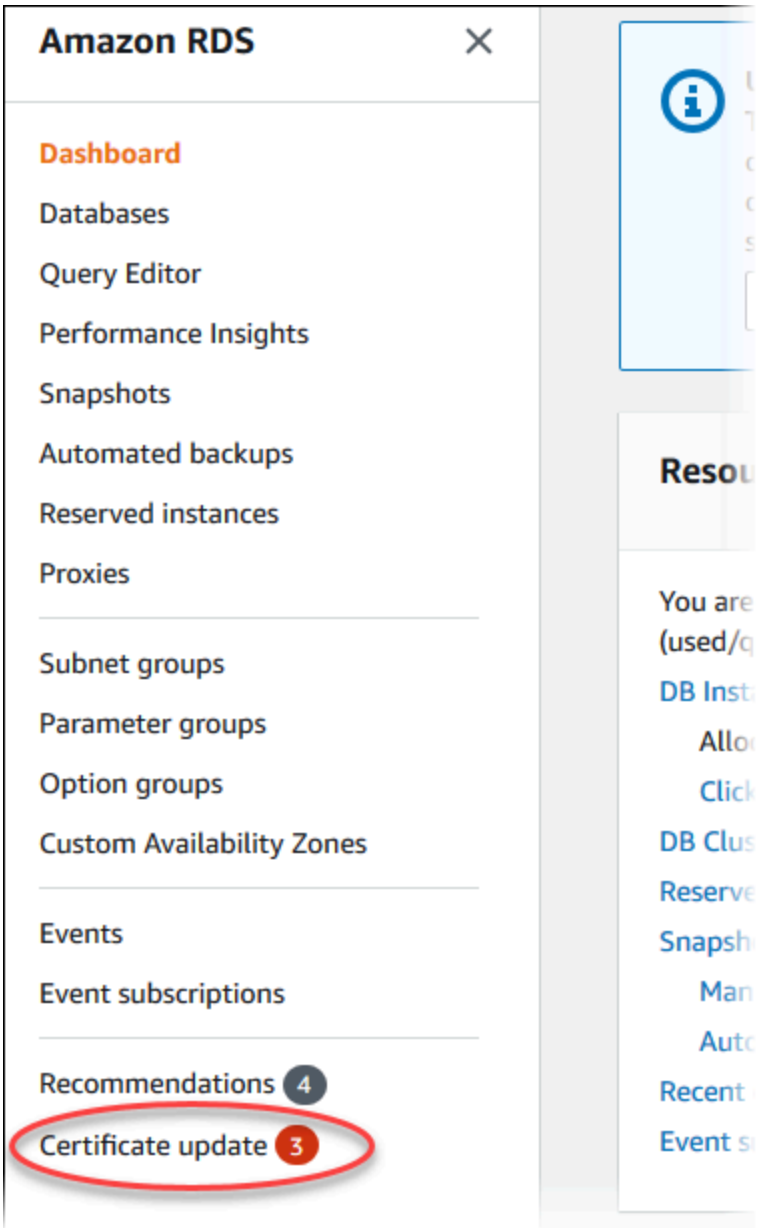
Aggiornamento del certificato CA mediante l'applicazione di manutenzione

Completa la procedura seguente per aggiornare il certificato CA applicando la manutenzione.

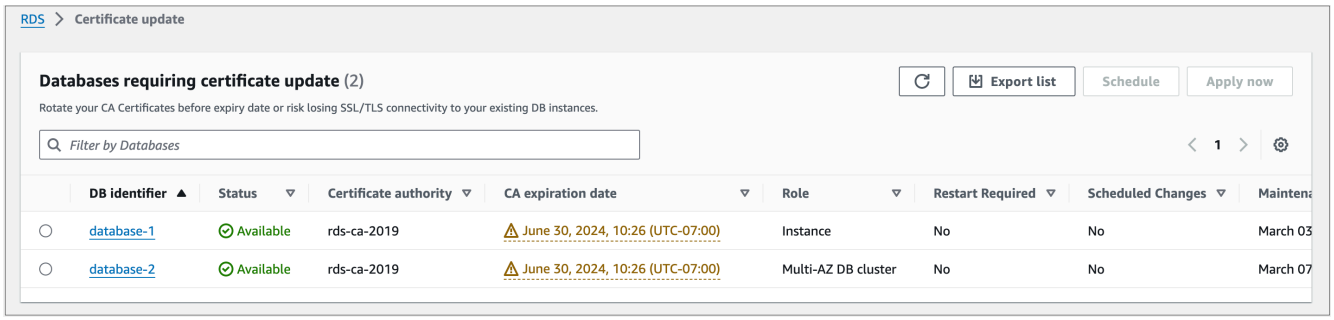
Console

Per aggiornare il certificato CA applicando la manutenzione

1. Accedi a Console di gestione AWS e apri la console Amazon RDS all'indirizzo https://console.aws.amazon.com/rds/.
2. Nel riquadro di navigazione, scegli Aggiornamento certificato.



Viene visualizzata la pagina Database con aggiornamento certificati richiesto.



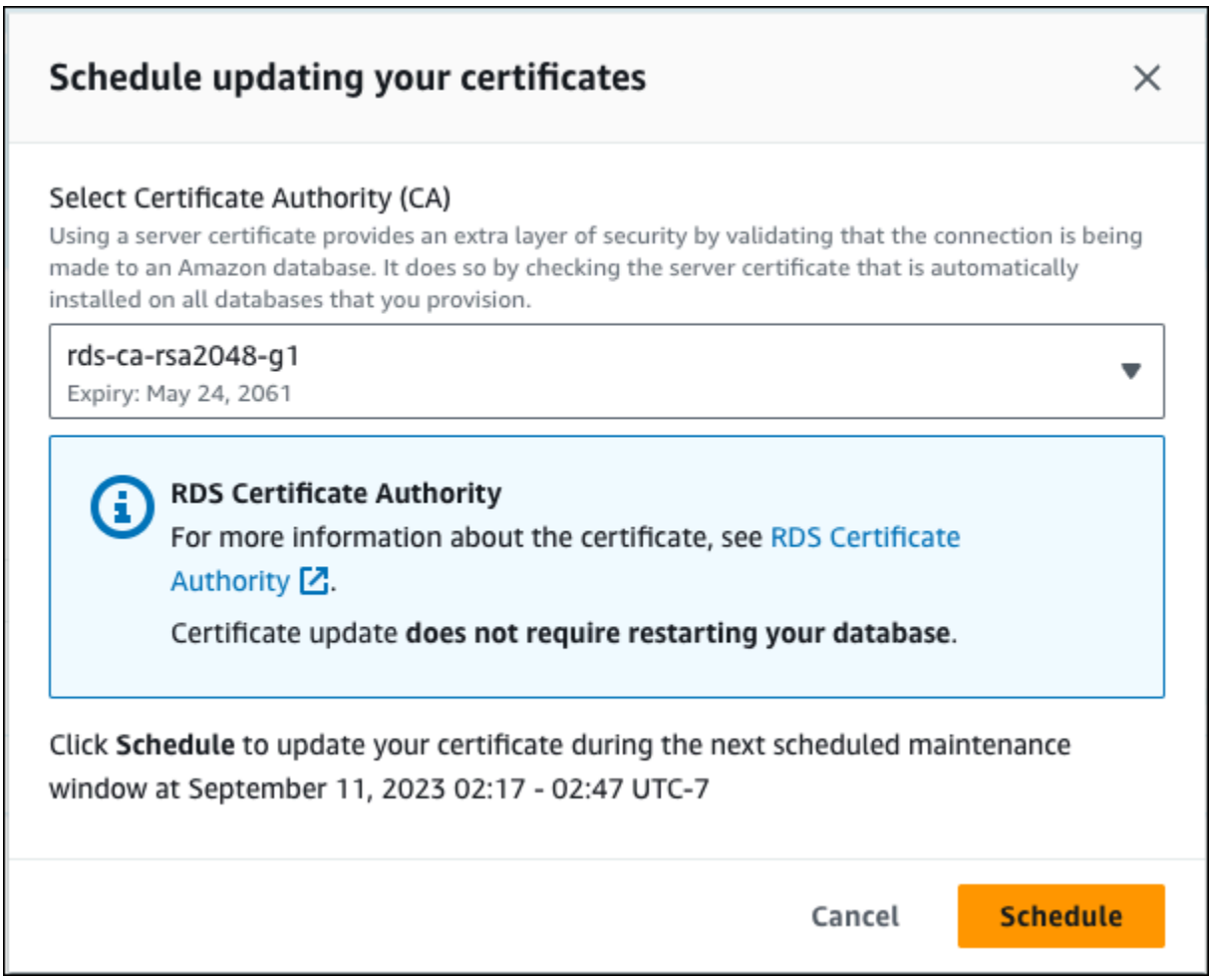
> **Note**
>
> Questa pagina mostra solo le istanze DB della versione corrente. Regione AWS Se disponi di database in più di uno Regione AWS, controlla questa pagina in ciascuno di essi Regione AWS per vedere tutte le istanze DB con vecchi certificati. SSL/TLS

3. Scegli l'istanza database da aggiornare.

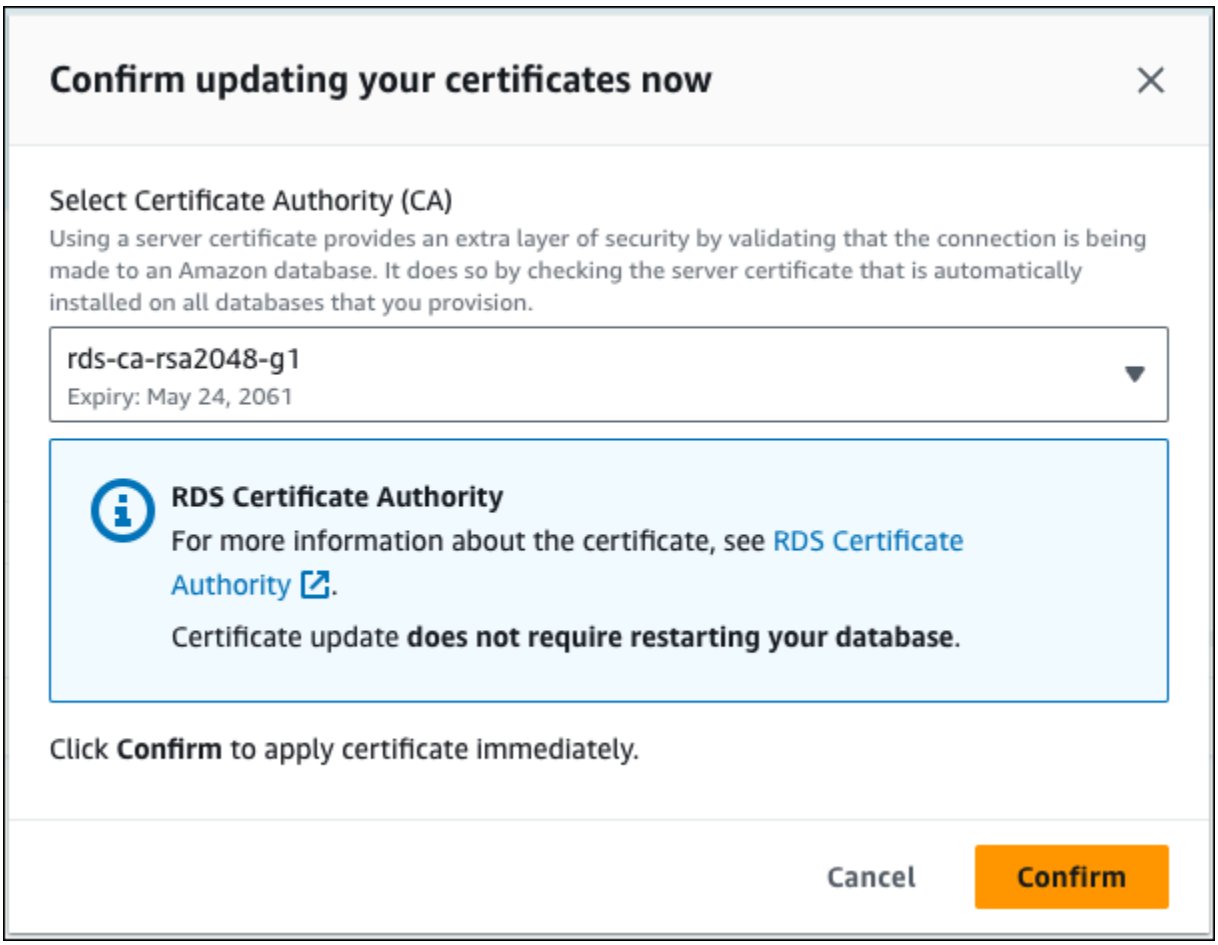
   È possibile pianificare la rotazione dei certificati per la finestra di manutenzione successiva scegliendo Pianifica. Applica immediatamente la rotazione scegliendo Applica ora.

> **Important**
>
> Se si verificano problemi di connettività dopo la scadenza del certificato, utilizza l'opzione Applica ora.

4. a. Se scegli Pianifica, ti viene richiesto di confermare la rotazione dei certificati CA. Nella richiesta viene indicato anche la finestra pianificata per l'aggiornamento.



b. Se scegli Applica ora, ti viene richiesto di confermare la rotazione dei certificati CA.



> **Important**
>
> Prima di pianificare la rotazione dei certificati CA sul database, aggiorna tutte le applicazioni client che utilizzano SSL/TLS e il certificato del server per la connessione. Questi aggiornamenti sono specifici per il motore DB. Dopo avere aggiornato queste applicazioni client, è possibile confermare la rotazione del certificato CA.

Per continuare, scegliere la casella di controllo e quindi scegliere Confirm (Conferma).

5. Ripetere i passaggi 3 e 4 per ogni istanza database da aggiornare.

Rotazione automatica dei certificati del server

Se la CA root supporta la rotazione automatica dei certificati del server, RDS gestisce automaticamente la rotazione dei certificati del server di database. Per questa rotazione automatica, RDS utilizza la stessa CA root e pertanto non è necessario scaricare un nuovo bundle CA. Per informazioni, consultare Autorità di certificazione.

La rotazione e la validità del certificato del server di database dipendono dal motore di database:

- Se il motore di database supporta la rotazione senza riavvio, RDS esegue automaticamente la rotazione del certificato del server di database senza richiedere alcuna azione da parte dell'utente. RDS tenta di eseguire la rotazione del certificato del server di database nella finestra di manutenzione preferita in corrispondenza della semivita del certificato del server di database. Il nuovo certificato del server di database è valido per 12 mesi.
- Se il tuo motore DB non supporta la rotazione senza riavvio, Amazon RDS rende visibile un'azione di manutenzione `server-certificate-rotation` in sospeso tramite Describe-pending-maintenance-actions API, al termine del periodo di dimezzamento del certificato o almeno 3 mesi prima della scadenza. Puoi applicare la rotazione utilizzando l'API. apply-pending-maintenance-action Il nuovo certificato del server di database è valido per 36 mesi.

Usa il describe-db-engine-versionscomando e controlla il `SupportsCertificateRotationWithoutRestart` flag per identificare se la versione del motore DB supporta la rotazione del certificato senza riavvio. Per ulteriori informazioni, consulta Impostazione della CA per il database.

Script di esempio per l'importazione di certificati nel tuo archivio di trust

Di seguito sono riportati script di shell di esempio che importano il bundle di certificati in un archivio di trust.

Ogni script di shell di esempio utilizza keytool, che fa parte del Java Development Kit (JDK). Per informazioni sull'installazione di JDK, consulta la Guida di installazione di JDK.

Linux

Il seguente script è uno script di esempio shell che importa il bundle di certificati in un archivio di trust su un sistema operativo Linux.

```
mydir=tmp/certs
```

```
if [ ! -e "${mydir}" ]
then
mkdir -p "${mydir}"
fi truststore=${mydir}/rds-truststore.jks storepassword=changeit

curl -sS "https://truststore.pki.rds.amazonaws.com/global/global-bundle.pem">
 ${mydir}/global-bundle.pem
awk 'split_after == 1 {n++;split_after=0} /-----END CERTIFICATE-----/
 {split_after=1}{print > "rds-ca-" n+1 ".pem"}' < ${mydir}/global-bundle.pem

for CERT in rds-ca-*; do alias=$(openssl x509 -noout -text -in $CERT | perl -ne
 'next unless /Subject:/; s/.*(CN=|CN = )//; print')
  echo "Importing $alias"
  keytool -import -file ${CERT} -alias "${alias}" -storepass ${storepassword} -
keystore ${truststore} -noprompt
  rm $CERT
done

rm ${mydir}/global-bundle.pem

echo "Trust store content is: "

keytool -list -v -keystore "$truststore" -storepass ${storepassword} | grep Alias |
 cut -d " " -f3- | while read alias
do expiry=`keytool -list -v -keystore "$truststore" -storepass ${storepassword} -
alias "${alias}" | grep Valid | perl -ne 'if(/until: (.*?)\n/) { print "$1\n"; }'`
   echo " Certificate ${alias} expires in '$expiry'"
done
```

macOS

Il seguente script è uno script di shell di esempio che importa il bundle di certificati in un archivio di trust su un sistema operativo Linux.

```
mydir=tmp/certs
if [ ! -e "${mydir}" ]
then
mkdir -p "${mydir}"
fi truststore=${mydir}/rds-truststore.jks storepassword=changeit

curl -sS "https://truststore.pki.rds.amazonaws.com/global/global-bundle.pem">
 ${mydir}/global-bundle.pem
```

```
split -p "-----BEGIN CERTIFICATE-----" ${mydir}/global-bundle.pem rds-ca-

for CERT in rds-ca-*; do alias=$(openssl x509 -noout -text -in $CERT | perl -ne
 'next unless /Subject:/; s/.*(CN=|CN = )//; print')
  echo "Importing $alias"
  keytool -import -file ${CERT} -alias "${alias}" -storepass ${storepassword} -
keystore ${truststore} -noprompt
  rm $CERT
done

rm ${mydir}/global-bundle.pem

echo "Trust store content is: "

keytool -list -v -keystore "$truststore" -storepass ${storepassword} | grep Alias |
 cut -d " " -f3- | while read alias
do expiry=`keytool -list -v -keystore "$truststore" -storepass ${storepassword} -
alias "${alias}" | grep Valid | perl -ne 'if(/until: (.*?)\n/) { print "$1\n"; }'`
   echo " Certificate ${alias} expires in '$expiry'"
done
```

Per ulteriori informazioni sulle best practice sull'utilizzo di SSL con Amazon RDS, consulta Best practices for successful SSL connections to Amazon RDS for Oracle.

Riservatezza del traffico Internet

Le connessioni sono protette tra Amazon Aurora e le applicazioni on-premises e tra Amazon Aurora e altre risorse AWS nella stessa Regione AWS.

Traffico tra servizio e applicazioni e client locali

Sono disponibili due opzioni di connettività tra la rete privata e AWS:

- Una connessione Site-to-Site VPN AWS Per ulteriori informazioni, consulta Che cos'è AWS Site-to-Site VPN?
- Una connessione Direct Connect. Per ulteriori informazioni, consulta Che cos'è Direct Connect?

Puoi ottenere l'accesso ad Amazon Aurora tramite la rete utilizzando le operazioni API pubblicate da AWS. I client devono supportare quanto segue:

- Transport Layer Security (TLS). È richiesto TLS 1.2 ed è consigliato TLS 1.3.

- Suite di cifratura con Perfect Forward Secrecy (PFS), ad esempio Ephemeral Diffie-Hellman (DHE) o Elliptic Curve Ephemeral Diffie-Hellman (ECDHE). La maggior parte dei sistemi moderni, come Java 7 e versioni successive, supporta tali modalità.

Inoltre, le richieste devono essere firmate utilizzando un ID chiave di accesso e una chiave di accesso segreta associata a un principale IAM. In alternativa, è possibile utilizzare AWS Security Token Service (AWS STS) per generare le credenziali di sicurezza temporanee per sottoscrivere le richieste.

Gestione accessi e identità per Amazon Aurora

AWS Identity and Access Management (IAM) è un programma Servizio AWS che aiuta un amministratore a controllare in modo sicuro l'accesso alle AWS risorse. Gli amministratori IAM controllano chi può essere autenticato (accesso effettuato) e autorizzato (dispone di autorizzazioni) per utilizzare le risorse Amazon RDS. IAM è un software Servizio AWS che puoi utilizzare senza costi aggiuntivi.

Argomenti

- Destinatari
- Autenticazione con identità
- Gestione dell’accesso tramite policy
- Funzionamento di Amazon Aurora con IAM
- Esempi di policy di Amazon Aurora basate su identità
- Policy gestite da AWS per Amazon RDS
- Aggiornamenti di Amazon RDS sulle policy gestite da AWS
- Prevenzione del problema "confused deputy" tra servizi
- Autenticazione del database IAM
- Risoluzione dei problemi di identità e accesso in Amazon Aurora

Destinatari

Il modo in cui utilizzi AWS Identity and Access Management (IAM) varia a seconda del lavoro svolto in Aurora.

Utente del servizio –Se utilizzi il servizio Aurora per eseguire il tuo lavoro, l’amministratore fornisce le credenziali e le autorizzazioni necessarie. All’aumentare del numero di funzionalità Aurora utilizzate per il lavoro, potrebbero essere necessarie ulteriori autorizzazioni. La comprensione della gestione dell’accesso consente di richiedere le autorizzazioni corrette all’amministratore. Se non riesci ad accedere a una funzionalità di Aurora, consulta Risoluzione dei problemi di identità e accesso in Amazon Aurora.

Amministratore del servizio – Se sei il responsabile delle risorse Aurorapresso la tua azienda, probabilmente disponi dell’accesso completo a Aurora. Il tuo compito è determinare le caratteristiche

e le risorse Aurora a cui i dipendenti devono accedere. Devi inviare le richieste all'amministratore per cambiare le autorizzazioni degli utenti del servizio. Esamina le informazioni contenute in questa pagina per comprendere le nozioni di base di IAM. Per ulteriori informazioni su come la tua azienda può utilizzare IAM con Aurora, consulta Funzionamento di Amazon Aurora con IAM.

Amministratore – Se sei un amministratore, potresti essere interessato a ottenere informazioni su come puoi scrivere policy per gestire l'accesso a Aurora. Per visualizzare policy basate su identità Aurora di esempio che puoi utilizzare in IAM, consulta Esempi di policy di Amazon Aurora basate su identità.

Autenticazione con identità

L'autenticazione è il modo in cui accedi AWS utilizzando le tue credenziali di identità. Devi autenticarti come utente IAM o assumendo un ruolo IAM. Utente root dell'account AWS

Puoi accedere come identità federata utilizzando credenziali provenienti da una fonte di identità come AWS IAM Identity Center (IAM Identity Center), autenticazione Single Sign-On o credenziali. Google/ Facebook Per maggiori informazioni sull'accesso, consultare la sezione Come accedere a Account AWS nella Guida per l'utente di Accedi ad AWS .

Per l'accesso programmatico, AWS fornisce un SDK e una CLI per firmare crittograficamente le richieste. Per ulteriori informazioni, consulta AWS Signature Version 4 per le richieste API nella Guida per l'utente IAM.

AWS account (utente root)

Quando si crea un Account AWS, si inizia con un'identità di accesso denominata utente Account AWS root che ha accesso completo a tutte Servizi AWS le risorse. Si consiglia vivamente di non utilizzare l'utente root per le attività quotidiane. Per le attività che richiedono le credenziali come utente root, consulta Attività che richiedono le credenziali dell'utente root nella Guida per l'utente IAM.

Identità federata

Come procedura ottimale, richiedi agli utenti umani di utilizzare la federazione con un provider di identità per accedere Servizi AWS utilizzando credenziali temporanee.

Un'identità federata è un utente della directory aziendale, del provider di identità Web o Directory Service che accede Servizi AWS utilizzando le credenziali di una fonte di identità. Le identità federate assumono ruoli che forniscono credenziali temporanee.

Per la gestione centralizzata degli accessi, consigliamo di utilizzare AWS IAM Identity Center. Per ulteriori informazioni, consulta Cos'è IAM Identity Center? nella Guida per l'utente di AWS IAM Identity Center .

Utenti e gruppi IAM

Un utente IAM è un'identità che dispone di autorizzazioni specifiche per una singola persona o applicazione. Ti consigliamo di utilizzare credenziali temporanee invece di utenti IAM con credenziali a lungo termine. Per ulteriori informazioni, consulta Richiedere agli utenti umani di utilizzare la federazione con un provider di identità per accedere AWS utilizzando credenziali temporanee nella Guida per l'utente IAM.

Un gruppo IAM specifica una raccolta di utenti IAM e semplifica la gestione delle autorizzazioni per gestire gruppi di utenti di grandi dimensioni. Per ulteriori informazioni, consulta Casi d'uso per utenti IAM nella Guida per l'utente IAM.

Puoi eseguire l'autenticazione nel cluster di database tramite l'autenticazione del database IAM.

L'autenticazione del database IAM funziona con Aurora. Per ulteriori informazioni sull'autenticazione al cluster di database tramite IAM, consulta Autenticazione del database IAM .

Ruoli IAM

Un ruolo IAM è un'identità interna all'utente Account AWS che dispone di autorizzazioni specifiche. È simile a un utente, ma non è associato a una persona specifica. Puoi assumere temporaneamente un ruolo IAM in Console di gestione AWS cambiando ruolo. Puoi assumere un ruolo chiamando un'operazione AWS CLI o AWS API o utilizzando un URL personalizzato. Per ulteriori informazioni sui metodi per l'utilizzo dei ruoli, consulta Utilizzo di ruoli IAM nella Guida per l'utente IAM.

I ruoli IAM con credenziali temporanee sono utili nelle seguenti situazioni:

- Autorizzazioni utente temporanee: un utente può assumere un ruolo IAM per ottenere temporaneamente autorizzazioni diverse per un'attività specifica.
- Accesso utente federato - Per assegnare le autorizzazioni a una identità federata, è possibile creare un ruolo e definire le autorizzazioni per il ruolo. Quando un'identità federata viene autenticata, l'identità viene associata al ruolo e ottiene le autorizzazioni da esso definite. Per ulteriori informazioni sulla federazione dei ruoli, consulta Creare un ruolo per un provider di identità di terza parte (federazione) nella Guida per l'utente IAM. Se si utilizza IAM Identity Center, configurare un set di autorizzazioni. IAM Identity Center mette in correlazione il set di autorizzazioni

con un ruolo in IAM per controllare a cosa possono accedere le identità dopo l'autenticazione. Per informazioni sui set di autorizzazioni, consulta Set di autorizzazioni nella Guida per l'utente AWS IAM Identity Center .

- Accesso multi-account: è possibile utilizzare un ruolo IAM per permettere a un utente (un principale affidabile) con un account diverso di accedere alle risorse nell'account. I ruoli sono lo strumento principale per concedere l'accesso multi-account. Tuttavia, con alcuni Servizi AWS, è possibile allegare una policy direttamente a una risorsa (anziché utilizzare un ruolo come proxy). Per informazioni sulle differenze tra ruoli e policy basate su risorse per l'accesso multi-account, consulta Differenza tra i ruoli IAM e le policy basate su risorse nella Guida per l'utente IAM.
- Accesso a più servizi: alcuni Servizi AWS utilizzano le funzionalità di altri Servizi AWS. Ad esempio, quando effettui una chiamata in un servizio, è normale che quel servizio esegua applicazioni in Amazon EC2 o archivi oggetti in Amazon S3. Un servizio può eseguire questa operazione utilizzando le autorizzazioni dell'entità chiamante, utilizzando un ruolo di servizio o utilizzando un ruolo collegato al servizio.
  - Sessioni di accesso inoltrato: le sessioni di accesso inoltrato (FAS) utilizzano le autorizzazioni del principale chiamante e Servizio AWS, in combinazione con la richiesta, Servizio AWS per effettuare richieste ai servizi downstream. Per i dettagli delle policy relative alle richieste FAS, consultare Forward access sessions.
  - Ruolo di servizio - Un ruolo di servizio è un ruolo IAM che un servizio assume per eseguire operazioni per conto dell'utente. Un amministratore IAM può creare, modificare ed eliminare un ruolo di servizio dall'interno di IAM. Per ulteriori informazioni, consulta la sezione Create a role to delegate permissions to an Servizio AWS nella Guida per l'utente IAM.
  - Ruolo collegato al servizio: un ruolo collegato al servizio è un tipo di ruolo di servizio collegato a un. Servizio AWS Il servizio può assumere il ruolo per eseguire un'azione per tuo conto. I ruoli collegati al servizio vengono visualizzati nel tuo account Account AWS e sono di proprietà del servizio. Un amministratore IAM può visualizzare le autorizzazioni per i ruoli collegati al servizio, ma non modificarle.
- Applicazioni in esecuzione su Amazon EC2: puoi utilizzare un ruolo IAM per gestire le credenziali temporanee per le applicazioni in esecuzione su un' EC2 istanza e che AWS CLI effettuano richieste AWS API. Questa soluzione è preferibile alla memorizzazione delle chiavi di accesso all'interno dell' EC2 istanza. Per assegnare un AWS ruolo a un' EC2 istanza e renderlo disponibile per tutte le sue applicazioni, create un profilo di istanza collegato all'istanza. Un profilo di istanza contiene il ruolo e consente ai programmi in esecuzione sull' EC2 istanza di ottenere credenziali temporanee. Per ulteriori informazioni, consulta Utilizzare un ruolo IAM per concedere le autorizzazioni alle applicazioni in esecuzione su EC2 istanze Amazon nella IAM User Guide.

Per informazioni sull'utilizzo di ruoli IAM, consulta Quando creare un ruolo IAM invece di un utente nella Guida per l'utente di IAM.

Gestione dell'accesso tramite policy

Puoi controllare l'accesso AWS creando policy e collegandole a identità o risorse IAM. AWS Una policy è un oggetto AWS che, se associato a un'identità o a una risorsa, ne definisce le autorizzazioni. AWS valuta queste politiche quando un'entità (utente root, utente o ruolo IAM) effettua una richiesta. Le autorizzazioni nelle policy determinano l'approvazione o il rifiuto della richiesta. La maggior parte delle politiche viene archiviata AWS come documenti JSON. Per ulteriori informazioni sulla struttura e sui contenuti dei documenti delle policy JSON, consulta Panoramica delle policy JSON nella Guida per l'utente IAM.

Un amministratore può utilizzare le policy per specificare chi ha accesso alle AWS risorse e quali azioni può eseguire su tali risorse. Ogni entità IAM (set di autorizzazioni o ruolo) inizialmente non dispone di autorizzazioni. Ovvero, di default, gli utenti non possono eseguire alcuna operazione, neppure modificare la propria password. Per autorizzare un utente a eseguire operazioni, un amministratore deve allegare una policy di autorizzazioni a tale utente. In alternativa, l'amministratore può aggiungere l'utente a un gruppo che dispone delle autorizzazioni desiderate. Quando un amministratore fornisce le autorizzazioni a un gruppo, le autorizzazioni vengono concesse a tutti gli utenti in tale gruppo.

Le policy IAM definiscono le autorizzazioni relative a un'operazione, indipendentemente dal metodo utilizzato per eseguirla. Ad esempio, supponiamo di disporre di una policy che consente l'operazione `iam:GetRole`. Un utente con tale policy può ottenere informazioni sul ruolo dall' Console di gestione AWS AWS CLI, dall'o dall' AWS API.

Policy basate sull'identità

Le policy basate su identità sono documenti di policy di autorizzazione JSON che è possibile collegare a un'identità, ad esempio un set di autorizzazioni o un ruolo. Tali policy definiscono le operazioni autorizzate per l'identità, nonché le risorse e le condizioni in cui possono essere eseguite. Per informazioni su come creare una policy basata su identità, consulta Creazione di policy IAM nella Guida per l'utente IAM.

Le policy basate su identità possono essere ulteriormente classificate come policy inline o policy gestite. Le policy inline sono incorporate direttamente in un singolo set di autorizzazioni o ruolo. Le politiche gestite sono politiche autonome che puoi allegare a più set di autorizzazioni e ruoli nel tuo

AWS account. Le politiche gestite includono politiche AWS gestite e politiche gestite dai clienti. Per informazioni su come scegliere tra una policy gestita o una policy inline, consulta Scelta fra policy gestite e policy inline nella Guida per l'utente IAM.

Per informazioni sulle policy AWS gestite specifiche di Aurora, consulta. Policy gestite da AWS per Amazon RDS

Altri tipi di policy

AWS supporta tipi di policy aggiuntivi e meno comuni. Questi tipi di policy possono impostare il numero massimo di autorizzazioni concesse dai più tipi di policy comuni.

- Limiti delle autorizzazioni: un limite delle autorizzazioni è una funzione avanzata nella quale si imposta il numero massimo di autorizzazioni che una policy basata su identità può concedere a un'entità IAM (set di autorizzazioni o ruolo). È possibile impostare un limite delle autorizzazioni per un'entità. Le autorizzazioni risultanti sono l'intersezione delle policy basate su identità dell'entità e i suoi limiti delle autorizzazioni. Le policy basate su risorse che specificano il set di autorizzazioni o il ruolo nel campo `Principal` sono condizionate dal limite delle autorizzazioni. Un rifiuto esplicito in una qualsiasi di queste policy sostituisce l'autorizzazione. Per ulteriori informazioni sui limiti delle autorizzazioni, consulta Limiti delle autorizzazioni per le entità IAM nella Guida per l'utente di IAM.
- Politiche di controllo del servizio (SCPs): SCPs sono politiche JSON che specificano le autorizzazioni massime per un'organizzazione o un'unità organizzativa (OU) in. AWS Organizations AWS Organizations è un servizio per il raggruppamento e la gestione centralizzata di più AWS account di proprietà dell'azienda. Se abiliti tutte le funzionalità di un'organizzazione, puoi applicare le politiche di controllo del servizio (SCPs) a uno o tutti i tuoi account. L'SCP limita le autorizzazioni per le entità presenti negli account dei membri, inclusa ciascuna di esse. Utente root dell'account AWS Per ulteriori informazioni su Organizations and SCPs, consulta How SCPs work nella AWS Organizations User Guide.
- Policy di sessione - Le policy di sessione sono policy avanzate che vengono trasmesse come parametro quando si crea in modo programmatico una sessione temporanea per un ruolo o un utente federato. Le autorizzazioni della sessione risultante sono l'intersezione delle policy basate sull'identità del set di autorizzazioni o del ruolo e le policy di sessione. Le autorizzazioni possono anche provenire da una policy basata su risorse. Un rifiuto esplicito in una qualsiasi di queste policy sostituisce l'autorizzazione. Per ulteriori informazioni, consulta Policy di sessione nella Guida per l'utente IAM.

Più tipi di policy

Quando a una richiesta si applicano più tipi di policy, le autorizzazioni risultanti sono più complicate da comprendere. Per scoprire come si AWS determina se consentire o meno una richiesta quando sono coinvolti più tipi di policy, consulta Logica di valutazione delle policy nella IAM User Guide.

Funzionamento di Amazon Aurora con IAM

Prima di utilizzare IAM per gestire l'accesso ad Amazon Aurora, è necessario comprendere quali funzionalità IAM sono disponibili per l'uso con Amazon Aurora.

Nella seguente tabella sono elencate le funzionalità di IAM che è possibile utilizzare con Amazon Aurora:

Funzionalità IAM	Supporto di Amazon Aurora
Policy basate sull'identità	Sì
Policy basate su risorse	No
Operazioni di policy	Sì
Risorse relative alle policy	Sì
Chiavi di condizione della policy (specifica del servizio)	Sì
ACLs	No
Controllo degli accessi basato su attributi (ABAC) (tag nelle policy)	Sì
Credenziali temporanee	Sì
Inoltro delle sessioni di accesso	Sì
Ruoli di servizio	Sì
Ruoli collegati al servizio	Sì

Per avere una visione di alto livello di come Aurora e AWS altri servizi funzionano con IAM, AWS consulta i servizi che funzionano con IAM nella IAM User Guide.

Argomenti

- Policy basate su identità Aurora
- Policy basate su risorse all'interno di Aurora
- Operazioni delle policy per Aurora
- Risorse delle policy per Aurora
- Chiavi di condizione delle policy per Aurora
- Elenchi di controllo degli accessi (ACLs) in
- Controllo degli accessi basato su attributi (ABAC) nelle policy con tag Aurora
- Utilizzo di credenziali temporanee con Aurora
- Inoltro delle sessioni di accesso per Aurora
- Ruoli di servizio per Aurora
- Ruoli collegati ai servizi per Aurora

Policy basate su identità Aurora

Supporta le policy basate su identità: sì.

Le policy basate sull'identità sono documenti di policy di autorizzazione JSON che è possibile allegare a un'identità (utente, gruppo di utenti o ruolo IAM). Tali policy definiscono le operazioni che utenti e ruoli possono eseguire, su quali risorse e in quali condizioni. Per informazioni su come creare una policy basata su identità, consultare Definizione di autorizzazioni personalizzate IAM con policy gestite dal cliente nella Guida per l'utente IAM.

Con le policy basate sull'identità di IAM, è possibile specificare quali operazioni e risorse sono consentite o respinte, nonché le condizioni in base alle quali le operazioni sono consentite o respinte. Per informazioni su tutti gli elementi utilizzabili in una policy JSON, consulta Guida di riferimento agli elementi delle policy JSON IAM nella Guida per l'utente IAM.

Esempi di policy basate su identità per Aurora

Per visualizzare esempi di policy basate su identità Aurora, consulta Esempi di policy di Amazon Aurora basate su identità.

Policy basate su risorse all'interno di Aurora

Supporta le policy basate su risorse: no.

Le policy basate su risorse sono documenti di policy JSON che è possibile collegare a una risorsa. Esempi di policy basate sulle risorse sono le policy di attendibilità dei ruoli di IAM e le policy dei bucket di Amazon S3. Nei servizi che supportano policy basate sulle risorse, gli amministratori dei servizi possono utilizzarli per controllare l'accesso a una risorsa specifica. Quando è collegata a una risorsa, una policy definisce le operazioni che un principale può eseguire su tale risorsa e a quali condizioni. In una policy basata sulle risorse è obbligatorio specificare un'entità principale. I principali possono includere account, utenti, ruoli, utenti federati o. Servizi AWS

Per consentire l'accesso multi-account, è possibile specificare un intero account o entità IAM in un altro account come entità principale in una policy basata sulle risorse. Per ulteriori informazioni, consulta Accesso a risorse multi-account in IAM nella Guida per l'utente IAM.

Operazioni delle policy per Aurora

Supporta le azioni di policy: sì.

Gli amministratori possono utilizzare le policy AWS JSON per specificare chi ha accesso a cosa. In altre parole, quale entità principale può eseguire operazioni su quali risorse e in quali condizioni.

L'elemento `Action` di una policy JSON descrive le operazioni che è possibile utilizzare per consentire o negare l'accesso a un criterio. Includere le operazioni in una policy per concedere le autorizzazioni a eseguire l'operazione associata.

Le operazioni delle policy in Aurora utilizzano il seguente prefisso prima dell'operazione: `rds:`. Ad esempio, per concedere a qualcuno l'autorizzazione per descrivere istanze database con l'operazione API Amazon RDS `DescribeDBInstances`, includi l'operazione `rds:DescribeDBInstances` nella policy. Le istruzioni della policy devono includere un elemento `Action` o `NotAction`. Aurora definisce un proprio set di operazioni che descrivono le attività che puoi eseguire con quel servizio.

Per specificare più operazioni in una singola istruzione, separarle con una virgola come mostrato di seguito.

```
"Action": [
      "rds:action1",
      "rds:action2"
```

Puoi specificare più operazioni tramite caratteri jolly (*). Ad esempio, per specificare tutte le operazioni che iniziano con la parola `Describe`, includi la seguente operazione.

```
"Action": "rds:Describe*"
```

Per visualizzare un elenco di operazioni di Aurora, consulta Operazioni definite da Amazon RDS nella Service Authorization Reference.

Risorse delle policy per Aurora

Supporta le risorse di policy: sì.

Gli amministratori possono utilizzare le policy AWS JSON per specificare chi ha accesso a cosa. In altre parole, quale entità principale può eseguire operazioni su quali risorse e in quali condizioni.

L'elemento JSON `Resource` della policy specifica l'oggetto o gli oggetti ai quali si applica l'operazione. Come best practice, specificare una risorsa utilizzando il suo nome della risorsa Amazon (ARN). Per le azioni che non supportano le autorizzazioni a livello di risorsa, utilizzare un carattere jolly (*) per indicare che l'istruzione si applica a tutte le risorse.

```
"Resource": "*"
```

La risorsa di un'istanza database ha il seguente nome della risorsa Amazon (ARN).

```
arn:${Partition}:rds:${Region}:${Account}:{ResourceType}/${Resource}
```

Per ulteriori informazioni sul formato di ARNs, consulta Amazon Resource Names (ARNs) e AWS service namespaces.

Ad esempio, per specificare l'istanza database `dbtest` nell'istruzione, utilizza l'ARN riportato di seguito.

```
"Resource": "arn:aws:rds:us-west-2:123456789012:db:dbtest"
```

Per specificare tutte le istanze database che appartengono a un account specifico, utilizza il carattere jolly (*).

```
"Resource": "arn:aws:rds:us-east-1:123456789012:db:*"
```

Alcune operazioni API RDS, ad esempio quelle per la creazione di risorse, non possono essere eseguite su una risorsa specifica. In questi casi, utilizza il carattere jolly (*).

```
"Resource": "*"
```

Molte operazioni API di Amazon RDS coinvolgono più risorse. Ad esempio, `CreateDBInstance` crea un'istanza database. Puoi specificare che un utente deve utilizzare un gruppo specifico di sicurezza e un gruppo di parametri quando crea un'istanza database. Per specificare più risorse in una singola istruzione, separale con virgole. ARNs

```
"Resource": [
      "resource1",
      "resource2"
```

Per visualizzare un elenco dei tipi di risorse di Aurora e ARNs relativi, consulta Resources Defined by Amazon RDS nel Service Authorization Reference. Per informazioni sulle operazioni con cui è possibile specificare l'ARN di ogni risorsa, consulta Operazioni definite da Amazon RDS.

Chiavi di condizione delle policy per Aurora

Supporta le chiavi di condizione delle policy specifiche del servizio: sì.

Gli amministratori possono utilizzare le policy AWS JSON per specificare chi ha accesso a cosa. In altre parole, quale entità principale può eseguire operazioni su quali risorse e in quali condizioni.

L'elemento `Condition` specifica quando le istruzioni vengono eseguite in base a criteri definiti. È possibile compilare espressioni condizionali che utilizzano operatori di condizione, ad esempio uguale a o minore di, per soddisfare la condizione nella policy con i valori nella richiesta. Per visualizzare tutte le chiavi di condizione AWS globali, consulta le chiavi di contesto delle condizioni AWS globali nella Guida per l'utente IAM.

Aurora definisce il proprio set di chiavi di condizione e, inoltre, supporta l'uso di alcune chiavi di condizione globali. Per vedere tutte le chiavi di condizione AWS globali, consulta le chiavi di contesto delle condizioni AWS globali nella Guida per l'utente IAM.

Tutte le operazioni API RDS supportano la chiave di condizione `aws:RequestedRegion`.

Per visualizzare un elenco di chiavi di condizione Aurora, consulta Chiavi di condizione per Amazon RDS nella Service Authorization Reference. Per informazioni su operazioni e risorse con cui è possibile utilizzare una chiave di condizione, consulta Operazioni definite da Amazon RDS.

Elenchi di controllo degli accessi (ACLs) in

Supporta gli elenchi di controllo degli accessi (ACLs): No

Le liste di controllo degli accessi (ACLs) controllano quali principali (membri dell'account, utenti o ruoli) dispongono delle autorizzazioni per accedere a una risorsa. ACLs sono simili alle politiche basate sulle risorse, sebbene non utilizzino il formato del documento di policy JSON.

Controllo degli accessi basato su attributi (ABAC) nelle policy con tag Aurora

Supporta tag di controllo degli accessi basato su attributi (ABAC) nelle policy: sì.

Il controllo degli accessi basato su attributi (ABAC) è una strategia di autorizzazione che definisce le autorizzazioni in base agli attributi, chiamati tag. È possibile allegare tag a entità e AWS risorse IAM, quindi progettare politiche ABAC per consentire operazioni quando il tag del principale corrisponde al tag sulla risorsa.

Per controllare l'accesso basato su tag, fornire informazioni sui tag nell'elemento condizione di una policy utilizzando le chiavi di condizione `aws:ResourceTag/`*`key-name`*, `aws:RequestTag/`*`key-name`* o `aws:TagKeys`.

Se un servizio supporta tutte e tre le chiavi di condizione per ogni tipo di risorsa, il valore per il servizio è Sì. Se un servizio supporta tutte e tre le chiavi di condizione solo per alcuni tipi di risorsa, allora il valore sarà Parziale.

Per maggiori informazioni su ABAC, consultare Definizione delle autorizzazioni con autorizzazione ABAC nella Guida per l'utente IAM. Per visualizzare un tutorial con i passaggi per l'impostazione di ABAC, consultare Utilizzo del controllo degli accessi basato su attributi (ABAC) nella Guida per l'utente IAM.

Per ulteriori informazioni sul tagging delle risorse di Aurora, consulta Specifica delle condizioni: Utilizzo di tag personalizzati. Per visualizzare una policy basata sulle identità di esempio per limitare l'accesso a una risorsa basata su tag su tale risorsa, consulta Concedere l'autorizzazione per le operazioni su una risorsa con un tag specifico con due valori diversi.

Utilizzo di credenziali temporanee con Aurora

Supporta le credenziali temporanee: sì.

Le credenziali temporanee forniscono l'accesso a breve termine alle AWS risorse e vengono create automaticamente quando si utilizza la federazione o si cambia ruolo. AWS consiglia di generare

dinamicamente credenziali temporanee anziché utilizzare chiavi di accesso a lungo termine. Per ulteriori informazioni, consulta Credenziali di sicurezza temporanee in IAM e Servizi AWS compatibili con IAM nelle Guida per l'utente IAM.

Inoltro delle sessioni di accesso per Aurora

Supporta l'inoltro delle sessioni di accesso: sì.

Le sessioni di accesso inoltrato (FAS) utilizzano le autorizzazioni del principale che chiama e, in combinazione con la richiesta Servizio AWS, Servizio AWS per effettuare richieste ai servizi downstream. Per i dettagli delle policy relative alle richieste FAS, consultare Forward access sessions.

Ruoli di servizio per Aurora

Supporta i ruoli di servizio: sì.

Un ruolo di servizio è un ruolo IAM che un servizio assume per eseguire operazioni per tuo conto. Un amministratore IAM può creare, modificare ed eliminare un ruolo di servizio dall'interno di IAM. Per ulteriori informazioni, consulta la sezione Create a role to delegate permissions to an Servizio AWS nella Guida per l'utente IAM.

> **Warning**
>
> La modifica delle autorizzazioni per un ruolo di servizio potrebbe compromettere la funzionalità di Aurora. Modificare i ruoli di servizio solo quando Aurora fornisce le indicazioni per farlo.

Ruoli collegati ai servizi per Aurora

Supporta i ruoli collegati ai servizi: sì.

Un ruolo collegato al servizio è un tipo di ruolo di servizio collegato a un. Servizio AWS Il servizio può assumere il ruolo per eseguire un'azione per tuo conto. I ruoli collegati al servizio vengono visualizzati nel tuo account Account AWS e sono di proprietà del servizio. Un amministratore IAM può visualizzare le autorizzazioni per i ruoli collegati al servizio, ma non modificarle.

Per ulteriori informazioni sull'utilizzo di ruoli collegati ai servizi Aurora, consulta Utilizzo di ruoli collegati ai servizi per Amazon Aurora.

Esempi di policy di Amazon Aurora basate su identità

Per impostazione predefinita, i set di autorizzazioni e i ruoli non dispongono dell'autorizzazione per creare o modificare risorse Aurora. Inoltre, non possono eseguire attività utilizzando l' AWS API Console di gestione AWS AWS CLI, o. Un amministratore deve creare policy IAM che concedono a set di autorizzazioni e ruoli l'autorizzazione per eseguire operazioni API specifiche sulle risorse specifiche di cui hanno bisogno. L'amministratore deve quindi collegare queste policy ai set di autorizzazioni o ai ruoli che richiedono tali autorizzazioni.

Per informazioni su come creare una policy basata su identità IAM utilizzando questi documenti di policy JSON di esempio, consultare Creazione di policy nella scheda JSON nella Guida per l'utente di IAM.

Argomenti

- Best practice delle policy
- Utilizzo della console di Aurora
- Autorizzazioni necessarie per l'uso della console
- Consentire agli utenti di visualizzare le loro autorizzazioni
- Policy di autorizzazione per creare, modificare ed eliminare risorse in Aurora
- Policy di esempio: Utilizzo di chiavi di condizione
- Specifica delle condizioni: Utilizzo di tag personalizzati
- Concessione dell’autorizzazione all’applicazione di tag Aurora per le risorse durante la creazione

Best practice delle policy

Le policy basate su identità determinano se qualcuno può creare, accedere o eliminare risorse Amazon RDS nell'account. Queste operazioni possono comportare costi aggiuntivi per l’ Account AWS. Quando si creano o modificano policy basate sull’identità, seguire queste linee guida e raccomandazioni:

- Inizia con le policy AWS gestite e passa alle autorizzazioni con privilegi minimi: per iniziare a concedere autorizzazioni a utenti e carichi di lavoro, utilizza le politiche AWS gestite che concedono le autorizzazioni per molti casi d'uso comuni. Sono disponibili nel tuo. Account AWS Ti consigliamo di ridurre ulteriormente le autorizzazioni definendo politiche gestite dai AWS clienti specifiche per i tuoi casi d'uso. Per maggiori informazioni, consultare Policy gestite da AWS o Policy gestite da AWS per le funzioni dei processi nella Guida per l’utente IAM.

- Applicazione delle autorizzazioni con privilegio minimo - Quando si impostano le autorizzazioni con le policy IAM, concedere solo le autorizzazioni richieste per eseguire un'attività. È possibile farlo definendo le azioni che possono essere intraprese su risorse specifiche in condizioni specifiche, note anche come autorizzazioni con privilegio minimo. Per maggiori informazioni sull'utilizzo di IAM per applicare le autorizzazioni, consultare Policy e autorizzazioni in IAM nella Guida per l'utente IAM.
- Condizioni d'uso nelle policy IAM per limitare ulteriormente l'accesso - Per limitare l'accesso a operazioni e risorse è possibile aggiungere una condizione alle policy. Ad esempio, è possibile scrivere una condizione di policy per specificare che tutte le richieste devono essere inviate utilizzando SSL. Puoi anche utilizzare le condizioni per concedere l'accesso alle azioni del servizio se vengono utilizzate tramite uno specifico Servizio AWS, ad esempio CloudFormation. Per maggiori informazioni, consultare la sezione Elementi delle policy JSON di IAM: condizione nella Guida per l'utente IAM.
- Utilizzo di IAM Access Analyzer per convalidare le policy IAM e garantire autorizzazioni sicure e funzionali - IAM Access Analyzer convalida le policy nuove ed esistenti in modo che aderiscano alla sintassi della policy IAM (JSON) e alle best practice di IAM. IAM Access Analyzer offre oltre 100 controlli delle policy e consigli utili per creare policy sicure e funzionali. Per maggiori informazioni, consultare Convalida delle policy per il Sistema di analisi degli accessi IAM nella Guida per l'utente IAM.
- Richiedi l'autenticazione a più fattori (MFA): se hai uno scenario che richiede utenti IAM o un utente root nel Account AWS tuo, attiva l'MFA per una maggiore sicurezza. Per richiedere la MFA quando vengono chiamate le operazioni API, aggiungere le condizioni MFA alle policy. Per maggiori informazioni, consultare Protezione dell'accesso API con MFA nella Guida per l'utente IAM.

Per maggiori informazioni sulle best practice in IAM, consultare Best practice di sicurezza in IAM nella Guida per l'utente IAM.

Utilizzo della console di Aurora

Per accedere alla console Amazon Aurora, è necessario disporre di un set di autorizzazioni minimo. Queste autorizzazioni devono consentirti di elencare e visualizzare i dettagli sulle risorse Amazon Aurora presenti nel tuo. Account AWS Se si cra una policy basata sull'identità più restrittiva rispetto alle autorizzazioni minime richieste, la console non funzionerà nel modo previsto per le entità (utenti o ruoli) associate a tale policy.

Non è necessario consentire autorizzazioni minime per la console per gli utenti che effettuano chiamate solo verso o l' AWS CLI API. AWS Al contrario, è possibile accedere solo alle operazioni che soddisfano l'operazione API che stai cercando di eseguire.

Per garantire che tali entità possano ancora utilizzare la console Aurora, allega anche la AWS seguente policy gestita alle entità.

```
AmazonRDSReadOnlyAccess
```

Per maggiori informazioni, consultare Aggiunta di autorizzazioni a un utente nella Guida per l'utente IAM.

Autorizzazioni necessarie per l'uso della console

Affinché un utente possa utilizzare la console, è necessario che disponga di un set minimo di autorizzazioni. Queste autorizzazioni consentono all'utente di descrivere le risorse Aurora per il AWS proprio account e di fornire altre informazioni correlate, tra cui informazioni sulla sicurezza e sulla rete di EC2 Amazon.

Se decidi di creare una policy IAM più restrittiva delle autorizzazioni minime richieste, la console non funzionerà come previsto per gli utenti con tale policy IAM. Per garantire che gli utenti possano continuare a usare la console, collega anche la policy gestita `AmazonRDSReadOnlyAccess` all'utente, come descritto in Gestione dell'accesso tramite policy.

Non sono necessarie le autorizzazioni minime della console per gli utenti che effettuano chiamate solo a AWS CLI o all'API di Amazon RDS.

La seguente politica garantisce l'accesso completo a tutte le risorse Amazon Aurora per l'account root: AWS

```
AmazonRDSFullAccess
```

Consentire agli utenti di visualizzare le loro autorizzazioni

Questo esempio mostra in che modo è possibile creare una policy che consente agli utenti IAM di visualizzare le policy inline e gestite che sono cpllegate alla relativa identità utente. Questa politica include le autorizzazioni per completare questa azione sulla console o utilizzando l'API o a livello di codice. AWS CLI AWS

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "ViewOwnUserInfo",
            "Effect": "Allow",
            "Action": [
                "iam:GetUserPolicy",
                "iam:ListGroupsForUser",
                "iam:ListAttachedUserPolicies",
                "iam:ListUserPolicies",
                "iam:GetUser"
            ],
            "Resource": ["arn:aws:iam::*:user/${aws:username}"]
        },
        {
            "Sid": "NavigateInConsole",
            "Effect": "Allow",
            "Action": [
                "iam:GetGroupPolicy",
                "iam:GetPolicyVersion",
                "iam:GetPolicy",
                "iam:ListAttachedGroupPolicies",
                "iam:ListGroupPolicies",
                "iam:ListPolicyVersions",
                "iam:ListPolicies",
                "iam:ListUsers"
            ],
            "Resource": "*"
        }
    ]
}
```

Policy di autorizzazione per creare, modificare ed eliminare risorse in Aurora

Le seguenti sezioni presentano esempi di policy di autorizzazione che concedono e limitano l'accesso alle risorse:

Consenti a un utente di creare istanze DB in un account AWS

Di seguito è riportato un esempio di politica che consente all'account con l'ID di `123456789012` creare istanze DB per il tuo AWS account. La policy richiede che il nome della nuova istanza database inizi con `test`. La nuova istanza database deve anche utilizzare il motore del database

MySQL e la classe di istanza database `db.t2.micro`. Inoltre, la nuova istanza database deve utilizzare un gruppo di opzioni e un gruppo di parametri database che inizia con `default`, e deve utilizzare il gruppo di sottoreti `default`.

JSON

```
{
   "Version":"2012-10-17",
   "Statement": [
      {
         "Sid": "AllowCreateDBInstanceOnly",
         "Effect": "Allow",
         "Action": [
            "rds:CreateDBInstance"
         ],
         "Resource": [
            "arn:aws:rds:*:123456789012:db:test*",
            "arn:aws:rds:*:123456789012:og:default*",
            "arn:aws:rds:*:123456789012:pg:default*",
            "arn:aws:rds:*:123456789012:subgrp:default"
         ],
         "Condition": {
            "StringEquals": {
               "rds:DatabaseEngine": "mysql",
               "rds:DatabaseClass": "db.t2.micro"
            }
         }
      }
   ]
}
```

La policy include una singola istruzione che specifica le autorizzazioni seguenti per l'utente :

- La policy consente all'account di creare un'istanza DB utilizzando l'operazione Create DBInstance API (ciò vale anche per il create-db-instanceAWS CLIcomando e ilConsole di gestione AWS).
- L'elemento `Resource` specifica che l'utente può eseguire azioni in o con altre risorse. Specifica le risorse usando Amazon Resource Name (ARN). Questo ARN include il nome del servizio a cui appartiene la risorsa (`rds`), la AWS regione (*indica qualsiasi regione in questo esempio), il numero di AWS account (`123456789012`è il numero di account in questo esempio) e il tipo di

risorsa. Per ulteriori informazioni sulla creazione ARNs, vedereNome della risorsa Amazon (ARN) in Amazon RDS.

L'elemento `Resource` nell'esempio specifica i seguenti vincoli della policy sulle risorse per l'utente:

- L'identificatore istanze DB per la nuova istanza database deve iniziare con `test` (per esempio, `testCustomerData1`, `test-region2-data`).
- Il gruppo di opzioni per la nuova istanza database deve iniziare con `default`.
- Il gruppo di parametri database per la nuova istanza database deve iniziare con `default`.
- Il gruppo di sottoreti per la nuova istanza database deve essere il gruppo di sottoreti `default`.

- L'elemento `Condition` specifica che il motore database deve essere MySQL e la classe di istanza database deve essere `db.t2.micro`. L'elemento `Condition` specifica le condizioni quando deve essere applicata una policy. È possibile aggiungere permessi o restrizioni aggiuntivi usando l'elemento `Condition`. Per ulteriori informazioni su come specificare le condizioni;, consulta Chiavi di condizione delle policy per Aurora. Questo esempio specifica le condizioni `rds:DatabaseEngine` e `rds:DatabaseClass`. Per informazioni sui valori delle condizioni validi per`rds:DatabaseEngine`, consultate l'elenco sotto il `Engine` parametro in Create DBInstance. Per informazioni sui valori di condizione validi per `rds:DatabaseClass`, consulta Motori di database supportati per classi di istanza database.

La policy non specifica l'elemento `Principal` poiché in una policy basata su identità l'entità che ottiene l'autorizzazione non viene specificata. Quando si collega una policy a un utente, quest'ultimo è l'entità implicita. Quando si collega una policy di autorizzazione a un ruolo IAM;, l'entità identificata nella policy di attendibilità del ruolo ottiene le autorizzazioni.

Per visualizzare un elenco di operazioni di Aurora, consulta Operazioni definite da Amazon RDS nella Service Authorization Reference.

Consentire a un utente di eseguire qualsiasi operazione Describe in qualsiasi risorsa RDS

La seguente policy di autorizzazione concede a un utente le autorizzazioni per eseguire tutte le operazioni che iniziano con `Describe`. Queste operazioni riportano informazioni su una risorsa RDS, ad esempio un'istanza database. Il carattere jolly (*) nell'elemento `Resource` indica che le operazioni sono permesse per tutte le risorse Amazon Aurora di proprietà dell'account.

JSON

```
{
```

```
   "Version":"2012-10-17",
   "Statement": [
      {
         "Sid": "AllowRDSDescribe",
         "Effect": "Allow",
         "Action": "rds:Describe*",
         "Resource": "*"
      }
   ]
}
```

Consentire a un utente di creare un'istanza database che utilizza il gruppo parametri del database e il gruppo di sottorete specificati

La seguente policy di autorizzazioni concede le autorizzazioni per consentire a un utente di creare un'istanza database che deve usare il gruppo di parametri database `mydbpg` e il gruppo di sottorete DB `mydbsubnetgroup`.

JSON

```
{
   "Version":"2012-10-17",
   "Statement": [
      {
         "Sid": "VisualEditor0",
         "Effect": "Allow",
         "Action": "rds:CreateDBInstance",
         "Resource": [
            "arn:aws:rds:*:*:pg:mydbpg",
            "arn:aws:rds:*:*:subgrp:mydbsubnetgroup"
         ]
      }
   ]
}
```

Concedere l'autorizzazione per le operazioni su una risorsa con un tag specifico con due valori diversi

Puoi utilizzare le condizioni nella policy basata sulle identità per controllare l'accesso alle risorse di Aurora in base ai tag. La seguente policy concede l'autorizzazione per eseguire l'operazione API `CreateDBSnapshot` sulle istanze database con il tag `stage` impostato su `development` o `test`.

JSON

```
{
   "Version":"2012-10-17",
   "Statement":[
      {
         "Sid":"AllowAnySnapshotName",
         "Effect":"Allow",
         "Action":[
            "rds:CreateDBSnapshot"
         ],
         "Resource":"arn:aws:rds:*:123456789012:snapshot:*"
      },
      {
         "Sid":"AllowDevTestToCreateSnapshot",
         "Effect":"Allow",
         "Action":[
            "rds:CreateDBSnapshot"
         ],
         "Resource":"arn:aws:rds:*:123456789012:db:*",
         "Condition":{
            "StringEquals":{
                "rds:db-tag/stage":[
                  "development",
                  "test"
               ]
            }
         }
      }
   ]
}
```

La seguente policy concede l'autorizzazione per eseguire l'operazione API `ModifyDBInstance` sulle istanze database con il tag `stage` impostato su `development` o `test`.

JSON

```
{
   "Version":"2012-10-17",
   "Statement":[
      {
         "Sid":"AllowChangingParameterOptionSecurityGroups",
         "Effect":"Allow",
         "Action":[
            "rds:ModifyDBInstance"
         ],
         "Resource": [
            "arn:aws:rds:*:123456789012:pg:*",
            "arn:aws:rds:*:123456789012:secgrp:*",
            "arn:aws:rds:*:123456789012:og:*"
         ]
      },
      {
         "Sid":"AllowDevTestToModifyInstance",
         "Effect":"Allow",
         "Action":[
            "rds:ModifyDBInstance"
         ],
         "Resource":"arn:aws:rds:*:123456789012:db:*",
         "Condition":{
            "StringEquals":{
                "rds:db-tag/stage":[
                  "development",
                  "test"
               ]
            }
         }
      }
   ]
}
```

Impedire a un utente di eliminare un'istanza database

La seguente policy di autorizzazione assegna le autorizzazioni per impedire a un utente di eliminare un'istanza database specifica. Ad esempio, potresti voler negare la possibilità di eliminare le istanze database di produzione a qualsiasi utente che non sia un amministratore.

JSON

```
{
   "Version":"2012-10-17",
   "Statement": [
      {
         "Sid": "DenyDelete1",
         "Effect": "Deny",
         "Action": "rds:DeleteDBInstance",
         "Resource": "arn:aws:rds:us-west-2:123456789012:db:my-mysql-instance"
      }
   ]
}
```

Negare tutti gli accessi a una risorsa

È anche possibile negare esplicitamente l'accesso a una risorsa. I criteri di negazione hanno la precedenza sui criteri di autorizzazione. La policy seguente nega esplicitamente a un utente la possibilità di gestire una risorsa:

JSON

```
{
   "Version":"2012-10-17",
   "Statement": [
      {
         "Effect": "Deny",
         "Action": "rds:*",
         "Resource": "arn:aws:rds:us-east-1:123456789012:db:mydb"
      }
   ]
}
```

Policy di esempio: Utilizzo di chiavi di condizione

Di seguito sono illustrati esempi di come è possibile utilizzare le chiavi di condizione nelle policy di autorizzazioni IAM di Amazon Aurora.

Esempio 1: Concessione dell'autorizzazione per creare un'istanza database che utilizza un motore del database specifico e non è Multi-AZ

La policy seguente utilizza una chiave di condizione RDS e permette a un utente di creare solo istanze database che utilizzano il motore del database MySQL e non utilizzano Multi-AZ. L'elemento `Condition` indica il requisito secondo cui il motore del database deve essere MySQL.

JSON

```
{
   "Version":"2012-10-17",
   "Statement": [
      {
         "Sid": "AllowMySQLCreate",
         "Effect": "Allow",
         "Action": "rds:CreateDBInstance",
         "Resource": "*",
         "Condition": {
            "StringEquals": {
               "rds:DatabaseEngine": "mysql"
            },
            "Bool": {
               "rds:MultiAz": false
            }
         }
      }
   ]
}
```

Esempio 2: Rifiuto esplicito dell'autorizzazione per creare istanze database per determinate classi di istanze database e per creare istanze database che utilizzano Provisioned IOPS

La policy seguente rifiuta in modo esplicito l'autorizzazione per creare istanze database che utilizzano le classi di istanze database `r3.8xlarge` e `m4.10xlarge`, che sono le istanze database di

dimensioni maggiori e più costose. Questa policy impedisce anche gli utenti di creare istanze database che utilizzano Provisioned IOPS, che comporta costi aggiuntivi.

Il rifiuto esplicito di un'autorizzazione prevale su qualsiasi altra autorizzazione concessa. In questo modo si evita che le identità ottengano accidentalmente un'autorizzazione che non desideravi concedere.

JSON

```
{
   "Version":"2012-10-17",
   "Statement": [
      {
         "Sid": "DenyLargeCreate",
         "Effect": "Deny",
         "Action": "rds:CreateDBInstance",
         "Resource": "*",
         "Condition": {
            "StringEquals": {
               "rds:DatabaseClass": [
                  "db.r3.8xlarge",
                  "db.m4.10xlarge"
               ]
            }
         }
      },
      {
         "Sid": "DenyPIOPSCreate",
         "Effect": "Deny",
         "Action": "rds:CreateDBInstance",
         "Resource": "*",
         "Condition": {
            "NumericNotEquals": {
               "rds:Piops": "0"
            }
         }
      }
   ]
}
```

Esempio 3: limita il set di chiavi di tag e valori che possono essere utilizzati per aggiungere tag a una risorsa

La policy seguente utilizza una chiave di condizione RDS che consente l'aggiunta di un tag con la chiave `stage` per essere aggiunta alla risorsa con i valori `test`, `qa` e `production`.

JSON

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Sid": "AllowTagEdits",
      "Effect": "Allow",
      "Action": [
        "rds:AddTagsToResource",
        "rds:RemoveTagsFromResource"
      ],
      "Resource": "arn:aws:rds:us-east-1:123456789012:db:db-123456",
      "Condition": {
        "StringEquals": {
          "rds:req-tag/stage": [
            "test",
            "qa",
            "production"
          ]
        }
      }
    }
  ]
}
```

Specifica delle condizioni: Utilizzo di tag personalizzati

Amazon Aurora permette di specificare condizioni in una policy IAM utilizzando tag personalizzati.

Ad esempio, supponi di aggiungere alle istanze database un tag denominato `environment` con valori quali `beta`, `staging`, `production` e così via. In questo modo, puoi creare una policy che limita alcuni utenti alle istanze database in base al valore del tag `environment`.

> **Note**
>
> Per gli identificatori di tag personalizzati viene fatta distinzione tra maiuscole e minuscole.

Nella tabella seguente sono elencati gli identificatori di tag RDS che è possibile utilizzare in un elemento `Condition`.

Identificatore di tag RDS	Si applica a
`db-tag`	Istanze database, incluse repliche di lettura
`snapshot-tag`	Snapshot DB
`ri-tag`	Istanze database riservate
`og-tag`	Gruppi di opzioni database
`pg-tag`	Gruppi di parametri database
`subgrp-tag`	Gruppi di sottoreti database
`es-tag`	Abbonamenti a eventi
`cluster-tag`	Cluster database
`cluster-pg-tag`	Gruppi di parametri di cluster database
`cluster-snapshot-tag`	Snapshot cluster database

La sintassi per una condizione di un tag personalizzato è la seguente:

```
"Condition":{"StringEquals":{"rds:rds-tag-identifier/tag-name":
["value"]} }
```

Ad esempio, l'elemento `Condition` seguente si applica alle istanze database con un tag denominato `environment` e un valore di tag corrispondente a `production`.

```
"Condition":{"StringEquals":{"rds:db-tag/environment": ["production"]} }
```

Per informazioni sulla creazione di tag, consulta Applicazione di tag alle risorse Amazon Aurora e Amazon RDS.

Important

Se gestisci l'accesso alle risorse RDS tramite tagging, è consigliabile proteggere l'accesso ai tag per le risorse RDS. È possibile gestire l'accesso ai tag creando policy per le operazioni `AddTagsToResource` e `RemoveTagsFromResource`. Ad esempio, la policy seguente nega agli utenti la possibilità di aggiungere o rimuovere tag per tutte le risorse. Puoi quindi creare policy per permettere a utenti specifici di aggiungere o rimuovere tag.

JSON

```
{
   "Version":"2012-10-17",
   "Statement":[
      {
         "Sid":"DenyTagUpdates",
         "Effect":"Deny",
         "Action":[
            "rds:AddTagsToResource",
            "rds:RemoveTagsFromResource"
         ],
         "Resource":"*"
      }
   ]
}
```

Per visualizzare un elenco di operazioni di Aurora, consulta Operazioni definite da Amazon RDS nella Service Authorization Reference.

Policy di esempio: Utilizzo di tag personalizzati

Di seguito sono illustrati esempi di come è possibile utilizzare i tag personalizzati nelle policy di autorizzazioni IAM di Amazon Aurora. Per ulteriori informazioni sull'aggiunta di tag a una risorsa Amazon Aurora, consulta Nome della risorsa Amazon (ARN) in Amazon RDS.

> **Note**
>
> Tutti gli esempi utilizzano la regione us-west-2 e contengono account fittizi. IDs

Esempio 1: Concessione dell'autorizzazione per le operazioni su una risorsa con un tag specifico con due valori diversi

La seguente policy concede l'autorizzazione per eseguire l'operazione API `CreateDBSnapshot` sulle istanze database con il tag `stage` impostato su `development` o `test`.

JSON

```
{
   "Version":"2012-10-17",
   "Statement":[
      {
         "Sid":"AllowAnySnapshotName",
         "Effect":"Allow",
         "Action":[
            "rds:CreateDBSnapshot"
         ],
         "Resource":"arn:aws:rds:*:123456789012:snapshot:*"
      },
      {
         "Sid":"AllowDevTestToCreateSnapshot",
         "Effect":"Allow",
         "Action":[
            "rds:CreateDBSnapshot"
         ],
         "Resource":"arn:aws:rds:*:123456789012:db:*",
         "Condition":{
            "StringEquals":{
                "rds:db-tag/stage":[
                  "development",
                  "test"
               ]
            }
         }
      }
   ]
```

```
}
```

La seguente policy concede l'autorizzazione per eseguire l'operazione API `ModifyDBInstance` sulle istanze database con il tag `stage` impostato su `development` o `test`.

JSON

```
{
   "Version":"2012-10-17",
   "Statement":[
      {
         "Sid":"AllowChangingParameterOptionSecurityGroups",
         "Effect":"Allow",
         "Action":[
            "rds:ModifyDBInstance"
            ],
          "Resource": [
            "arn:aws:rds:*:123456789012:pg:*",
            "arn:aws:rds:*:123456789012:secgrp:*",
            "arn:aws:rds:*:123456789012:og:*"
            ]
       },
       {
         "Sid":"AllowDevTestToModifyInstance",
         "Effect":"Allow",
         "Action":[
            "rds:ModifyDBInstance"
            ],
         "Resource":"arn:aws:rds:*:123456789012:db:*",
         "Condition":{
            "StringEquals":{
               "rds:db-tag/stage":[
                  "development",
                  "test"
                  ]
               }
            }
       }
    ]
}
```

Esempio 2: Rifiuto esplicito dell'autorizzazione per creare un'istanza database che utilizza gruppi di parametri database specificati

La policy seguente rifiuta in modo esplicito l'autorizzazione per creare un'istanza database che utilizza gruppi di parametri database con valori di tag specifici. Puoi applicare questa policy se desideri che uno specifico gruppo di parametri database creato dal cliente venga sempre utilizzato nella creazione di istanze database. Le policy che utilizzano `Deny` servono per lo più a limitare l'accesso concesso da una policy più ampia.

Il rifiuto esplicito di un'autorizzazione prevale su qualsiasi altra autorizzazione concessa. In questo modo si evita che le identità ottengano accidentalmente un'autorizzazione che non desideravi concedere.

JSON

```
{
   "Version":"2012-10-17",
   "Statement":[
      {
         "Sid":"DenyProductionCreate",
         "Effect":"Deny",
         "Action":"rds:CreateDBInstance",
         "Resource":"arn:aws:rds:*:123456789012:pg:*",
         "Condition":{
            "StringEquals":{
               "rds:pg-tag/usage":"prod"
            }
         }
      }
   ]
}
```

Esempio 3: Concessione dell'autorizzazione per le operazioni in un'istanza database con un nome di istanza che ha come prefisso un nome utente

La policy seguente concede l'autorizzazione per chiamare qualsiasi API (ad eccezione di `AddTagsToResource` o `RemoveTagsFromResource`) in un'istanza database con un nome di istanza che ha come prefisso il nome dell'utente e che dispone di un tag denominato `stage` equivalente a `devo` o che non dispone di un tag `stage`.

La riga `Resource` nella policy identifica una risorsa in base al relativo Amazon Resource Name (ARN). Per ulteriori informazioni sull'utilizzo ARNs con le risorse Amazon Aurora, consulta. Nome della risorsa Amazon (ARN) in Amazon RDS

JSON

```
{
   "Version":"2012-10-17",
   "Statement":[
      {
         "Sid":"AllowFullDevAccessNoTags",
         "Effect":"Allow",
         "NotAction":[
            "rds:AddTagsToResource",
            "rds:RemoveTagsFromResource"
         ],
         "Resource":"arn:aws:rds:*:123456789012:db:${aws:username}*",
         "Condition":{
            "StringEqualsIfExists":{
               "rds:db-tag/stage":"devo"
            }
         }
      }
   ]
}
```

Concessione dell’autorizzazione all’applicazione di tag Aurora per le risorse durante la creazione

Alcune operazioni dell’API RDS consentono di specificare tag al momento della creazione delle risorse. È possibile utilizzare i tag delle risorse per implementare il controllo basato sugli attributi (ABAC). Per ulteriori informazioni, consulta A cosa serve ABAC? AWS e Controllo dell'accesso alle AWS risorse tramite tag.

Per essere abilitati a taggare le risorse in fase di creazione, gli utenti devono disporre delle autorizzazioni per utilizzare l’azione che crea la risorsa, ad esempio `rds:CreateDBCluster`. Se i tag vengono specificati nell’azione di creazione, RDS implementa autorizzazioni aggiuntive per l’azione `rds:AddTagsToResource` al fine di verificare se gli utenti dispongono delle autorizzazioni

per creare tag. Pertanto, gli utenti devono disporre anche di autorizzazioni esplicite a utilizzare l'azione `rds:AddTagsToResource`.

Nella definizione della policy IAM per l'azione `rds:AddTagsToResource`, è possibile utilizzare la chiave di condizione `aws:RequestTag` per richiedere tag in una richiesta di applicare tag a una risorsa.

Ad esempio, la seguente policy consente agli utenti di creare istanze database e applicare tag durante la creazione di un'istanza database, ma solo con chiavi di tag specifiche (`environment` o `project`):

JSON

```
{
   "Version":"2012-10-17",
   "Statement": [
       {
           "Effect": "Allow",
           "Action": [
               "rds:CreateDBInstance"
           ],
           "Resource": "*"
       },
       {
           "Effect": "Allow",
           "Action": [
               "rds:AddTagsToResource"
           ],
           "Resource": "*",
           "Condition": {
               "StringEquals": {
                   "aws:RequestTag/environment": ["production", "development"],
                   "aws:RequestTag/project": ["dataanalytics", "webapp"]
               },
               "ForAllValues:StringEquals": {
                   "aws:TagKeys": ["environment", "project"]
               }
           }
       }
   ]
}
```

Questa policy respinge qualsiasi richiesta di creazione di istanza database che includa tag diversi dai tag `environment` o `project` oppure che non specifichi nessuno di questi tag. Inoltre, gli utenti devono specificare i valori per i tag che corrispondono ai valori consentiti nella policy.

La seguente policy consente gli utenti di creare cluster di database e applicare tag durante la creazione ad eccezione del tag `environment=prod`:

JSON

```
{
   "Version":"2012-10-17",
   "Statement": [
       {
           "Effect": "Allow",
           "Action": [
               "rds:CreateDBCluster"
           ],
           "Resource": "*"
       },
       {
           "Effect": "Allow",
           "Action": [
               "rds:AddTagsToResource"
           ],
           "Resource": "*",
           "Condition": {
               "StringNotEquals": {
                   "aws:RequestTag/environment": "prod"
               }
           }
       }
   ]
}
```

Azioni API RDS supportate per l’applicazione di tag in fase di creazione

Le seguenti azioni dell’API RDS supportano l’applicazione di tag durante la creazione di una risorsa. Per queste azioni, puoi specificare i tag al momento della creazione della risorsa:

- `CreateBlueGreenDeployment`

- CreateCustomDBEngineVersion
- CreateDBCluster
- CreateDBClusterEndpoint
- CreateDBClusterParameterGroup
- CreateDBClusterSnapshot
- CreateDBInstance
- CreateDBInstanceReadReplica
- CreateDBParameterGroup
- CreateDBProxy
- CreateDBProxyEndpoint
- CreateDBSecurityGroup
- CreateDBShardGroup
- CreateDBSnapshot
- CreateDBSubnetGroup
- CreateEventSubscription
- CreateGlobalCluster
- CreateIntegration
- CreateOptionGroup
- CreateTenantDatabase
- CopyDBClusterParameterGroup
- CopyDBClusterSnapshot
- CopyDBParameterGroup
- CopyDBSnapshot
- CopyOptionGroup
- RestoreDBClusterFromS3
- RestoreDBClusterFromSnapshot
- RestoreDBClusterToPointInTime
- RestoreDBInstanceFromDBSnapshot
- RestoreDBInstanceFromS3
- RestoreDBInstanceToPointInTime

- `PurchaseReservedDBInstancesOffering`

Se si utilizza l'API AWS CLI or per creare una risorsa con tag, il `Tags` parametro viene utilizzato per applicare i tag alle risorse durante la creazione.

Per queste azioni dell'API, se l'applicazione di tag non riesce, la risorsa non viene creata, la richiesta ha esito negativo e viene restituito un errore. Questo garantisce che le risorse vengano create con i tag o che non vengano create affatto, impedendo così la creazione di risorse senza i tag previsti.

Policy gestite da AWS per Amazon RDS

Per aggiungere le autorizzazioni a set di autorizzazioni e ruoli, è più semplice utilizzare le policy gestite da AWS che scrivere autonomamente le policy. La creazione di policy gestite dai clienti IAM che forniscono al tuo team solo le autorizzazioni di cui ha bisogno richiede tempo e competenza. Per iniziare rapidamente, utilizza le nostre policy gestite da AWS. Queste policy coprono i casi d'uso comuni e sono disponibili nel tuo Account AWS. Per ulteriori informazioni sulle policy gestite da AWS, consulta Policy gestite da AWS nella Guida per l'utente di IAM.

Servizi AWS mantengono e aggiornano le policy gestite da AWS. Non è possibile modificare le autorizzazioni nelle policy gestite da AWS. I servizi occasionalmente aggiungono altre autorizzazioni a una policy gestita da AWS per supportare nuove funzionalità. Questo tipo di aggiornamento interessa tutte le identità (set di autorizzazioni e ruoli) a cui è collegata la policy. È più probabile che i servizi aggiornino una policy gestita da AWS quando viene avviata una nuova funzionalità o quando diventano disponibili nuove operazioni. I servizi non rimuovono le autorizzazioni da una policy gestita da AWS, pertanto gli aggiornamenti delle policy non interrompono le autorizzazioni esistenti.

Inoltre, AWS supporta policy gestite per le funzioni di processi che coprono più servizi. Ad esempio, la policy `ReadOnlyAccess` gestita da AWS fornisce l'accesso in sola lettura sia ad Servizi AWS che a tutte le risorse. Quando un servizio avvia una nuova funzionalità, AWS aggiunge autorizzazioni di sola lettura per nuove operazioni e risorse. Per l'elenco e la descrizione delle policy di funzione dei processi, consulta la sezione Policy gestite da AWS per funzioni di processi nella Guida per l'utente di IAM.

Argomenti

- Policy gestita AWS: AmazonRDSReadOnlyAccess
- Policy gestita AWS: AmazonRDSFullAccess
- Policy gestita AWS: AmazonRDSDataFullAccess
- Policy gestita AWS: AmazonRDSEnhancedMonitoringRole
- Policy gestita AWS: AmazonRDSPerformanceInsightsReadOnly
- Policy gestita da AWS: AmazonRDSPerformanceInsightsFullAccess
- Policy gestita AWS: AmazonRDSDirectoryServiceAccess
- Policy gestita da AWS: AmazonRDSServiceRolePolicy
- Policy gestita da AWS: AmazonRDSPreviewServiceRolePolicy
- Politicy gestita da AWS: AmazonRDSBetaServiceRolePolicy

Policy gestita AWS: AmazonRDSReadOnlyAccess

Questa policy consente l'accesso in sola lettura ad Amazon RDS tramite la Console di gestione AWS.

Dettagli dell'autorizzazione

Questa policy include le seguenti autorizzazioni:

- `rds`: consente ai principali di descrivere le risorse Amazon RDS e di elencare i tag per le risorse Amazon RDS.
- `cloudwatch`: consente ai principali di ottenere statistiche relative ai parametri di Amazon CloudWatch.
- `ec2`: consente ai principali di descrivere le zone di disponibilità e le risorse di rete.
- `logs`: consente ai principali di descrivere i flussi di log di CloudWatch Logs per i gruppi di registri e di ottenere registri eventi di CloudWatch Logs.
- `devops-guru`: consente ai principali di descrivere le risorse che hanno la copertura di Amazon DevOps Guru, specificata dai nomi di stack o dai tag di risorse di CloudFormation.

Per ulteriori informazioni su questa policy, incluso il documento sulla policy JSON, consulta AmazonRDSReadOnlyAccess nella Guida di riferimento sulle policy gestite da AWS.

Policy gestita AWS: AmazonRDSFullAccess

Questa policy fornisce accesso completo ad Amazon RDS tramite Console di gestione AWS.

Dettagli dell'autorizzazione

Questa policy include le seguenti autorizzazioni:

- `rds`: consente ai principali l'accesso completo ad Amazon RDS.
- `application-autoscaling`: consente ai principali di descrivere e gestire gli obiettivi e le policy di dimensionamento di Application Auto Scaling.
- `cloudwatch`: consente ai principali di ottenere statistiche relative ai parametri di CloudWatch e di gestire gli allarmi di CloudWatch.
- `ec2`: consente ai principali di descrivere le zone di disponibilità e le risorse di rete.
- `logs`: consente ai principali di descrivere i flussi di log di CloudWatch Logs per i gruppi di registri e di ottenere registri eventi di CloudWatch Logs.

- `outposts`: consente ai principali di ottenere i tipi di istanza AWS Outposts.
- `pi`: consente ai principali di ottenere i parametri di Performance Insights.
- `sns`: consente ai principali di accedere a iscrizioni e argomenti Amazon Simple Notification Service (Amazon SNS) e di pubblicare messaggi Amazon SNS.
- `devops-guru`: consente ai principali di descrivere le risorse che hanno la copertura di Amazon DevOps Guru, specificata dai nomi di stack o dai tag di risorse di CloudFormation.

Per ulteriori informazioni su questa policy, incluso il documento sulla policy JSON, consulta AmazonRDSFullAccess nella Guida di riferimento sulle policy gestite da AWS.

Policy gestita AWS: AmazonRDSDataFullAccess

Questa policy consente l'accesso completo all'utilizzo dell'API dati e dell'editor di query su cluster Aurora Serverless in uno specifico Account AWS. Questa policy consente all'Account AWS di ottenere il valore di un segreto da Gestione dei segreti AWS.

È possibile allegare la policy `AmazonRDSDataFullAccess` alle identità IAM.

Dettagli dell'autorizzazione

Questa policy include le seguenti autorizzazioni:

- `dbqms`: consente ai principali di accedere, creare, eliminare, descrivere e aggiornare le query. Il Database Query Metadata Service (`dbqms`) è un servizio solo interno. Fornisce le query recenti e salvate per l'editor di query nella Console di gestione AWS per più Servizi AWS, tra cui Amazon RDS.
- `rds-data`: consente ai principali di eseguire istruzioni SQL su database Aurora Serverless.
- `secretsmanager`: consente ai principali di ottenere il valore di un segreto da Gestione dei segreti AWS.

Per ulteriori informazioni su questa policy, incluso il documento sulla policy JSON, consulta AmazonRDSDataFullAccess nella Guida di riferimento sulle policy gestite da AWS.

Policy gestita AWS: AmazonRDSEnhancedMonitoringRole

Questa policy fornisce l'accesso ad Amazon CloudWatch Logs per l'utilizzo della funzionalità di monitoraggio avanzato di Amazon RDS.

Dettagli dell'autorizzazione

Questa policy include le seguenti autorizzazioni:

- `logs`: consente ai principali di creare gruppi di registri e policy di conservazione di CloudWatch Logs e di creare e descrivere i flussi di log di CloudWatch Logs per tali gruppi. Consente inoltre ai principali di inserire e ottenere registri eventi di CloudWatch Logs.

Per ulteriori informazioni su questa policy, incluso il documento sulla policy JSON, consulta AmazonRDSEnhancedMonitoringRole nella Guida di riferimento sulle policy gestite da AWS.

Policy gestita AWS: AmazonRDSPerformanceInsightsReadOnly

Questa policy fornisce accesso in sola lettura alle istanze Amazon RDS Performance Insights per istanze database Amazon RDS e cluster di database Amazon Aurora.

Questa policy ora include `Sid` (ID istruzione) come identificativo per l'istruzione della policy.

Dettagli dell'autorizzazione

Questa policy include le seguenti autorizzazioni:

- `rds`: consente ai principali di descrivere le istanze database Amazon RDS e i cluster di database Amazon Aurora.
- `pi`: consente ai principali di effettuare chiamate all'API Amazon RDS Performance Insights e accedere ai parametri di Performance Insights.

Per ulteriori informazioni su questa policy, incluso il documento sulla policy JSON, consulta AmazonRDSPerformanceInsightsReadOnly nella Guida di riferimento sulle policy gestite da AWS.

Policy gestita da AWS: AmazonRDSPerformanceInsightsFullAccess

Questa policy fornisce l'accesso completo ad Approfondimenti sulle prestazioni di Amazon RDS per istanze database Amazon RDS e cluster database Amazon Aurora.

Questa policy ora include `Sid` (ID istruzione) come identificativo per l'istruzione della policy.

Dettagli dell'autorizzazione

Questa policy include le seguenti autorizzazioni:

- `rds`: consente ai principali di descrivere le istanze database Amazon RDS e i cluster di database Amazon Aurora.
- `pi`: consente ai principali di effettuare chiamate all'API Approfondimenti sulle prestazioni di Amazon RDS e di creare, visualizzare ed eliminare report di analisi delle prestazioni.
- `cloudwatch`: consente ai principali di elencare i parametri di Amazon CloudWatch e di ottenere dati dei parametri e statistiche.

Per ulteriori informazioni su questa policy, incluso il documento sulla policy JSON, consulta AmazonRDSPerformanceInsightsFullAccess nella Guida di riferimento sulle policy gestite da AWS.

Policy gestita AWS: AmazonRDSDirectoryServiceAccess

Questa policy permette ad Amazon RDS di effettuare chiamate alla Directory Service.

Dettagli dell'autorizzazione

Questa policy include la seguente autorizzazione:

- `ds`: consente ai principali di descrivere le directory Directory Service e le autorizzazioni di controllo alle directory Directory Service.

Per ulteriori informazioni su questa policy, incluso il documento sulla policy JSON, consulta AmazonRDSDirectoryServiceAccess nella Guida di riferimento sulle policy gestite da AWS.

Policy gestita da AWS: AmazonRDSServiceRolePolicy

Non è possibile allegare la policy `AmazonRDSServiceRolePolicy` alle entità IAM. Questa policy è allegata a un ruolo collegato ai servizi che consente ad Amazon RDS di eseguire operazioni per conto dell'utente. Per ulteriori informazioni, consulta Autorizzazioni del ruolo collegato ai servizi per Amazon Aurora.

Policy gestita da AWS: AmazonRDSPreviewServiceRolePolicy

Non bisogna collegare `AmazonRDSPreviewServiceRolePolicy` alle entità IAM. Questa policy è allegata a un ruolo collegato ai servizi che consente ad Amazon RDS di effettuare chiamate AWS per conto delle risorse database RDS. Per ulteriori informazioni, consulta Ruolo collegato ai servizi per Amazon RDS Preview.

Dettagli dell'autorizzazione

Questa policy include le seguenti autorizzazioni:

- `ec2`: consente ai principali di descrivere le zone di disponibilità e le risorse di rete.
- `secretsmanager`: consente ai principali di ottenere il valore di un segreto da Gestione dei segreti AWS.
- `cloudwatch`, `logs`: consente ad Amazon RDS di caricare le metriche e i log delle istanze database su CloudWatch tramite l'agente CloudWatch.

Per ulteriori informazioni su questa policy, incluso il documento sulla policy JSON, consulta AmazonRDSPreviewServiceRolePolicy nella Guida di riferimento sulle policy gestite da AWS.

Politicy gestita da AWS: AmazonRDSBetaServiceRolePolicy

Non bisogna collegare `AmazonRDSBetaServiceRolePolicy` alle entità IAM. Questa policy è allegata a un ruolo collegato ai servizi che consente ad Amazon RDS di effettuare chiamate AWS per conto delle risorse database RDS. Per ulteriori informazioni, consulta Autorizzazioni del ruolo collegato ai servizi per Amazon RDS Beta.

Dettagli dell'autorizzazione

Questa policy include le seguenti autorizzazioni:

- `ec2`: consente ad Amazon RDS di eseguire operazioni di backup sull'istanza database che fornisce funzionalità di ripristino point-in-time.
- `secretsmanager`: consente ad Amazon RDS di gestire i segreti specifici delle istanze database creati da Amazon RDS.
- `cloudwatch`, `logs`: consente ad Amazon RDS di caricare le metriche e i log delle istanze database su CloudWatch tramite l'agente CloudWatch.

Per ulteriori informazioni su questa policy, incluso il documento sulla policy JSON, consulta AmazonRDSBetaServiceRolePolicy nella Guida di riferimento sulle policy gestite da AWS.

Aggiornamenti di Amazon RDS sulle policy gestite da AWS

Visualizza i dettagli sugli aggiornamenti alle policy gestite da AWS per Amazon RDS da quando questo servizio ha iniziato a tenere traccia delle modifiche. Per gli avvisi automatici sulle modifiche apportate a questa pagina, sottoscrivere il feed RSS nella pagina di Cronologia dei documenti di Amazon RDS.

Modifica	Descrizione	Data
Policy gestita da AWS: AmazonRDSPreviewServiceRolePolicy: aggiornamento a policy esistente	Amazon RDS ha rimosso l'autorizzazione `sns:Publish` dalla policy `AmazonRDSPreviewServiceRolePolicy` del ruolo collegato al servizio `AWSServiceRoleForRDSPreview` . Per ulteriori informazioni, consulta Policy gestita da AWS: AmazonRDSPreviewServiceRolePolicy.	7 agosto 2024
Politicy gestita da AWS: AmazonRDSBetaServiceRolePolicy: aggiornamento a policy esistente	Amazon RDS ha rimosso l'autorizzazione `sns:Publish` dalla policy `AmazonRDSBetaServiceRolePolicy` del ruolo collegato al servizio `AWSServiceRoleForRDSBeta` . Per ulteriori informazioni, consulta Politicy gestita da AWS: AmazonRDSBetaServiceRolePolicy.	7 agosto 2024
Policy gestita da AWS: AmazonRDSServiceRolePolicy: aggiornamento a policy esistente	Amazon RDS ha rimosso l'autorizzazione `sns:Publish` dalla policy `AmazonRDSServiceRolePolicy` del	2 luglio 2024

Modifica	Descrizione	Data
	ruolo collegato al servizio `AWSServiceRoleForRDS` . Per ulteriori informazioni, consulta Policy gestita da AWS: AmazonRDSServiceRolePolicy.	
Policy gestite da AWS per Amazon RDS: aggiornamento a policy esistente	Amazon RDS ha aggiunto una nuova autorizzazione alla policy `AmazonRDSCustomServiceRolePolicy` del ruolo collegato al servizio `AWSServiceRoleForRDSCustom` per consentire a RDS Custom per SQL Server di modificare il tipo di istanza host del database sottostante. RDS ha inoltre aggiunto l'autorizzazione `ec2:DescribeInstanceTypes` per ottenere informazioni sul tipo di istanza per l'host del database. Per ulteriori informazioni, consulta Policy gestite da AWS per Amazon RDS.	8 aprile 2024

Modifica	Descrizione	Data
Policy gestite da AWS per Amazon RDS: nuova policy	Amazon RDS ha aggiunto una nuova policy gestita denominata `AmazonRDSCustomInstanceProfileRolePolicy` per consentire a RDS Custom di eseguire azioni di automazione e attività di gestione del database tramite un profilo dell'istanza EC2. Per ulteriori informazioni, consulta Policy gestite da AWS per Amazon RDS.	27 febbraio 2024
Autorizzazioni del ruolo collegato ai servizi per Amazon Aurora: aggiornamento a una policy esistente	Amazon RDS ha aggiunto nuovi ID istruzione alla policy `AmazonRDSServiceRolePolicy` del ruolo collegato al servizio `AWSServiceRoleForRDS` . Per ulteriori informazioni, consulta Autorizzazioni del ruolo collegato ai servizi per Amazon Aurora.	19 gennaio 2024

Modifica	Descrizione	Data
Policy gestite da AWS per Amazon RDS: aggiornamento a policy esistenti	Le policy gestite `AmazonRDSPerformanceInsightsReadOnly` e `AmazonRDSPerformanceInsightsFullAccess` ora includono `Sid` (ID istruzione) come identificativo nell'istruzione della policy. Per ulteriori informazioni, consulta Policy gestita AWS: AmazonRDSPerformanceInsightsReadOnly e Policy gestita da AWS: AmazonRDSPerformanceInsightsFullAccess.	23 ottobre 2023
Policy gestite da AWS per Amazon RDS: aggiornamento a policy esistente	Amazon RDS ha aggiunto nuove autorizzazioni alla policy gestita `AmazonRDSFullAccess`. Le autorizzazioni consentono di generare, visualizzare ed eliminare il report di analisi delle prestazioni per un periodo di tempo. Per ulteriori informazioni sulla configurazione delle policy di accesso per Performance Insights, consulta Configurazione delle policy di accesso per Performance Insights	17 agosto 2023

Modifica	Descrizione	Data
Policy gestite da AWS per Amazon RDS: nuova policy e aggiornamento a una policy esistente	Amazon RDS ha aggiunto nuove autorizzazioni alla policy gestita `AmazonRDSPerformanceInsightsReadOnly` e una nuova policy gestita denominata `AmazonRDSPerformanceInsightsFullAccess`. Queste autorizzazioni consentono di analizzare Performance Insights per un periodo di tempo, visualizzare i risultati dell'analisi insieme ai suggerimenti ed eliminare i report. Per ulteriori informazioni sulla configurazione delle policy di accesso per Performance Insights, consulta Configurazione delle policy di accesso per Performance Insights	16 agosto 2023

Modifica	Descrizione	Data
Policy gestite da AWS per Amazon RDS: aggiornamento a una policy esistente	Amazon RDS ha aggiunto un nuovo spazio dei nomi Amazon CloudWatch `ListMetrics` a `AmazonRDSFullAccess` e `AmazonRDSReadOnlyAccess`. Questo spazio dei nomi è necessario affinché Amazon RDS elenchi specifici parametri di utilizzo delle risorse. Per ulteriori informazioni, consulta Panoramica sulla gestione delle autorizzazioni di accesso alle risorse CloudWatch nella Guida per l'utente di Amazon CloudWatch.	4 aprile 2023

Modifica	Descrizione	Data
Autorizzazioni del ruolo collegato ai servizi per Amazon Aurora: aggiornamento a una policy esistente	Amazon RDS ha aggiunto nuove autorizzazioni alla `AmazonRDSServiceRolePolicy` del ruolo collegato al servizio `AWSServiceRoleForRDS` per l'integrazione con Gestione dei segreti AWS. RDS richiede l'integrazione con Secrets Manager per la gestione delle password degli utenti master in Secrets Manager. Il segreto utilizza una convenzione di denominazione riservata e limita gli aggiornamenti del cliente. Per ulteriori informazioni, consulta Gestione delle password con Amazon Aurora e Gestione dei segreti AWS.	22 dicembre 2022

Modifica	Descrizione	Data
Policy gestite da AWS per Amazon RDS: aggiornamento a policy esistenti	Amazon RDS ha aggiunto una nuova autorizzazione alle policy gestite `AmazonRDSFullAccess` e `AmazonRDSReadOnlyAccess` per consentire di attivare Amazon DevOps Guru nella console RDS. Questa autorizzazione è necessaria per verificare se DevOps Guru è attivato. Per ulteriori informazioni, consulta Configurazione delle politiche di accesso IAM per Guru for RDS DevOps.	19 dicembre 2022

Modifica	Descrizione	Data
Autorizzazioni del ruolo collegato ai servizi per Amazon Aurora: aggiornamento a una policy esistente	Amazon RDS ha aggiunto un nuovo spazio dei nomi Amazon CloudWatch ad `AmazonRDSPreviewServiceRolePolicy` per `PutMetricData` . Questo spazio dei nomi è necessario affinché Amazon RDS pubblichi i parametri di utilizzo delle risorse. Per ulteriori informazioni, consultare Utilizzo delle chiavi di condizione per limitare l'accesso agli spazi dei nomi di CloudWatch nella Guida per l'utente di Amazon CloudWatch.	7 luglio 2022

Modifica	Descrizione	Data
Autorizzazioni del ruolo collegato ai servizi per Amazon Aurora: aggiornamento a una policy esistente	Amazon RDS ha aggiunto un nuovo spazio dei nomi Amazon CloudWatch ad `AmazonRDSBetaServiceRolePolicy` per `PutMetricData` . Questo spazio dei nomi è necessario affinché Amazon RDS pubblichi i parametri di utilizzo delle risorse. Per ulteriori informazioni, consultare Utilizzo delle chiavi di condizione per limitare l'accesso agli spazi dei nomi di CloudWatch nella Guida per l'utente di Amazon CloudWatch.	7 luglio 2022

Modifica	Descrizione	Data
Autorizzazioni del ruolo collegato ai servizi per Amazon Aurora: aggiornamento a una policy esistente	Amazon RDS ha aggiunto un nuovo spazio dei nomi Amazon CloudWatch ad `AWSServiceRoleForRDS` per `PutMetricData` . Questo spazio dei nomi è necessario affinché Amazon RDS pubblichi i parametri di utilizzo delle risorse. Per ulteriori informazioni, consultare Utilizzo delle chiavi di condizione per limitare l'accesso agli spazi dei nomi di CloudWatch nella Guida per l'utente di Amazon CloudWatch.	22 aprile 2022
Policy gestite da AWS per Amazon RDS: nuova policy	Amazon RDS ha aggiunto una nuova policy gestita denominata `AmazonRDSPerformanceInsightsReadOnly` per consentire ad Amazon RDS di chiamare i servizi AWS per conto delle istanze database. Per ulteriori informazioni sulla configurazione delle policy di accesso per Performance Insights, consulta Configurazione delle policy di accesso per Performance Insights	10 marzo 2022

Modifica	Descrizione	Data
Autorizzazioni del ruolo collegato ai servizi per Amazon Aurora: aggiornamento a una policy esistente	Amazon RDS ha aggiunto nuovi namespace Amazon CloudWatch a `AWSServiceRoleForRDS` per `PutMetricData` . Questi spazi dei nomi servono ad Amazon DocumentDB (con compatibilità MongoDB) e Amazon Neptune per pubblicare i parametri di CloudWatch. Per ulteriori informazioni, consultare Utilizzo delle chiavi di condizione per limitare l'accesso agli spazi dei nomi di CloudWatch nella Guida per l'utente di Amazon CloudWatch.	4 marzo 2022
Amazon RDS ha iniziato a monitorare le modifiche	Amazon RDS ha iniziato a tenere traccia delle modifiche per le sue policy gestite da AWS.	26 ottobre 2021

Prevenzione del problema "confused deputy" tra servizi

Il problema confused deputy è un problema di sicurezza in cui un'entità che non dispone dell'autorizzazione per eseguire un'azione può costringere un'entità maggiormente privilegiata a eseguire l'azione. InoltreAWS, l'impersonificazione tra diversi servizi può portare alla confusione del vicesceriffo.

La rappresentazione tra servizi può verificarsi quando un servizio (il servizio chiamante) effettua una chiamata a un altro servizio (il servizio chiamato). Il servizio chiamante può essere manipolato per utilizzare le proprie autorizzazioni e agire sulle risorse di un altro cliente, a cui normalmente non avrebbe accesso. Per evitare che ciò accada, AWS fornisce strumenti che possono aiutarvi a proteggere i vostri dati per tutti i servizi, con responsabili del servizio a cui è stato concesso l'accesso alle risorse del vostro account. Per ulteriori informazioni, consultare Problema del "confused deputy" nella Guida per l'utente di IAM.

Per limitare le autorizzazioni alle risorse che Amazon RDS fornisce a un altro servizio per una risorsa specifica, si consiglia di utilizzare le chiavi di contesto delle condizioni globali `aws:SourceArn` e `aws:SourceAccount` nelle policy delle risorse.

In alcuni casi, il valore `aws:SourceArn` non contiene l'ID dell'account, ad esempio quando utilizzi Amazon Resource Name (ARN) per un bucket Simple Storage Service (Amazon S3). In questi casi, assicurati di utilizzare entrambe le chiavi di contesto delle condizioni globali per limitare le autorizzazioni. In alcuni casi, si utilizzano le chiavi di contesto delle condizioni globali e il valore `aws:SourceArn` contiene l'ID dell'account. In questi casi, assicurati che il valore `aws:SourceAccount` e l'account in `aws:SourceArn` utilizzano lo stesso ID dell'account quando vengono utilizzati nella stessa dichiarazione di policy. Utilizza `aws:SourceArn` se desideri consentire l'associazione di una sola risorsa all'accesso tra servizi. Se desideri consentire l'associazione di qualsiasi risorsa nell'AWSaccount specificato all'utilizzo tra servizi, utilizza. `aws:SourceAccount`

Assicurati che il valore di `aws:SourceArn` sia un ARN per un tipo di risorsa Amazon RDS. Per ulteriori informazioni, consulta Nome della risorsa Amazon (ARN) in Amazon RDS.

Il modo più efficace per proteggersi dal problema "confused deputy" è quello di usare la chiave di contesto della condizione globale `aws:SourceArn` con l'ARN completo della risorsa. In alcuni casi, potresti non conoscere l'ARN completo della risorsa o potresti specificare più risorse. In questi casi, utilizza la chiave di contesto delle condizioni globali `aws:SourceArn` con caratteri jolly (*) per le parti sconosciute dell'ARN. Un esempio è `arn:aws:rds:*:123456789012:*`.

L'esempio seguente mostra il modo in cui puoi utilizzare le chiavi di contesto `aws:SourceArn` e `aws:SourceAccount` delle condizioni globali in Amazon RDS per prevenire il problema “confused deputy”.

JSON

```
{
  "Version":"2012-10-17",
  "Statement": {
    "Sid": "ConfusedDeputyPreventionExamplePolicy",
    "Effect": "Allow",
    "Principal": {
      "Service": "rds.amazonaws.com"
    },
    "Action": "sts:AssumeRole",
    "Condition": {
      "ArnLike": {
        "aws:SourceArn": "arn:aws:rds:us-east-1:123456789012:db:mydbinstance"
      },
      "StringEquals": {
        "aws:SourceAccount": "123456789012"
      }
    }
  }
}
```

Per altri esempi di policy che utilizzano le chiavi di contesto delle condizioni globali `aws:SourceArn` e `aws:SourceAccount`, consulta le sezioni seguenti:

- Concessione di autorizzazioni per pubblicare le notifiche in un argomento Amazon SNS
- Configurazione dell'accesso a un bucket Simple Storage Service (Amazon S3) (importazione PostgreSQL)
- Configurazione dell'accesso a un bucket Amazon S3 (esportazione PostgreSQL)

Autenticazione del database IAM

Puoi autenticarti nel tuo cluster di DB utilizzando l'autenticazione del database AWS Identity and Access Management (IAM). L'autenticazione del database IAM funziona con Aurora MySQL e Aurora PostgreSQL. Con questo metodo di autenticazione, non devi utilizzare una password quando esegui la connessione al cluster database. Utilizzi invece un token di autenticazione.

Un token di autenticazione è una stringa univoca di caratteri generata da Amazon Aurora su richiesta. I token di autenticazione vengono generati utilizzando AWS Signature Version 4. Ciascun token ha un ciclo di vita di 15 minuti. Non devi archiviare le credenziali dell'utente nel database, perché l'autenticazione è gestita esternamente utilizzando IAM. Puoi anche utilizzare ancora l'autenticazione standard del database. Il token viene utilizzato solo per l'autenticazione e non influisce sulla sessione dopo che è stato stabilito.

L'autenticazione del database IAM fornisce i seguenti vantaggi:

- Il traffico di rete da e verso il database viene crittografato utilizzando Secure Socket Layer (SSL) o Transport Layer Security (TLS). Per ulteriori informazioni sull'utilizzo SSL/TLS con Aurora, consulta. Utilizzo SSL/TLS per crittografare una connessione a un'
- Puoi usare IAM per gestire in modo centralizzato l'accesso alle risorse del database invece di gestire l'accesso singolarmente in ogni cluster database.
- Per le applicazioni in esecuzione su Amazon EC2, puoi utilizzare credenziali di profilo specifiche per la tua EC2 istanza per accedere al database anziché una password, per una maggiore sicurezza.

In generale, prendi in considerazione l'utilizzo dell'autenticazione del database IAM quando le applicazioni creano meno di 200 connessioni al secondo e non desideri gestire nomi utente e password direttamente nel codice dell'applicazione.

Il driver JDBC Amazon Web Services (AWS) supporta l’autenticazione del database IAM. Per ulteriori informazioni, consulta AWS IAM Authentication Plugin nel repository di driver GitHub JDBC di Amazon Web Services (AWS).

Il driver Python Amazon Web Services (AWS) supporta l’autenticazione del database IAM. Per ulteriori informazioni, consulta AWS IAM Authentication Plugin nel repository Python Driver GitHub di Amazon Web Services (AWS).

Per apprendere il processo di impostazione di IAM per l’autenticazione del database, consulta i seguenti argomenti:

- Abilitazione e disabilitazione dell'autenticazione database IAM
- Creazione e utilizzo di una policy IAM per l'accesso al database IAM
- Creazione di un account database tramite l'autenticazione IAM
- Connessione al cluster di database tramite l'autenticazione IAM

Disponibilità di regioni e versioni

La disponibilità e il supporto della funzionalità varia a seconda delle versioni specifiche di ciascun motore di database Aurora e in tutte le Regioni AWS. Per ulteriori informazioni sulla disponibilità di versioni e Regioni per l'autenticazione del database Aurora e IAM, consulta Regioni e motori di database Aurora supportati per l'autenticazione del database IAM.

Per Aurora MySQL, tutte le classi di istanza database supportate supportano l'autenticazione del database IAM, ad eccezione di db.t2.small e db.t3.small. Per informazioni sulle classi di istanza database supportate, consulta Motori di database supportati per classi di istanza database.

Supporto per CLI e SDK

L'autenticazione del database IAM è disponibile per AWS CLIe per i seguenti linguaggi: AWS SDKs

- AWS SDK per .NET
- AWS SDK per C++
- AWS SDK per Go
- AWS SDK per Java
- AWS SDK per JavaScript
- AWS SDK per PHP
- AWS SDK per Python (Boto3)
- AWS SDK per Ruby

Limitazioni per l'autenticazione database IAM

Quando utilizzi l'autenticazione database IAM, tieni presenti le seguenti limitazioni:

- Attualmente, l'autenticazione database IAM non supporta nessuna delle chiavi di contesto delle condizioni globali.

Per ulteriori informazioni sulle chiavi di contesto delle condizioni globali, consulta Chiavi di contesto delle condizioni globali AWS nella Guida per l'utente di IAM.

- Per PostgreSQL, se il ruolo IAM (`rds_iam`) viene aggiunto a un utente (incluso l'utente principale RDS), l'autenticazione IAM ha la precedenza sull'autenticazione tramite password, quindi l'utente deve accedere come un utente IAM.
- Non è possibile utilizzare un record DNS Route 53 personalizzato anziché l'endpoint del cluster di databaseper generare il token di autenticazione.
- CloudWatch e CloudTrail non registrate l'autenticazione IAM. Questi servizi non tengono traccia delle chiamate API `generate-db-auth-token` che autorizzano il ruolo IAM a consentire la connessione al database.
- L'autenticazione del database IAM richiede risorse di calcolo sul cluster di database. Per una connettività affidabile è necessario disporre di una memoria aggiuntiva compresa tra 300 e 1000 MiB nel database. Per individuare la memoria necessaria per il carico di lavoro, si confronta la colonna RES per i processi RDS nell'elenco dei processi di monitoraggio avanzato prima e dopo aver abilitato l'autenticazione del database IAM. Per informazioni, consulta Visualizzazione delle metriche nella console RDS.

  Se si utilizza un'istanza di classe espandibile, evitare di esaurire la memoria riducendo della stessa quantità la memoria utilizzata da altri parametri, come buffer e cache.
- Per Aurora MySQL, non è possibile utilizzare l'autenticazione basata su password per un utente del database configurato con l'autenticazione IAM.
- L'autenticazione del database IAM non è supportata per RDS su Outposts per tutti i motori.

Consigli per l'autenticazione del database IAM

Quando si utilizza l'autenticazione del database IAM, è consigliabile procedere come segue:

- Utilizzare l'autenticazione del database IAM quando l'applicazione richiede meno di 200 nuove connessioni di autenticazione del database IAM al secondo.

  I motori di database che funzionano con Amazon Aurora non prevedono limitazioni per i tentativi di autenticazione al secondo. Tuttavia, quando utilizzi un'autenticazione database IAM, l'applicazione deve generare un token di autenticazione. L'applicazione usa quindi il token per connettersi al cluster database. Se eccedi il limite massimo di nuove connessioni al secondo, la gestione extra dell'autenticazione database IAM può causare throttling della connessione.

Valuta la possibilità di utilizzare il pool di connessioni nelle applicazioni per mitigare la creazione continua di connessioni. Questo può ridurre il sovraccarico derivante dall'autenticazione DB IAM e consentire alle applicazioni di riutilizzare le connessioni esistenti. In alternativa, per questi casi d'uso considera l'utilizzo di Server proxy per RDS. Per Server proxy per RDS sono previsti costi aggiuntivi. Consulta i prezzi per Server proxy per RDS.

- La dimensione di un token di autenticazione del database IAM dipende da molti fattori, tra cui il numero di tag IAM, le policy di servizio IAM, la lunghezza del nome della risorsa Amazon (ARN) e altre proprietà IAM e del database. La dimensione minima del token è generalmente di circa 1 KB, ma può essere maggiore. Poiché questo token viene utilizzato come password nella stringa di connessione al database mediante l'autenticazione IAM, è necessario assicurarsi che il driver del database (ad esempio ODBC) and/or non limiti o altrimenti tronchi questo token a causa delle sue dimensioni. Un token troncato causa l'esito negativo della convalida dell'autenticazione effettuata dal database e da IAM.
- Se si utilizzano credenziali temporanee durante la creazione di un token di autenticazione del database IAM, le credenziali temporanee devono essere ancora valide quando si utilizza il token di autenticazione del database IAM per effettuare una richiesta di connessione.

Chiavi di contesto relative alle condizioni globali non supportate AWS

L'autenticazione del database IAM non supporta il seguente sottoinsieme di chiavi di contesto delle condizioni AWS globali.

- `aws:Referer`
- `aws:SourceIp`
- `aws:SourceVpc`
- `aws:SourceVpce`
- `aws:UserAgent`
- `aws:VpcSourceIp`

Per ulteriori informazioni, consultare Chiavi di contesto delle condizioni globali AWS nella Guida per l'utente IAM.

Abilitazione e disabilitazione dell’autenticazione database IAM

Per impostazione predefinita, l’autenticazione database IAM è disabilitata nei cluster database. Puoi abilitare o disabilitare l'autenticazione del database IAM utilizzando Console di gestione AWS AWS CLI, o l'API.

Puoi abilitare l’autenticazione database IAM quando esegui una delle seguenti operazioni:

- Per creare un nuovo cluster di database con l’autenticazione database IAM abilitata, consulta Creazione di un cluster database Amazon Aurora.
- Per modificare un cluster di database per abilitare l’autenticazione database IAM, consulta Modifica di un cluster database Amazon Aurora.
- Per ripristinare un cluster di database da uno snapshot con l’autenticazione del database IAM abilitata, consulta Ripristino da uno snapshot cluster database.
- Per ripristinare un cluster database a un momento specifico con l’autenticazione del database IAM abilitata, consulta Ripristino di un cluster di database a un determinato momento.

Console

Ogni flusso di lavoro di creazione o modifica include una sezione Database authentication (Autenticazione database) in cui puoi abilitare o disabilitare l’autenticazione database IAM. In questa sezione scegli Password and IAM database authentication (Autenticazione password e database IAM) per abilitare l’autenticazione database IAM.

Per abilitare o disabilitare l’autenticazione database IAM per un cluster di database esistente

1. Aprire la console Amazon RDS all'indirizzo https://console.aws.amazon.com/rds/.
2. Nel pannello di navigazione, scegliere Databases (Database).
3. Scegliere il cluster database che si vuole modificare.

   > **Note**
   >
   > Puoi abilitare l’autenticazione IAM solo se tutte le istanze database nel cluster di database sono compatibili con IAM. Controllare i requisiti di compatibilità in Disponibilità di regioni e versioni.

4. Scegliere Modify (Modifica).

5. Nella sezione Autenticazione del database, scegli Autenticazione del database IAM per abilitare l'autenticazione del database IAM. Scegli Autenticazione password o Autenticazione di password e Kerberos per disabilitare l'autenticazione IAM.
6. Puoi anche scegliere di abilitare la pubblicazione dei log di autenticazione IAM DB su Logs. CloudWatch In Esportazioni di log, scegli l'opzione iam-db-auth-error log. La pubblicazione dei log in CloudWatch Logs consuma spazio di archiviazione e comporta l'addebito di costi per tale archiviazione. Assicurati di eliminare tutti i CloudWatch log che non ti servono più.
7. Scegli Continue (Continua).
8. Per applicare immediatamente le modifiche, scegli Immediately (Immediatamente) nella sezione Scheduling of modifications (Pianificazione delle modifiche).
9. Scegliere Modify cluster (Modifica cluster).

AWS CLI

Per creare un nuovo cluster DB con autenticazione IAM utilizzando il AWS CLI, usa il `create-db-cluster`comando. Specifica l'opzione `--enable-iam-database-authentication`.

Per aggiornare un cluster di database esistente in modo da abilitare o disabilitare l'autenticazione IAM, utilizzare il comando AWS CLI `modify-db-cluster`. Specifica l'opzione `--enable-iam-database-authentication` o `--no-enable-iam-database-authentication`, come appropriato.

> **Note**
>
> Puoi abilitare l'autenticazione IAM solo se tutte le istanze database nel cluster di database sono compatibili con IAM. Controllare i requisiti di compatibilità in Disponibilità di regioni e versioni.

Per impostazione predefinita, Aurora esegue la modifica durante la finestra di manutenzione successiva. Se desideri sostituire ciò e abilitare l'autenticazione database IAM il prima possibile, utilizza il parametro `--apply-immediately`.

Se stai ripristinando un cluster di DB, usa uno dei seguenti AWS CLI comandi:

- `restore-db-cluster-to-point-in-time`
- `restore-db-cluster-from-db-snapshot`

L'impostazione di autenticazione del database IAM corrisponde a quella della snapshot origine. Per modificare questa impostazione, imposta l'opzione `--enable-iam-database-authentication` or `--no-enable-iam-database-authentication`, come appropriato.

API RDS

Per creare una nuova istanza database con autenticazione IAM tramite l'API, utilizzare l'operazione API `CreateDBCluster`. Imposta il parametro `EnableIAMDatabaseAuthentication` su `true`.

Per aggiornare un cluster di database esistente in modo da abilitare l'autenticazione IAM, utilizzare l'operazione API `ModifyDBCluster`. Imposta il parametro `EnableIAMDatabaseAuthentication` su `true` per abilitare l'autenticazione IAM o su `false` per disabilitarla.

> **Note**
>
> Puoi abilitare l'autenticazione IAM solo se tutte le istanze database nel cluster di database sono compatibili con IAM. Controllare i requisiti di compatibilità in Disponibilità di regioni e versioni.

Se ripristini un cluster database, usa una delle operazioni API seguenti:

- `RestoreDBClusterFromSnapshot`
- `RestoreDBClusterToPointInTime`

L'impostazione di autenticazione del database IAM corrisponde a quella della snapshot origine. Per modificare questa impostazione, imposta il parametro `EnableIAMDatabaseAuthentication` su `true` per abilitare l'autenticazione IAM o su `false` per disabilitarla.

Creazione e utilizzo di una policy IAM per l'accesso al database IAM

Per permettere a un utente o un ruolo di connettersi al cluster database, devi creare una policy IAM. Dopo questa operazione, collega la policy a un set di autorizzazioni o un ruolo.

> **Note**
>
> Per ulteriori informazioni sulle policy IAM, consulta Gestione accessi e identità per Amazon Aurora.

La policy nell'esempio seguente permette a un utente di connettersi a un cluster database tramite l'autenticazione database IAM.

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "rds-db:connect"
            ],
            "Resource": [
                "arn:aws:rds-db:us-east-2:111122223333:dbuser:db-
ABCDEFGHIJKL01234/db_user"
            ]
        }
    ]
}
```

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "rds-db:connect"
            ],
            "Resource": [
                "arn:aws:rds-db:us-east-2:111122223333:dbuser:cluster-
ABCDEFGHIJKL01234/db_user"
            ]
        }
    ]
}
```

Important

Un utente con le autorizzazioni di amministratore può accedere ai cluster database senza una policy IAM esplicita. Se vuoi limitare l'accesso come amministratore a cluster di , puoi creare un ruolo IAM con le autorizzazioni appropriate, con privilegi minori e assegnarlo all'amministratore.

Note

Non confondere il prefisso `rds-db:` con altri prefissi di operazione API RDS che iniziano con `rds:`. Utilizzi il prefisso `rds-db:` e l'operazione `rds-db:connect` solo per l'autenticazione database IAM. Non sono validi in altri contesti.

La policy di esempio include una singola istruzione con i seguenti elementi:

- `Effect` – Specifica `Allow` per concedere l'accesso al cluster database. Se non si concede esplicitamente l'accesso, questo viene rifiutato per impostazione predefinita.
- `Action` – Specifica `rds-db:connect` per consentire le connessioni al cluster database.
- `Resource` – Specifica un Amazon Resource Name (ARN) che descrive un account database in un cluster database. Di seguito è riportato il formato ARN.

```
arn:aws:rds-db:region:account-id:dbuser:DbClusterResourceId/db-user-name
```

In questo formato, sostituisci quanto segue:

- *`region`*è la AWS regione per il cluster di DB. Nella politica di esempio, la AWS regione è`us-east-2`.
- *`account-id`*è il numero di AWS account per il cluster di DB. Nella policy di esempio, il numero di account è `1234567890`. L'utente deve essere nello stesso account dell'account del cluster di database.

  Per eseguire l'accesso multi-account, crea un ruolo IAM con la policy mostrata in precedenza nell'account per il cluster di database e consenti all'altro account di assumere il ruolo.

- *DbClusterResourceId* è l'identificatore del cluster database. Questo identificatore è unico per una AWS regione e non cambia mai. Nella policy di esempio, l'identificatore è `cluster-ABCDEFGHIJKL01234`.

  Per trovare un ID di risorsa del Console di gestione AWScluster di DB in Amazon Aurora, scegli il cluster di istanze per visualizzarne i dettagli. Quindi seleziona la scheda Configuration (Configurazione). Resource ID (ID risorsa) è visualizzato nella sezione Configuration (Configurazione).

  In alternativa, puoi utilizzare il AWS CLI comando per elencare gli identificatori e le risorse IDs per tutto il cluster di DB nella AWS regione corrente, come illustrato di seguito.

  ```
  aws rds describe-db-clusters --query "DBClusters[*].
  [DBClusterIdentifier,DbClusterResourceId]"
  ```

  > **Note**
  >
  > Se si sta effettuando la connessione a un database tramite proxy RDS, specificare l'ID risorsa proxy, ad esempio `prx-ABCDEFGHIJKL01234`. Per informazioni sull'utilizzo dell'autenticazione del database IAM con proxy RDS, vedere Connessione a un database tramite l'autenticazione IAM.

- *db-user-name* è il nome dell'account database da associare all'autenticazione IAM. Nella policy di esempio, l'account database è `db_user`.

È possibile crearne altri ARNs per supportare vari modelli di accesso. La policy seguente permette l'accesso a due diversi account database in un cluster database.

JSON

```
{
   "Version":"2012-10-17",
   "Statement": [
      {
         "Effect": "Allow",
         "Action": [
```

```
            "rds-db:connect"
        ],
        "Resource": [
            "arn:aws:rds-db:us-east-2:123456789012:dbuser:db-ABCDEFGHIJKL01234/
jane_doe",
            "arn:aws:rds-db:us-east-2:123456789012:dbuser:db-ABCDEFGHIJKL01234/
mary_roe"
        ]
      }
   ]
}
```

JSON

```
{
   "Version":"2012-10-17",
   "Statement": [
      {
         "Effect": "Allow",
         "Action": [
             "rds-db:connect"
         ],
         "Resource": [
             "arn:aws:rds-db:us-east-2:123456789012:dbuser:cluster-
ABCDEFGHIJKL01234/jane_doe",
             "arn:aws:rds-db:us-east-2:123456789012:dbuser:cluster-
ABCDEFGHIJKL01234/mary_roe"
         ]
      }
   ]
}
```

La seguente politica utilizza il carattere «*» per abbinare tutte le DB, i cluster e gli account di database per un account e una regione particolariAWS. AWS

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "rds-db:connect"
            ],
            "Resource": [
                "arn:aws:rds-db:us-east-2:111122223333:dbuser:*/*"
            ]
        }
    ]
}
```

La seguente politica corrisponde a tutti i cluster di DB per un account e una regione particolariAWS. AWS Tuttavia, la policy concede l'accesso solo a cluster database che hanno un account database `jane_doe`.

JSON

```
{
   "Version":"2012-10-17",
   "Statement": [
      {
         "Effect": "Allow",
         "Action": [
             "rds-db:connect"
         ],
         "Resource": [
             "arn:aws:rds-db:us-east-2:123456789012:dbuser:*/jane_doe"
         ]
      }
   ]
}
```

L'utente o il ruolo ha accesso solo a quei database ai quali l'utente ha accesso. Supponiamo, ad esempio, che il cluster database abbia un database denominato dev e un altro database denominato test. Se l'utente del database `jane_doe` ha accesso solo a dev, anche qualsiasi utente o ruolo che accede al cluster database con l'utente `jane_doe` ha accesso solo a dev. Questa restrizione dell'accesso è anche valida per altri oggetti database, come tabelle, visualizzazioni e così via.

Un amministratore deve creare policy IAM che concedono alle entità l'autorizzazione per eseguire operazioni API specifiche sulle risorse specificate di cui hanno bisogno. L'amministratore deve quindi collegare queste policy ai set di autorizzazioni o ai ruoli che richiedono tali autorizzazioni. Per esempi di policy, consulta Esempi di policy di Amazon Aurora basate su identità.

Collegamento di una policy IAM a un set di autorizzazioni o un ruolo

Dopo aver creato una policy IAM per consentire l'autenticazione del database, devi collegare la policy a un set di autorizzazioni o un ruolo. Per un tutorial su questo argomento, consulta Creare e collegare la tua prima policy gestita dal cliente nella Guida per l'utente IAM.

Durante il tutorial, puoi utilizzare uno degli esempi di policy indicati in questa sezione come punto di partenza e personalizzarli in base alle tue esigenze. Al termine del tutorial, avrai un set di autorizzazioni con una policy collegata che può utilizzare l'operazione `rds-db:connect`.

> **Note**
>
> Puoi mappare più set di autorizzazioni o ruoli allo stesso account utente del database. Ad esempio, supponiamo che la policy IAM abbia specificato la seguente risorsa ARN.
>
> ```
> arn:aws:rds-db:us-east-2:123456789012:dbuser:cluster-12ABC34DEFG5HIJ6KLMNOP78QR/
> jane_doe
> ```
>
> Se colleghi la policy agli utenti Jane, Bob e Diego, ciascuno di quegli utenti potrà connettersi al cluster di database specificato utilizzando l'account di database `jane_doe`.

Creazione di un account database tramite l'autenticazione IAM

Con l'autenticazione database IAM, non devi assegnare password di database agli account utente che crei. Se rimuovi un utente mappato a un account database, devi anche rimuovere l'account database con l'istruzione `DROP USER`.

> **Note**
>
> Il nome utente utilizzato per l'autenticazione IAM deve avere lo stesso formato maiuscolo/minuscolo del nome utente nel database.

Argomenti

- Utilizzo dell'autenticazione IAM con Aurora MySQL
- Utilizzo dell'autenticazione IAM con Aurora PostgreSQL

Utilizzo dell'autenticazione IAM con Aurora MySQL

Con Aurora MySQL, l'autenticazione viene gestita da `AWSAuthenticationPlugin`, un plug-in fornito da AWS che funziona perfettamente con IAM per autenticare gli utenti. Connettiti al cluster database come utente master o altro utente che può creare utenti e concedere privilegi. Dopo la connessione, immetti l'istruzione `CREATE USER`, come mostrato nell'esempio seguente.

```
CREATE USER 'jane_doe' IDENTIFIED WITH AWSAuthenticationPlugin AS 'RDS';
```

La clausola `IDENTIFIED WITH` permette a Aurora MySQL di utilizzare `AWSAuthenticationPlugin` per autenticare l'account database (`jane_doe`). La clausola `AS 'RDS'` fa riferimento al metodo di autenticazione. Assicurarsi che il nome utente del database specificato sia lo stesso di una risorsa nella policy IAM per l'accesso al database IAM. Per ulteriori informazioni, consulta Creazione e utilizzo di una policy IAM per l'accesso al database IAM.

> **Note**
>
> Se viene visualizzato il messaggio seguente, significa che il plug-in fornito da AWS non è disponibile per il cluster di database corrente.
> `ERROR 1524 (HY000): Plugin 'AWSAuthenticationPlugin' is not loaded`
> Per correggere questo errore, assicurati di usare una configurazione supportata e di aver abilitato l'autenticazione database IAM nel cluster database. Per ulteriori informazioni, consulta Disponibilità di regioni e versioni e Abilitazione e disabilitazione dell'autenticazione database IAM.

Dopo aver creato un account utilizzando `AWSAuthenticationPlugin`, lo gestisci nello stesso modo di altri account database. Ad esempio, puoi modificare i privilegi di account con le istruzioni `GRANT` e `REVOKE` o modificare vari attributi di account con l'istruzione `ALTER USER`.

Il traffico di rete del database viene crittografato utilizzando SSL/TLS quando si utilizza IAM. Per consentire le connessioni SSL, modifica l'account utente con il comando seguente.

```
ALTER USER 'jane_doe'@'%' REQUIRE SSL;
```

Utilizzo dell'autenticazione IAM con Aurora PostgreSQL

Per utilizzare l'autenticazione IAM con Aurora PostgreSQL, connettiti al cluster database come utente master o altro utente che può creare utenti e concedere privilegi. Dopo la connessione, crea gli utenti di database e autorizza il ruolo `rds_iam`, come mostrato nell'esempio seguente.

```
CREATE USER db_userx;
GRANT rds_iam TO db_userx;
```

Assicurarsi che il nome utente del database specificato sia lo stesso di una risorsa nella policy IAM per l'accesso al database IAM. Per ulteriori informazioni, consulta Creazione e utilizzo di una policy IAM per l'accesso al database IAM. È necessario fornire il ruolo `rds_iam` per utilizzare l'autenticazione IAM. È possibile utilizzare anche appartenenze nidificate o concessioni indirette del ruolo.

Tieni presente che un utente del database PostgreSQL può utilizzare l'autenticazione IAM o Kerberos ma non entrambe, quindi questo utente non può avere anche il ruolo `rds_ad`. Questo vale anche per le appartenenze nidificate. Per ulteriori informazioni, consulta Fase 7: creazione di utenti PostgreSQL per i principali Kerberos .

Connessione al cluster di database tramite l'autenticazione IAM

Con l'autenticazione database IAM devi usare un token di autenticazione per la connessione al cluster di database. Un token di autenticazione è una stringa unica di caratteri che utilizzi invece di una password. Trascorsi 15 minuti dalla sua creazione, un token di autenticazione scade. Se cerchi di eseguire la connessione utilizzando un token scaduto la richiesta di connessione viene negata.

Ogni token di autenticazione deve essere accompagnato da una firma valida, utilizzando AWS Signature Version 4. Per ulteriori informazioni, consulta Processo di firma Signature Version 4 in

Riferimenti generali di AWS. La AWS CLI e un SDK AWS, come AWS SDK per Java o AWS SDK per Python (Boto3), possono firmare automaticamente ogni token che viene creato.

Poi usare un token di autenticazione quando ti connetti ad Amazon Aurora da un altro servizio AWS, ad esempio AWS Lambda. Utilizzando un token, eviti di inserire una password nel codice. In alternativa, puoi utilizzare l'SDK AWS per creare e firmare in modo programmatico un token di autenticazione.

Quando hai un token di autenticazione IAM firmato, puoi connetterti a un cluster di database Aurora. Di seguito vengono fornite informazioni su come eseguire questa operazione tramite uno strumento a riga di comando o un SDK AWS, come AWS SDK per Java o AWS SDK per Python (Boto3).

Per ulteriori informazioni, consulta il seguente post sul blog:

- Uso dell'autenticazione IAM per la connessione con SQL Workbench/J a Aurora MySQL o Amazon RDS for MySQL.
- Utilizzo dell'autenticazione IAM per connettersi con pgAdmin Amazon Aurora PostgreSQL o Amazon RDS for PostgreSQL

Prerequisiti

Di seguito sono riportati i prerequisiti per la connessione alcluster di DB utilizzando l'autenticazione IAM:

- Abilitazione e disabilitazione dell’autenticazione database IAM
- Creazione e utilizzo di una policy IAM per l'accesso al database IAM
- Creazione di un account database tramite l’autenticazione IAM

Argomenti

- Connessione al cluster dell’ database utilizzando l’autenticazione IAM con i driver AWS
- Connessione al cluster database tramite l'autenticazione IAM dalla riga di comando: AWS CLI e il client MySQL
- Connessione al cluster di database tramite Autenticazione IAM dalla riga di comando: AWS CLI e client psql
- Connessione al dell'istanza database tramite Autenticazione IAM e AWS SDK per .NET
- Connessione al dell'istanza database tramite Autenticazione IAM e AWS SDK per Go

- Connessione al dell’istanza database tramite Autenticazione IAM e AWS SDK per Java
- Connessione al dell'istanza database tramite Autenticazione IAM e AWS SDK per Python (Boto3)

Connessione al cluster dell’ database utilizzando l’autenticazione IAM con i driver AWS

La suite di driver AWS è stata progettata per consentire tempi di switchover e failover più rapidi e per eseguire l’autenticazione con Gestione dei segreti AWS, AWS Identity and Access Management (IAM) e l’identità federata. I driver AWS si basano sul monitoraggio dello stato dell’ del cluster di database e conoscono la topologia dell’ del cluster per determinare la nuova istanza di scrittura. Questo approccio riduce i tempi di switchover e failover a meno di dieci secondi, rispetto alle decine di secondi necessari per i driver open source.

Per ulteriori informazioni sui driver AWS, consulta il driver del linguaggio corrispondente per il cluster di database Aurora MySQL o Aurora PostgreSQL.

Connessione al cluster database tramite l'autenticazione IAM dalla riga di comando: AWS CLI e il client MySQL

Puoi eseguire la connessione dalla riga di comando a un cluster database Aurora con AWS CLI e lo strumento a riga di comando `mysql` come descritto di seguito.

Prerequisiti

Di seguito sono riportati i prerequisiti per la connessione alcluster di DB utilizzando l'autenticazione IAM:

- Abilitazione e disabilitazione dell’autenticazione database IAM
- Creazione e utilizzo di una policy IAM per l'accesso al database IAM
- Creazione di un account database tramite l’autenticazione IAM

> **Note**
>
> Per informazioni sulla connessione al database tramite SQL Workbench/J con autenticazione IAM, consulta il post del blog Utilizzo dell'autenticazione IAM per connettersi con SQL Workbench/J a Aurora MySQL o Amazon RDS for MySQL.

Argomenti

- Generazione di un token di autenticazione IAM
- Connessione a un cluster database

Generazione di un token di autenticazione IAM

L'esempio seguente mostra come ottenere un token di autenticazione firmato utilizzando AWS CLI.

```
aws rds generate-db-auth-token \
   --hostname rdsmysql.123456789012.us-west-2.rds.amazonaws.com \
   --port 3306 \
   --region us-west-2 \
   --username jane_doe
```

Nell'esempio, i parametri sono come segue:

- `--hostname` – Nome host del cluster database cui vuoi accedere.
- `--port` – Numero di porta usato per la connessione al dell'istanza database
- `--region`: la regione AWS in cui è in esecuzione il cluster database
- `--username` – L'account database cui vuoi accedere.

I primi caratteri del token hanno il seguente aspetto.

```
rdsmysql.123456789012.us-west-2.rds.amazonaws.com:3306/?
Action=connect&DBUser=jane_doe&X-Amz-Algorithm=AWS4-HMAC-SHA256&X-Amz-Expires=900...
```

> **Note**
>
> Non è possibile utilizzare un record DNS Route 53 personalizzato o un endpoint personalizzato Aurora anziché l'endpoint del cluster database per generare il token di autenticazione.

Connessione a un cluster database

Il formato generale per la connessione è visualizzato di seguito.

```
mysql --host=hostName --port=portNumber --ssl-ca=full_path_to_ssl_certificate --enable-
cleartext-plugin --user=userName --password=authToken
```

I parametri sono i seguenti:

- `--host` – Nome host del cluster database cui vuoi accedere.
- `--port` – Numero di porta usato per la connessione al dell'istanza database
- `--ssl-ca` – Il percorso completo del file del certificato SSL che contiene la chiave pubblica

  Per ulteriori informazioni, consulta Connessioni TLS a cluster di database Aurora MySQL.

  Per scaricare un certificato SSL consulta Utilizzo SSL/TLS per crittografare una connessione a un'
- `--enable-cleartext-plugin` – Valore che specifica che per la connessione deve essere usato `AWSAuthenticationPlugin`.

  Se si utilizza un client MariaDB, l'opzione `--enable-cleartext-plugin` non è richiesta.
- `--user` – L'account database cui vuoi accedere.
- `--password` – Token di autenticazione IAM firmato.

Il token di autenticazione consiste in diverse centinaia di caratteri. Può essere macchinoso nella riga di comando. Una soluzione è di salvare il token in una variabile di ambiente e utilizzare quella variabile durante la connessione. L'esempio seguente mostra un modo per eseguire questa soluzione. Nell'esempio,*/sample_dir/* è il percorso completo del file del certificato SSL che contiene la chiave pubblica.

```
RDSHOST="mysqlcluster.cluster-123456789012.us-east-1.rds.amazonaws.com"
TOKEN="$(aws rds generate-db-auth-token --hostname $RDSHOST --port 3306 --region us-
west-2 --username jane_doe )"

mysql --host=$RDSHOST --port=3306 --ssl-ca=/sample_dir/global-bundle.pem --enable-
cleartext-plugin --user=jane_doe --password=$TOKEN
```

Quando si esegue la connessione utilizzando `AWSAuthenticationPlugin`, la connessione viene protetta utilizzando SSL. Per verificare ciò, digita quanto segue al prompt del comando `mysql>`.

```
show status like 'Ssl%';
```

Le righe seguenti nell'output mostrano più dettagli.

```
+---------------+-------------+
```

```
| Variable_name | Value

                                                                    |
+---------------+-------------+
| ...           | ...
| Ssl_cipher    | AES256-SHA

                                                                    |
| ...           | ...
| Ssl_version   | TLSv1.1

                                                                    |
| ...           | ...
+-----------------------------+
```

Se desideri connetterti a un cluster di database tramite un proxy, consulta Connessione a un database tramite l'autenticazione IAM.

Connessione al cluster di database tramite Autenticazione IAM dalla riga di comando: AWS CLI e client psql

Puoi eseguire la connessione dalla riga di comando a un cluster database Aurora PostgreSQL con AWS CLI e lo strumento a riga di comando psql come descritto di seguito.

Prerequisiti

Di seguito sono riportati i prerequisiti per la connessione alcluster di DB utilizzando l'autenticazione IAM:

- Abilitazione e disabilitazione dell’autenticazione database IAM
- Creazione e utilizzo di una policy IAM per l'accesso al database IAM
- Creazione di un account database tramite l’autenticazione IAM

> **Note**
>
> Per informazioni sulla connessione al database tramite pgAdmin con autenticazione IAM, consulta il post sul blog Utilizzo dell'autenticazione IAM per connettersi con PGadmin Amazon Aurora PostgreSQL o Amazon RDS for PostgreSQL.

Argomenti

- Generazione di un token di autenticazione IAM
- Connessione a un cluster Aurora PostgreSQL

Generazione di un token di autenticazione IAM

Il token di autenticazione è costituito da diverse centinaia di caratteri, quindi può essere complesso nella riga di comando. Una soluzione è di salvare il token in una variabile di ambiente e utilizzare quella variabile durante la connessione. Il seguente esempio mostra come usare l'AWS CLI per ottenere un token di autenticazione firmato utilizzando il comando `generate-db-auth-token` e archiviarlo in una variabile di ambiente `PGPASSWORD`.

```
export RDSHOST="mypostgres-cluster.cluster-123456789012.us-west-2.rds.amazonaws.com"
export PGPASSWORD="$(aws rds generate-db-auth-token --hostname $RDSHOST --port 5432 --
region us-west-2 --username jane_doe )"
```

Nell'esempio, i parametri per il comando `generate-db-auth-token` sono i seguenti:

- `--hostname` – Nome host del cluster (endpoint del cluster) di database cui desideri accedere.
- `--port` – Numero di porta usato per la connessione al dell'istanza database
- `--region`: la regione AWS in cui è in esecuzione il cluster database
- `--username` – L'account database cui vuoi accedere.

I primi caratteri del token generato hanno il seguente aspetto.

```
mypostgres-cluster.cluster-123456789012.us-west-2.rds.amazonaws.com:5432/?
Action=connect&DBUser=jane_doe&X-Amz-Algorithm=AWS4-HMAC-SHA256&X-Amz-Expires=900...
```

> **Note**
>
> Non è possibile utilizzare un record DNS Route 53 personalizzato o un endpoint personalizzato Aurora anziché l'endpoint del cluster database per generare il token di autenticazione.

Connessione a un cluster Aurora PostgreSQL

Di seguito è mostrato il formato generale per l'utilizzo di psql per la connessione.

```
psql "host=hostName port=portNumber sslmode=verify-full
 sslrootcert=full_path_to_ssl_certificate dbname=DBName user=userName
 password=authToken"
```

I parametri sono i seguenti:

- `host` – Nome host del cluster (endpoint del cluster) di database cui desideri accedere.
- `port` – Numero di porta usato per la connessione al dell'istanza database
- `sslmode` – Modalità SSL da utilizzare.

  Quando si utilizza `sslmode=verify-full`, la connessione SSL verifica l'endpoint del cluster database rispetto a quello nel certificato SSL.
- `sslrootcert` – Il percorso completo del file del certificato SSL che contiene la chiave pubblica

  Per ulteriori informazioni, consulta Sicurezza dei dati Aurora PostgreSQL con SSL/TLS.

  Per scaricare un certificato SSL consulta Utilizzo SSL/TLS per crittografare una connessione a un'
- `dbname` – Database a cui accedere.
- `user` – L'account database cui vuoi accedere.
- `password` – Token di autenticazione IAM firmato.

> **Note**
>
> Non è possibile utilizzare un record DNS Route 53 personalizzato o un endpoint personalizzato Aurora anziché l'endpoint del cluster database per generare il token di autenticazione.

L'esempio seguente mostra l'utilizzo di psql per la connessione. Nell'esempio, psql utilizza la variabile d'ambiente `RDSHOST` per l'host e la variabile d'ambiente `PGPASSWORD` per il token generato. Inoltre, */sample_dir/* è il percorso completo al file del certificato SSL che contiene la chiave pubblica.

```
export RDSHOST="mypostgres-cluster.cluster-123456789012.us-west-2.rds.amazonaws.com"
```

```
export PGPASSWORD="$(aws rds generate-db-auth-token --hostname $RDSHOST --port 5432 --
region us-west-2 --username jane_doe )"

psql "host=$RDSHOST port=5432 sslmode=verify-full sslrootcert=/sample_dir/global-
bundle.pem dbname=DBName user=jane_doe password=$PGPASSWORD"
```

Se desideri connetterti a un cluster di database tramite un proxy, consulta Connessione a un database tramite l'autenticazione IAM.

Connessione al dell'istanza database tramite Autenticazione IAM e AWS SDK per .NET

Puoi connetterti a un cluster di database Aurora MySQL o Aurora PostgreSQL con la AWS SDK per .NET come descritto di seguito.

Prerequisiti

Di seguito sono riportati i prerequisiti per la connessione alcluster di DB utilizzando l'autenticazione IAM:

- Abilitazione e disabilitazione dell'autenticazione database IAM
- Creazione e utilizzo di una policy IAM per l'accesso al database IAM
- Creazione di un account database tramite l'autenticazione IAM

Esempi

Il seguente esempio di codice mostra come generare un token di autenticazione e utilizzarlo per eseguire la connessione a un cluster del database.

Per eseguire questo codice di esempio è necessario AWS SDK per .NET, disponibile sul sito AWS. I pacchetti `AWSSDK.CORE` e `AWSSDK.RDS` sono obbligatori. Per connetterti a un cluster database, utilizza il connettore di database .NET per il motore di database, ad esempio MySQLConnector per MariaDB o MySQL o Npgsql per PostgreSQL.

Questo codice si connette a un cluster di database Aurora MySQL. Modifica i valori delle variabili seguenti in base alle esigenze.

- `server`: l'endpoint del cluster database cui vuoi accedere
- `user` – L'account database cui vuoi accedere.
- `database` – Database a cui accedere.
- `port` – Numero di porta usato per la connessione al dell'istanza database

- `SslMode` – Modalità SSL da utilizzare.

  Quando si utilizza `SslMode=Required`, la connessione SSL verifica l'endpoint del cluster database rispetto a quello nel certificato SSL.
- `SslCa` - Percorso completo del certificato SSL per Amazon Aurora

  Per scaricare un certificato, consultare Utilizzo SSL/TLS per crittografare una connessione a un'.

> **Note**
>
> Non è possibile utilizzare un record DNS Route 53 personalizzato o un endpoint personalizzato Aurora anziché l'endpoint del cluster database per generare il token di autenticazione.

```
using System;
using System.Data;
using MySql.Data;
using MySql.Data.MySqlClient;
using Amazon;

namespace ubuntu
{
  class Program
  {
    static void Main(string[] args)
    {
      var pwd =
 Amazon.RDS.Util.RDSAuthTokenGenerator.GenerateAuthToken(RegionEndpoint.USEast1,
 "mysqlcluster.cluster-123456789012.us-east-1.rds.amazonaws.com", 3306, "jane_doe");
      // for debug only Console.Write("{0}\n", pwd);  //this verifies the token is
 generated

      MySqlConnection conn = new
 MySqlConnection($"server=mysqlcluster.cluster-123456789012.us-
east-1.rds.amazonaws.com;user=jane_doe;database=mydB;port=3306;password={pwd};SslMode=Required;
      conn.Open();

      // Define a query
      MySqlCommand sampleCommand = new MySqlCommand("SHOW DATABASES;", conn);
```

```
      // Execute a query
      MySqlDataReader mysqlDataRdr = sampleCommand.ExecuteReader();

      // Read all rows and output the first column in each row
      while (mysqlDataRdr.Read())
        Console.WriteLine(mysqlDataRdr[0]);

      mysqlDataRdr.Close();
      // Close connection
      conn.Close();
    }
  }
}
```

Questo codice si connette a un cluster di database Aurora PostgreSQL.

Modifica i valori delle variabili seguenti in base alle esigenze.

- `Server`: l'endpoint del cluster database cui vuoi accedere
- `User ID` – L'account database cui vuoi accedere.
- `Database` – Database a cui accedere.
- `Port` – Numero di porta usato per la connessione al dell'istanza database
- `SSL Mode` – Modalità SSL da utilizzare.

  Quando si utilizza `SSL Mode=Required`, la connessione SSL verifica l'endpoint del cluster database rispetto a quello nel certificato SSL.
- `Root Certificate` - Percorso completo del certificato SSL per Amazon Aurora

  Per scaricare un certificato, consultare Utilizzo SSL/TLS per crittografare una connessione a un'.

> **Note**
>
> Non è possibile utilizzare un record DNS Route 53 personalizzato o un endpoint personalizzato Aurora anziché l'endpoint del cluster database per generare il token di autenticazione.

```
using System;
using Npgsql;
```

```
using Amazon.RDS.Util;

namespace ConsoleApp1
{
    class Program
    {
        static void Main(string[] args)
        {
            var pwd =
 RDSAuthTokenGenerator.GenerateAuthToken("postgresmycluster.cluster-123456789012.us-
east-1.rds.amazonaws.com", 5432, "jane_doe");
// for debug only Console.Write("{0}\n", pwd);  //this verifies the token is generated

            NpgsqlConnection conn = new
 NpgsqlConnection($"Server=postgresmycluster.cluster-123456789012.us-
east-1.rds.amazonaws.com;User Id=jane_doe;Password={pwd};Database=mydb;SSL
 Mode=Require;Root Certificate=full_path_to_ssl_certificate");
            conn.Open();

            // Define a query
                   NpgsqlCommand cmd = new NpgsqlCommand("select count(*) FROM
 pg_user", conn);

            // Execute a query
            NpgsqlDataReader dr = cmd.ExecuteReader();

            // Read all rows and output the first column in each row
            while (dr.Read())
                Console.Write("{0}\n", dr[0]);

            // Close connection
            conn.Close();
        }
    }
}
```

Se desideri connetterti a un cluster di database tramite un proxy, consulta Connessione a un database tramite l'autenticazione IAM.

Connessione al dell'istanza database tramite Autenticazione IAM e AWS SDK per Go

Puoi connetterti a un cluster di database Aurora MySQL o Aurora PostgreSQL con la AWS SDK per Go come descritto di seguito.

Prerequisiti

Di seguito sono riportati i prerequisiti per la connessione alcluster di DB utilizzando l'autenticazione IAM:

- Abilitazione e disabilitazione dell'autenticazione database IAM
- Creazione e utilizzo di una policy IAM per l'accesso al database IAM
- Creazione di un account database tramite l'autenticazione IAM

Esempi

Per eseguire questo codice di esempio è necessario AWS SDK per Go, disponibile sul sito AWS.

Modifica i valori delle variabili seguenti in base alle esigenze.

- `dbName` – Database a cui accedere.
- `dbUser` – L'account database cui vuoi accedere.
- `dbHost`: l'endpoint del cluster database cui vuoi accedere

> **Note**
>
> Non è possibile utilizzare un record DNS Route 53 personalizzato o un endpoint personalizzato Aurora anziché l'endpoint del cluster database per generare il token di autenticazione.

- `dbPort` – Numero di porta usato per la connessione al dell'istanza database
- `region`: la regione AWS in cui è in esecuzione il cluster database

Inoltre, assicurarsi che le librerie importate nel codice di esempio esistano nel sistema.

> **Important**
>
> Negli esempi riportati in questa sezione viene utilizzato il codice seguente per fornire credenziali che accedono a un database da un ambiente locale:
> `creds := credentials.NewEnvCredentials()`
> Se si accede a un database da un servizio AWS, ad esempio Amazon EC2 o Amazon ECS, è possibile sostituire il codice con il seguente:
> `sess := session.Must(session.NewSession())`

`creds := sess.Config.Credentials`

Se si apporta questa modifica, assicurarsi di aggiungere la seguente importazione: `"github.com/aws/aws-sdk-go/aws/session"`

Argomenti

- Connessione tramite Autenticazione IAM e AWS SDK per Go V2
- Connessione mediante IAM e AWS SDK per Go V1.

Connessione tramite Autenticazione IAM e AWS SDK per Go V2

È possibile connettersi a uncluster di database utilizzando l'autenticazione IAM e AWS SDK per Go V2.

Il seguente esempio di codice mostra come generare un token di autenticazione e utilizzarlo per eseguire la connessione a un cluster del database.

Questo codice si connette a un cluster di database Aurora MySQL.

```
package main

import (
     "context"
     "database/sql"
     "fmt"

     "github.com/aws/aws-sdk-go-v2/config"
     "github.com/aws/aws-sdk-go-v2/feature/rds/auth"
     _ "github.com/go-sql-driver/mysql"
)

func main() {

     var dbName string = "DatabaseName"
     var dbUser string = "DatabaseUser"
     var dbHost string = "mysqlcluster.cluster-123456789012.us-
east-1.rds.amazonaws.com"
     var dbPort int = 3306
     var dbEndpoint string = fmt.Sprintf("%s:%d", dbHost, dbPort)
     var region string = "us-east-1"
```

```
    cfg, err := config.LoadDefaultConfig(context.TODO())
    if err != nil {
     panic("configuration error: " + err.Error())
    }

    authenticationToken, err := auth.BuildAuthToken(
     context.TODO(), dbEndpoint, region, dbUser, cfg.Credentials)
    if err != nil {
     panic("failed to create authentication token: " + err.Error())
    }

    dsn := fmt.Sprintf("%s:%s@tcp(%s)/%s?tls=true&allowCleartextPasswords=true",
        dbUser, authenticationToken, dbEndpoint, dbName,
    )

    db, err := sql.Open("mysql", dsn)
    if err != nil {
        panic(err)
    }

    err = db.Ping()
    if err != nil {
        panic(err)
    }
}
```

Questo codice si connette a un cluster di database Aurora PostgreSQL.

```
package main

import (
     "context"
     "database/sql"
     "fmt"

     "github.com/aws/aws-sdk-go-v2/config"
     "github.com/aws/aws-sdk-go-v2/feature/rds/auth"
     _ "github.com/lib/pq"
)

func main() {

     var dbName string = "DatabaseName"
```

```
     var dbUser string = "DatabaseUser"
     var dbHost string = "postgresmycluster.cluster-123456789012.us-
east-1.rds.amazonaws.com"
     var dbPort int = 5432
     var dbEndpoint string = fmt.Sprintf("%s:%d", dbHost, dbPort)
     var region string = "us-east-1"

    cfg, err := config.LoadDefaultConfig(context.TODO())
    if err != nil {
     panic("configuration error: " + err.Error())
    }

    authenticationToken, err := auth.BuildAuthToken(
     context.TODO(), dbEndpoint, region, dbUser, cfg.Credentials)
    if err != nil {
     panic("failed to create authentication token: " + err.Error())
    }

    dsn := fmt.Sprintf("host=%s port=%d user=%s password=%s dbname=%s",
        dbHost, dbPort, dbUser, authenticationToken, dbName,
    )

    db, err := sql.Open("postgres", dsn)
    if err != nil {
        panic(err)
    }

    err = db.Ping()
    if err != nil {
        panic(err)
    }
}
```

Se desideri connetterti a un cluster di database tramite un proxy, consulta Connessione a un database tramite l'autenticazione IAM.

Connessione mediante IAM e AWS SDK per Go V1.

È possibile connettersi a uncluster di database utilizzando l'autenticazione IAM e AWS SDK per Go V1

Il seguente esempio di codice mostra come generare un token di autenticazione e utilizzarlo per eseguire la connessione a un cluster del database.

Questo codice si connette a un cluster di database Aurora MySQL.

```
package main

import (
    "database/sql"
    "fmt"
    "log"

    "github.com/aws/aws-sdk-go/aws/credentials"
    "github.com/aws/aws-sdk-go/service/rds/rdsutils"
    _ "github.com/go-sql-driver/mysql"
)

func main() {
    dbName := "app"
    dbUser := "jane_doe"
    dbHost := "mysqlcluster.cluster-123456789012.us-east-1.rds.amazonaws.com"
    dbPort := 3306
    dbEndpoint := fmt.Sprintf("%s:%d", dbHost, dbPort)
    region := "us-east-1"

    creds := credentials.NewEnvCredentials()
    authToken, err := rdsutils.BuildAuthToken(dbEndpoint, region, dbUser, creds)
    if err != nil {
        panic(err)
    }

    dsn := fmt.Sprintf("%s:%s@tcp(%s)/%s?tls=true&allowCleartextPasswords=true",
        dbUser, authToken, dbEndpoint, dbName,
    )

    db, err := sql.Open("mysql", dsn)
    if err != nil {
        panic(err)
    }

    err = db.Ping()
    if err != nil {
        panic(err)
    }
}
```

Questo codice si connette a un cluster di database Aurora PostgreSQL.

```
package main

import (
 "database/sql"
 "fmt"

 "github.com/aws/aws-sdk-go/aws/credentials"
 "github.com/aws/aws-sdk-go/service/rds/rdsutils"
 _ "github.com/lib/pq"
)

func main() {
    dbName := "app"
    dbUser := "jane_doe"
    dbHost := "postgresmycluster.cluster-123456789012.us-east-1.rds.amazonaws.com"
    dbPort := 5432
    dbEndpoint := fmt.Sprintf("%s:%d", dbHost, dbPort)
    region := "us-east-1"

    creds := credentials.NewEnvCredentials()
    authToken, err := rdsutils.BuildAuthToken(dbEndpoint, region, dbUser, creds)
    if err != nil {
        panic(err)
    }

    dsn := fmt.Sprintf("host=%s port=%d user=%s password=%s dbname=%s",
        dbHost, dbPort, dbUser, authToken, dbName,
    )

    db, err := sql.Open("postgres", dsn)
    if err != nil {
        panic(err)
    }

    err = db.Ping()
    if err != nil {
        panic(err)
    }
}
```

Se desideri connetterti a un cluster di database tramite un proxy, consulta Connessione a un database tramite l'autenticazione IAM.

Connessione al dell'istanza database tramite Autenticazione IAM e AWS SDK per Java

Puoi connetterti a un cluster di database Aurora MySQL o Aurora PostgreSQL con la AWS SDK per Java come descritto di seguito.

Prerequisiti

Di seguito sono riportati i prerequisiti per la connessione alcluster di DB utilizzando l'autenticazione IAM:

- Abilitazione e disabilitazione dell'autenticazione database IAM
- Creazione e utilizzo di una policy IAM per l'accesso al database IAM
- Creazione di un account database tramite l'autenticazione IAM
- Configura l'SDK AWS per Java

Per esempi su come utilizzare l'SDK per Java 2.x, consulta Esempi per Amazon RDS con SDK per Java 2.x. È anche possibile utilizzare il wrapper JDBC avanzato AWS, per informazioni consulta AWS Advanced JDBC Wrapper documentation.

Argomenti

- Generazione di un token di autenticazione IAM
- Creazione manuale di un token di autenticazione IAM
- Connessione a un cluster database

Generazione di un token di autenticazione IAM

Se scrivi programmi usando l'AWS SDK per Java, puoi ottenere un token di autenticazione firmato tramite la classe `RdsIamAuthTokenGenerator`. L'utilizzo di questa classe richiede che vengano fornite le credenziali AWS. A questo scopo, devi creare un'istanza della classe `DefaultAWSCredentialsProviderChain`. `DefaultAWSCredentialsProviderChain` utilizza la prima chiave di accesso e la chiave segreta AWS che trova nella catena di provider delle credenziali predefinite. Per ulteriori informazioni sulle chiavi di accesso AWS, consulta Gestione delle chiavi di accesso per gli utenti IAM.

> **Note**
>
> Non è possibile utilizzare un record DNS Route 53 personalizzato o un endpoint personalizzato Aurora anziché l'endpoint del cluster database per generare il token di autenticazione.

Dopo aver creato un'istanza di `RdsIamAuthTokenGenerator`, puoi chiamare il metodo `getAuthToken` per ottenere un token firmato. Specifica la regione AWS, il nome host, il numero di porta e il nome utente. Il seguente esempio di codice dimostra come eseguire questa operazione.

```
package com.amazonaws.codesamples;

import com.amazonaws.auth.DefaultAWSCredentialsProviderChain;
import com.amazonaws.services.rds.auth.GetIamAuthTokenRequest;
import com.amazonaws.services.rds.auth.RdsIamAuthTokenGenerator;

public class GenerateRDSAuthToken {

    public static void main(String[] args) {

     String region = "us-west-2";
     String hostname = "rdsmysql.123456789012.us-west-2.rds.amazonaws.com";
     String port = "3306";
     String username = "jane_doe";

     System.out.println(generateAuthToken(region, hostname, port, username));
    }

    static String generateAuthToken(String region, String hostName, String port, String
 username) {

     RdsIamAuthTokenGenerator generator = RdsIamAuthTokenGenerator.builder()
      .credentials(new DefaultAWSCredentialsProviderChain())
      .region(region)
      .build();

     String authToken = generator.getAuthToken(
      GetIamAuthTokenRequest.builder()
      .hostname(hostName)
      .port(Integer.parseInt(port))
      .userName(username)
```

```
      .build());

     return authToken;
    }

}
```

Creazione manuale di un token di autenticazione IAM

In Java, il modo più semplice di generare un token di autenticazione è di utilizzare `RdsIamAuthTokenGenerator`. Questa classe crea un token di autenticazione e lo firma utilizzando AWS Signature Version 4. Per ulteriori informazioni, consulta la pagina relativa al processo di firma Signature Version 4 nella Riferimenti generali di AWS.

Tuttavia, puoi anche creare e firmare un token di autenticazione manualmente, come visualizzato nel seguente esempio di codice.

```
package com.amazonaws.codesamples;

import com.amazonaws.SdkClientException;
import com.amazonaws.auth.DefaultAWSCredentialsProviderChain;
import com.amazonaws.auth.SigningAlgorithm;
import com.amazonaws.util.BinaryUtils;
import org.apache.commons.lang3.StringUtils;

import javax.crypto.Mac;
import javax.crypto.spec.SecretKeySpec;
import java.nio.charset.Charset;
import java.security.MessageDigest;
import java.text.SimpleDateFormat;
import java.util.Date;
import java.util.SortedMap;
import java.util.TreeMap;

import static com.amazonaws.auth.internal.SignerConstants.AWS4_TERMINATOR;
import static com.amazonaws.util.StringUtils.UTF8;

public class CreateRDSAuthTokenManually {
    public static String httpMethod = "GET";
    public static String action = "connect";
    public static String canonicalURIParameter = "/";
    public static SortedMap<String, String> canonicalQueryParameters = new TreeMap();
    public static String payload = StringUtils.EMPTY;
```

```
    public static String signedHeader = "host";
    public static String algorithm = "AWS4-HMAC-SHA256";
    public static String serviceName = "rds-db";
    public static String requestWithoutSignature;

    public static void main(String[] args) throws Exception {

        String region = "us-west-2";
        String instanceName = "rdsmysql.123456789012.us-west-2.rds.amazonaws.com";
        String port = "3306";
        String username = "jane_doe";

        Date now = new Date();
        String date = new SimpleDateFormat("yyyyMMdd").format(now);
        String dateTimeStamp = new
SimpleDateFormat("yyyyMMdd'T'HHmmss'Z'").format(now);
       DefaultAWSCredentialsProviderChain creds = new
DefaultAWSCredentialsProviderChain();
     String awsAccessKey = creds.getCredentials().getAWSAccessKeyId();
     String awsSecretKey = creds.getCredentials().getAWSSecretKey();
        String expiryMinutes = "900";

        System.out.println("Step 1:  Create a canonical request:");
        String canonicalString = createCanonicalString(username, awsAccessKey, date,
dateTimeStamp, region, expiryMinutes, instanceName, port);
        System.out.println(canonicalString);
        System.out.println();

        System.out.println("Step 2:  Create a string to sign:");
        String stringToSign = createStringToSign(dateTimeStamp, canonicalString,
awsAccessKey, date, region);
        System.out.println(stringToSign);
        System.out.println();

        System.out.println("Step 3:  Calculate the signature:");
        String signature = BinaryUtils.toHex(calculateSignature(stringToSign,
newSigningKey(awsSecretKey, date, region, serviceName)));
        System.out.println(signature);
        System.out.println();

        System.out.println("Step 4:  Add the signing info to the request");

        System.out.println(appendSignature(signature));
        System.out.println();
```

```
   }

   //Step 1: Create a canonical request date should be in format YYYYMMDD and dateTime
 should be in format YYYYMMDDTHHMMSSZ
   public static String createCanonicalString(String user, String accessKey, String
 date, String dateTime, String region, String expiryPeriod, String hostName, String
 port) throws Exception {
       canonicalQueryParameters.put("Action", action);
       canonicalQueryParameters.put("DBUser", user);
       canonicalQueryParameters.put("X-Amz-Algorithm", "AWS4-HMAC-SHA256");
       canonicalQueryParameters.put("X-Amz-Credential", accessKey + "%2F" + date +
"%2F" + region + "%2F" + serviceName + "%2Faws4_request");
       canonicalQueryParameters.put("X-Amz-Date", dateTime);
       canonicalQueryParameters.put("X-Amz-Expires", expiryPeriod);
       canonicalQueryParameters.put("X-Amz-SignedHeaders", signedHeader);
       String canonicalQueryString = "";
       while(!canonicalQueryParameters.isEmpty()) {
           String currentQueryParameter = canonicalQueryParameters.firstKey();
           String currentQueryParameterValue =
 canonicalQueryParameters.remove(currentQueryParameter);
           canonicalQueryString = canonicalQueryString + currentQueryParameter + "=" +
 currentQueryParameterValue;
           if (!currentQueryParameter.equals("X-Amz-SignedHeaders")) {
               canonicalQueryString += "&";
           }
       }
       String canonicalHeaders = "host:" + hostName + ":" + port + '\n';
       requestWithoutSignature = hostName + ":" + port + "/?" + canonicalQueryString;

       String hashedPayload = BinaryUtils.toHex(hash(payload));
       return httpMethod + '\n' + canonicalURIParameter + '\n' + canonicalQueryString
 + '\n' + canonicalHeaders + '\n' + signedHeader + '\n' + hashedPayload;

   }

   //Step 2: Create a string to sign using sig v4
   public static String createStringToSign(String dateTime, String canonicalRequest,
 String accessKey, String date, String region) throws Exception {
       String credentialScope = date + "/" + region + "/" + serviceName + "/
aws4_request";
       return algorithm + '\n' + dateTime + '\n' + credentialScope + '\n' +
 BinaryUtils.toHex(hash(canonicalRequest));
```

```
    }

    //Step 3: Calculate signature
    /**
     * Step 3 of the &AWS; Signature version 4 calculation. It involves deriving
     * the signing key and computing the signature. Refer to
     * http://docs.aws.amazon
     * .com/general/latest/gr/sigv4-calculate-signature.html
     */
    public static byte[] calculateSignature(String stringToSign,
                                            byte[] signingKey) {
        return sign(stringToSign.getBytes(Charset.forName("UTF-8")), signingKey,
                SigningAlgorithm.HmacSHA256);
    }

    public static byte[] sign(byte[] data, byte[] key,
                          SigningAlgorithm algorithm) throws SdkClientException {
        try {
            Mac mac = algorithm.getMac();
            mac.init(new SecretKeySpec(key, algorithm.toString()));
            return mac.doFinal(data);
        } catch (Exception e) {
            throw new SdkClientException(
                    "Unable to calculate a request signature: "
                            + e.getMessage(), e);
        }
    }

    public static byte[] newSigningKey(String secretKey,
                                   String dateStamp, String regionName, String
serviceName) {
        byte[] kSecret = ("AWS4" + secretKey).getBytes(Charset.forName("UTF-8"));
        byte[] kDate = sign(dateStamp, kSecret, SigningAlgorithm.HmacSHA256);
        byte[] kRegion = sign(regionName, kDate, SigningAlgorithm.HmacSHA256);
        byte[] kService = sign(serviceName, kRegion,
                SigningAlgorithm.HmacSHA256);
        return sign(AWS4_TERMINATOR, kService, SigningAlgorithm.HmacSHA256);
    }

    public static byte[] sign(String stringData, byte[] key,
                      SigningAlgorithm algorithm) throws SdkClientException {
        try {
            byte[] data = stringData.getBytes(UTF8);
            return sign(data, key, algorithm);
```

```
        } catch (Exception e) {
            throw new SdkClientException(
                    "Unable to calculate a request signature: "
                            + e.getMessage(), e);
        }
    }

    //Step 4: append the signature
    public static String appendSignature(String signature) {
        return requestWithoutSignature + "&X-Amz-Signature=" + signature;
    }

    public static byte[] hash(String s) throws Exception {
        try {
            MessageDigest md = MessageDigest.getInstance("SHA-256");
            md.update(s.getBytes(UTF8));
            return md.digest();
        } catch (Exception e) {
            throw new SdkClientException(
                    "Unable to compute hash while signing request: "
                            + e.getMessage(), e);
        }
    }
}
```

Connessione a un cluster database

Il seguente esempio di codice mostra come generare un token di autenticazione e utilizzarlo per eseguire la connessione a un cluster eseguendo Aurora MySQL.

Per eseguire questo codice di esempio è necessario AWS SDK per Java, disponibile sul sito AWS. Inoltre, hai bisogno di quanto segue:

- MySQL Connector/J. Questo esempio di codice è stato testato con `mysql-connector-java-5.1.33-bin.jar`.
- Un certificato intermedio per Amazon Aurora specifico per una regione AWS. Per ulteriori informazioni, consulta Utilizzo SSL/TLS per crittografare una connessione a un'. Durante il runtime, il loader della classe cerca un certificato nella stessa directory di questo esempio di codice Java, per permettere al loader della classe di trovarlo.
- Modifica i valori delle variabili seguenti in base alle esigenze.
  - `RDS_INSTANCE_HOSTNAME`: il nome host del cluster database cui vuoi accedere.

- `RDS_INSTANCE_PORT`: il numero di porta usato per la connessione al cluster database PostgreSQL.
- `REGION_NAME`: la regione AWS in cui è in esecuzione il cluster database.
- `DB_USER`: l'account database cui vuoi accedere.
- `SSL_CERTIFICATE`: un certificato intermedio per Amazon Aurora specifico per una regione AWS.

  Per scaricare un certificato per la regione AWS, consulta Utilizzo SSL/TLS per crittografare una connessione a un'. Inserisci il certificato SSL nella stessa directory di questo file di programma Java, per permettere al loader di classe di trovare il certificato durante il runtime.

Questo esempio di codice ottiene le credenziali AWS dalla catena di provider delle credenziali predefinita.

> **Note**
>
> Specifica una password per `DEFAULT_KEY_STORE_PASSWORD` diversa da quella mostrata qui come best practice di sicurezza.

```
package com.amazonaws.samples;

import com.amazonaws.services.rds.auth.RdsIamAuthTokenGenerator;
import com.amazonaws.services.rds.auth.GetIamAuthTokenRequest;
import com.amazonaws.auth.BasicAWSCredentials;
import com.amazonaws.auth.DefaultAWSCredentialsProviderChain;
import com.amazonaws.auth.AWSStaticCredentialsProvider;

import java.io.File;
import java.io.FileOutputStream;
import java.io.InputStream;
import java.security.KeyStore;
import java.security.cert.CertificateFactory;
import java.security.cert.X509Certificate;

import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.ResultSet;
import java.sql.Statement;
```

```
import java.util.Properties;

import java.net.URL;

public class IAMDatabaseAuthenticationTester {
    //&AWS; Credentials of the IAM user with policy enabling IAM Database Authenticated
 access to the db by the db user.
    private static final DefaultAWSCredentialsProviderChain creds = new
 DefaultAWSCredentialsProviderChain();
    private static final String AWS_ACCESS_KEY =
 creds.getCredentials().getAWSAccessKeyId();
    private static final String AWS_SECRET_KEY =
 creds.getCredentials().getAWSSecretKey();

    //Configuration parameters for the generation of the IAM Database Authentication
 token
    private static final String RDS_INSTANCE_HOSTNAME = "rdsmysql.123456789012.us-
west-2.rds.amazonaws.com";
    private static final int RDS_INSTANCE_PORT = 3306;
    private static final String REGION_NAME = "us-west-2";
    private static final String DB_USER = "jane_doe";
    private static final String JDBC_URL = "jdbc:mysql://" + RDS_INSTANCE_HOSTNAME +
 ":" + RDS_INSTANCE_PORT;

    private static final String SSL_CERTIFICATE = "rds-ca-2019-us-west-2.pem";

    private static final String KEY_STORE_TYPE = "JKS";
    private static final String KEY_STORE_PROVIDER = "SUN";
    private static final String KEY_STORE_FILE_PREFIX = "sys-connect-via-ssl-test-
cacerts";
    private static final String KEY_STORE_FILE_SUFFIX = ".jks";
    private static final String DEFAULT_KEY_STORE_PASSWORD = "changeit";

    public static void main(String[] args) throws Exception {
        //get the connection
        Connection connection = getDBConnectionUsingIam();

        //verify the connection is successful
        Statement stmt= connection.createStatement();
        ResultSet rs=stmt.executeQuery("SELECT 'Success!' FROM DUAL;");
        while (rs.next()) {
             String id = rs.getString(1);
            System.out.println(id); //Should print "Success!"
        }
```

```
        //close the connection
        stmt.close();
        connection.close();

        clearSslProperties();

    }

    /**
     * This method returns a connection to the db instance authenticated using IAM
 Database Authentication
     * @return
     * @throws Exception
     */
    private static Connection getDBConnectionUsingIam() throws Exception {
        setSslProperties();
        return DriverManager.getConnection(JDBC_URL, setMySqlConnectionProperties());
    }

    /**
     * This method sets the mysql connection properties which includes the IAM Database
 Authentication token
     * as the password. It also specifies that SSL verification is required.
     * @return
     */
    private static Properties setMySqlConnectionProperties() {
        Properties mysqlConnectionProperties = new Properties();
        mysqlConnectionProperties.setProperty("verifyServerCertificate","true");
        mysqlConnectionProperties.setProperty("useSSL", "true");
        mysqlConnectionProperties.setProperty("user",DB_USER);
        mysqlConnectionProperties.setProperty("password",generateAuthToken());
        return mysqlConnectionProperties;
    }

    /**
     * This method generates the IAM Auth Token.
     * An example IAM Auth Token would look like follows:
     * btusi123.cmz7kenwo2ye.rds.cn-north-1.amazonaws.com.cn:3306/?
Action=connect&DBUser=iamtestuser&X-Amz-Algorithm=AWS4-HMAC-SHA256&X-Amz-
Date=20171003T010726Z&X-Amz-SignedHeaders=host&X-Amz-Expires=899&X-Amz-
Credential=AKIAPFXHGVDI5RNFO4AQ%2F20171003%2Fcn-north-1%2Frds-db%2Faws4_request&X-Amz-
Signature=f9f45ef96c1f770cdad11a53e33ffa4c3730bc03fdee820cfdf1322eed15483b
     * @return
```

```
     */
    private static String generateAuthToken() {
        BasicAWSCredentials awsCredentials = new BasicAWSCredentials(AWS_ACCESS_KEY,
AWS_SECRET_KEY);

        RdsIamAuthTokenGenerator generator = RdsIamAuthTokenGenerator.builder()
                .credentials(new
AWSStaticCredentialsProvider(awsCredentials)).region(REGION_NAME).build();
        return generator.getAuthToken(GetIamAuthTokenRequest.builder()

.hostname(RDS_INSTANCE_HOSTNAME).port(RDS_INSTANCE_PORT).userName(DB_USER).build());
    }

    /**
     * This method sets the SSL properties which specify the key store file, its type
and password:
     * @throws Exception
     */
    private static void setSslProperties() throws Exception {
        System.setProperty("javax.net.ssl.trustStore", createKeyStoreFile());
        System.setProperty("javax.net.ssl.trustStoreType", KEY_STORE_TYPE);
        System.setProperty("javax.net.ssl.trustStorePassword",
DEFAULT_KEY_STORE_PASSWORD);
    }

    /**
     * This method returns the path of the Key Store File needed for the SSL
verification during the IAM Database Authentication to
     * the db instance.
     * @return
     * @throws Exception
     */
    private static String createKeyStoreFile() throws Exception {
        return createKeyStoreFile(createCertificate()).getPath();
    }

    /**
     *  This method generates the SSL certificate
     * @return
     * @throws Exception
     */
    private static X509Certificate createCertificate() throws Exception {
        CertificateFactory certFactory = CertificateFactory.getInstance("X.509");
        URL url = new File(SSL_CERTIFICATE).toURI().toURL();
```

```
        if (url == null) {
            throw new Exception();
        }
        try (InputStream certInputStream = url.openStream()) {
            return (X509Certificate) certFactory.generateCertificate(certInputStream);
        }
    }

    /**
     * This method creates the Key Store File
     * @param rootX509Certificate - the SSL certificate to be stored in the KeyStore
     * @return
     * @throws Exception
     */
    private static File createKeyStoreFile(X509Certificate rootX509Certificate) throws
 Exception {
        File keyStoreFile = File.createTempFile(KEY_STORE_FILE_PREFIX,
 KEY_STORE_FILE_SUFFIX);
        try (FileOutputStream fos = new FileOutputStream(keyStoreFile.getPath())) {
            KeyStore ks = KeyStore.getInstance(KEY_STORE_TYPE, KEY_STORE_PROVIDER);
            ks.load(null);
            ks.setCertificateEntry("rootCaCertificate", rootX509Certificate);
            ks.store(fos, DEFAULT_KEY_STORE_PASSWORD.toCharArray());
        }
        return keyStoreFile;
    }

    /**
     * This method clears the SSL properties.
     * @throws Exception
     */
    private static void clearSslProperties() throws Exception {
           System.clearProperty("javax.net.ssl.trustStore");
           System.clearProperty("javax.net.ssl.trustStoreType");
           System.clearProperty("javax.net.ssl.trustStorePassword");
    }

}
```

Se desideri connetterti a un cluster di database tramite un proxy, consulta Connessione a un database tramite l'autenticazione IAM.

Connessione al dell'istanza database tramite Autenticazione IAM e AWS SDK per Python (Boto3)

Puoi connetterti a un cluster di database Aurora MySQL o Aurora PostgreSQL con la AWS SDK per Python (Boto3) come descritto di seguito.

Prerequisiti

Di seguito sono riportati i prerequisiti per la connessione alcluster di DB utilizzando l'autenticazione IAM:

- Abilitazione e disabilitazione dell'autenticazione database IAM
- Creazione e utilizzo di una policy IAM per l'accesso al database IAM
- Creazione di un account database tramite l'autenticazione IAM

Inoltre, assicurarsi che le librerie importate nel codice di esempio esistano nel sistema.

Esempi

Gli esempi di codice utilizzano i profili per le credenziali condivise. Per informazioni sulla specifica delle credenziali, vedere Credenziali nella documentazione di AWS SDK per Python (Boto3).

Il seguente esempio di codice mostra come generare un token di autenticazione e utilizzarlo per eseguire la connessione a un cluster del database.

Per eseguire questo codice di esempio è necessario AWS SDK per Python (Boto3), disponibile sul sito AWS.

Modifica i valori delle variabili seguenti in base alle esigenze.

- `ENDPOINT`: l'endpoint del cluster di database cui vuoi accedere
- `PORT` – Numero di porta usato per la connessione al dell'istanza database
- `USER` – L'account database cui vuoi accedere.
- `REGION`: la regione AWS in cui è in esecuzione il cluster di database
- `DBNAME` – Database a cui accedere.
- `SSLCERTIFICATE` - Percorso completo del certificato SSL per Amazon Aurora

  Per `ssl_ca`, specificare un certificato SSL. Per scaricare un certificato SSL consulta Utilizzo SSL/TLS per crittografare una connessione a un'

> **Note**
>
> Non è possibile utilizzare un record DNS Route 53 personalizzato o un endpoint personalizzato Aurora anziché l'endpoint del cluster database per generare il token di autenticazione.

Questo codice si connette a un cluster di database Aurora MySQL.

Prima di eseguire questo codice, installa il driver PyMySQL seguendo le istruzioni in Python Package Index (Indice dei pacchetti Python).

```
import pymysql
import sys
import boto3
import os

ENDPOINT="mysqlcluster.cluster-123456789012.us-east-1.rds.amazonaws.com"
PORT="3306"
USER="jane_doe"
REGION="us-east-1"
DBNAME="mydb"
os.environ['LIBMYSQL_ENABLE_CLEARTEXT_PLUGIN'] = '1'

#gets the credentials from .aws/credentials
session = boto3.Session(profile_name='default')
client = session.client('rds')

token = client.generate_db_auth_token(DBHostname=ENDPOINT, Port=PORT, DBUsername=USER,
 Region=REGION)

try:
    conn =
 pymysql.connect(auth_plugin_map={'mysql_clear_password':None},host=ENDPOINT,
 user=USER, password=token, port=PORT, database=DBNAME, ssl_ca='SSLCERTIFICATE',
 ssl_verify_identity=True, ssl_verify_cert=True)
    cur = conn.cursor()
    cur.execute("""SELECT now()""")
    query_results = cur.fetchall()
    print(query_results)
except Exception as e:
    print("Database connection failed due to {}".format(e))
```

Questo codice si connette a un cluster di database Aurora PostgreSQL.

Prima di eseguire questo codice, installa`psycopg2`seguendo le istruzioni in Documentazione di Psycopg.

```
import psycopg2
import sys
import boto3
import os

ENDPOINT="postgresmycluster.cluster-123456789012.us-east-1.rds.amazonaws.com"
PORT="5432"
USER="jane_doe"
REGION="us-east-1"
DBNAME="mydb"

#gets the credentials from .aws/credentials
session = boto3.Session(profile_name='RDSCreds')
client = session.client('rds')

token = client.generate_db_auth_token(DBHostname=ENDPOINT, Port=PORT, DBUsername=USER,
 Region=REGION)

try:
    conn = psycopg2.connect(host=ENDPOINT, port=PORT, database=DBNAME, user=USER,
 password=token, sslrootcert="SSLCERTIFICATE")
    cur = conn.cursor()
    cur.execute("""SELECT now()""")
    query_results = cur.fetchall()
    print(query_results)
except Exception as e:
    print("Database connection failed due to {}".format(e))
```

Se desideri connetterti a un cluster di database tramite un proxy, consulta Connessione a un database tramite l'autenticazione IAM.

Risoluzione dei problemi relativi all'autenticazione database IAM

Di seguito, è possibile trovare idee per la risoluzione dei problemi comuni relativi all'autenticazione database IAM e informazioni sui log e sulle metriche di CloudWatch relativi all'autenticazione database IAM.

Esportazione di log degli errori di autenticazione database IAM su CloudWatch Logs

I log degli errori di autenticazione database IAM sono archiviati sull'host del database e li puoi esportare nel tuo account CloudWatch Logs. Utilizza i log e i metodi di correzione in questa pagina per risolvere i problemi relativi all'autenticazione database IAM.

È possibile abilitare le esportazioni dei log verso CloudWatch Logs dalla console, dalla AWS CLI e dall'API RDS. Per istruzioni relative alla console, consulta Pubblicazione di log di database su Amazon CloudWatch Logs.

Per esportare i log degli errori di autenticazione database IAM in CloudWatch Logs durante la modifica di un cluster di database dalla AWS CLI, usa il seguente comando:

```
aws rds create-db-cluster --db-cluster-identifier mydbinstance \
--region us-east-1 \
--engine postgres \
--engine-version 16 \
--master-username master \
--master-user-password password \
--publicly-accessible \
--enable-iam-database-authentication \
--enable-cloudwatch-logs-exports=iam-db-auth-error
```

Per esportare i log degli errori di autenticazione database IAM in CloudWatch Logs durante la modifica di un cluster di database dalla AWS CLI, usa il seguente comando:

```
aws rds modify-db-cluster --db-cluster-identifier mydbcluster \
--region us-east-1 \
--cloudwatch-logs-export-configuration '{"EnableLogTypes":["iam-db-auth-error"]}'
```

Per verificare se il cluster di database sta esportando i log di autenticazione database IAM su CloudWatch Logs, controlla se il parametro `EnabledCloudwatchLogsExports` è impostato a `iam-db-auth-error` nell'output del comando `describe-db-instances`.

```
aws rds describe-db-cluster --region us-east-1 --db-cluster-identifier mydbcluster
```

```
            ...

             "EnabledCloudwatchLogsExports": [
                "iam-db-auth-error"
            ],
            ...
```

Metriche CloudWatch relative all'autenticazione database IAM

Amazon Aurora fornisce metriche quasi in tempo reale sull'autenticazione database IAM al tuo account Amazon CloudWatch. Nella tabella seguente sono elencati le metriche di autenticazione database IAM e le dimensioni disponibili con CloudWatch:

Parametro	Descrizione
`IamDbAuthConnectionRequests`	Numero totale di richieste di connessione effettuate con l'autenticazione database IAM.
`IamDbAuthConnectionSuccess`	Numero totale di richieste di autenticazione database IAM riuscite.
`IamDbAuthConnectionFailure`	Numero totale di richieste di autenticazione database IAM non riuscite.
`IamDbAuthConnectionFailureInvalidToken`	Numero totale di richieste di autenticazione database IAM non riuscite a causa di un token non valido.
`IamDbAuthConnectionFailureInsufficientPermissions`	Numero totale di richieste di autenticazione database IAM non riuscite a causa di policy o autorizzazioni errate.
`IamDbAuthConnectionFailureThrottling`	Numero totale di richieste di autenticazione database IAM non riuscite a causa della limitazione (della larghezza di banda della rete) per l'autenticazione database IAM.
`IamDbAuthConnectionFailureServerError`	Numero totale di richieste di autenticazione database IAM non riuscite a causa di un errore

Parametro	Descrizione
	interno del server nella funzionalità di autentica zione database IAM.

Problemi e soluzioni comuni

È possibile che si verifichino i seguenti problemi durante l'utilizzo dell'autenticazione database IAM. Utilizza i passaggi di correzione indicati nella tabella per risolvere i problemi:

Errore	Metrica/metriche	Causa	Soluzione

`[ERROR] Failed to authenticate the connectio n request for user db_user because the provided token is malformed or otherwise invalid. (Status Code: 400, Error Code: InvalidTo ken)`	`IamDbAuth Connectio nFailure`
	
	`IamDbAuth Connectio nFailureI nvalidToken`
		Il token di autentica zione database IAM nella richiesta di connessione non è un token SigV4a valido o non è formattato correttamente.	Controlla la tua strategia di generazio ne di token nella tua applicazione. In alcuni casi, assicurati di passare il token con una formattazione valida. La troncatur a del token (o la formattazione errata della stringa) renderà il token non valido.

`[ERROR] Failed to authenticate the connection request for user db_user because the token age is longer than 15 minutes. (Status Code: 400,`	`IamDbAuth Connectio nFailure`
	
	`IamDbAuth Connectio nFailureI nvalidToken`
		Il token di autentica zione database IAM è scaduto. I token sono validi solo per 15 minuti.	Controlla la logica di memorizzazione nella cache dei token e/o di riutilizzo dei token nell'applicazione. Non riutilizzare token più vecchi di 15 minuti.

Errore	Metrica/metriche	Causa	Soluzione
`Error Code:ExpiredToken)`			

`[ERROR] Failed to authorize the connection request for user` *`db_user`* `because the IAM policy assumed by the caller 'arn:aws:sts::123456789012:assumed-role/ <RoleName>/ <RoleSession>' is not authorized to perform` `` `rds-db:connect` `` `on the DB instance. (Status Code: 403, Error Code:NotAuthorized)`	`IamDbAuthConnectionFailure`
	
	`IamDbAuthConnectionFailureInsufficientPermissions`
		Questo errore potrebbe essere dovuto ai seguenti fattori:
		• La policy IAM assunta dall'applicazione non autorizza l'operazione `rds-db:connect` .
		• Stai assumendo il ruolo/la policy errati per la connessione di *`db_user`* al database.
		• Stai assumendo la policy corretta per *`db_user`*, ma non ti stai connettendo al database corretto.
			Verifica il ruolo e/o la policy IAM che stai assumendo nella tua applicazione. Assicurati di assumere la stessa policy per generare il token utilizzata per la connessione al database.

Errore	Metrica/metriche	Causa	Soluzione

`[ERROR] Failed to authorize the connection request for user db_user due to IAM DB authentication throttling. (Status Code: 429, Error Code: ThrottlingException)`	`IamDbAuthConnectionFailure`
	
	`IamDbAuthConnectionFailureThrottling`
		Stai effettuando troppe richieste di connessione al tuo database in un breve lasso di tempo. La limitazione (della larghezza di banda della rete) per l'autenticazione database IAM è di 200 connessioni al secondo.	Riduci la velocità di creazione di nuove connessioni con l'autenticazione IAM. Prendi in considerazione l'implementazione del pool di connessioni utilizzando Server proxy per RDS per riutilizzare le connessioni stabilite nell'applicazione.

`[ERROR] Failed to authorize the connection request for user db_user due to an internal IAM DB authentication error. (Status Code: 500, Error Code: InternalError)`	`IamDbAuthConnectionFailure`
	
	`IamDbAuthConnectionFailureThrottling`
		Si è verificato un errore interno durante l'autorizzazione della connessione database con l'autenticazione database IAM.	Contatta https://aws.amazon.com/premiumsupport/ per indagare sul problema.

Risoluzione dei problemi di identità e accesso in Amazon Aurora

Utilizza le informazioni seguenti per diagnosticare e risolvere i problemi comuni che possono verificarsi durante l'utilizzo di Aurora e IAM.

Argomenti

- Non sono autorizzato a eseguire un'operazione in Aurora
- Non sono autorizzato a eseguire: iam:PassRole
- Voglio consentire a persone esterne al mio account AWS di accedere alle mie risorse Aurora

Non sono autorizzato a eseguire un'operazione in Aurora

Se la Console di gestione AWS indica che non sei autorizzato a eseguire un'azione, devi contattare l'amministratore per ricevere assistenza. L'amministratore è colui che ti ha fornito le credenziali di accesso.

L'errore di esempio seguente si verifica quando l'utente `mateojackson` cerca di utilizzare la console per visualizzare i dettagli relativi a un *`widget`*, ma non dispone di autorizzazioni `rds:`*`GetWidget`*.

```
User: arn:aws:iam::123456789012:user/mateojackson is not authorized to perform:
 rds:GetWidget on resource: my-example-widget
```

In questo caso, Mateo richiede al suo amministratore di aggiornare le sue policy per poter accedere alla risorsa *`my-example-widget`* utilizzando l'operazione `rds:`*`GetWidget`*.

Non sono autorizzato a eseguire: iam:PassRole

Se ricevi un errore che indica che non sei autorizzato a eseguire l'operazione `iam:PassRole`, devi contattare il tuo amministratore per ricevere assistenza. L'amministratore è colui che ti ha fornito le credenziali di accesso. Richiedi a tale persona di aggiornare le tue policy per poter passare un ruolo a Aurora.

Alcuni servizi AWS consentono di passare un ruolo esistente a tale servizio, invece di creare un nuovo ruolo del servizio o ruolo collegato ai servizi. Per eseguire questa operazione, è necessario disporre delle autorizzazioni per passare il ruolo al servizio.

L'errore di esempio seguente si verifica quando un utente denominato `marymajor` cerca di utilizzare la console per eseguire un'operazione in Aurora. Tuttavia, l'operazione richiede che il servizio disponga delle autorizzazioni concesse da un ruolo di servizio. Mary non dispone di autorizzazioni per passare il ruolo al servizio.

```
User: arn:aws:iam::123456789012:user/marymajor is not authorized to perform:
 iam:PassRole
```

In questo caso, Mary chiede all'amministratore di aggiornare la sue policy per poter eseguire l'operazione `iam:PassRole`.

Voglio consentire a persone esterne al mio account AWS di accedere alle mie risorse Aurora

È possibile creare un ruolo con il quale utenti in altri account o persone esterne all'organizzazione possono accedere alle tue risorse. È possibile specificare chi è attendibile per l'assunzione del ruolo. Per servizi che supportano policy basate su risorse o liste di controllo degli accessi (ACL), utilizza tali policy per concedere alle persone l'accesso alle tue risorse.

Per ulteriori informazioni, consulta gli argomenti seguenti:

- Per capire se Aurora supporta queste funzionalità, consulta Funzionamento di Amazon Aurora con IAM.
- Per informazioni su come garantire l'accesso alle risorse negli account AWS che possiedi, consulta Fornire l'accesso a un utente IAM in un altro account AWS che si possiede nella Guida per l'utente di IAM.
- Per capire come fornire l'accesso alle risorse ad account AWS di terze parti, consulta Concessione dell'accesso agli account AWS di proprietà di terze parti nella Guida per l'utente di IAM.
- Per capire come fornire l'accesso tramite la federazione delle identità, consulta Fornire accesso a utenti autenticati esternamente (Federazione delle identità) nella Guida per l'utente di IAM.
- Per informazioni sulle differenze tra l'utilizzo di ruoli e policy basate su risorse per l'accesso multi-account, consultare Differenza tra i ruoli IAM e le policy basate su risorse nella Guida per l'utente di IAM.

Registrazione e monitoraggio in Amazon Aurora

Il monitoraggio è una parte importante per mantenere l'affidabilità, la disponibilità e le prestazioni di Amazon Aurora e delle soluzioni AWS. È consigliabile raccogliere i dati di monitoraggio da tutte le parti della soluzione AWS in modo da poter eseguire più facilmente il debug di errori in più punti nel caso si verifichino. AWS fornisce diversi strumenti per il monitoraggio delle risorse Amazon Aurora e la risposta a potenziali incidenti:

Allarmi di Amazon CloudWatch

> Utilizzando gli allarmi di Amazon CloudWatch, è possibile osservare un singolo parametro per il periodo di tempo specificato. Se il parametro supera una determinata soglia, viene inviata una notifica a un argomento Amazon SNS o alla policy AWS Auto Scaling. Gli allarmi CloudWatch non

richiamano le operazioni perché si trovano in uno stato particolare. È necessario invece cambiare lo stato e mantenerlo per un numero di periodi specificato.

AWS CloudTrailLog di

CloudTrail fornisce un record delle operazioni eseguite da un utente, un ruolo o un servizio AWS in Amazon RDS. CloudTrail acquisisce tutte le chiamate API per Amazon Aurora come eventi, incluse le chiamate dalla console e dal codice alle operazioni API di Amazon RDS. Le informazioni raccolte da CloudTrail consentono di determinare la richiesta effettuata ad Amazon Aurora, l'indirizzo IP di origine da cui è stata eseguita la richiesta, l'autore della richiesta, il momento in cui è stata eseguita e altri dettagli. Per ulteriori informazioni, consulta Monitoraggio delle chamate API di Amazon Aurora in AWS CloudTrail.

Monitoraggio avanzato

Amazon Aurora fornisce le metriche in tempo reale per il sistema operativo sul quale è in esecuzione il cluster di database. È possibile visualizzare le metriche per il cluster di database mediante la console oppure utilizzare l'output JSON del monitoraggio avanzato di Amazon CloudWatch Logs nel sistema di monitoraggio desiderato. Per ulteriori informazioni, consulta Monitoraggio dei parametri del sistema operativo con il monitoraggio avanzato.

Performance Insights di Amazon RDS

Approfondimenti sulle prestazioni amplia le funzionalità di monitoraggio esistenti di Amazon Aurora per illustrare le prestazioni del database e consentire l'analisi di eventuali problemi che lo riguardano. Con il pannello di controllo di Performance Insights, è possibile visualizzare il carico del database e filtrare il carico in base alle attese, alle istruzioni SQL, agli host o agli utenti. Per ulteriori informazioni, consulta Monitoraggio del carico DB con Performance Insights su Amazon Aurora.

Log di database

Puoi visualizzare, scaricare e controllare i log di database tramite Console di gestione AWS, AWS CLI o l'API RDS. Per ulteriori informazioni, consulta Monitoraggio dei file di log di Amazon Aurora.

Suggerimenti di Amazon Aurora

Amazon Aurora offre suggerimenti automatici per le risorse di database. Questi suggerimenti forniscono indicazioni sulle best practice analizzando la configurazione, l'utilizzo e i dati sulle prestazioni del cluster di database. Per ulteriori informazioni, consulta Suggerimenti di Amazon Aurora.

Notifiche di eventi Amazon Aurora

Amazon Aurora utilizza Amazon Simple Notification Service (Amazon SNS) per fornire una notifica quando si verifica un evento Amazon Aurora. Queste notifiche possono essere in qualsiasi forma supportata da Amazon SNS per una regione AWS, ad esempio un'e-mail, un SMS o una chiamata a un endpoint HTTP. Per ulteriori informazioni, consulta Utilizzo della notifica degli eventi di Amazon RDS.

AWS Trusted Advisor

Trusted Advisor sfrutta le best practice acquisite servendo centinaia di migliaia di clienti AWS. Trusted Advisor controlla l'ambiente AWS, quindi fornisce suggerimenti nel caso in cui vi siano opportunità di risparmio, di miglioramento delle prestazioni e della disponibilità dei sistemi o di risoluzione dei problemi di sicurezza. Tutti i clienti AWS hanno accesso a cinque controlli di Trusted Advisor. I clienti che hanno sottoscritto un piano di supporto Business o Enterprise possono visualizzare tutti i controlli di Trusted Advisor.

Trusted Advisor dispone dei seguenti controlli correlati ad Amazon Aurora:

- Istanze database Amazon Aurora inattive
- Rischio accesso gruppo di sicurezza Amazon Aurora
- Backup Amazon Aurora
- Multi-AZ Amazon Aurora
- Accessibilità dell’istanza database Aurora

Per ulteriori informazioni su questi controlli, consulta best practice Trusted Advisor (Controlli).

Flussi di attività di database

I flussi di attività di database proteggono i database da minacce interne controllando l'accesso DBA ai flussi di attività di database. Pertanto, la raccolta, trasmissione, storage e successiva elaborazione del flusso di attività di database non rientra nell'ambito dell'accesso dei DBA che gestiscono il database. Gli stream dell'attività di database consentono di fornire protezioni per il database e soddisfare requisiti normativi e di conformità. Per ulteriori informazioni, consulta Monitoraggio di Amazon Aurora tramite i flussi di attività del database.

Per ulteriori informazioni sul monitoraggio di Aurora, consulta Monitoraggio dei parametri in un cluster di database Amazon Aurora.

Convalida della conformità per Amazon Aurora

Revisori di terze parti valutano la sicurezza e la conformità di Amazon Aurora come parte di più programmi di conformità di AWS. Questi includono SOC, PCI, FedRAMP, HIPAA e altri.

Per un elenco dei AWS servizi che rientrano nell'ambito di specifici programmi di conformità, consulta la sezione AWSServizi rientranti nell'ambito di un programma di conformità. Per informazioni generali, consultare Programmi per la conformità di AWS.

È possibile scaricare report di audit di terze parti utilizzandoAWS Artifact. Per ulteriori informazioni, consulta Scaricamento dei report in AWS Artifact.

La tua responsabilità di conformità quando usi Amazon Aurora è determinata dalla sensibilità dei tuoi dati, dagli obiettivi di conformità dell'organizzazione e dalle leggi e dai regolamenti applicabili. AWSfornisce le seguenti risorse per contribuire alla conformità:

- Guide introduttive su sicurezza e conformità: queste guide all'implementazione illustrano considerazioni sull'architettura e forniscono passaggi per implementare ambienti di base incentrati sulla sicurezza e la conformità. AWS
- Progettazione per la sicurezza e la conformità HIPAA su Amazon Web Services: questo white paper descrive come le aziende possono utilizzare AWS per creare applicazioni conformi allo standard HIPAA.
- AWSrisorse per la conformità: questa raccolta di cartelle di lavoro e guide che potrebbero riguardare il tuo settore e la tua area geografica.
- AWS Config— Questo AWS servizio valuta la conformità delle configurazioni delle risorse alle pratiche interne, alle linee guida del settore e alle normative.
- AWS Security Hub CSPM— Ciò Servizio AWS fornisce una visione completa dello stato di sicurezza dell'utente all'interno. AWS Security Hub CSPM utilizza i controlli di sicurezza per valutare le AWS risorse e verificare la conformità rispetto agli standard e alle migliori pratiche del settore della sicurezza. Per un elenco dei servizi e dei controlli supportati, consulta il riferimento ai controlli CSPM di Security Hub.

Resilienza in Amazon Aurora

L'infrastruttura globale di AWS è basata su regioni AWS e zone di disponibilità. AWS forniscono più zone di disponibilità fisicamente separate e isolate che sono connesse tramite reti altamente ridondanti, a bassa latenza e con throughput elevato. Con le zone di disponibilità, è possibile progettare e gestire applicazioni e database che eseguono il failover automatico tra zone di disponibilità senza interruzioni. Le zone di disponibilità sono più disponibili, tolleranti ai guasti e scalabili rispetto alle infrastrutture tradizionali a data center singolo o multiplo.

Per ulteriori informazioni sulle regioni e le zone di disponibilità AWS, consulta Infrastruttura globale di AWS.

Oltre all'infrastruttura globale AWS, Aurora offre numerose funzionalità per supportare la resilienza dei dati e le esigenze di backup.

Backup e ripristino

Aurora esegue automaticamente il backup del volume del cluster e conserva i dati per la durata del tempo di conservazione del backup. I backup Aurora sono continui e incrementali, in modo da poter eseguire rapidamente un ripristino da qualsiasi punto del tempo di conservazione del backup. Durante la scrittura dei dati di backup, non si verifica alcun impatto sulle prestazioni o interruzione del funzionamento del servizio del database. Puoi specificare un tempo di conservazione del backup, da 1 a 35 giorni, durante la creazione o la modifica di un cluster di database.

Se desideri conservare un backup oltre il tempo di conservazione del backup, puoi anche eseguire uno snapshot dei dati del volume del cluster. Aurora mantiene i dati di ripristino incrementali per l'intero tempo di conservazione del backup. Quindi, è sufficiente creare uno snapshot per i dati che desideri conservare oltre il tempo di conservazione del backup. Puoi creare un nuovo cluster di database dalla snapshot.

Puoi recuperare i dati creando un nuovo cluster di database Aurora dai dati di backup conservati da Aurora o da uno snapshot del cluster di database salvata. Puoi ripristinare rapidamente una nuova copia di un cluster di database creato dai dati di backup a un momento qualsiasi del tempo di conservazione del backup. Data la natura continua e incrementale dei backup di Aurora a durante il tempo di conservazione del backup, non dovrai effettuare snapshot frequenti dei dati per migliorare i tempi di ripristino.

Per ulteriori informazioni, consulta Backup e ripristino di un cluster DB Amazon Aurora.

Replica

Le repliche Aurora sono endpoint indipendenti in un cluster di database Aurora, utilizzati al meglio per operazioni di dimensionamento della lettura e maggiore disponibilità. È possibile distribuire fino a 15 repliche di Aurora nelle zone di disponibilità sulle quali si estende un cluster di database in una regione AWS. Il volume del cluster di database si compone di più copie dei dati per il cluster di database. Tuttavia, i dati nel volume del cluster sono rappresentati come singolo volume logico all'istanza database primaria e alle repliche di Aurora nel cluster di database. Se l'istanza database primaria non va a buon fine, una replica di Aurora viene promossa a istanza primaria.

Aurora supporta inoltre opzioni di replica specifiche per Aurora MySQL e Aurora PostgreSQL.

Per ulteriori informazioni, consulta Replica con Amazon Aurora.

Failover

Aurora memorizza copie di dati in un cluster di database in più zone di disponibilità in una singola regione AWS. Lo storage avviene indipendentemente dal fatto che le istanze database nel cluster di database comprendano più zone di disponibilità. Quando si creano repliche di Aurora tra le zone di disponibilità, Aurora effettua automaticamente il provisioning e le mantiene in modo sincrono. L'istanza database primaria viene replicata in modo sincrono nelle zone di disponibilità sulle repliche di Aurora per fornire ridondanza dati, eliminare blocchi I/O e ridurre al minimo i picchi di latenza durante i backup di sistema. Eseguendo un cluster di database con disponibilità elevata, è possibile migliorare la disponibilità durante la manutenzione pianificata del sistema e consentire di proteggere i database da errori e interruzioni relative alle zone di disponibilità.

Per ulteriori informazioni, consulta Elevata disponibilità di Amazon Aurora.

Sicurezza dell'infrastruttura in Amazon Aurora

In quanto servizio gestito, Amazon Relational Database Service è protetto AWS dalla sicurezza di rete globale. Per informazioni sui servizi di AWS sicurezza e su come AWS protegge l'infrastruttura, consulta AWS Cloud Security. Per progettare il tuo AWS ambiente utilizzando le migliori pratiche per la sicurezza dell'infrastruttura, vedi Infrastructure Protection in Security Pillar AWS Well-Architected Framework.

Utilizzi chiamate API AWS pubblicate per accedere ad Amazon RDS attraverso la rete. I client devono supportare quanto segue:

- Transport Layer Security (TLS). È richiesto TLS 1.2 ed è consigliato TLS 1.3.
- Suite di cifratura con Perfect Forward Secrecy (PFS), ad esempio Ephemeral Diffie-Hellman (DHE) o Elliptic Curve Ephemeral Diffie-Hellman (ECDHE). La maggior parte dei sistemi moderni, come Java 7 e versioni successive, supporta tali modalità.

Inoltre, Aurora offre funzionalità che contribuiscono a supportare la sicurezza dell'infrastruttura.

Gruppi di sicurezza

I gruppi di sicurezza controllano l'accesso del traffico in entrata e in uscita di un cluster database. Per impostazione predefinita, l'accesso alla rete è disattivato per un cluster database. Puoi specificare delle norme in un gruppo di sicurezza che consente l'accesso da un intervallo di indirizzi IP, dalla porta o da un gruppo di sicurezza. Una volta configurate le regole in ingresso, si applicano le stesse regole a tutti i cluster database con associazione a tale gruppo di sicurezza.

Per ulteriori informazioni, consulta Controllo dell'accesso con i gruppi di sicurezza.

Public accessibility (Accesso pubblico)

Quando si avvia un'istanza database all'interno di un cloud privato virtuale (VPC) in base al servizio Amazon VPC, è possibile attivare o disattivare l'accessibilità pubblica per tale istanza database. Per stabilire se l'istanza database creata ha un nome DNS che si risolve in un indirizzo IP pubblico, utilizza il parametro Public accessibility (Accessibilità pubblica). Questo parametro consente di stabilire se esiste un accesso pubblico all'istanza database. È possibile modificare un'istanza database per attivare o disattivare l'accessibilità pubblica modificando il parametro Public accessibility (Accessibilità pubblica).

Per ulteriori informazioni, consulta Nascondere cluster database in un VPC da Internet.

> **Note**
>
> Se la tua istanza DB si trova in un VPC ma non è accessibile pubblicamente, puoi anche utilizzare una connessione AWS Site-to-Site VPN o una Direct Connect connessione per accedervi da una rete privata. Per ulteriori informazioni, consulta Riservatezza del traffico Internet.

API Amazon RDS ed endpoint VPC dell'interfaccia (AWS PrivateLink)

È possibile stabilire una connessione privata tra gli endpoint VPC e l'API Amazon RDS creando un endpoint VPC di interfaccia. Endpoint di interfaccia con tecnologia AWS PrivateLink.

AWS PrivateLink consente di accedere in modo privato alle operazioni delle API di Amazon RDS senza un gateway Internet, un dispositivo NAT, una connessione VPN o una connessione. Direct Connect Le istanze database nel VPC non necessitano di indirizzi IP pubblici per comunicare con gli endpoint dell'API Amazon RDS per avviare, modificare o terminare le istanze database e i cluster DB. Inoltre, le istanze database non richiedono indirizzi IP pubblici per utilizzare le operazioni dell'API RDS disponibili. Il traffico di rete tra il VPC e Amazon RDS non esce dalla rete Amazon.

Ogni endpoint dell'interfaccia è rappresentato da una o più interfacce di rete elastiche nelle sottoreti. Per ulteriori informazioni sulle interfacce di rete elastiche, consulta Interfacce di rete elastiche nella Amazon EC2 User Guide.

Per ulteriori informazioni sugli endpoint dei VPC, consulta Endpoint VPC di interfaccia (AWS PrivateLink) nella Guida per l'utente di Amazon VPC. Per informazioni sulle operazioni API RDS, consulta Documentazione di riferimento delle API di Amazon RDS.

Non è necessaria un'interfaccia endpoint VPC per connettersi a un cluster database. Per ulteriori informazioni, consulta Scenari per accedere a un cluster database in un VPC.

Considerazioni sugli endpoint VPC

Prima di impostare un endpoint VPC di interfaccia per endpoint Amazon RDS API, verificare di esaminare le proprietà e le limitazioni degli endpoint di interfaccia in Guida per l'utente di Amazon VPC.

Tutte le operazioni API RDS rilevanti per la gestione delle risorse Amazon Aurora sono disponibili dal VPC utilizzando AWS PrivateLink.

Le policy degli endpoint VPC sono supportate per gli endpoint dell'API RDS. Per impostazione predefinita, l'accesso completo alle operazioni API RDS è consentito attraverso l'endpoint. Per ulteriori informazioni, consulta Controllo degli accessi ai servizi con endpoint VPC in Guida per l'utente di Amazon VPC.

Disponibilità

L'API Amazon RDS attualmente supporta gli endpoint VPC nelle seguenti regioni: AWS

- Stati Uniti orientali (Ohio)
- Stati Uniti orientali (Virginia settentrionale)
- Stati Uniti occidentali (California settentrionale)
- Stati Uniti occidentali (Oregon)
- Africa (Città del Capo)
- Asia Pacific (Hong Kong)
- Asia Pacifico (Mumbai)
- Asia Pacifico (Nuova Zelanda)
- Asia Pacifico (Osaka-Locale)
- Asia Pacific (Seul)
- Asia Pacifico (Singapore)
- Asia Pacifico (Sydney)
- Asia Pacifico (Taipei)
- Asia Pacifico (Thailandia)
- Asia Pacifico (Tokyo)
- Canada (Centrale)
- Canada occidentale (Calgary)
- Cina (Pechino)
- China (Ningxia)
- Europa (Francoforte)
- Europa (Zurigo)
- Europa (Irlanda)
- Europe (London)
- Europe (Paris)
- Europe (Stockholm)
- Europa (Milano)

- Israele (Tel Aviv)
- Messico (centrale)
- Medio Oriente (Bahrein)
- Sud America (San Paolo)
- AWS GovCloud (Stati Uniti orientali)
- AWS GovCloud (Stati Uniti occidentali)

Creazione di un endpoint VPC di interfaccia per l'API Amazon RDS

Puoi creare un endpoint VPC per l'API Amazon RDS utilizzando la console Amazon VPC o il (). AWS Command Line Interface AWS CLI Per ulteriori informazioni, consultare Creazione di un endpoint di interfaccia nella Guida per l'utente di Amazon VPC.

Crea un endpoint VPC per l'API Amazon RDS utilizzando il nome del servizio `com.amazonaws.`*`region`*`.rds`.

Escludendo AWS le regioni in Cina, se abiliti il DNS privato per l'endpoint, puoi effettuare richieste API ad Amazon RDS con l'endpoint VPC utilizzando ad esempio il nome DNS predefinito per la regione. AWS `rds.us-east-1.amazonaws.com` Per le AWS regioni Cina (Pechino) e Cina (Ningxia), puoi effettuare richieste API con l'endpoint VPC utilizzando e, rispettivamente. `rds-api.cn-north-1.amazonaws.com.cn` `rds-api.cn-northwest-1.amazonaws.com.cn`

Per ulteriori informazioni, consultare Accesso a un servizio tramite un endpoint di interfaccia in Guida per l'utente di Amazon VPC.

Creazione di una policy di endpoint VPC per l'API Amazon RDS

Puoi allegare una policy di endpoint all'endpoint VPC che controlla l'accesso all'API Amazon RDS. Questo codice specifica le informazioni riportate di seguito:

- Il principale che può eseguire operazioni.
- Le azioni che possono essere eseguite.
- Le risorse sui cui si possono eseguire operazioni.

Per ulteriori informazioni, consulta Controllo degli accessi ai servizi con endpoint VPC in Guida per l'utente di Amazon VPC.

Ad esempio, policy di endpoint VPC per le operazioni API Amazon RDS

Di seguito è riportato un esempio di una policy endpoint per l'API Amazon RDS. Se collegato a un endpoint, questa policy concede l'accesso alle operazioni API Amazon RDS riportate per tutte le entità su tutte le risorse.

```
{
   "Statement":[
      {
         "Principal":"*",
         "Effect":"Allow",
         "Action":[
            "rds:CreateDBInstance",
            "rds:ModifyDBInstance",
            "rds:CreateDBSnapshot"
         ],
         "Resource":"*"
      }
   ]
}
```

Esempio: policy degli endpoint VPC che nega tutti gli accessi da un account specificato AWS

La seguente politica degli endpoint VPC nega all' AWS account `123456789012` tutti gli accessi alle risorse che utilizzano l'endpoint. La policy consente tutte le operazioni da altri account.

```
{
  "Statement": [
    {
      "Action": "*",
      "Effect": "Allow",
      "Resource": "*",
      "Principal": "*"
    },
    {
      "Action": "*",
      "Effect": "Deny",
      "Resource": "*",
      "Principal": { "AWS": [ "123456789012" ] }
     }
   ]
}
```

Best practice relative alla sicurezza di Amazon Aurora

Usa gli account AWS Identity and Access Management (IAM) per controllare l'accesso alle operazioni dell'API Amazon RDS, in particolare le operazioni che creano, modificano o eliminano risorse Amazon Aurora. Tali risorse includono i cluster di database, i gruppi di sicurezza e i gruppi di parametri. Utilizza anche IAM per controllare le azioni che eseguono azioni amministrative comuni come il backup e il ripristino di cluster di database.

- Crea un singolo utente per ogni persona, te compresa, che gestisce le risorse Amazon Aurora. Non utilizzare credenziali AWS root per gestire le risorse Amazon Aurora.
- Assegna a ciascun utente un set minimo di autorizzazioni richieste per eseguire le proprie mansioni.
- Utilizza gruppi IAM per gestire in modo efficace le autorizzazioni per più utenti.
- Ruota periodicamente le credenziali IAM.
- Configura Gestione dei segreti AWS per ruotare automaticamente i segreti per Aurora. Per ulteriori informazioni, consulta Rotazione dei segreti Gestione dei segreti AWS nella Guida per l'utente di Gestione dei segreti AWS. Puoi anche recuperare le credenziali da programmaticamente. Gestione dei segreti AWS Per ulteriori informazioni, consulta Recupero del valore segreto nella Guida per l'utente di Gestione dei segreti AWS.

Per ulteriori informazioni sulla sicurezza di Amazon Aurora, consulta Sicurezza in Amazon Aurora. Per ulteriori informazioni su IAM, consulta AWS Identity and Access Management. Per informazioni sulle best practice di IAM, consulta Best practice di IAM.

AWS Security Hub CSPMutilizza controlli di sicurezza per valutare le configurazioni delle risorse e gli standard di sicurezza per aiutarvi a rispettare vari quadri di conformità. Per ulteriori informazioni sull'utilizzo di Security Hub CSPM per valutare le risorse RDS, consulta i controlli di Amazon Relational Database Service nella Guida per l'utente. AWS Security Hub

È possibile monitorare l'utilizzo di RDS in relazione alle migliori pratiche di sicurezza utilizzando Security Hub CSPM. Per ulteriori informazioni, vedi Cos'è? AWS Security Hub CSPM .

Usa l'API Console di gestione AWSAWS CLI, the o RDS per modificare la password per il tuo utente principale. Se usi uno strumento diverso, ad esempio un client SQL, per modificare la password dell'utente master, i privilegi per l'utente potrebbero venire revocati involontariamente.

Amazon GuardDuty è un servizio di monitoraggio continuo della sicurezza che analizza ed elabora varie fonti di dati, inclusa l'attività di accesso ad Amazon RDS. Utilizza i feed di intelligence sulle

minacce e il machine learning per identificare attività inattese, potenzialmente non autorizzate, comportamenti di accesso sospetti e operazioni dannose nell'ambiente AWS.

Quando Amazon GuardDuty RDS Protection rileva un tentativo di accesso potenzialmente sospetto o anomalo che indica una minaccia per il database, GuardDuty genera una nuova scoperta con dettagli sul database potenzialmente compromesso. Per ulteriori informazioni, consulta Monitoraggio delle minacce con la protezione RDS Amazon GuardDuty per Amazon Aurora.

Controllo dell'accesso con i gruppi di sicurezza

I gruppi di sicurezza VPC controllano l'accesso che il traffico ha in entrata e in uscita su un cluster di database. Per impostazione predefinita, l'accesso alla rete è disattivato per un cluster di database. Puoi specificare delle norme in un gruppo di sicurezza che consente l'accesso da un intervallo di indirizzi IP, dalla porta o da un gruppo di sicurezza. Una volta configurate le regole in ingresso, si applicano le stesse regole a tutti i cluster di database con associazione a tale gruppo di sicurezza. Puoi specificare fino a 20 norme in un gruppo di sicurezza.

Panoramica dei gruppi di sicurezza VPC

Ogni regola del gruppo di sicurezza VPC consente a un'origine specifica di accedere a un cluster di database in un VPC con associazione al gruppo di sicurezza VPC specifico. L'origine può essere una serie di indirizzi (ad esempio, 203.0.113.0/24) oppure un altro gruppo di sicurezza VPC. Specificando un gruppo di sicurezza VPC come origine, consenti il traffico in entrata da tutte le istanze (in genere i server dell'applicazione) che usano il gruppo di sicurezza VPC. I gruppi di sicurezza VPC possono avere regole che gestiscono sia il traffico in entrata che in uscita. Tuttavia, le regole del traffico in uscita in genere non si applicano ai cluster di database. Le regole del traffico in uscita si applicano solo se il cluster di database funge da client. Per creare gruppi di sicurezza VPC, è necessario utilizzare l'API Amazon EC2 o l'opzione Security Group (Gruppo di sicurezza) nella console VPC.

Quando si creano regole per il gruppo di sicurezza VPC che consentono l'accesso ai cluster nel VPC, è necessario specificare una porta per ciascun intervallo di indirizzi per i quali la regola consente l'accesso. Ad esempio, se si desidera abilitare l'accesso SSH (Secure Shell) alle istanze nel VPC, devi creare una regola che consenta l'accesso alla porta TCP 22 per l'intervallo di indirizzi specificato.

È possibile configurare più gruppi di sicurezza VPC che consentono l'accesso a porte diverse per istanze diverse nel VPC. Ad esempio, è possibile creare un gruppo di sicurezza VPC che consenta l'accesso alla porta TCP 80 per i server web nel VPC. Quindi è possibile creare un altro gruppo di sicurezza VPC che consenta l'accesso alla porta TCP 3306 per le istanze database Aurora MySQL nel VPC.

> **Note**
>
> In un cluster database Aurora, il gruppo di sicurezza VPC associato al cluster database è anche associato a tutte le istanze database nel cluster database. Se modifichi il gruppo di sicurezza VPC per il cluster database o per un'istanza database, la modifica viene applicata automaticamente a tutte le istanze database nel cluster database.

Per ulteriori informazioni sui gruppi di sicurezza VPC, consulta la pagina Gruppi di sicurezza nella Guida per l'utente di Amazon Virtual Private Cloud.

> **Note**
>
> Se il cluster di database si trova in un VPC ma non è accessibile pubblicamente, è possibile anche utilizzare una connessione VPN AWS sito-sito o una connessione Direct Connect per accedervi da una rete privata. Per ulteriori informazioni, consulta Riservatezza del traffico Internet.

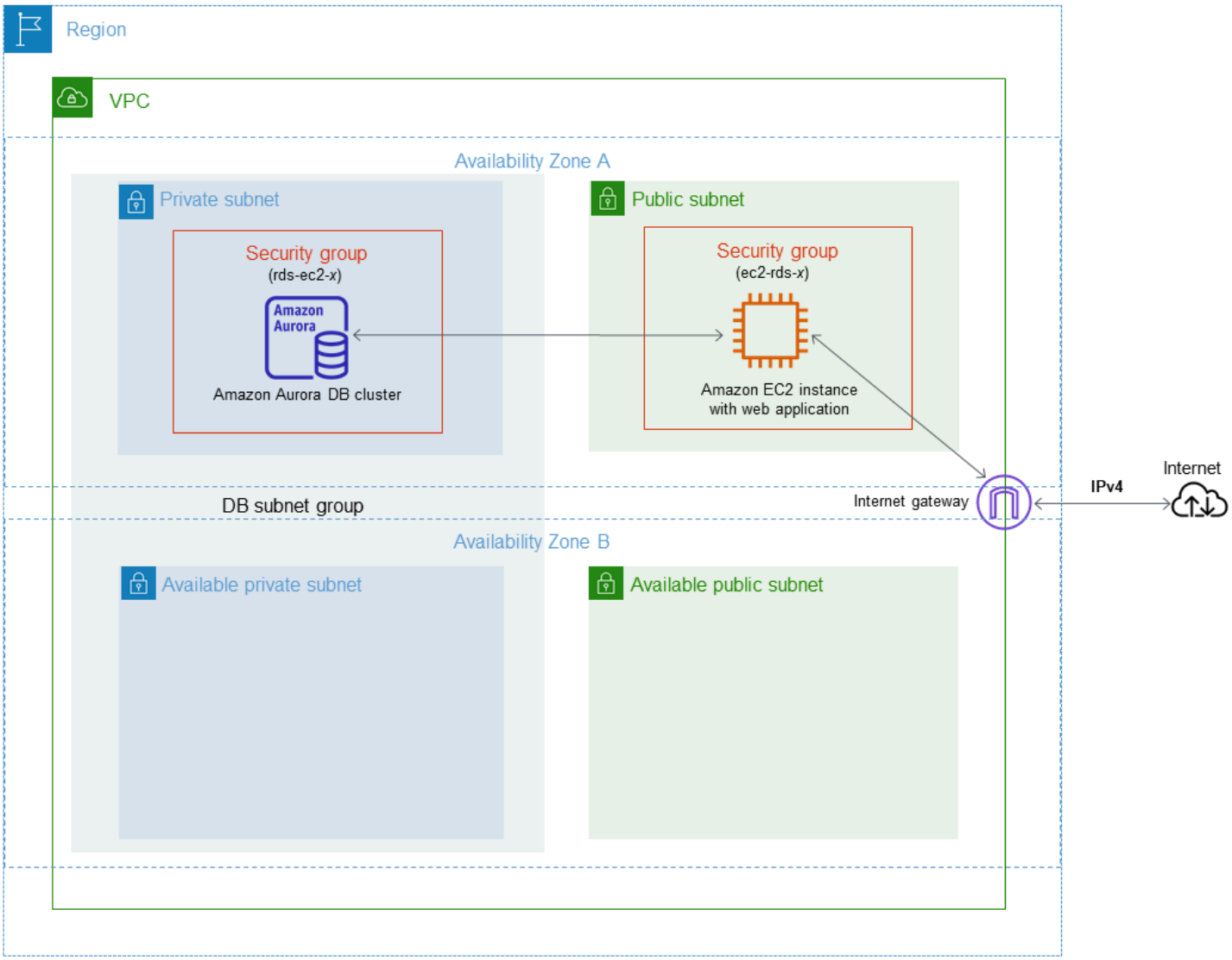
Scenario del gruppo di sicurezza

Un uso comune di un cluster di database in un VPC è quello di condividere dati con un server di applicazione in esecuzione in un'istanza Amazon EC2 nello stesso VPC, al quale si accede tramite un'applicazione client esterna al VPC. Per questo scenario, si utilizzano le pagine RDS e VPC sulla Console di gestione AWS oppure le azioni API RDS ed EC2 per creare le istanze e i gruppi di sicurezza necessari:

1. Creare un gruppo di sicurezza VPC (ad esempio, `sg-0123ec2example`) e definire le regole in entrata che utilizzano l'indirizzo IP dell'applicazione client usato nell'indirizzo IP dell'applicazione del client come origine. Questo gruppo di sicurezza consente all'applicazione del client di collegarsi alle istanze EC2 in una VPC che utilizza questo gruppo di sicurezza.
2. Creare un'istanza EC2 per l'applicazione e aggiungere l'istanza EC2 al gruppo di sicurezza VPC (`sg-0123ec2example`) creato nel passaggio precedente.
3. Creare un secondo gruppo di sicurezza VPC (ad esempio, `sg-6789rdsexample`) e creare una nuova regola specificando il gruppo di sicurezza VPC creato nel passaggio 1 (`sg-0123ec2example`) come l'origine.

4. Creare un nuovo cluster di database e aggiungere il cluster di database al gruppo di sicurezza VPC (`sg-6789rdsexample`) creato nel passaggio precedente. Quando si crea il cluster di database, utilizzare lo stesso numero di porta di quello specificato per la regola del gruppo di sicurezza VPC (`sg-6789rdsexample`) creato nel passaggio 3.

Il seguente diagramma mostra questo scenario.



Per istruzioni dettagliate sulla configurazione di un VPC per questo scenario, consulta Tutorial: Creazione di un Amazon VPC da utilizzare con un cluster database (solo IPv4). Per ulteriori informazioni sull'utilizzo di un VPC, consulta VPC Amazon e Amazon Aurora.

Creazione di un gruppo di sicurezza VPC

Puoi creare un gruppo di sicurezza VPC per un'istanza database tramite la console VPC. Per informazioni sulla creazione di un gruppo di sicurezza, consulta le pagine Fornitura dell'accesso al cluster di database nel VPC creando un gruppo di sicurezza e Gruppi di sicurezza nella Guida per l'utente di Amazon Virtual Private Cloud.

Associazione di un gruppo di sicurezza a un cluster database

È possibile associare un gruppo di sicurezza a un cluster database utilizzando l'opzione Modifica cluster nella console RDS, l'API `ModifyDBCluster` Amazon RDS o il comando `modify-db-cluster` AWS CLI.

L'esempio dell'interfaccia della riga di comando seguente associa un gruppo VPC specifico e rimuove i gruppi di sicurezza database dal cluster di database

```
aws rds modify-db-cluster --db-cluster-identifier dbName --vpc-security-group-ids sg-ID
```

Per informazioni sulla modifica di un cluster di database, consulta Modifica di un cluster database Amazon Aurora.

Privilegi dell'account utente master

Quando si crea un nuovo cluster di database, l'utente master predefinito utilizzato ottiene determinati privilegi per tale cluster di database. Non è possibile modificare il nome utente master dopo aver creato il cluster di database.

> **Important**
>
> Si consiglia di non utilizzare l'utente master direttamente nelle applicazioni. Rispetta piuttosto la best practice di utilizzare un utente del database creato con i privilegi minimi richiesti per l'applicazione.

> **Note**
>
> Se si cancellano per errore le autorizzazioni per l'utente master è possibile ripristinarle modificando il cluster di database e impostando una nuova password per l'utente master. Per

ulteriori informazioni sulla modifica di un cluster di database, consulta Modifica di un cluster database Amazon Aurora .

La tabella seguente mostra i privilegi e i ruoli di database ottenuti dall'utente master per ciascuno dei motori di database.

Motore del database	Privilegio del sistema	Ruolo di database
Aurora MySQL	Versione 2: `ALTER,ALTER ROUTINE, CREATE,CREATE ROUTINE,CREATE TEMPORARY TABLES,CREATE USER,CREATE VIEW,DELETE,DROP, EVENT,EXECUTE,GRANT OPTION,INDEX,INSERT, LOAD FROM S3,LOCK TABLES, PROCESS,REFERENCES , RELOAD,REPLICATI ON CLIENT , REPLICATION SLAVE ,SELECT, SELECT INTO S3,SHOW DATABASES , SHOW VIEW,TRIGGER, UPDATE`	—
	Versione 3: `ALTER,APPLICATION_PASSWORD_ADMIN , ALTER ROUTINE,CONNECTION_ADMIN , CREATE,CREATE ROLE,CREATE ROUTINE,CREATE TEMPORARY TABLES, CREATE USER,CREATE VIEW, DELETE,DROP,DROP ROLE, EVENT,EXECUTE,INDEX, INSERT,LOCK TABLES, PROCESS,REFERENCES , RELOAD,REPLICATION CLIENT , REPLICATI ON SLAVE ,ROLE_ADMIN , SET_USER_ ID ,SELECT,SHOW DATABASES ,SHOW VIEW,TRIGGER, UPDATE,XA_RECOVER_ADMIN`  A partire da Aurora MySQL versione 3.04.0, anche l'utente master ottiene il privilegio `SHOW_ROUTINE` .	`rds_superuser_role` Per ulteriori informazioni su rds_superuser_role, consultare Privilegio basato sui ruoli.

Motore del database	Privilegio del sistema	Ruolo di database
	A partire da Aurora MySQL versione 3.09.0, anche l'utente master ottiene i privilegi `FLUSH_OPTIMIZER_COSTS`, `FLUSH_STATUS`, `FLUSH_TABLES` e `FLUSH_USER_RESOURCES`.	
Aurora PostgreSQL	`LOGIN`,`NOSUPERUSER`, `INHERIT`,`CREATEDB`, `CREATEROLE`,`NOREPLICATION`,`VALID UNTIL 'infinity'`	`RDS_SUPERUSER` Per ulteriori informazioni su RDS_SUPERUSER, consultare Informazioni su ruoli e autorizzazioni di PostgreSQL.

Utilizzo di ruoli collegati ai servizi per Amazon Aurora

Amazon Aurora utilizza i ruoli collegati ai servizi AWS Identity and Access Management (IAM). Un ruolo collegato ai servizi è un tipo univoco di ruolo IAM collegato direttamente a Amazon Aurora. I ruoli collegati ai servizi sono definiti automaticamente da Amazon Aurora e includono tutte le autorizzazioni richieste dal servizio per eseguire chiamate agli altri servizi AWS per tuo conto.

Un ruolo collegato ai servizi semplifica l'uso di Amazon Aurora perché non sarà più necessario aggiungere manualmente le autorizzazioni. Amazon Aurora definisce le autorizzazioni dei ruoli collegati ai servizi e, salvo diversamente definito, solo Amazon Aurora può assumere il ruolo. Le autorizzazioni definite includono la policy di trust e la policy delle autorizzazioni. Una policy delle autorizzazioni specifica non può essere collegata a un'altra entità IAM.

È possibile eliminare i ruoli solo dopo aver eliminato le risorse correlate. Questa procedura protegge le risorse di Amazon Aurora perché impedisce la rimozione involontaria delle autorizzazioni di accesso alle risorse.

Per informazioni sugli altri servizi che supportano i ruoli collegati ai servizi, consultare i servizi AWS che funzionano con IAM e cercare i servizi che riportano Yes (Sì) nella colonna Service-Linked Role (Ruolo associato ai servizi). Scegli Sì in corrispondenza di un link per visualizzare la documentazione relativa al ruolo collegato ai servizi per tale servizio.

Autorizzazioni del ruolo collegato ai servizi per Amazon Aurora

Amazon Aurora usa il ruolo collegato al servizio denominato AWSServiceRoleForRDS per consentire ad Amazon RDS di chiamare i servizi AWS per conto del tuo cluster di database.

Ai fini dell'assunzione del ruolo, il ruolo collegato ai servizi AWSServiceRoleForRDS considera attendibili i seguenti servizi:

- `rds.amazonaws.com`

A questo ruolo collegato ai servizi è collegata un policy di autorizzazione denominata `AmazonRDSServiceRolePolicy` che concede le autorizzazioni per operare nell'account.

Per ulteriori informazioni su questa policy, incluso il documento sulla policy JSON, consulta AmazonRDSServiceRolePolicy nella Guida di riferimento sulle policy gestite da AWS.

> **Note**
>
> Per consentire a un'entità IAM (ad esempio un utente, un gruppo o un ruolo) di creare, modificare o eliminare un ruolo collegato ai servizi, devi configurare le autorizzazioni. Se viene visualizzato il messaggio di errore seguente:
> Unable to create the resource. (Impossibile creare la risorsa. Verify that you have permission to create service linked role. (Verifica di possedere le autorizzazioni necessarie per creare un ruolo collegato ai servizi.) Otherwise wait and try again later. (In caso contrario, attendi e riprova più tardi.
> Accertati che le seguenti autorizzazioni siano abilitate:
>
> ```
> {
>     "Action": "iam:CreateServiceLinkedRole",
>     "Effect": "Allow",
>     "Resource": "arn:aws:iam::*:role/aws-service-role/rds.amazonaws.com/
> AWSServiceRoleForRDS",
>     "Condition": {
>         "StringLike": {
>             "iam:AWSServiceName":"rds.amazonaws.com"
>         }
>     }
> }
> ```
>
> Per ulteriori informazioni, consulta Autorizzazioni del ruolo collegato ai servizi nella Guida per l'utente di IAM.

Creazione di un ruolo collegato ai servizi per Amazon Aurora

Non devi creare manualmente un ruolo collegato ai servizi. Quando si crea un cluster di database, Amazon Aurora crea nuovamente il ruolo collegato al servizio per conto dell'utente.

> **Important**
>
> Se usavi il servizio Amazon Aurora prima del 1 dicembre 2017, data da cui è disponibile il supporto dei ruoli collegati ai servizi, allora Amazon Aurora ha creato il ruolo AWSServiceRoleForRDS nel tuo account. Per ulteriori informazioni, consulta Un nuovo ruolo appare nell'account AWS.

Se elimini questo ruolo collegato ai servizi e quindi devi ricrearlo di nuovo, puoi utilizzare lo stesso processo per ricreare il ruolo nel tuo account. Quando si crea un cluster di database, Amazon Aurora crea nuovamente il ruolo collegato al servizio per tuo conto.

Modifica di un ruolo collegato ai servizi per Amazon Aurora

Amazon Aurora non consente di modificare il ruolo collegato ai servizi AWSServiceRoleForRDS. Dopo aver creato un ruolo collegato ai servizi, non potrai modificarne il nome perché varie entità potrebbero farvi riferimento. È possibile tuttavia modificarne la descrizione utilizzando IAM. Per ulteriori informazioni, consulta Modifica di un ruolo collegato ai servizi nella Guida per l'utente di IAM.

Eliminazione di un ruolo collegato ai servizi per Amazon Aurora

Se non è più necessario utilizzare una caratteristica o un servizio che richiede un ruolo collegato ai servizi, ti consigliamo di eliminare quel ruolo. In questo modo non sarà più presente un'entità non utilizzata che non viene monitorata e gestita attivamente. Tuttavia, prima di poter eliminare il ruolo collegato al servizio, devi eliminare tutti i cluster database.

Pulizia di un ruolo collegato ai servizi

Prima di utilizzare IAM per eliminare un ruolo collegato ai servizi, devi innanzitutto verificare che il ruolo non abbia sessioni attive ed eliminare tutte le risorse utilizzate dal ruolo.

Per verificare se il ruolo collegato ai servizi dispone di una sessione attiva nella console IAM

1. Accedi alla Console di gestione AWSe apri la console IAM all'indirizzo https://console.aws.amazon.com/iam/.
2. Nel pannello di navigazione della console IAM seleziona Roles (Ruoli). Quindi, scegli il nome (non la casella di controllo) del ruolo AWSServiceRoleForRDS.
3. Nella pagina Riepilogo per il ruolo selezionato, scegli la scheda Ultimo accesso.
4. Nella scheda Ultimo accesso, esamina l'attività recente per il ruolo collegato ai servizi.

> **Note**
>
> Se non si ha la certezza che Amazon Aurora stia utilizzando il ruolo AWSServiceRoleForRDS, è possibile provare a eliminarlo. Se il servizio sta utilizzando il ruolo, l'eliminazione non andrà a buon fine e potrai visualizzare le regioni AWS in cui il ruolo viene utilizzato. Se il ruolo è in uso, prima di poterlo eliminare dovrai attendere il termine della sessione. Non puoi revocare la sessione per un ruolo collegato ai servizi.

Se desideri rimuovere il ruolo AWSServiceRoleForRDS, devi prima eliminare tutti gli oggetti di tipo cluster database.

Eliminazione di tutti i cluster

Utilizza una delle procedure seguenti per eliminare un singolo cluster. Ripeti la procedura per ogni cluster.

Per eliminare un cluster (console)

1. Apri la console Amazon RDS all'indirizzo https://console.aws.amazon.com/rds/.
2. Nell'elenco Databases (Database), scegliere il cluster da eliminare.
3. Per Cluster Actions (Operazioni cluster), scegliere Delete (Elimina).
4. Scegliere Delete (Elimina).

Per eliminare un cluster (CLI)

Consulta `delete-db-cluster` in Riferimento ai comandi AWS CLI.

Per eliminare un cluster (API)

Consulta `DeleteDBCluster` nella Amazon RDS API Reference.

Per eliminare il ruolo collegato al servizio AWSServiceRoleForRDS, puoi usare la console IAM, la CLI IAM o l'API IAM. Per ulteriori dettagli, consulta Eliminazione di un ruolo collegato al servizio nella Guida per l'utente di IAM.

Autorizzazioni del ruolo collegato ai servizi per Amazon RDS Beta

Amazon Aurora usa il ruolo collegato al servizio denominato `AWSServiceRoleForRDSBeta` per consentire ad Amazon Aurora di chiamare i servizi AWS per conto delle tue risorse di database RDS.

Ai fini dell'assunzione del ruolo, il ruolo collegato ai servizi AWSServiceRoleForRDSBeta considera attendibili i seguenti servizi:

- `rds.amazonaws.com`

A questo ruolo collegato ai servizi è collegata un policy di autorizzazione denominata `AmazonRDSBetaServiceRolePolicy` che concede le autorizzazioni per operare nell'account. Per ulteriori informazioni, consulta Politicy gestita da AWS: AmazonRDSBetaServiceRolePolicy.

> **Note**
>
> Per consentire a un'entità IAM (ad esempio un utente, un gruppo o un ruolo) di creare, modificare o eliminare un ruolo collegato ai servizi, devi configurare le autorizzazioni. Se viene visualizzato il messaggio di errore seguente:
> Unable to create the resource. (Impossibile creare la risorsa. Verify that you have permission to create service linked role. (Verifica di possedere le autorizzazioni necessarie per creare un ruolo collegato ai servizi.) Otherwise wait and try again later. (In caso contrario, attendi e riprova più tardi.
> Accertati che le seguenti autorizzazioni siano abilitate:
>
> ```
> {
>     "Action": "iam:CreateServiceLinkedRole",
>     "Effect": "Allow",
>     "Resource": "arn:aws:iam::*:role/aws-service-role/custom.rds.amazonaws.com/
> AmazonRDSBetaServiceRolePolicy",
>     "Condition": {
>         "StringLike": {
>             "iam:AWSServiceName":"custom.rds.amazonaws.com"
>         }
>     }
> }
> ```
>
> Per ulteriori informazioni, consulta Autorizzazioni del ruolo collegato ai servizi nella Guida per l'utente di IAM.

Ruolo collegato ai servizi per Amazon RDS Preview

Amazon Aurora usa il ruolo collegato al servizio denominato `AWSServiceRoleForRDSPreview` per consentire ad Amazon Aurora di chiamare i servizi AWS delle tue risorse database.

Ai fini dell'assunzione del ruolo, il ruolo collegato ai servizi AWSServiceRoleForRDSPreview considera attendibili i seguenti servizi:

- `rds.amazonaws.com`

A questo ruolo collegato ai servizi è collegata un policy di autorizzazione denominata `AmazonRDSPreviewServiceRolePolicy` che concede le autorizzazioni per operare nell'account. Per ulteriori informazioni, consulta Policy gestita da AWS: AmazonRDSPreviewServiceRolePolicy.

> **Note**
>
> Per consentire a un'entità IAM (ad esempio un utente, un gruppo o un ruolo) di creare, modificare o eliminare un ruolo collegato ai servizi, devi configurare le autorizzazioni. Se viene visualizzato il messaggio di errore seguente:
> Unable to create the resource. (Impossibile creare la risorsa. Verify that you have permission to create service linked role. (Verifica di possedere le autorizzazioni necessarie per creare un ruolo collegato ai servizi.) Otherwise wait and try again later. (In caso contrario, attendi e riprova più tardi.
> Accertati che le seguenti autorizzazioni siano abilitate:
>
> ```
> {
>     "Action": "iam:CreateServiceLinkedRole",
>     "Effect": "Allow",
>     "Resource": "arn:aws:iam::*:role/aws-service-role/custom.rds.amazonaws.com/
> AmazonRDSPreviewServiceRolePolicy",
>     "Condition": {
>         "StringLike": {
>             "iam:AWSServiceName":"custom.rds.amazonaws.com"
>         }
>     }
> }
> ```
>
> Per ulteriori informazioni, consulta Autorizzazioni del ruolo collegato ai servizi nella Guida per l'utente di IAM.

VPC Amazon e Amazon Aurora

Amazon Virtual Private Cloud (Amazon VPC) consente di avviare le risorse AWS, come i cluster database Aurora, in un cloud privato virtuale (VPC).

Quando utilizzi un VPC, hai il controllo completo sull'ambiente virtuale di rete. Puoi scegliere il tuo intervallo di indirizzi IP, creare sottoreti e configurare liste di routing e di controllo accessi. Non è previsto alcun costo aggiuntivo per eseguire il cluster database in Amazon VPC.

Gli account hanno un VPC predefinito. Tutti i nuovi cluster database vengono creati nel VPC predefinito, salvo diversamente specificato.

Argomenti

- Uso di un cluster database in un VPC
- Scenari per accedere a un cluster database in un VPC
- Tutorial: Creazione di un Amazon VPC da utilizzare con un cluster database (solo IPv4)
- Tutorial: Creazione di un VPC per l'utilizzo con un cluster database (modalità dual-stack)

Di seguito vengono descritte le funzionalità VPC relative ai cluster database Amazon Aurora. Per ulteriori informazioni su Amazon VPC, consulta Guida alle operazioni di base di Amazon VPC e Guida per l'utente di Amazon VPC.

Uso di un cluster database in un VPC

Il cluster database deve essere all'interno di un cloud privato virtuale (VPC). Un VPC è una rete virtuale logicamente isolata dalle altre reti virtuali nel cloud. AWS Amazon VPC ti consente di avviare AWS risorse, come un di istanze Amazon o un'istanza Amazon EC2 , in un VPC. Il VPC può essere un VPC predefinito fornito con l'account o uno creato da te. Tutti VPCs sono associati al tuo account. AWS

Il VPC predefinito ha tre sottoreti che è possibile usare per isolare le risorse all'interno del VPC. Il VPC predefinito ha anche un gateway Internet che può essere usato per fornire l'accesso alle risorse all'interno del VPC dall'esterno del VPC.

Per un elenco di scenari relativi ai cluster database Amazon Aurora in un VPC , consulta Scenari per accedere a un cluster database in un VPC.

Argomenti

- Uso di un cluster database in un VPC
- Controllo della crittografia VPC
- Utilizzo di gruppi di sottoreti database
- Sottoreti condivise
- Assegnazione di indirizzi IP in Amazon Aurora
- Nascondere cluster database in un VPC da Internet
- Creazione di un cluster database in un VPC

Nei seguenti tutorial puoi imparare a creare un VPC da usare per uno scenario Amazon Aurora comune:

- Tutorial: Creazione di un Amazon VPC da utilizzare con un cluster database (solo IPv4)
- Tutorial: Creazione di un VPC per l'utilizzo con un cluster database (modalità dual-stack)

Uso di un cluster database in un VPC

Ecco alcuni consigli per l'uso di un cluster database in un VPC:

- Il VPC deve avere almeno due sottoreti. Una sottorete è un segmento dell'intervallo di indirizzi IP di un VPC che è possibile specificare e utilizzare per raggruppare i cluster database in base alle esigenze operative e di sicurezza.
- Se desideri che il cluster database nel VPC sia accessibile a livello pubblico, verifica di aver attivato gli attributi VPC DNS hostnames (Nomi host DNS) e DNS resolution (Risoluzione DNS).
- Il VPC deve includere un gruppo di sottoreti database creato da te. Puoi creare un gruppo di sottoreti di database, specificando le sottoreti create. Amazon Aurora sceglie una sottorete e un indirizzo IP all'interno di quella stessa sottorete da associare all'istanza database primaria nel cluster DB. L'istanza database primaria utilizza la zona di disponibilità che contiene la sottorete.
- Il VPC deve avere un gruppo di sicurezza VPC che consente l'accesso al cluster database.

  Per ulteriori informazioni, consulta Scenari per accedere a un cluster database in un VPC.
- I blocchi CIDR in ciascuna delle sottoreti devono essere sufficientemente grandi da contenere gli indirizzi IP di riserva per Amazon Aurora da utilizzare durante le attività di manutenzione, inclusi il failover e il dimensionamento delle risorse di calcolo. Ad esempio, un intervallo come 10.0.0.0/24 e 10.0.1.0/24 è in genere sufficientemente ampio.

- Un VPC può avere un attributo di tenancy di istanza o predefinito o dedicato. Tutte le impostazioni predefinite VPCs hanno l'attributo di tenancy dell'istanza impostato su default e un VPC predefinito può supportare qualsiasi classe di istanza DB.

  Se scegli di avere il tuo cluster di DB in un VPC dedicato in cui l'attributo di tenancy dell'istanza è impostato su dedicato, la classe di istanza DB del tuo cluster di DB deve essere uno dei tipi di istanza EC2 dedicata Amazon approvati. Ad esempio, l'istanza EC2 dedicata r5.large corrisponde alla classe di istanza DB db.r5.large. Per informazioni sulla tenancy di istanza in un VPC, consulta Istanze dedicate nella Guida per l'utente Amazon Elastic Compute Cloud.

  Per ulteriori informazioni sui tipi di istanze che possono essere presenti in un'istanza dedicata, consulta le istanze EC2 dedicate di Amazon nella pagina EC2 dei prezzi di Amazon.

  > **Note**
  >
  > Quando si imposta l'attributo di locazione dell'istanza su dedicato per un cluster database, non è garantita l'esecuzione del cluster database su un host dedicato.

Controllo della crittografia VPC

I controlli di crittografia VPC ti consentono di applicare tutto il traffico di rete encryption-in-transit all'interno del tuo. VPCs Utilizzate il controllo della crittografia per soddisfare i requisiti di conformità normativa assicurando che solo l'hardware basato su Nitro con funzionalità di crittografia possa essere fornito nelle aree designate. VPCs Il controllo della crittografia rileva inoltre i problemi di compatibilità al momento della richiesta dell'API anziché durante il provisioning. I carichi di lavoro esistenti continuano a funzionare e vengono bloccate solo le nuove richieste incompatibili.

Imposta i controlli di crittografia VPC impostando la modalità di controllo VPC su:

- disabilitato (impostazione predefinita)
- monitor
- forzato

Per verificare la modalità di controllo corrente per il tuo VPC, usa il comando Console di gestione AWS o DescribeVpcsCLI o API.

Se il tuo VPC applica la crittografia, puoi fornire solo basati su Nitro che supportano la crittografia in transito in quel VPC. Per ulteriori informazioni, consultare Tipi di classi di istanza database. Per informazioni sulle istanze Nitro, consulta Instances built on the AWS Nitro System nella Amazon User Guide. EC2

> **Note**
>
> `VpcEncryptionControlViolationException`

Aurora ServerlessPer MySQL e PostreSQL, il controllo della crittografia richiede la versione 3 o successiva della piattaforma.

Utilizzo di gruppi di sottoreti database

Le sottoreti sono segmenti di un intervallo di indirizzi IP di un VPC che si designa per raggruppare le risorse in base alle esigenze operative e di sicurezza. Un gruppo di sottoreti DB è una raccolta di sottoreti (generalmente private) creata in un VPC e che è possibile indicare per i cluster database. Utilizzando un gruppo di sottoreti DB, è possibile specificare un particolare VPC durante la creazione di cluster di istanze utilizzando AWS CLI l'API o RDS. Se utilizzi la console, puoi scegliere il VPC e i gruppi di sottorete che desideri usare.

Ogni gruppo di sottoreti database deve avere almeno due zone di disponibilità in una determinata Regione AWS. Quando si crea un cluster database in un VPC, è necessario selezionare anche un gruppo di sottoreti DB. Dal gruppo di sottoreti DB, Amazon Aurora sceglie una sottorete e un indirizzo IP all'interno di essa da associare all'istanza database primaria nel cluster database. Il database utilizza la zona di disponibilità che contiene la sottorete. Aurora Amazon assegna sempre un indirizzo IP da una sottorete con spazio libero per gli indirizzi IP.

Le sottoreti di un gruppo di sottoreti DB sono pubbliche o private. Le sottoreti sono pubbliche o private, a seconda della configurazione impostata per le relative liste di controllo degli accessi alla rete (rete) e le tabelle di routing. ACLs Affinché un cluster database possa essere accessibile a livello pubblico, tutte le sottoreti nel gruppo di sottoreti database devono essere pubbliche. Se una sottorete associata a un cluster database accessibile pubblicamente cambia da pubblica a privata, ciò può avere ripercussioni sulla disponibilità del cluster database.

Per creare un gruppo di sottoreti DB che supporti la modalità dual-stack, assicuratevi che a ogni sottorete che aggiungete al gruppo di sottorete DB sia associato un blocco CIDR del protocollo

Internet versione 6 (). IPv6 Per ulteriori informazioni, consulta Assegnazione di indirizzi IP in Amazon Aurora la sezione «Migrazione verso» IPv6 nella Amazon VPC User Guide.

Quando Amazon Aurora crea un cluster database in un VPC, assegna un'interfaccia di rete al cluster database usando un indirizzo IP dal gruppo di sottoreti del database. Tuttavia, consigliamo vivamente di usare il nome del sistema dei nomi di dominio (DNS) per collegare il cluster database, perché l'indirizzo IP sottostante varia durante un failover.

> **Note**
>
> Per ogni cluster database in esecuzione in un VPC, assicurati di riservare almeno un indirizzo in ogni sottorete nel gruppo di sottoreti DB per l'utilizzo da parte di Amazon Aurora per le operazioni di ripristino.

Sottoreti condivise

Puoi creare un cluster database in un VPC condiviso.

Alcune considerazioni da tenere a mente durante l'utilizzo di shared: VPCs

- È possibile spostare un cluster database da una sottorete VPC condivisa a una sottorete VPC non condivisa e viceversa.
- I membri di un VPC condiviso devono creare un gruppo di sicurezza nel VPC per consentire loro di creare un cluster database.
- I proprietari e i membri di un VPC condiviso possono accedere al database utilizzando le query SQL. Tuttavia, solo il creatore di una risorsa può effettuare chiamate API sulla risorsa.

Assegnazione di indirizzi IP in Amazon Aurora

Gli indirizzi IP permettono alle risorse nel VPC di comunicare tra loro e con le risorse su Internet. Amazon Aurora supporta IPv4 entrambi i protocolli di IPv6 indirizzamento. Per impostazione predefinita, Amazon Aurora e Amazon VPC utilizzano il protocollo di indirizzamento. IPv4 Non puoi disattivare questo comportamento. Quando crei un VPC, assicurati di specificare un blocco IPv4 CIDR (un intervallo di indirizzi privati IPv4).

Il supporto per il IPv6 protocollo amplia il numero di indirizzi IP supportati. Utilizzando il IPv6 protocollo, ti assicuri di disporre di indirizzi disponibili sufficienti per la crescita futura di Internet. Le

risorse RDS nuove ed esistenti possono utilizzare IPv4 e IPv6 indirizzare all'interno del tuo VPC. La configurazione, la protezione e la traduzione del traffico di rete tra i due protocolli utilizzati in diverse parti di un'applicazione può causare sovraccarico operativo. Puoi standardizzare il IPv6 protocollo per le risorse Amazon RDS per semplificare la configurazione della rete.

Argomenti

- IPv4 indirizzi
- IPv6 indirizzi
- Modalità dual-stack

IPv4 indirizzi

Quando si crea un VPC, è necessario specificare un intervallo di IPv4 indirizzi per il VPC sotto forma di blocco CIDR, ad esempio. `10.0.0.0/16` Un gruppo di sottoreti DB definisce l'intervallo di indirizzi IP in questo blocco CIDR che può essere usato da cluster database. Questi indirizzi IP possono essere privati o pubblici.

Un IPv4 indirizzo privato è un indirizzo IP non raggiungibile su Internet. Puoi utilizzare IPv4 indirizzi privati per la comunicazione tra il tuo cluster di DB e altre risorse, come EC2 le istanze Amazon, nello stesso VPC. Ogni cluster database dispone di un indirizzo IP privato per la comunicazione nel VPC.

Un indirizzo IP pubblico è un IPv4 indirizzo raggiungibile da Internet. Puoi utilizzare gli indirizzi pubblici per la comunicazione tra cluster database e risorse su Internet, ad esempio un client SQL. Puoi controllare se cluster database ricevono un indirizzo IP pubblico.

Per un tutorial che mostra come creare un VPC con solo IPv4 indirizzi privati da utilizzare per uno scenario Amazon comune, consulta. Tutorial: Creazione di un Amazon VPC da utilizzare con un cluster database (solo IPv4)

IPv6 indirizzi

Facoltativamente, puoi associare un blocco IPv6 CIDR al tuo VPC e alle sottoreti e assegnare IPv6 gli indirizzi di quel blocco alle risorse del tuo VPC. Ogni indirizzo è unico a livello globale. IPv6

Il blocco IPv6 CIDR per il tuo VPC viene assegnato automaticamente dal pool IPv6 di indirizzi di Amazon. Non è possibile scegliere l'intervallo in modo autonomo.

Quando ti connetti a un IPv6 indirizzo, assicurati che siano soddisfatte le seguenti condizioni:

- Il client è configurato in modo da consentire il trasferimento del traffico tra IPv6 client e database.
- I gruppi di sicurezza RDS utilizzati dall'istanza DB sono configurati correttamente in modo da consentire il trasferimento del traffico tra client e database. IPv6
- Lo stack del sistema operativo client consente il traffico sull' IPv6 indirizzo e i driver e le librerie del sistema operativo sono configurati per scegliere l'endpoint di istanza DB predefinito corretto (uno o due IPv4). IPv6

Per ulteriori informazioni IPv6, consulta la sezione Indirizzamento IP nella Amazon VPC User Guide.

Modalità dual-stack

Quando un cluster di DB è in grado di comunicare sia tramite il protocollo di IPv4 IPv6 indirizzamento che tramite quello di indirizzamento, viene eseguito in modalità dual-stack. Pertanto, le risorse possono comunicare con il cluster di DB su o IPv4 entrambi IPv6. RDS disabilita l'accesso a Internet Gateway per gli IPv6 endpoint delle istanze DB private in modalità dual-stack. RDS esegue questa operazione per garantire che gli IPv6 endpoint siano privati e accessibili solo dall'interno del VPC.

Argomenti

- Modalità dual-stack e gruppi di sottoreti database
- Utilizzo di istanze database in modalità dual-stack
- Modifica dei cluster di IPv4 sole DB per utilizzare la modalità dual-stack
- Disponibilità di cluster database di rete dual-stack
- Limitazioni per cluster database di rete dual-stack

Per un tutorial che mostra come creare un VPC con entrambi IPv4 IPv6 gli indirizzi da utilizzare per uno scenario Amazon comune, consulta. Tutorial: Creazione di un VPC per l'utilizzo con un cluster database (modalità dual-stack)

Modalità dual-stack e gruppi di sottoreti database

Per utilizzare la modalità dual-stack, assicurati che a ogni sottorete del gruppo di sottoreti DB associato al cluster di DB sia associato un blocco CIDR. IPv6 Per soddisfare questo requisito, puoi creare un nuovo gruppo di sottoreti database o modificare un gruppo di sottoreti database esistente. Quando cluster database sono in modalità dual-stack, i client possono connettersi normalmente. Assicurati che i firewall di sicurezza dei client e i gruppi di sicurezza delle istanze RDS DB siano configurati accuratamente per consentire il trasferimento del traffico. IPv6 Per connettersi, i client

utilizzano l'endpoint dell'istanza principale del cluster database. Le applicazioni client possono specificare il protocollo preferito per la connessione a un database. In modalità dual-stack, il cluster di DB rileva il protocollo di rete preferito del client IPv6, IPv4 o lo utilizza per la connessione.

Se un gruppo di sottoreti database smette di supportare la modalità dual-stack a causa dell'eliminazione della sottorete o dell'annullamento dell'associazione con il blocco CIDR, si può verificare una situazione di stato di rete incompatibile per le istanze database associate al gruppo di sottoreti database. Inoltre, non puoi utilizzare il gruppo di sottoreti database quando crei un nuovo cluster database in modalità dual-stack.

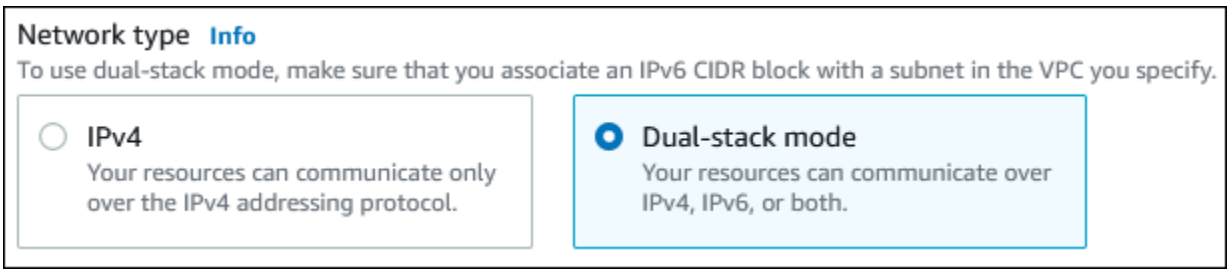
Per determinare se un gruppo di sottoreti DB supporta la modalità dual-stack utilizzando il Console di gestione AWS, visualizza il Tipo di rete nella pagina dei dettagli del gruppo di sottoreti DB. Per determinare se un gruppo di sottoreti DB supporta la modalità dual-stack utilizzando il AWS CLI, esegui il comando e visualizza nell'output. describe-db-subnet-groups`SupportedNetworkTypes`

Le repliche di lettura vengono trattate come istanze database indipendenti e possono avere un tipo di rete diverso dall'istanza database principale. Se si modifica il tipo di rete dell'istanza database principale di una replica di lettura, tale modifica non interessa la replica di lettura. Quando si ripristina un'istanza database, è possibile ripristinarla su qualsiasi tipo di rete supportato.

Utilizzo di istanze database in modalità dual-stack

Quando si crea o si modifica un cluster di DB, è possibile specificare la modalità dual-stack per consentire alle risorse di comunicare con il cluster di o entrambi. IPv4 IPv6

Quando si utilizza Console di gestione AWS per creare o modificare un'istanza DB, è possibile specificare la modalità dual-stack nella sezione Tipo di rete. L'immagine seguente mostra la sezione Network type (Tipo di rete) nella console.



Quando utilizzi il AWS CLI per creare o modificare un cluster di DB, imposta l'`--network-type`opzione per utilizzare la modalità `DUAL` dual-stack. Quando usi l'API RDS per creare o modificare un cluster database, per utilizzare la modalità dual-stack imposta il parametro `NetworkType` su `DUAL`. Quando modifichi il tipo di rete di un'istanza database, è possibile che

si verifichi un periodo di inattività. Se la modalità dual-stack non è supportata dalla versione del motore del database o dal gruppo di sottoreti database in uso, viene restituito l'errore `NetworkTypeNotSupported`.

Ulteriori informazioni sulla creazione di un cluster di database sono disponibili in Creazione di un cluster database Amazon Aurora. Per ulteriori informazioni sulla modifica di un cluster di database, consultare Modifica di un cluster database Amazon Aurora.

Per determinare se un cluster database è in modalità dual-stack utilizzando la console, visualizza l'opzione Network type (Tipo di rete) nella scheda Connectivity & security (Connettività e sicurezza) per il cluster database in questione.

Modifica dei cluster di IPv4 sole DB per utilizzare la modalità dual-stack

A tale scopo, devi modificare il tipo di rete del cluster database. La modifica potrebbe comportare tempi di inattività.

Ti consigliamo di modificare il tipo di rete dei cluster Amazon Aurora durante una finestra di manutenzione. L'impostazione predefinita del tipo di rete delle nuove istanze sulla modalità dual stack non è al momento supportata. È possibile impostare il tipo di rete manualmente utilizzando il comando `modify-db-cluster`.

Prima di modificare un cluster database per utilizzare la modalità dual-stack, assicurati che il gruppo di sottoreti DB supporti la modalità dual-stack. Se il gruppo di sottoreti DB associato al cluster database non supporta la modalità dual-stack, specifica un gruppo di sottoreti database diverso che supporta tale modalità quando modifichi il cluster database. La modifica del gruppo di sottoreti di database di un cluster di database può causare tempi di inattività.

Se si modifica il gruppo di sottoreti di database di un cluster di database prima di modificare il cluster di database per l'utilizzo della modalità Dual stack, assicurati che il gruppo di sottoreti di database sia valido per il cluster di database prima e dopo la modifica.

Ti consigliamo di eseguire l'modify-db-clusterAPI solo con il `--network-type` parametro con valore `DUAL` per modificare la rete di un cluster Amazon Aurora in modalità dual-stack. L'aggiunta di altri parametri al parametro `--network-type` nella stessa chiamata API potrebbe causare tempi di inattività.

Se non riesci a connetterti al cluster di DB dopo la modifica, assicurati che i firewall di sicurezza del client e del database e le tabelle di routing siano configurati accuratamente per consentire il traffico

verso il database sulla rete selezionata (IPv4 o IPv6). Potrebbe inoltre essere necessario modificare i parametri del sistema operativo, le librerie o i driver per connetterti utilizzando un IPv6 indirizzo.

Per modificare un cluster IPv4 di sole DB per utilizzare la modalità dual-stack

1. Modificare un gruppo di sottoreti database per supportare la modalità dual-stack o creare un gruppo di sottoreti database che supporti la modalità dual-stack:

   a. Associa un blocco IPv6 CIDR al tuo VPC.

      Per istruzioni, consulta Aggiungere un blocco IPv6 CIDR al tuo VPC nella Amazon VPC User Guide.

   b. Collega il blocco IPv6 CIDR a tutte le sottoreti del gruppo di sottoreti DB.

      Per istruzioni, consulta Aggiungere un blocco IPv6 CIDR alla sottorete nella Amazon VPC User Guide.

   c. Verificare che il gruppo di sottoreti database supporti la modalità dual-stack.

      Se utilizzi il Console di gestione AWS, seleziona il gruppo di sottoreti DB e assicurati che il valore Supported network types sia Dual,. IPv4

      Se utilizzi il AWS CLI, esegui il describe-db-subnet-groupscomando e assicurati che il `SupportedNetworkType` valore per l'istanza DB sia`Dual, IPv4`.

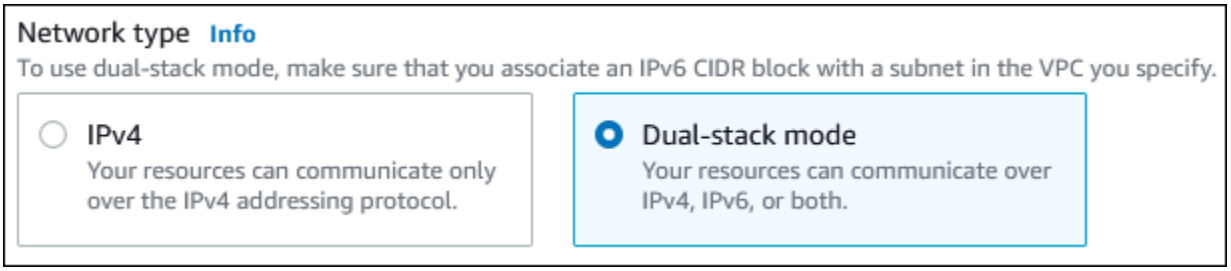
2. Modifica il gruppo di sicurezza associato al cluster di DB per consentire IPv6 le connessioni al database o crea un nuovo gruppo di sicurezza che consenta IPv6 le connessioni.

   Per le istruzioni, vedere Regole del gruppo di sicurezza nella Guida per l'utente di Amazon VPC.

3. Modifica il cluster database in modo che supporti la modalità dual-stack. A questo scopo, imposta l'opzione Network type (Tipo di rete) su Dual-stack mode (Modalità dual-stack).

   In caso di utilizzo della console, accertati che le impostazioni seguenti siano corrette:

   - Tipo di rete–Dual-stack mode (Modalità dual-stack)



- DB subnet group (Gruppo di sottoreti DB): gruppo di sottoreti database configurato in un passaggio precedente
- Security group (Gruppo di sicurezza): gruppo di sicurezza di default configurato in un passaggio precedente

Se utilizzi il AWS CLI, assicurati che le seguenti impostazioni siano corrette:

- `--network-type` – `dual`
- `--db-subnet-group-name`: gruppo di sottoreti database configurato in un passaggio precedente
- `--vpc-security-group-ids`: gruppo di sicurezza VPC configurato in un passaggio precedente

Ad esempio:

```
aws rds modify-db-cluster --db-cluster-identifier my-cluster --network-type "DUAL"
```

4. Verifica che il cluster database supporti la modalità dual-stack.

   Se utilizzi la console, scegli la scheda Configuration (Configurazione) per il cluster database. In quella scheda, assicurati che il valore dell'opzione Network type (Tipo di rete) sia Dual-stack mode (Modalità dual-stack).

   Se utilizzi il AWS CLI, esegui il describe-db-clusterscomando e assicurati che il `NetworkType` valore per il cluster DB sia`dual`.

   Esegui il `dig` comando sull'endpoint dell'istanza Writer DB per identificare l' IPv6indirizzo ad esso associato.

   ```
   dig db-instance-endpoint AAAA
   ```

   Utilizza l'endpoint dell'istanza Writer DB, non l' IPv6 indirizzo, per connetterti al cluster di DB.

Disponibilità di cluster database di rete dual-stack

I cluster DB di rete dual-stack sono disponibili in tutti Regioni AWS gli ambienti ad eccezione dei seguenti:

- Asia Pacific (Hyderabad)
- Asia Pacifico (Malesia)
- Asia Pacifico (Melbourne)
- Asia Pacifico (Thailandia)
- Canada occidentale (Calgary)
- Europa (Spagna)
- Europa (Zurigo)
- Israele (Tel Aviv)
- Messico (centrale)
- Medio Oriente (Emirati Arabi Uniti)

Le seguenti versioni del motore del database supportano i cluster di database di rete dual-stack:

- Versioni Aurora MySQL:
  - 3.02 o versioni successive alla 3
  - 2.09.1 o versioni successive alla 2

  Per ulteriori informazioni sulle versioni di Aurora MySQL, consulta la sezione Note di rilascio di Aurora MySQL.
- Versioni di Aurora PostgreSQL:
  - 15.2 e tutte le versioni successive
  - 14.3 o versioni successive alla 14
  - 13.7 o versioni successive alla 13

  Per ulteriori informazioni sulle versioni di Aurora PostgreSQL, consulta Note di rilascio di Aurora PostgreSQL.

Limitazioni per cluster database di rete dual-stack

Le seguenti limitazioni si applicano ai cluster database di rete dual-stack:

- I cluster di DB non possono utilizzare esclusivamente il protocollo. IPv6 Possono utilizzare IPv4 esclusivamente oppure possono utilizzare il IPv6 protocollo IPv4 and (modalità dual-stack).
- Amazon RDS non supporta IPv6 sottoreti native.

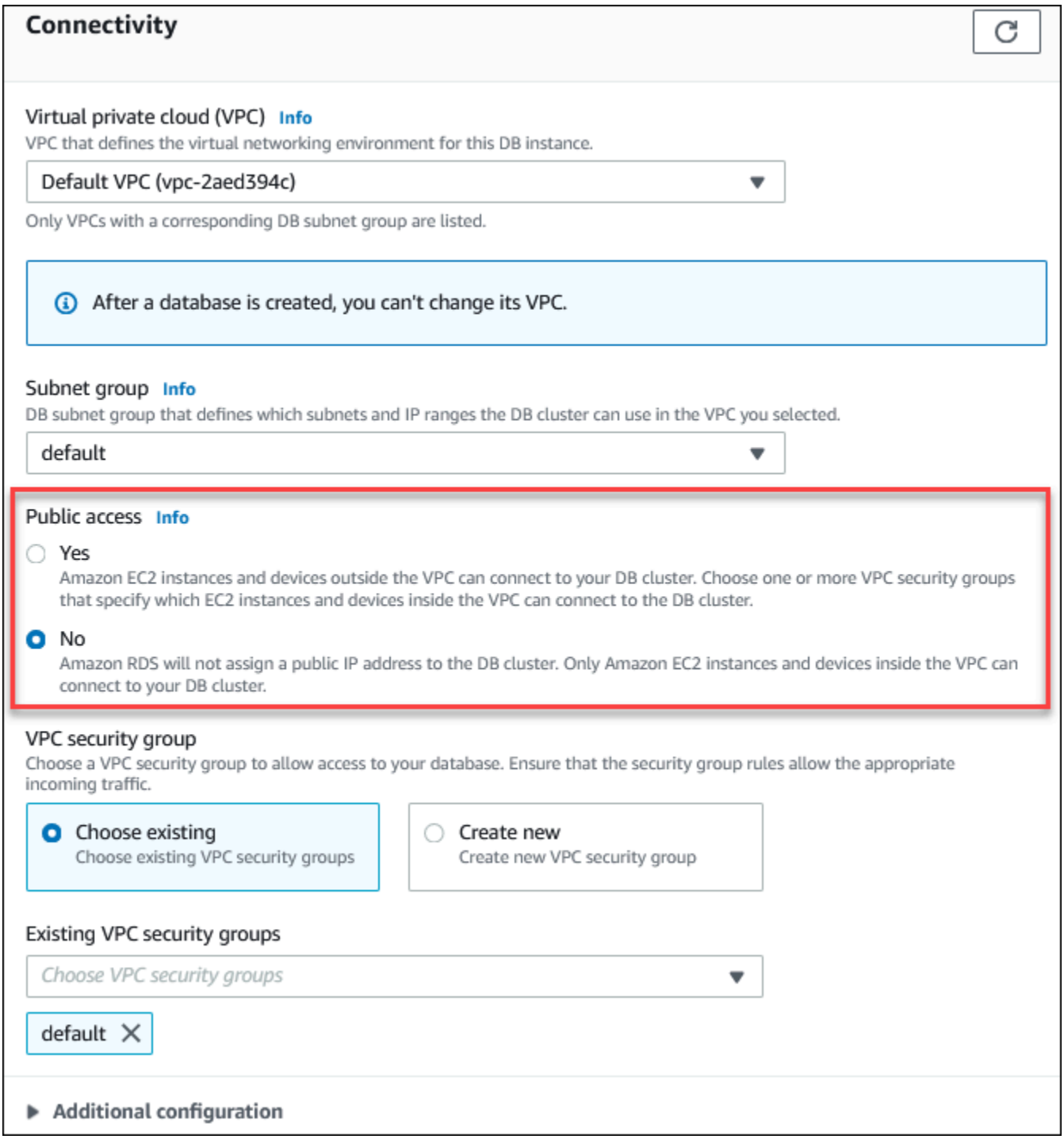
- I cluster database che utilizzano la modalità dual-stack devono essere di tipo privato. Non possono essere accessibili pubblicamente.
- Non è possibile utilizzare il proxy RDS con cluster database in modalità dual-stack.

Nascondere cluster database in un VPC da Internet

Uno scenario comune di Amazon Aurora consiste nell'avere un VPC in cui è presente un'istanza EC2 Amazon con un'applicazione Web rivolta al pubblico e un cluster di istanze non accessibile al pubblico. Ad esempio, puoi creare un VPC con una sottorete pubblica e una sottorete privata. EC2 le istanze che funzionano come server Web possono essere distribuite nella sottorete pubblica. L'implementazione di cluster database viene invece eseguita nella sottorete privata. In tale implementazione, solo i server Web hanno accesso ai cluster database. Per un'illustrazione di questo scenario, consulta Un cluster di database in un VPC a cui accede un'istanza EC2 nello stesso VPC.

Quando si avvia un cluster database all'interno di un VPC, il cluster database dispone di un indirizzo IP privato per il traffico all'interno del VPC. Questo indirizzo IP privato non è accessibile pubblicamente. Puoi utilizzare l'opzione Public access (Accesso pubblico) per indicare se il cluster database dispone anche di un indirizzo IP pubblico oltre all'indirizzo IP privato. Se il cluster database è definito come accessibile pubblicamente, il relativo endpoint DNS utilizza l'indirizzo IP privato dall'interno del VPC. Utilizza invece l'indirizzo IP pubblico dall'esterno del VPC. L'accesso al cluster database è in ultima analisi controllato dal gruppo di sicurezza in uso. Questo accesso pubblico non è consentito se il gruppo di sicurezza assegnato al cluster database non include regole in entrata che lo consentono. Per un cluster database che deve essere accessibile pubblicamente, le sottoreti nel relativo gruppo di sottoreti DB devono disporre di un gateway Internet. Per ulteriori informazioni, consulta Impossibile connettersi all'istanza database di Amazon RDS

Puoi modificare un cluster database per attivare o disattivare l'accessibilità pubblica modificando l'opzione Public access (Accesso pubblico). Nella figura seguente viene illustrata l'opzione Public access (Accesso pubblico) nella sezione Additional connectivity configuration (Configurazioni di connettività aggiuntiva) . Per impostare l'opzione, apri la sezione Additional connectivity configuration (Configurazioni di connettività aggiuntiva) nella sezione Connectivity (Connettività) .



Per informazioni sulla modifica di un'istanza database per impostare l'opzione Public access (Accesso pubblico), consulta Modifica di un'istanza database in un cluster database.

Creazione di un cluster database in un VPC

Le procedure seguenti aiutano a creare un cluster database in un VPC. Per utilizzare il VPC predefinito, puoi iniziare con il passaggio 2 e utilizzare il VPC e il gruppo di sottoreti DB creati automaticamente. Se desideri creare un VPC aggiuntivo, puoi creare un nuovo VPC.

> **Note**
>
> Se desideri che il cluster database nel VPC sia pubblicamente accessibile, devi aggiornare le informazioni DNS per il VPC attivando gli attributi VPC DNS hostnames (Nomi host DNS) e DNS resolution (Risoluzione DNS). Per informazioni sull'aggiornamento delle informazioni DNS per un'istanza VPC, consulta Aggiornamento del supporto DNS per il VPC.

Segui questa procedura per creare un'istanza database in un VPC:

- Fase 1. Creazione di un VPC
- Fase 2: creazione di un gruppo di sottoreti database
- Fase 3: creazione di un gruppo di sicurezza VPC
- Passaggio 4: creazione di un'istanza database nel VPC

Fase 1. Creazione di un VPC

Crea un VPC con sottoreti in almeno due zone di disponibilità. Usi queste sottoreti quando crei un gruppo di sottoreti database. Se disponi di un VPC di default, viene creata automaticamente una sottorete in ciascuna zona di disponibilità nella Regione AWS.

Per ulteriori informazioni, consulta Creazione di un VPC con sottoreti pubbliche e private oppure Creazione di un VPC nella Guida per l'utente di Amazon VPC..

Fase 2: creazione di un gruppo di sottoreti database

Un gruppo di sottoreti DB è una raccolta di sottoreti (generalmente private) creata per un VPC e che è possibile definire per i cluster database. Un gruppo di sottoreti DB consente di specificare un particolare VPC quando si creano cluster di istanze utilizzando AWS CLI l'API o RDS. Se utilizzi la console, puoi scegliere il VPC e le sottoreti che desideri usare. Ogni gruppo di sottoreti database deve avere almeno una sottorete in almeno due zone di disponibilità nella Regione AWS. Come best

practice, ogni gruppo di sottoreti database deve disporre di almeno una sottorete per ogni zona di disponibilità nella Regione AWS.

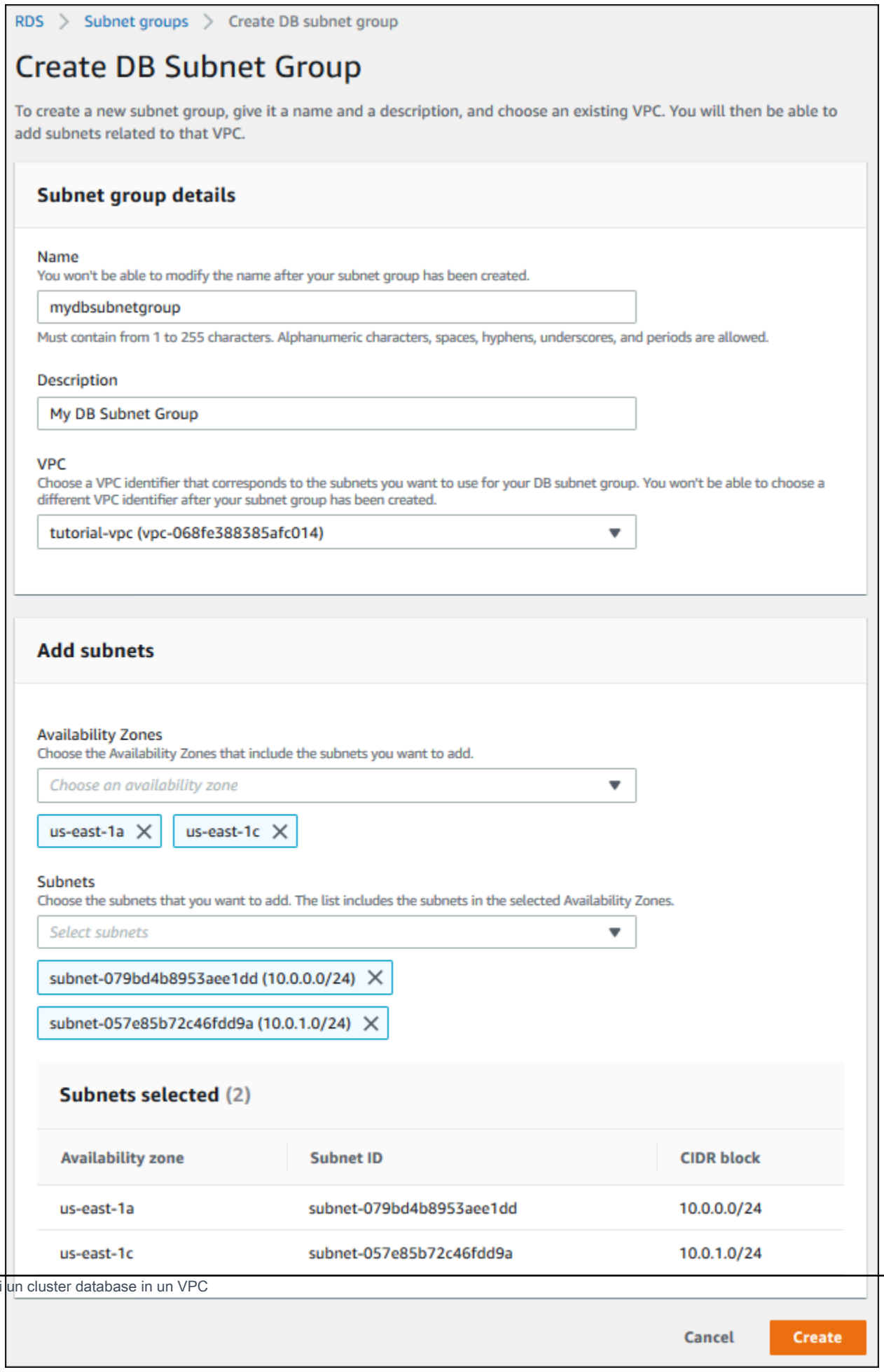
Per un cluster database che deve essere accessibile pubblicamente, le sottoreti nel gruppo di sottoreti database devono disporre di un gateway Internet. Per ulteriori informazioni sui gateway Internet, consulta Eseguire la connessione a Internet utilizzando un gateway Internet nella Guida per l'utente di Amazon VPC.

Quando crei un cluster database in un VPC, puoi selezionare un gruppo di sottoreti DB. Amazon Aurora sceglie una sottorete e un indirizzo IP al suo interno da associare al cluster database. Se non esistono gruppi di sottoreti DB, Amazon Aurora crea un gruppo di sottoreti predefinito quando crei un cluster database. Amazon Aurora crea e associa un'interfaccia di rete elastica al cluster database con tale indirizzo IP. Il cluster database utilizza la zona di disponibilità contenente la sottorete.

In questo passaggio, si crea un gruppo di sottoreti database e si aggiungono le sottoreti create per il VPC.

Creare un gruppo di sottoreti database

1. Aprire la console Amazon RDS all'indirizzo https://console.aws.amazon.com/rds/.
2. Nel pannello di navigazione selezionare Subnet groups (Gruppi di sottoreti).
3. Scegli Create DB Subnet Group (Crea gruppo di sottoreti del database).
4. Per Name (Nome), digita il nome del gruppo di sottoreti database.
5. Per Description Descrizione), digita una descrizione per il gruppo di sottoreti database.
6. In VPC, scegli il VPC predefinito o il VPC creato in precedenza.
7. Nella sezione Aggiungi sottoreti, scegliere le zone di disponibilità che includono le sottoreti da Zone di disponibilità, quindi scegliere le sottoreti da Sottoreti.


RDS > Subnet groups > Create DB subnet group
Create DB Subnet Group
To create a new subnet group, give it a name and a description, and choose an existing VPC. You will then be able to add subnets related to that VPC.
Subnet group details
Name
You won't be able to modify the name after your subnet group has been created.
mydbsubnetgroup
Must contain from 1 to 255 characters. Alphanumeric characters, spaces, hyphens, underscores, and periods are allowed.
Description
My DB Subnet Group
VPC
Choose a VPC identifier that corresponds to the subnets you want to use for your DB subnet group. You won't be able to choose a different VPC identifier after your subnet group has been created.
tutorial-vpc (vpc-068fe388385afc014)
Add subnets
Availability Zones
Choose the Availability Zones that include the subnets you want to add.
Choose an availability zone
us-east-1a
us-east-1c
Subnets
Choose the subnets that you want to add. The list includes the subnets in the selected Availability Zones.
Select subnets
subnet-079bd4b8953aee1dd (10.0.0.0/24)
subnet-057e85b72c46fdd9a (10.0.1.0/24)
Subnets selected (2)
Availability zone
Subnet ID
CIDR block
us-east-1a
subnet-079bd4b8953aee1dd
10.0.0.0/24
us-east-1c
subnet-057e85b72c46fdd9a
10.0.1.0/24
Cancel
Create

8. Scegliere Create (Crea).

   Il nuovo gruppo di sottoreti database viene visualizzato nell'elenco dei gruppi di sottoreti database sulla console RDS. Puoi scegliere il gruppo di sottoreti database per visualizzare i dettagli, comprese tutte le sottoreti associate al gruppo, nel riquadro dei dettagli nella parte inferiore della finestra.

Fase 3: creazione di un gruppo di sicurezza VPC

Prima di creare un cluster database, devi creare un gruppo di sicurezza VPC da associare tale cluster database. Se non crei un gruppo di sicurezza VPC, puoi utilizzare il gruppo di sicurezza predefinito quando crei un cluster database. Per istruzioni su come creare un gruppo di sicurezza per il cluster database, consulta Creazione di un gruppo di sicurezza VPC per un cluster database privato oppure Controlla il traffico verso le risorse utilizzando gruppi di sicurezza nella Guida per l'utente di Amazon VPC.

Passaggio 4: creazione di un'istanza database nel VPC

In questo passaggio, si crea un cluster database e si utilizza il nome VPC, il gruppo di sottoreti DB e il gruppo di sicurezza VPC creato nel passaggio precedente.

> **Note**
>
> Se desideri che il cluster database nel VPC sia pubblicamente accessibile, devi abilitare gli attributi VPC DNS hostnames (Nomi host DNS) e DNS resolution (Risoluzione DNS). Per ulteriori informazioni, consulta Attributi DNS per il VPC nella Guida per l'utente di Amazon VPC.

Per informazioni dettagliate su come creare un cluster di database, consulta Creazione di un cluster database Amazon Aurora.

Quando richiesto nella sezione Connectivity (Connettività), inserisci il nome VPC, il gruppo di sottoreti DB e il gruppo di sicurezza VPC.

> **Note**
>
> L'aggiornamento VPCs non è attualmente supportato per i cluster Aurora DB.

Scenari per accedere a un cluster database in un VPC

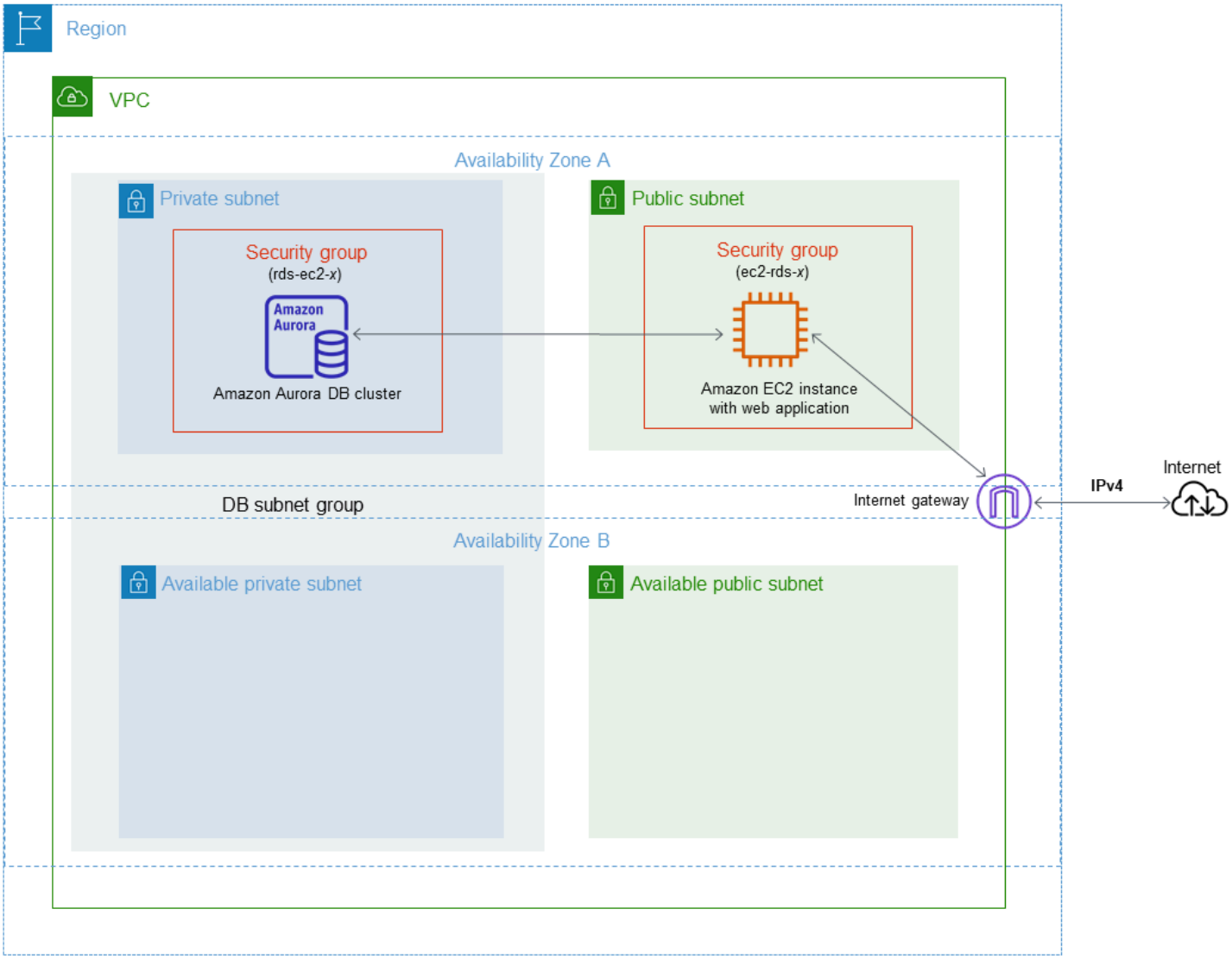
Amazon Aurora supporta i seguenti scenari per accedere a un cluster database in un VPC:

- Un’istanza Amazon EC2 nella stessa VPC
- Un'istanza EC2 in un VPC diverso
- Un'applicazione client tramite internet
- Una rete privata

Un cluster di database in un VPC a cui accede un’istanza EC2 nello stesso VPC

Un uso comune di un cluster di database in un VPC è quello di condividere dati con un server di applicazione in esecuzione in un’istanza EC2 nello stesso VPC.

Il seguente diagramma mostra questo scenario.



Il modo più semplice per gestire l'accesso tra istanze EC2 e cluster di database nello stesso VPC consiste nel fare quanto segue:

- Creare un gruppo di sicurezza VPC in cui si troveranno i cluster database. Questo gruppo di sicurezza può essere usato per limitare l'accesso ai cluster database. Ad esempio, puoi creare una regola personalizzata per questo gruppo di sicurezza. Ciò potrebbe consentire l'accesso TCP usando la porta assegnata al cluster database al momento della creazione della stessa e un indirizzo IP utilizzato per accedere al cluster database per lo sviluppo o per altri scopi.
- Crea un gruppo di sicurezza VPC in cui si troveranno le istanze EC2 (server Web e client). Questo gruppo di sicurezza può, se necessario, consentire l'accesso all'istanza EC2 da Internet tramite la tabella di routing del VPC. Ad esempio, può impostare regole in questo gruppo di sicurezza per consentire l'accesso TCP all'istanza EC2 sulla porta 22.

- Creare regole personalizzate nel gruppo di sicurezza per i cluster database che consentono connessioni dal gruppo di sicurezza creato per le istanze EC2. Queste regole potrebbero consentire a qualsiasi membro del gruppo di sicurezza di accedere ai cluster database.

È disponibile una sottorete pubblica e privata aggiuntiva in una zona di disponibilità separata. Un gruppo di sottoreti DB RDS richiede una sottorete in almeno due zone di disponibilità. La sottorete aggiuntiva semplifica il passaggio a un'implementazione Multi-AZ di un'istanza DB in futuro.

Per una dimostrazione che mostri come creare un VPC con sottoreti pubbliche e private per questo scenario, consulta Tutorial: Creazione di un Amazon VPC da utilizzare con un cluster database (solo IPv4).

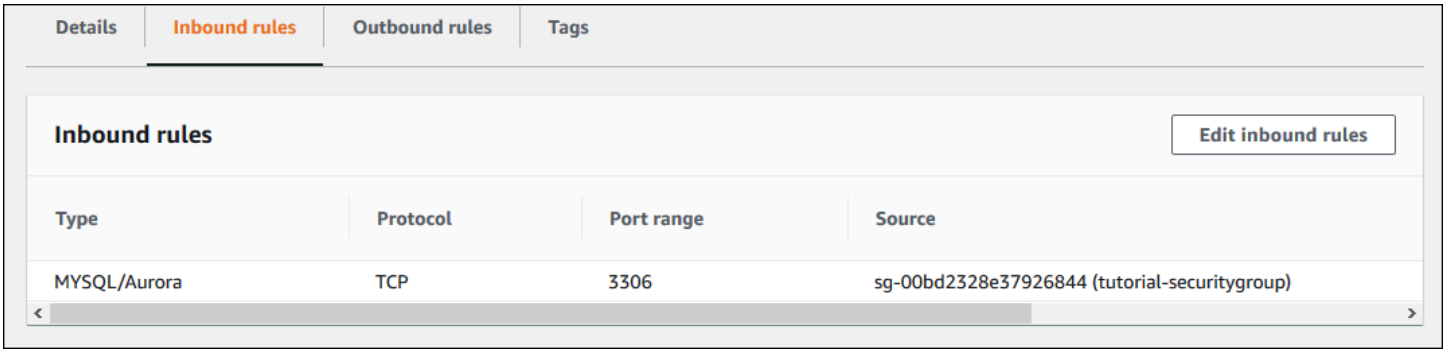
> **Tip**
>
> Quando crei un cluster database, puoi configurare automaticamente la connettività di rete tra un'istanza Amazon EC2 e un cluster database. Per ulteriori informazioni, consulta Configurazione della connettività di rete automatica con un'istanza EC2 .

Per creare una regola in un gruppo di sicurezza VPC che consente delle connessioni da un altro gruppo di sicurezza, esegui la procedura seguente:

1. Accedere ad Console di gestione AWS e aprire la console Amazon VPC all'indirizzo https://console.aws.amazon.com/vpc.
2. Fai clic su Security Groups (Gruppi di sicurezza) nel pannello di navigazione.
3. Scegli o crea un gruppo di sicurezza per il quale desideri concedere l'accesso ai membri di un altro gruppo di sicurezza. Nello scenario precedente, questo è il gruppo di sicurezza utilizzato per i cluster database. Seleziona la scheda Regole in entrata, quindi scegli Modifica regola.
4. Nella scheda Modifica regole in entrata, seleziona Aggiungi regola.
5. Per Tipo, scegli la voce che corrisponde alla porta utilizzata durante la creazione del cluster database, ad esempio MySQL/Aurora.
6. Nella casella Origine iniziare a digitare l'ID del gruppo di sicurezza, che elenca i gruppi di sicurezza corrispondenti. Scegli il gruppo di sicurezza con i membri che desideri abbiano accesso alle risorse protette da questo gruppo di sicurezza. Nello scenario precedente, questo è il gruppo di sicurezza utilizzato per le istanze EC2.

7. Se necessario, ripeti i passaggi per il protocollo TCP creando una regola con All TCP (Tutti i TCP) come Tipo e il gruppo di sicurezza nella casella Source (Origine). Se desideri usare il protocollo UDP, crea una regola con All UDP (Tutti i UDP) come Type (Tipo) e il gruppo di sicurezza nella casella Source (Origine).
8. Scegliere Salva regole.

Nella schermata seguente viene illustrata una regola in entrata con un gruppo di sicurezza per la relativa origine.

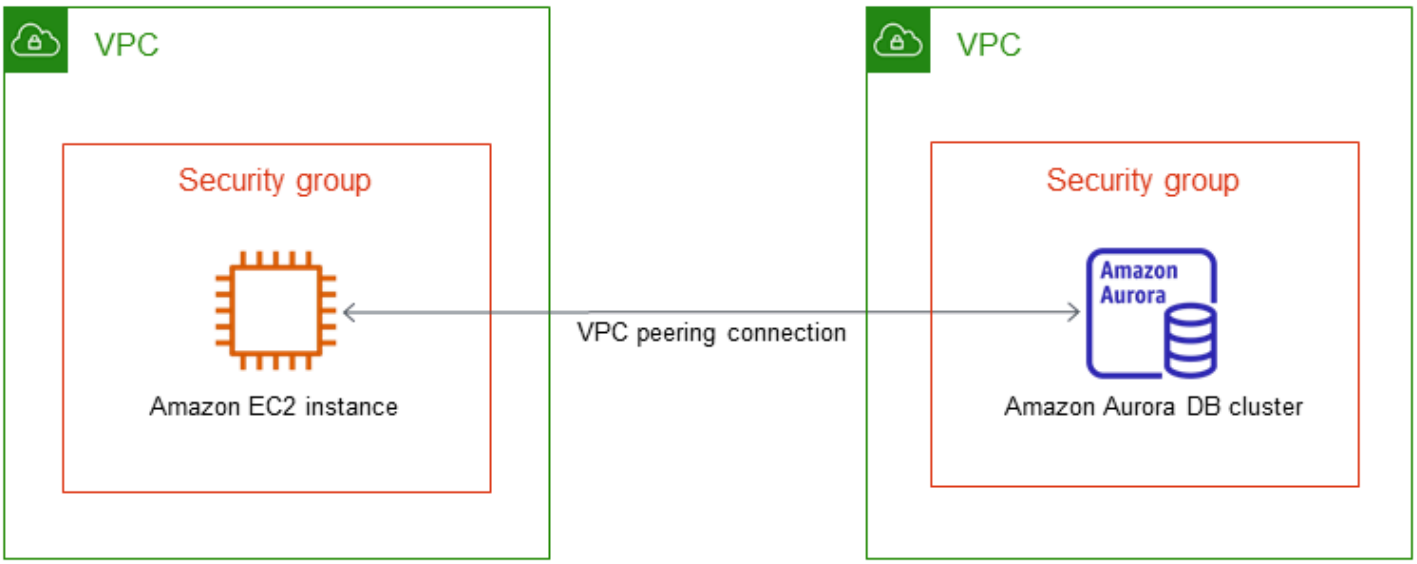


Per ulteriori informazioni sulla connessione al cluster di database dall'istanza EC2, consulta Connessione a un cluster database Amazon Aurora.

Un cluster database in un VPC a cui accede un'istanza EC2 in un VPC diverso

Quando i cluster database si trova in un VPC diverso dall'istanza EC2 che si sta utilizzando per accedervi, puoi utilizzare il peering VPC per accedere al cluster database.

Il seguente diagramma mostra questo scenario.

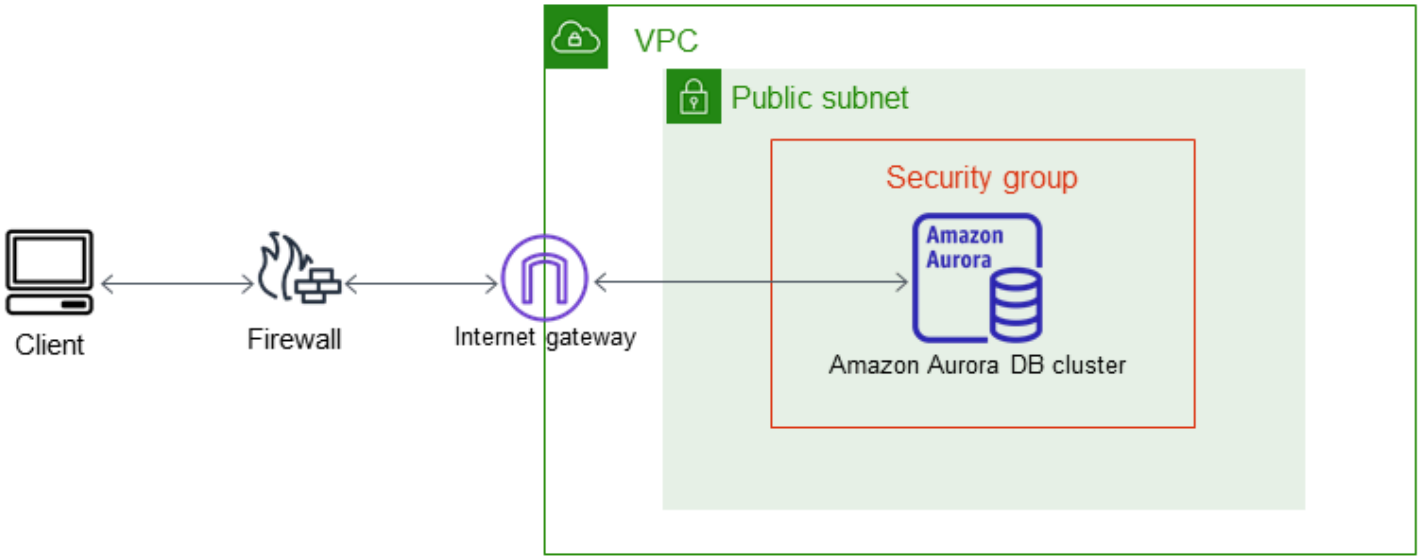


Una connessione di peering di VPC è una connessione di rete tra due VPC che consentono di instradare il traffico tra loro utilizzando degli indirizzi IP privati. Le risorse in uno qualsiasi dei VPC possono comunicare tra loro come se fossero nella stessa rete. Puoi anche creare una connessione di peering VPC tra VPC, con un VPC in un altro account AWS o con un VPC in una Regione AWS diversa. Per ulteriori informazioni su VPC in peering, consulta Peering di VPC nella Guida per l'utente di Amazon Virtual Private Cloud.

Un cluster database in un VPC a cui accede un'applicazione client tramite Internet

Per accedere a cluster database in un VPC da un'applicazione client tramite Internet, configura un VPC con una sottorete pubblica singola e un gateway Internet per abilitare la comunicazione in Internet.

Il seguente diagramma mostra questo scenario.



È consigliabile utilizzare la seguente configurazione:

- Un VPC di dimensione /16 (ad esempio, CIDR: 10.0.0.0/16). Questa dimensione fornisce indirizzi 65.536 indirizzi IP privati.
- Una sottorete di dimensione /24 (ad esempio, CIDR: 10.0.0.0/24). Questa dimensione fornisce 256 indirizzi IP privati.
- Un cluster di database Amazon Aurora che è in associazione al VPC e alla sottorete. Amazon RDS assegna un indirizzo IP nella sottorete al cluster database.
- Un gateway Internet che collega il VPC a Internet e agli altri prodotti AWS.
- Un gruppo di sicurezza associato al cluster database. Le regole in entrata del gruppo di sicurezza consentono all'applicazione client di accedere al cluster database.

Per informazioni su come creare un cluster database in un VPC, consulta Creazione di un cluster database in un VPC.

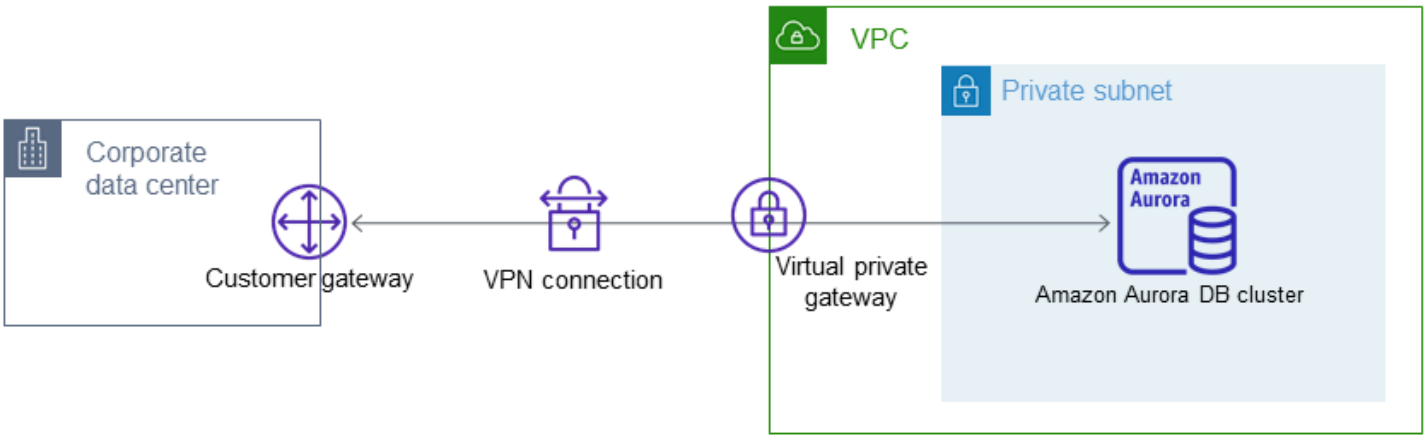
Un cluster database in un VPC a cui si accede da una rete privata

Se il cluster database non è accessibile pubblicamente, sono disponibili le seguenti opzioni per consentire l'accesso da una rete privata:

- Una connessione Site-to-Site VPN AWS Per ulteriori informazioni, consulta Che cos'è AWS Site-to-Site VPN?
- Una connessione Direct Connect. Per ulteriori informazioni, consulta Che cos'è Direct Connect?

- Una connessione AWS Client VPN. Per ulteriori informazioni, consulta Che cos'è AWS Client VPN?

Il diagramma seguente mostra uno scenario con una connessione Site-to-Site VPN AWS.

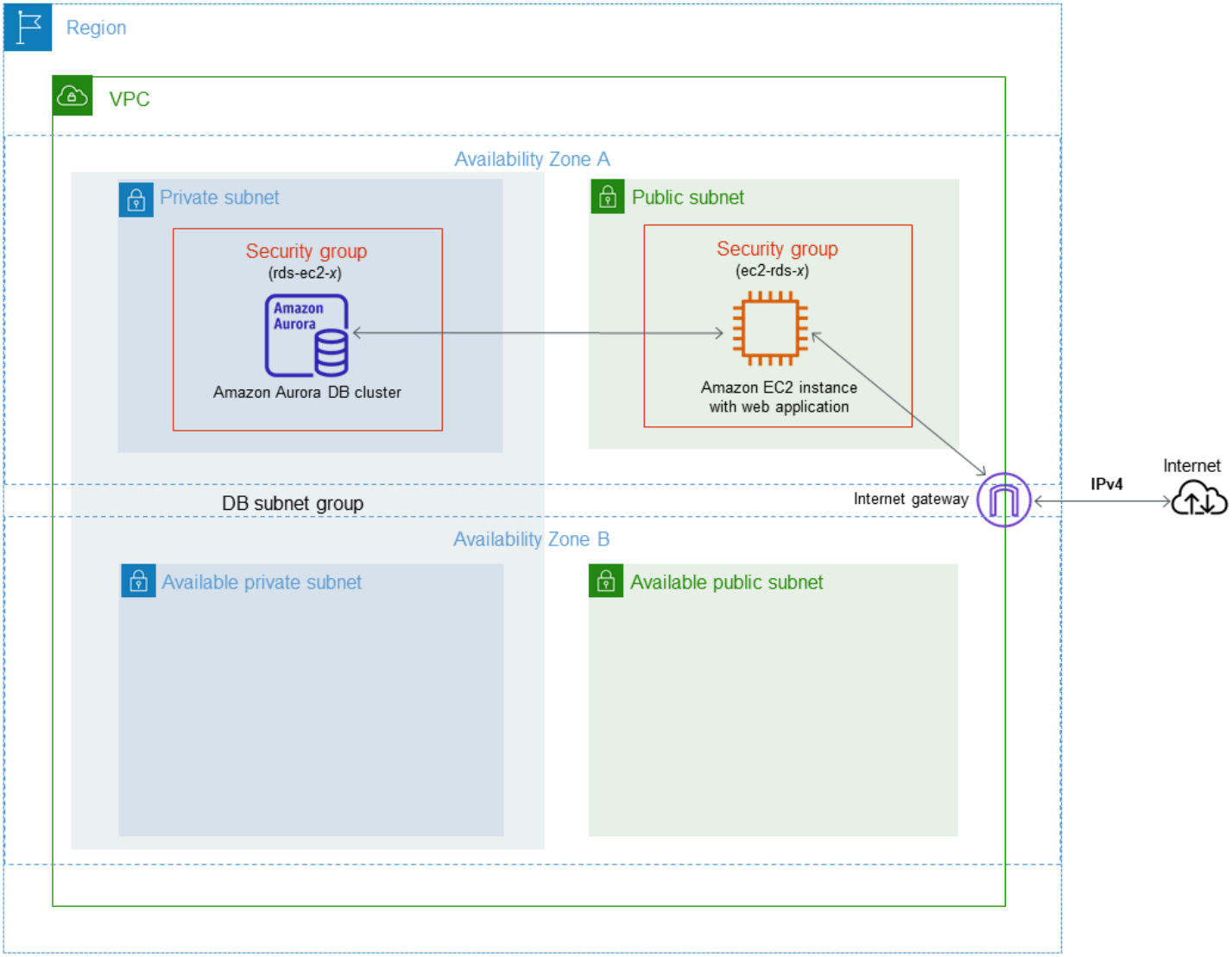


Per ulteriori informazioni, consulta Riservatezza del traffico Internet.

Tutorial: Creazione di un Amazon VPC da utilizzare con un cluster database (solo IPv4)

Uno scenario comune include un cluster database in un cloud privato virtuale (VPC) basato sul servizio Amazon VPC. Questo VPC condivide i dati con un server Web in esecuzione nello stesso VPC. In questo tutorial eseguirai la creazione del VPC in questo scenario.

Il seguente diagramma mostra questo scenario. Per informazioni su altri scenari, consulta Scenari per accedere a un cluster database in un VPC.



Il cluster database deve essere disponibile solo per il server Web e non per la rete Internet pubblica. Pertanto, occorre creare un VPC con sottoreti pubbliche e private. Il server Web è ospitato nella sottorete pubblica, in modo da poter raggiungere l'Internet pubblico. Il cluster database è ospitato in

una sottorete privata. Il server Web può connettersi al cluster database perché è ospitato nello stesso VPC. Tuttavia, il cluster database non è disponibile per la rete Internet pubblica e ciò garantisce una maggiore sicurezza.

Questo tutorial configura una sottorete pubblica e privata aggiuntiva in una zona di disponibilità separata. Queste sottoreti non vengono utilizzate dal tutorial. Un gruppo di sottoreti DB RDS richiede una sottorete in almeno due zone di disponibilità. La sottorete aggiuntiva semplifica la configurazione di più istanze database Aurora.

In questa esercitazione viene descritta la configurazione di un VPC per cluster di database Amazon Aurora. Per un'esercitazione che illustra come creare un server Web per questo scenario VPC, consulta Tutorial: creazione di un server Web e un cluster database Amazon Aurora. Per ulteriori informazioni su Amazon VPC, consulta Guida alle operazioni di base di Amazon VPC e Guida per l'utente di Amazon VPC.

> **Tip**
>
> Quando crei un cluster database, puoi configurare automaticamente la connettività di rete tra un'istanza Amazon EC2 e un cluster database. La configurazione di rete è simile a quella descritta in questo tutorial. Per ulteriori informazioni, consulta Configurazione della connettività di rete automatica con un'istanza EC2.

Creazione di un VPC con sottoreti pubbliche e private

Utilizza la procedura seguente per creare un VPC con sottoreti pubbliche e private.

Per creare un VPC e sottoreti

1. Accedi alla console Amazon VPC all'indirizzo https://console.aws.amazon.com/vpc/.
2. Nell'angolo superiore destro della Console di gestione AWS scegliere la regione in cui creare il VPC. In questo esempio viene utilizzata la regione Stati Uniti occidentali (Oregon).
3. Nell'angolo in alto a sinistra, scegli VPC Dashboard (Pannello di controllo VPC). Per iniziare a creare un VPC, scegli Create VPC (Crea VPC).
4. In Resources to create (Risorse da creare) nell'area VPC settings (Impostazioni VPC), scegli VPC and more (VPC e altro).
5. In VPC settings (Impostazioni VPC) restanti, imposta i seguenti valori:

- Name tag auto-generation (Generazione automatica del tag del nome): **`tutorial`**
- IPv4 CIDR block (Blocco CIDR IPv4): **`10.0.0.0/16`**
- IPv6 CIDR block (Blocco IPv6 CIDR): No IPv6 CIDR Block (Nessun blocco IPv6 CIDR)
- Tenancy (Locazione): Default
- Number of Availability Zones (AZs) (Numero di zone di disponibilità): 2
- Customize AZs (Personalizza le zone di disponibilità): mantieni i valori predefiniti.
- Number of public subnet (Numero di sottoreti pubbliche): 2
- Number of private subnets (Numero di sottoreti private): 2
- Customize subnets CIDR blocks (Personalizza blocchi CIDR delle sottoreti): mantieni i valori predefiniti.
- NAT gateways ($) (Gateway NAT): nessuna
- VPC endpoints (Endpoint VPC): nessuno
- DNS options (Opzioni DNS): mantieni i valori predefiniti.

6. Seleziona Create VPC (Crea VPC).

Creazione di un gruppo di sicurezza VPC per un server Web pubblico

È possibile a questo punto aggiungere un gruppo di sicurezza per l'accesso pubblico. Per connetterti alle istanze EC2 nel VPC, aggiungi regole in entrata al gruppo di sicurezza VPC. Queste consentono al traffico di connettersi da Internet.

Per creare un gruppo di sicurezza VPC

1. Aprire la console Amazon VPC all'indirizzo https://console.aws.amazon.com/vpc/.
2. Scegliere VPC Dashboard (Pannello di controllo VPC), Security Groups (Gruppi di sicurezza) e quindi Create security group (Crea gruppo di sicurezza).
3. Nella pagina Create security group (Crea gruppo di sicurezza) impostare questi valori:
   - Security group name: (Nome del gruppo di sicurezza: **`tutorial-securitygroup`**
   - Descrizione: **`Tutorial Security Group`**
   - VPC: scegliere il VPC creato in precedenza, ad esempio vpc-***`identifier`*** (tutorial-vpc)
4. Aggiungere regole in entrata al gruppo di sicurezza

a. Determina l'indirizzo IP da utilizzare per connettersi alle istanze EC2 nel VPC utilizzando Secure Shell (SSH). Per determinare l'indirizzo IP pubblico, in una finestra o una scheda del browser diversa, è possibile utilizzare il servizio all'indirizzo https://checkip.amazonaws.com. Un esempio di indirizzo IP è `203.0.113.25/32`.

   In molti casi, è possibile eseguire la connessione tramite un fornitore di servizi Internet (ISP) o con la protezione di un firewall senza un indirizzo IP statico. In tal caso, trova l'intervallo di indirizzi IP utilizzati dai computer client.

   > ⚠ Warning
   >
   > Se utilizzi `0.0.0.0/0` per l'accesso SSH, consenti a tutti gli indirizzi IP di accedere alle istanze pubbliche utilizzando SSH. Questo approccio è accettabile per un breve periodo di tempo in un ambiente di test, ma non è sicuro per gli ambienti di produzione. In produzione, autorizza solo un determinato indirizzo IP o un intervallo di indirizzi per accedere alle istanze utilizzando SSH.

b. Nella sezione Regole in entrata, scegliere Aggiungi regola.

c. Imposta i seguenti valori per la nuova regola in entrata per consentire l'accesso SSH all'istanza Amazon EC2. In tal caso, è possibile eseguire la connessione all'istanza Amazon EC2 per installare il server Web e altre utility. Puoi connetterti all'istanza EC2 anche per caricare contenuto per il server Web.

   - Tipo: **`SSH`**
   - Origine: indirizzo IP o intervallo ottenuto nella fase 1, ad esempio **`203.0.113.25/32`**.

d. Scegli Aggiungi regola.

e. Imposta i seguenti valori per la nuova regola in entrata per consentire l'accesso HTTP al server Web:

   - Tipo: **`HTTP`**
   - Origine: **`0.0.0.0/0`**

5. Per creare il gruppo di sicurezza, scegli Create security group (Crea gruppo di sicurezza).

   Prendi nota dell'ID del gruppo di sicurezza perché sarà necessario in seguito in questo tutorial.

Creazione di un gruppo di sicurezza VPC per un cluster database privato

Per mantenere il cluster database privato, crea un secondo gruppo di sicurezza per l'accesso privato. Per connetterti ai cluster di database privati nel VPC, aggiungi regole in entrata al gruppo di sicurezza VPC per consentire il traffico solo dal server web.

Per creare un gruppo di sicurezza VPC

1. Aprire la console Amazon VPC all'indirizzo https://console.aws.amazon.com/vpc/.
2. Scegliere VPC Dashboard (Pannello di controllo VPC), Security Groups (Gruppi di sicurezza) e quindi Create security group (Crea gruppo di sicurezza).
3. Nella pagina Create security group (Crea gruppo di sicurezza) impostare questi valori:
   - Security group name: (Nome del gruppo di sicurezza: **`tutorial-db-securitygroup`**
   - Descrizione: **`Tutorial DB Instance Security Group`**
   - VPC: scegliere il VPC creato in precedenza, ad esempio vpc-***`identifier`*** (tutorial-vpc)
4. Aggiungere regole in entrata al gruppo di sicurezza
   a. Nella sezione Regole in entrata, scegliere Aggiungi regola.
   b. Imposta i valori seguenti per la nuova regola in entrata per consentire il traffico MySQL sulla porta 3306 dall'istanza Amazon EC2. In tal caso, puoi connetterti dal server Web al cluster database ed eseguire l'archiviazione e il recupero dei dati dall'applicazione Web nel database.
      - Tipo: **`MySQL/Aurora`**
      - Source (Origine): l'identificatore del gruppo di sicurezza tutorial-securitygroup creato in precedenza in questo tutorial, ad esempio sg-9edd5cfb.
5. Per creare il gruppo di sicurezza, scegli Create security group (Crea gruppo di sicurezza).

Per creare un gruppo di sottoreti del database

Un gruppo di sottoreti DB è una raccolta di sottoreti creata in un VPC e che è possibile indicare per i cluster database. Un gruppo di sottoreti DB consente di specificare un determinato VPC quando si creano cluster database.

Creare un gruppo di sottoreti database

1. Identifica le sottoreti private per il database nel VPC.

   a. Accedi alla console Amazon VPC all'indirizzo https://console.aws.amazon.com/vpc/.

   b. Scegli VPC Dashboard (Pannello di controllo VPC), quindi seleziona Subnets (Sottoreti).

   c. Prendi nota degli ID sottorete delle sottoreti denominati tutorial-subnet-private1-us-west-2a e tutorial-subnet-private2-us-west-2b.

      Gli ID sottorete sono necessari quando si crea il gruppo di sottoreti DB.

2. Apri la console di Amazon RDS all'indirizzo https://console.aws.amazon.com/rds/.

   Assicurarsi di connettersi alla console Amazon RDS e non alla console Amazon VPC.

3. Nel pannello di navigazione selezionare Subnet groups (Gruppi di sottoreti).

4. Scegli Create DB Subnet Group (Crea gruppo di sottoreti del database).

5. Nella pagina Create DB subnet group (Crea gruppo di sottoreti del database) impostare questi valori in Subnet group details (Dettagli gruppi di sottoreti):

   - Nome: **`tutorial-db-subnet-group`**
   - Descrizione: **`Tutorial DB Subnet Group`**
   - VPC: tutorial-vpc (vpc-***`identifier`***)

6. Nella sezione Aggiungi sottoreti, scegliere Zone di disponibilità e Sottoreti.

   Per questo tutorial, scegli us-west-2a e us-west-2b per l'opzione Availability Zones (Zone di disponibilità). In Subnets (Sottoreti), scegli le sottoreti private identificate nella fase precedente.

7. Scegli Create (Crea).

   Il nuovo gruppo di sottoreti database viene visualizzato nell'elenco dei gruppi di sottoreti database sulla console RDS. Puoi scegliere il gruppo di sottoreti DB per visualizzare i dettagli nel riquadro dei dettagli nella parte inferiore della finestra. Questi dettagli includono tutte le sottoreti associate al gruppo.

> **Note**
>
> Se questo VPC è stato creato per il completamento di Tutorial: creazione di un server Web e un cluster database Amazon Aurora, creare il cluster di database seguendo le istruzioni riportate in Creazione di un cluster di database Amazon Aurora.

Eliminazione del VPC

Dopo aver creato il VPC e altre risorse per questo tutorial, è possibile eliminarle se non sono più necessarie.

> **Note**
>
> Se hai aggiunto risorse nel VPC creato per questo tutorial, potrebbe essere necessario eliminarle prima di poter eliminare il VPC. Ad esempio, queste risorse potrebbero includere istanze Amazon EC2 o cluster database Amazon RDS. Per ulteriori informazioni, consulta Eliminazione del VPC nella Guida per l'utente di Amazon VPC.

Per eliminare un VPC e le risorse correlate

1. Eliminare il gruppo di sottoreti di database.
    a. Apri la console Amazon RDS all'indirizzo https://console.aws.amazon.com/rds/.
    b. Nel pannello di navigazione selezionare Subnet groups (Gruppi di sottoreti).
    c. Selezionare il gruppo di sottoreti di database che si desidera eliminare, ad esempio tutorial-db-subnet-group.
    d. Scegliere Delete (Elimina) e quindi scegliere Delete (Elimina) nella finestra di conferma.
2. Nota l'ID VPC.
    a. Accedi alla console Amazon VPC all'indirizzo https://console.aws.amazon.com/vpc/.
    b. ScegliereVPC Dashboard (Pannello di controllo VPC) e quindi scegliere VPC.
    c. Nell'elenco, identifica il VPC creato, ad esempio tutorial-vpc.
    d. Prendi nota del valore ID VPC del VPC creato. L'ID VPC è richiesto nei passaggi successivi.
3. Eliminare i gruppi di sicurezza.

a. Accedi alla console Amazon VPC all'indirizzo https://console.aws.amazon.com/vpc/.
b. Scegliere VPC Dashboard (Pannello di controllo VPC) e quindi scegliere Security Groups (Gruppi di sicurezza).
c. Seleziona il gruppo di sicurezza per l'istanza database di Amazon RDS, ad esempio tutorial-db-securitygroup.
d. In Actions (Operazioni), scegli Delete security groups (Elimina gruppi di sicurezza) e quindi scegli Delete (Elimina) nella pagina di conferma.
e. Sulla pagina Security Groups (Gruppi di sicurezza), selezionare il gruppo di sicurezza per l'istanza Amazon EC2, ad esempio tutorial-securitygroup.
f. In Actions (Operazioni), scegli Delete security groups (Elimina gruppi di sicurezza) e quindi scegli Delete (Elimina) nella pagina di conferma.

4. Eliminare il VPC.

a. Accedi alla console Amazon VPC all'indirizzo https://console.aws.amazon.com/vpc/.
b. ScegliereVPC Dashboard (Pannello di controllo VPC) e quindi scegliere VPC.
c. Selezionare il VPC che si desidera eliminare, ad esempio tutorial-vpc.
d. In Operazioni, scegli Elimina VPC.

   Nella pagina di conferma vengono visualizzate altre risorse associate al VPC che verranno eliminate, incluse le subnet associate.
e. Nella pagina di conferma, immetti **`delete`** e scegliere Delete (Elimina).

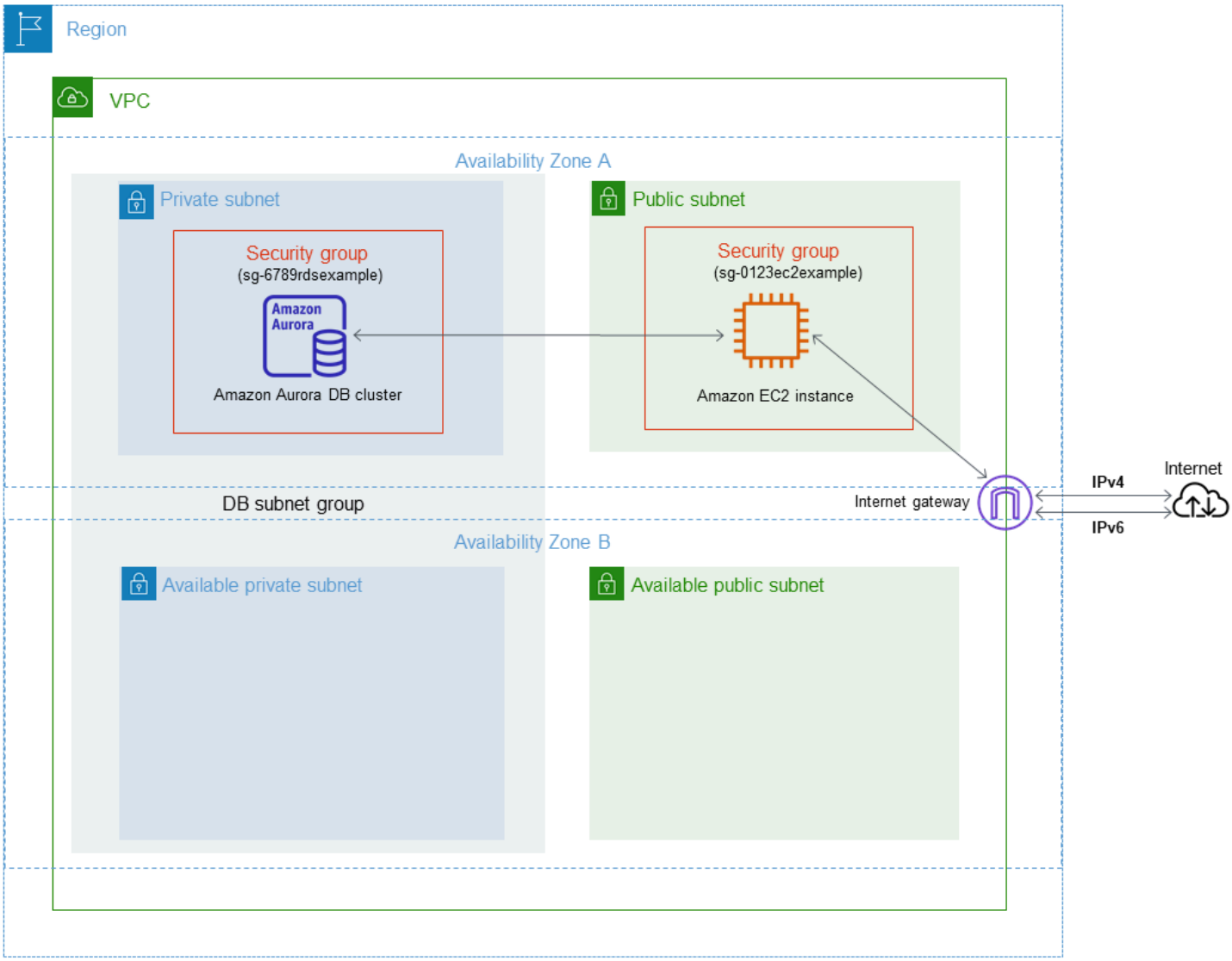
Tutorial: Creazione di un VPC per l'utilizzo con un cluster database (modalità dual-stack)

Uno scenario comune include un cluster database in un cloud privato virtuale (VPC) basato sul servizio Amazon VPC. Questo VPC condivide i dati con un'istanza Amazon EC2 pubblica in esecuzione nello stesso VPC.

In questo tutorial, si crea il VPC per questo scenario che funziona con un database in esecuzione in modalità dual-stack. Modalità dual-stack per abilitare la connessione tramite il protocollo di indirizzamento IPv6. Per ulteriori informazioni sull'indirizzamento IP, consulta Assegnazione di indirizzi IP in Amazon Aurora.

I cluster di rete dual-stack sono supportati nella maggior parte delle regioni. Per ulteriori informazioni, consulta Disponibilità di cluster database di rete dual-stack. Per esaminare le limitazioni della modalità dual-stack, consulta Limitazioni per cluster database di rete dual-stack.

Il seguente diagramma mostra questo scenario.



Per informazioni su altri scenari, consulta Scenari per accedere a un cluster database in un VPC.

Il cluster database deve essere disponibile solo per l'istanza Amazon EC2 e non per la rete Internet pubblica. Pertanto, occorre creare un VPC con sottoreti pubbliche e private. Il server Web è ospitato nella sottorete pubblica, in modo da poter raggiungere l'Internet pubblico. Il cluster database è ospitato in una sottorete privata. L'istanza Amazon EC2 può connettersi al cluster database perché è ospitata nello stesso VPC. Tuttavia, il cluster database non è disponibile per la rete Internet pubblica, fornendo una maggiore sicurezza.

Questo tutorial configura una sottorete pubblica e privata aggiuntiva in una zona di disponibilità separata. Queste sottoreti non vengono utilizzate dal tutorial. Un gruppo di sottoreti DB RDS richiede una sottorete in almeno due zone di disponibilità. La sottorete aggiuntiva semplifica la configurazione di più istanze database Aurora.

Per creare un cluster database che utilizza la modalità dual-stack, specifica Dual-stack mode (Modalità dual-stack) per l'opzione Network type (Tipo di rete). Puoi, inoltre, modificare un cluster database usando la stessa impostazione. Ulteriori informazioni sulla creazione di un cluster di database sono disponibili in Creazione di un cluster database Amazon Aurora. Per ulteriori informazioni sulla modifica di un cluster di database, consultare Modifica di un cluster database Amazon Aurora.

In questa esercitazione viene descritta la configurazione di un VPC per cluster di database Amazon Aurora. Per ulteriori informazioni su Amazon VPC, consulta la Guida per l'utente di Amazon VPC.

Creazione di un VPC con sottoreti pubbliche e private

Utilizza la procedura seguente per creare un VPC con sottoreti pubbliche e private.

Per creare un VPC e sottoreti

1. Accedi alla console Amazon VPC all'indirizzo https://console.aws.amazon.com/vpc/.
2. Nell'angolo in alto a destra della Console di gestione AWS scegliere la regione in cui creare il VPC. Questo esempio utilizza la regione Stati Uniti orientali (Ohio).
3. Nell'angolo in alto a sinistra, scegli VPC Dashboard (Pannello di controllo VPC). Per iniziare a creare un VPC, scegli Create VPC (Crea VPC).
4. In Resources to create (Risorse da creare) nell'area VPC settings (Impostazioni VPC), scegli VPC and more (VPC e altro).
5. Nei campi VPC settings (Impostazioni VPC) restanti, imposta i seguenti valori:
   - Name tag auto-generation (Generazione automatica del tag del nome): **`tutorial-dual-stack`**
   - IPv4 CIDR block (Blocco CIDR IPv4): **`10.0.0.0/16`**
   - IPv6 CIDR block (Blocco CIDR IPv6): blocco CIDR IPv6 fornito da Amazon
   - Tenancy (Locazione): Default
   - Number of Availability Zones (AZs) (Numero di zone di disponibilità): 2
   - Customize AZs (Personalizza le zone di disponibilità): mantieni i valori predefiniti.
   - Number of public subnet (Numero di sottoreti pubbliche): 2
   - Number of private subnets (Numero di sottoreti private): 2
   - Customize subnets CIDR blocks (Personalizza blocchi CIDR delle sottoreti): mantieni i valori predefiniti.

- NAT gateways ($) (Gateway NAT): nessuna
- Egress only internet gateway (Gateway Internet solo in uscita): No
- VPC endpoints (Endpoint VPC): nessuno
- DNS options (Opzioni DNS): mantieni i valori predefiniti.

> **Note**
>
> Amazon RDS richiede almeno due sottoreti in due zone di disponibilità diverse per supportare le implementazioni Multi-AZ di un'istanza DB. In questo tutorial viene creata crea un'implementazione Single-AZ, ma il requisito semplifica la conversione in un'implementazione Multi-AZ di un'istanza DB in futuro.

6. Seleziona Crea VPC.

Per creare un gruppo di sicurezza VPC per un'istanza Amazon EC2 pubblica

È possibile a questo punto aggiungere un gruppo di sicurezza per l'accesso pubblico. Per connettersi alle istanze EC2 nel VPC, aggiungi regole in entrata al gruppo di sicurezza VPC per consentire il traffico da Internet.

Per creare un gruppo di sicurezza VPC

1. Aprire la console Amazon VPC all'indirizzo https://console.aws.amazon.com/vpc/.
2. Scegliere VPC Dashboard (Pannello di controllo VPC), Security Groups (Gruppi di sicurezza) e quindi Create security group (Crea gruppo di sicurezza).
3. Nella pagina Create security group (Crea gruppo di sicurezza) impostare questi valori:
   - Security group name: (Nome del gruppo di sicurezza: **`tutorial-dual-stack-securitygroup`**
   - Descrizione: **`Tutorial Dual-Stack Security Group`**
   - VPC: scegli il VPC creato in precedenza, ad esempio vpc-***`identifier`*** (tutorial-dual-stack-vpc)
4. Aggiungere regole in entrata al gruppo di sicurezza
   a. Determina l'indirizzo IP da utilizzare per connettersi alle istanze EC2 nel VPC utilizzando Secure Shell (SSH).

Un esempio di indirizzo Internet Protocol versione 4 (IPv4) è `203.0.113.25/32`.
Un esempio di intervalli di indirizzi Internet Protocol versione 6 (IPv6) è `2001:db8:1234:1a00::/64`.

In molti casi, è possibile eseguire la connessione tramite un fornitore di servizi Internet (ISP) o con la protezione di un firewall senza un indirizzo IP statico. In tal caso, trova l'intervallo di indirizzi IP utilizzati dai computer client.

> **Warning**
>
> Se utilizzi `0.0.0.0/0` per IPv4 o `::0` per IPv6, consenti a tutti gli indirizzi IP di accedere alle istanze pubbliche utilizzando SSH. Questo approccio è accettabile per un breve periodo di tempo in un ambiente di test, ma non è sicuro per gli ambienti di produzione. In produzione, è preferibile autorizzare l'accesso alle istanze solo a indirizzo IP o a un intervallo di indirizzi specifico.

b. Nella sezione Regole in entrata, scegliere Aggiungi regola.

c. Imposta i seguenti valori per la nuova regola in entrata per consentire l'accesso Secure Shell (SSH) all'istanza Amazon EC2. In questo caso, è possibile connettersi all'istanza EC2 per installare client SQL e altre applicazioni. Specifica un indirizzo IP per poter accedere all'istanza EC2:

- Tipo: **`SSH`**
- Origine: indirizzo IP o intervallo ottenuto nel passaggio a. Un esempio di indirizzo IP IPv4 è **`203.0.113.25/32`**. Un esempio di indirizzo IP IPv6 è **`2001:DB8::/32`**.

5. Per creare il gruppo di sicurezza, scegli Create security group (Crea gruppo di sicurezza).

   Prendi nota dell'ID del gruppo di sicurezza perché sarà necessario in seguito in questo tutorial.

Creazione di un gruppo di sicurezza VPC per un cluster database privato

Per mantenere il cluster database privato, crea un secondo gruppo di sicurezza per l'accesso privato. Per connetterti ai cluster database privati nel VPC, aggiungi le regole in entrata al gruppo di sicurezza VPC. Queste consentono il traffico solo dall'istanza Amazon EC2.

Per creare un gruppo di sicurezza VPC

1. Aprire la console Amazon VPC all'indirizzo https://console.aws.amazon.com/vpc/.
2. Scegliere VPC Dashboard (Pannello di controllo VPC), Security Groups (Gruppi di sicurezza) e quindi Create security group (Crea gruppo di sicurezza).
3. Nella pagina Create security group (Crea gruppo di sicurezza) impostare questi valori:
   - Security group name: (Nome del gruppo di sicurezza: **`tutorial-dual-stack-db-securitygroup`**
   - Descrizione: **`Tutorial Dual-Stack DB Instance Security Group`**
   - VPC: scegli il VPC creato in precedenza, ad esempio vpc-***`identifier`*** (tutorial-dual-stack-vpc)
4. Aggiungere regole in entrata al gruppo di sicurezza
   a. Nella sezione Regole in entrata, scegliere Aggiungi regola.
   b. Imposta i valori seguenti per la nuova regola in entrata per consentire il traffico MySQL sulla porta 3306 dall'istanza Amazon EC2. In tal modo, puoi eseguire la connessione dall'istanza EC2 al cluster database ed Questa operazione consente di inviare dati dall'istanza EC2 al database.
      - Tipo: MySQL/Aurora
      - Source (Origine): l'identificatore del gruppo di sicurezza tutorial-dual-stack-securitygroup creato in precedenza in questo tutorial, ad esempio sg-9edd5cfb.
5. Per creare il gruppo di sicurezza, scegliere Crea gruppo di sicurezza.

Per creare un gruppo di sottoreti del database

Un gruppo di sottoreti DB è una raccolta di sottoreti creata in un VPC e che è possibile indicare per i cluster database. L'utilizzo di un gruppo di sottoreti DB consente di specificare un determinato VPC durante la creazione di cluster database. Per creare un gruppo di sottoreti database compatibile con `DUAL`, tutte le sottoreti devono essere compatibili con `DUAL`. Per essere compatibile con `DUAL`, a una sottorete deve essere associato un CIDR IPv6.

Creare un gruppo di sottoreti database

1. Identifica le sottoreti private per il database nel VPC.

a. Accedi alla console Amazon VPC all'indirizzo https://console.aws.amazon.com/vpc/.
b. Scegli VPC Dashboard (Pannello di controllo VPC), quindi seleziona Subnets (Sottoreti).
c. Prendi nota degli ID sottorete delle sottoreti denominati tutorial-dual-stack-subnet-private1-us-west-2a e tutorial-dual-stack-subnet-private2-us-west-2b.

   Gli ID sottorete saranno necessari quando si crea il gruppo di sottoreti DB.

2. Apri la console di Amazon RDS all'indirizzo https://console.aws.amazon.com/rds/.

   Assicurarsi di connettersi alla console Amazon RDS e non alla console Amazon VPC.
3. Nel pannello di navigazione selezionare Subnet groups (Gruppi di sottoreti).
4. Scegli Create DB Subnet Group (Crea gruppo di sottoreti del database).
5. Nella pagina Create DB subnet group (Crea gruppo di sottoreti del database) impostare questi valori in Subnet group details (Dettagli gruppi di sottoreti):

   - Nome: **`tutorial-dual-stack-db-subnet-group`**
   - Descrizione: **`Tutorial Dual-Stack DB Subnet Group`**
   - VPC: tutorial-dual-stack-vpc (vpc-***`identifier`***)
6. Nella sezione Add subnets (Aggiungi sottoreti), scegli i valori desiderati per le opzioni Availability Zones (Zone di disponibilità) e Subnets (Sottoreti).

   Per questo tutorial, scegli us-east-2b e us-east-2c per l'opzione Availability Zones (Zone di disponibilità). In Subnets (Sottoreti), scegli le sottoreti private identificate nella fase precedente.
7. Scegli Create (Crea).

Il nuovo gruppo di sottoreti database viene visualizzato nell'elenco dei gruppi di sottoreti database sulla console RDS. È possibile scegliere il gruppo di sottoreti database per visualizzarne i dettagli. I dettagli includono i protocolli di indirizzamento supportati e tutte le sottoreti associate al gruppo e il tipo di rete supportato dal gruppo di sottoreti database.

Creare un'istanza Amazon EC2 in modalità dual-stack

Per creare un'istanza Amazon EC2, segui le istruzioni riportate in Launch an instance using the new launch instance wizard nella Guida per l'utente di Amazon EC2.

Nella pagina Configure Instance Details (Configura i dettagli dell'istanza), imposta questi valori e lascia gli altri valori come predefiniti:

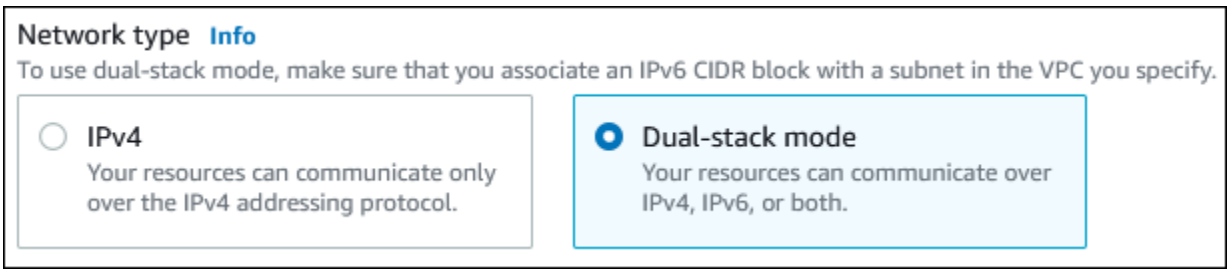
- Rete: scegli un VPC esistente con sottoreti sia pubbliche che private, ad esempio tutorial-dual-stack-vpc (vpc-*identifier*) creato in Creazione di un VPC con sottoreti pubbliche e private.
- Subnet (Sottorete): scegli una sottorete pubblica esistente, ad esempio subnet-***identifier*** | tutorial-dual-stack-subnet-public1-us-east-2a | us-east-2a creata in Per creare un gruppo di sicurezza VPC per un'istanza Amazon EC2 pubblica.
- Auto-assign Public IP (Assegna automaticamente IP pubblico): scegli Enable (Abilita).
- Auto-assign IPv6 IP (Assegna automaticamente IP IPv6): scegli Enable (Abilita).
- Firewall (security groups) (Firewall (gruppi di sicurezza)): scegli Select an existing security group (Seleziona un gruppo di sicurezza esistente).
- Gruppi di sicurezza comuni: scegli un gruppo di sicurezza esistente, ad esempio il `tutorial-securitygroup` creato in Per creare un gruppo di sicurezza VPC per un'istanza Amazon EC2 pubblica. Assicurarsi che il gruppo di sicurezza scelto includa regole in ingresso per Secure Shell (SSH) e l'accesso HTTP.

Creazione di un cluster database in modalità dual-stack

In questo passaggio, viene creata un cluster database che viene eseguito in modalità dual-stack.

Per creare un'istanza database.

1. Accedi alla Console di gestione AWS e apri la console Amazon RDS all'indirizzo https://console.aws.amazon.com/rds/.
2. Nell'angolo in alto a destra della console, scegli la Regione AWS in cui desideri creare il cluster database. Questo esempio utilizza la regione Stati Uniti orientali (Ohio).
3. Nel riquadro di navigazione, scegliere Databases (Database).
4. Scegliere Create database (Crea database).
5. Nella pagina Create database (Crea database), assicurati che l'opzione Standard create (Creazione standard) sia selezionata, quindi scegli il tipo di motore DB Aurora MySQL.
6. Nella sezione Connettività, imposta i seguenti valori:
   - Network type (Tipo di rete): scegli Dual-stack mode (Modalità dual-stack).



- Virtual private cloud (VPC) Cloud privato virtuale (VPC): scegli un VPC esistente con sottoreti sia pubbliche che private, ad esempio tutorial-dual-stack-vpc (vpc-*identifier*) creato in Creazione di un VPC con sottoreti pubbliche e private.

  Il VPC deve avere sottoreti in diverse zone di disponibilità.
- DB subnet group (Gruppo di sottoreti DB): scegli un gruppo di sottoreti DB per il VPC, ad esempio tutorial-dual-stack-db-subnet-group creato in Per creare un gruppo di sottoreti del database.
- Public access (Accesso pubblico): scegli No.
- VPC security group (firewall) (Gruppo di sicurezza VPC (firewall)): seleziona Choose existing (Sceglie esistente).
- Gruppi di sicurezza VPC esistenti: scegli un gruppo di sicurezza VPC esistente che sia configurato per accesso privato, ad esempio tutorial-dual-stack-db-securitygroup creato in Creazione di un gruppo di sicurezza VPC per un cluster database privato.

  Rimuovere gli altri gruppi di sicurezza, come quello predefinito, selezionando la X associata a esso.
- Availability Zone (Zona di disponibilità): scegli us-west-2a.

  Per evitare il traffico tra AZ, assicurati che l'istanza database e l'istanza EC2 si trovino nella stessa zona di disponibilità.

7. Per le restanti sezioni, specifica le impostazioni del cluster di database. Per informazioni su ciascuna impostazione, consulta Impostazioni per cluster di database Aurora.

Connessione all'istanza Amazon EC2 e al cluster database

Dopo aver creato l'istanza Amazon EC2 e il cluster database in modalità dual-stack, puoi eseguire la connessione utilizzando il protocollo IPv6. Per connetterti a un'istanza Amazon EC2 utilizzando il protocollo IPv6, segui le istruzioni riportate in Connect to your Linux instance nella Guida per l'utente di Amazon EC2.

Per connetterti al cluster database Aurora MySQL dall'istanza Amazon EC2, segui le istruzioni in Connessione a un cluster di database Aurora MySQL.

Eliminazione del VPC

Dopo aver creato il VPC e altre risorse per questo tutorial, è possibile eliminarle se non sono più necessarie.

Se hai aggiunto risorse nel VPC creato per questo tutorial, potrebbe essere necessario eliminarle prima di poter eliminare il VPC. Esempi di risorse sono le istanze Amazon EC2 o i cluster database. Per ulteriori informazioni, consulta Eliminazione del VPC nella Guida per l'utente di Amazon VPC.

Per eliminare un VPC e le risorse correlate

1. Eliminare il gruppo di sottoreti database:
    a. Apri la console Amazon RDS all'indirizzo https://console.aws.amazon.com/rds/.
    b. Nel pannello di navigazione selezionare Subnet groups (Gruppi di sottoreti).
    c. Seleziona il gruppo di sottoreti database da eliminare, ad esempio tutorial-db-subnet-group.
    d. Scegliere Delete (Elimina) e quindi scegliere Delete (Elimina) nella finestra di conferma.
2. Annotare l'ID VPC:
    a. Accedi alla console Amazon VPC all'indirizzo https://console.aws.amazon.com/vpc/.
    b. ScegliereVPC Dashboard (Pannello di controllo VPC) e quindi scegliere VPC.
    c. Nell'elenco, individua il VPC creato, ad esempio tutorial-dual-stack-vpc.
    d. Annotare il valore ID VPC del VPC creato. Sarà necessario usare questo ID VPC nei passaggi successivi.
3. Eliminare i gruppi di sicurezza:
    a. Accedi alla console Amazon VPC all'indirizzo https://console.aws.amazon.com/vpc/.
    b. Scegliere VPC Dashboard (Pannello di controllo VPC) e quindi scegliere Security Groups (Gruppi di sicurezza).
    c. Selezionare il gruppo di sicurezza per l'istanza database di Amazon RDS, ad esempio tutorial-dual-stack-db-securitygroup.
    d. In Actions (Operazioni), scegli Delete security groups (Elimina gruppi di sicurezza) e quindi scegli Delete (Elimina) nella pagina di conferma.

e. Nella pagina Security Groups (Gruppi di sicurezza), seleziona il gruppo di sicurezza per l'istanza Amazon EC2, ad esempio tutorial-dual-stack-securitygroup..

f. In Actions (Operazioni), scegli Delete security groups (Elimina gruppi di sicurezza) e quindi scegli Delete (Elimina) nella pagina di conferma.

4. Elliminare il gateway NAT:

a. Accedi alla console Amazon VPC all'indirizzo https://console.aws.amazon.com/vpc/.

b. Scegliere VPC Dashboard (Pannello di controllo VPC) e quindi scegliere NAT Gateways (Gateway NAT).

c. Seleziona il gateway NAT del VPC creato. Utilizzare l'ID VPC per identificare il gateway NAT corretto.

d. In Actions (Operazioni), scegli Delete NAT gateway (Elimina gateway NAT).

e. Nella pagina di conferma, immetti **`delete`** e scegliere Delete (Elimina).

5. Eliminare il VPC

a. Accedi alla console Amazon VPC all'indirizzo https://console.aws.amazon.com/vpc/.

b. ScegliereVPC Dashboard (Pannello di controllo VPC) e quindi scegliere VPC.

c. Seleziona il VPC da eliminare, ad esempio tutorial-dual-stack-vpc.

d. In Operazioni, scegli Elimina VPC.

Nella pagina di conferma vengono visualizzate altre risorse associate al VPC che verranno eliminate, incluse le subnet associate.

e. Nella pagina di conferma, immetti **`delete`** e scegliere Delete (Elimina).

6. Rilasciare gli indirizzi IP elastici:

a. Aprire la console Amazon EC2 all'indirizzo https://console.aws.amazon.com/ec2/.

b. Scegliere EC2 Dashboard (Pannello di controllo EC2) e quindi scegliere Elastic IPs (IP elastici).

c. Seleziona l'indirizzo IP elastico da rilasciare.

d. In Actions (Operazioni), scegli Release Elastic IP addresses (Rilascia indirizzi IP elastici).

e. Nella pagina di conferma scegli Release (Rilasciare).

Quote e vincoli per Amazon Aurora

Di seguito, è possibile trovare una descrizione delle quote di risorse e dei vincoli di denominazione per Amazon Aurora.

Argomenti

- Quote in Amazon Aurora
- Vincoli per la denominazione in Amazon Aurora
- Limiti di dimensione Amazon Aurora

Quote in Amazon Aurora

Ogni AWS account dispone di quote, per ogni AWS regione, sul numero di risorse Amazon Aurora che è possibile creare. Una volta raggiunta la quota della risorsa, le ulteriori richieste di creazione di tale risorsa restituiranno un errore con un'eccezione.

La tabella seguente elenca le risorse e le relative quote per regione. AWS

Nome	Predefinita	Adatta e	Description
Autorizzazioni per gruppo di sicurezza DB	Ogni regione supportata: 20	No	Numero di autorizzazioni per gruppo di sicurezza DB
Endpoint personalizzati per cluster DB	Ogni regione supportata: 5	Sì	Il numero massimo di endpoint personalizzati che è possibile creare per cluster di database Aurora in questo account nella Regione corrente. Questo valore riflette il numero più elevato di endpoint personali zzati in un cluster di database nell'account.

Nome	Predefinita	Adatta e	Description
			Altri cluster di database nell'account possono avere un numero inferiore di endpoint personalizzati.
Versioni personalizzate del motore	Ogni regione supportata: 40	Sì	Il numero massimo di versioni personalizzate del motore consentite in questo account nella regione corrente
Gruppi di parametri del cluster di database	Ogni Regione supportata: 50	Sì	Il numero massimo di gruppi di parametri del cluster di database
Cluster database	Ogni regione supportata: 40	Sì	Il numero massimo di cluster Aurora consentiti in questo account nella regione corrente
Istanze DB	ap-south-1: 20 Ogni altra Regione supportata: 40	Sì	Il numero massimo di istanze database consentite in questo account nella regione corrente
Gruppi di shard del database	Ogni regione supportata: 5	Sì	Il numero massimo di gruppi di shard del database per Aurora Limitless Database in questo account nella Regione corrente

Nome	Predefinita	Adatta e	Description
Gruppi di sottoreti database	ap-south-1: 20 Ogni altra regione supportata: 50	Sì	Il numero massimo di gruppi di sottoreti di database
Dimensione del corpo della richiesta HTTP dell'API dati	Ogni regione supportata: 4 MB	No	La dimensione massima consentita per il corpo della richiesta HTTP.
Numero massimo di coppie segrete del cluster simultanee dell'API dati	Ogni regione supportata: 30	No	Il numero massimo di coppie uniche di cluster e segreti DB Aurora Serverless v1 nelle richieste simultanee di Data API per questo account nella regione corrente. AWS
Numero massimo di richieste simultanee dell'API dati	Ogni regione supportata: 500	No	Il numero massimo di richieste API dati a un cluster di database Aurora Serverless v1 che utilizzano lo stesso segreto e possono essere elaborate contempor aneamente. Le richieste aggiuntive vengono messe in coda ed elaborate al termine delle richieste in corso.
Dimensione massima del set di risultati dell'API dati	Ogni regione supportata: 1 MB	No	La dimensione massima del set di risultati del database che può essere restituito dall'API dati.

Nome	Predefinita	Adatta e	Description
Dimensione massima dell'API dati della stringa di risposta JSON	Ogni regione supportata: 10 megabyte	No	La dimensione massima della stringa di risposta JSON semplificata restituita dall'API dati RDS.
Richieste API dati al secondo	Ogni regione supportata: 1.000 al secondo	No	Il numero massimo di richieste all'API Data al secondo consentito per questo account nella regione corrente. AWS Questa quota si applica solo ai cluster Amazon Aurora Serverless v1.
Abbonamenti a eventi	Ogni regione supportata: 20	Sì	Il numero massimo di sottoscrizioni di eventi
Ruoli IAM per cluster di database	Ogni regione supportata: 5	Sì	Il numero massimo di ruoli IAM associati a un cluster di database
Ruoli IAM per istanza database	Ogni regione supportata: 5	Sì	Il numero massimo di ruoli IAM associati a un'istanza database
Integrazioni	Ogni regione supportata: 100	No	Il numero massimo di integrazioni consentite in questo account nella regione corrente AWS
Snapshot del cluster di database manuale	Ogni regione supportata: 100	Sì	Il numero massimo di snapshot manuali del cluster di database

Nome	Predefinita	Adattabile	Description
Snapshot manuali dell'istanza database	Ogni regione supportata: 100	Sì	Il numero massimo di snapshot manuali di istanza database
Gruppi di opzioni	Ogni regione supportata: 20	Sì	Il numero massimo di gruppi di opzioni
Gruppi di parametri	ap-south-1: 20 Ogni altra regione supportata: 50	Sì	Il numero massimo di gruppi di parametri
Proxy	Ogni regione supportata: 20	Sì	Il numero massimo di proxy consentito in questo account nella regione corrente AWS
Lettura delle repliche per primario	Ogni regione supportata: 15	Sì	Il numero massimo di repliche di lettura per istanza database primario. Questa quota non può essere modificata per Amazon Aurora.
Istanze database riservate	ap-south-1: 20 Ogni altra Regione supportata: 40	Sì	Il numero massimo di istanze DB riservate consentite in questo account nella regione corrente AWS
Gruppi di sicurezza	ap-south-1: 20 Ogni altra regione supportata: 25	Sì	Il numero massimo di gruppi di sicurezza DB

Nome	Predefinita	Adatta e	Description
Sottoreti per gruppo di sottoreti del database	Ogni regione supportata: 20	No	Il numero massimo di sottoreti per gruppo di sottoreti di database
Totale storage per tutte le istanze database	Ogni regione supportata: 100.000	Sì	Lo spazio di archiviazione massimo totale (in GB) per tutte le istanze database di Amazon RDS aggiunte insieme. Questa quota non si applica ad Amazon Aurora, che ha un volume massimo di cluster di 128 TiB per ogni cluster di database.

Note

Per impostazione predefinita, puoi avere fino a 40 istanze database. Le istanze database RDS, le istanze database Aurora, le istanze Amazon Neptune e le istanze Amazon DocumentDB si applicano a questa quota.

Se l'applicazione richiede più istanze database, puoi richiedere altre istanze database aprendo la console Service Quotas. Nel pannello di navigazione, scegliere servizi AWS . Scegli Amazon Relational Database Service (Amazon RDS), scegli una quota e segui le istruzioni per richiedere un aumento della quota. Per ulteriori informazioni, consulta Richiesta di un aumento di quota nella Guida per l'utente per Service Quotas.

Per RDS per Oracle, è possibile creare fino a 15 repliche di lettura per istanza database di origine in ogni Regione, ma è consigliabile limitare le repliche a 5 per ridurre al minimo il ritardo di replica.

I backup gestiti da AWS Backup sono considerati istantanee manuali del cluster DB, ma non vengono conteggiati ai fini della quota di snapshot del cluster manuale. Per informazioni in merito AWS Backup, consulta la Guida per gli sviluppatori AWS Backup .

Se si utilizza una delle operazioni dell'API RDS e si supera la quota predefinita del numero di chiamate al secondo, l'API Amazon RDS emette un errore simile al seguente.

ClientError: Si è verificato un errore (ThrottlingException) durante la chiamata dell'*API_name*operazione: Frequenza superata.

Qui, riduci il numero di chiamate al secondo. La quota è destinata a coprire la maggior parte dei casi d'uso. Se necessario, è possibile richiedere un aumento di quota utilizzando una delle seguenti opzioni:

- Dalla console, aprire la console Service Quotas.
- Da AWS CLI, utilizzare il request-service-quota-increase AWS CLI comando.

Per maggiori informazioni, consulta Guida per l'utente di Service Quotas.

Vincoli per la denominazione in Amazon Aurora

I vincoli per la denominazione in Amazon Aurora sono:

- Identificatore cluster di database:
  - Devono contenere da 1 a 63 caratteri alfanumerici o trattini.
  - Il primo carattere deve essere una lettera.
  - Non può terminare con un trattino o contenere due trattini consecutivi.
  - Deve essere unico per tutte le istanze DB per AWS account, per AWS regione.
- Nome database iniziale: i vincoli del nome del database differiscono tra Aurora MySQL e Aurora PostgreSQL. Per ulteriori informazioni, consultare le impostazioni disponibili durante la creazione di ogni cluster di database.
- Nome utente master: i vincoli del nome utente master variano in base al motore del database. Per ulteriori informazioni, consultare le impostazioni disponibili durante la creazione del cluster di database.
- Password master:
  - La password dell'utente master del database può includere qualsiasi carattere ASCII stampabile, tranne `/`, `'`, `"`, `@` o uno spazio.
  - La password può contenere il seguente numero di caratteri ASCII stampabili, a seconda del motore del database:

    - Aurora MySQL: 8 - 41
    - Aurora PostgreSQL: 8 - 99
- Gruppo di parametri database:
    - Devono contenere da 1 a 255 caratteri alfanumerici.
    - Il primo carattere deve essere una lettera.
    - I trattini sono consentiti, ma il nome non può terminare con un trattino o contenere due trattini consecutivi.
- Gruppo di sottoreti di database:
    - Devono contenere da 1 a 255 caratteri.
    - Sono consentiti caratteri alfanumerici, spazi, trattini, trattini bassi e punti.

Limiti di dimensione Amazon Aurora

Limiti delle dimensioni dello storage

La dimensione massima del volume del cluster Aurora varia in base alla versione del motore:

256 TiB (massimo):

- Versioni di Aurora PostgreSQL:
    - 17.5 e tutte le versioni successive
    - 16.9 e versioni successive
    - 15.13 e versioni successive
- Aurora MySQL versione 3.10 (compatibile con MySQL 8.0.42) e versioni successive

128 TiB (massimo):

- Tutte le versioni precedenti di Aurora PostgreSQL
- Tutte le versioni di Aurora MySQL 3 disponibili, Aurora MySQL versione 2, versione 2.09 e successive

Per ulteriori informazioni sul dimensionamento automatico dell'archiviazione, consulta Ridimensionamento automatico dello storage Aurora.

Per monitorare lo spazio di archiviazione rimanente, è possibile utilizzare il parametro `AuroraVolumeBytesLeftTotal`. Per ulteriori informazioni, consulta Parametri a livello di cluster per Amazon Aurora.

Limiti delle dimensioni delle tabelle SQL

Per un cluster di database Aurora MySQL, la dimensione massima della tabella è 64 tebibyte (TiB). Per un cluster database Aurora PostgreSQL, la dimensione massima della tabella è 32 tebibyte (TiB). Consigliamo di seguire le best practice per la progettazione delle tabelle, ad esempio partizionando le tabelle di grandi dimensioni.

Limiti di ID spazio delle tabelle

L'ID spazio delle tabelle massimo per Aurora MySQL è 2147483647. Se crei e elimini spesso tabelle, assicurati di prestare attenzione allo spazio a disposizione sulle tabelle IDs e pianifica di utilizzare i dump logici. Per ulteriori informazioni, consulta Migrazione logica da MySQL ad Amazon Aurora MySQL mediante mysqldump.

Risoluzione dei problemi per Amazon Aurora

Utilizza le sezioni seguenti per risolvere i problemi che riscontri con le istanze database in Amazon RDS e Amazon Aurora.

Argomenti

- Impossibile connettersi all'istanza database di Amazon RDS
- Problemi relativi alla sicurezza di Amazon RDS
- Reimpostazione della password del ruolo di proprietario dell'istanza di database
- Errore o riavvio di un'istanza di database Amazon RDS
- Modifiche ai parametri di database Amazon RDS che non hanno effetto
- Problemi di memoria liberabile in Amazon Aurora
- Problemi di replica relativi a Amazon Aurora MySQL

Per informazioni sui problemi di debug con l'API Amazon RDS, consulta Risoluzione dei problemi delle applicazioni in Aurora.

Impossibile connettersi all'istanza database di Amazon RDS

Quando non è possibile connettersi a un'istanza database, le cause comuni sono le seguenti:

- Regole in entrata: le regole di accesso applicate dal firewall locale e gli indirizzi IP autorizzati per accedere all'istanza database potrebbero non corrispondere. Il problema è probabilmente correlato alle regole in entrata del gruppo di sicurezza.

  Per impostazione predefinita, le istanze database non consentono l'accesso. L'accesso viene concesso tramite un gruppo di sicurezza associato al VPC che consente il traffico in entrata e in uscita dall'istanza database. Se necessario, aggiungi regole in entrata e in uscita per la situazione particolare al gruppo di sicurezza. Puoi specificare un indirizzo IP, un intervallo di indirizzi IP o un altro gruppo di sicurezza VPC.

> **Note**
>
> Quando si aggiunge una nuova regola in entrata, puoi scegliere My IP (Il mio IP) per Source (Origine) per consentire l'accesso all'istanza database dall'indirizzo IP rilevato nel browser.

Per ulteriori informazioni sulla configurazione di un gruppo di sicurezza, consulta Fornitura dell'accesso al cluster di database nel VPC creando un gruppo di sicurezza.

> **Note**
>
> Le connessioni client da indirizzi IP all'interno dell'intervallo 169.254.0.0/16 non sono consentite. Si tratta di un intervallo APIPA (Automatic Private IP Addressing), utilizzato per gli indirizzi con collegamenti locali.

- Public accessibility (Accesso pubblico): per connettersi all'istanza database dall'esterno del VPC, ad esempio utilizzando un'applicazione client, occorre assegnare all'istanza un indirizzo IP pubblico.

  Per rendere l'istanza accessibile pubblicamente, modificarla e scegliere Yes (Sì) in Public accessibility (Accesso pubblico). Per ulteriori informazioni, consulta Nascondere cluster database in un VPC da Internet.
- Porta: la porta specificata al momento della creazione dell'istanza database non può essere utilizzata per inviare o ricevere comunicazioni a causa delle restrizioni del firewall locale. Per determinare se la rete consente l'utilizzo della porta specificata per le comunicazioni in entrata e in uscita, consulta il tuo amministratore di rete.
- Disponibilità: per una nuova istanza database creata, lo stato dell'istanza database è `creating` finché non è pronta per essere utilizzata. Quando lo stato cambia in `available`, puoi connetterti all'istanza database. A seconda delle dimensioni dell'istanza di database, potresti dover attendere circa 20 minuti prima che questa diventi disponibile.
- Gateway Internet – Per un'istanza database che deve essere accessibile pubblicamente, le sottoreti nel gruppo di sottoreti del database devono disporre di un gateway Internet.

Per configurare un gateway Internet per una sottorete

1. Accedi alla Console di gestione AWS e apri la console Amazon RDS all'indirizzo https://console.aws.amazon.com/rds/.
2. Nel riquadro di navigazione, scegliere Databases (Database) e selezionare il nome dell'istanza database.
3. Nella scheda Connectivity & security (Connettività e sicurezza) annotare i valori dell'ID VPC in VPC e l'ID della sottorete in Subnets (Sottoreti).
4. Accedi alla console Amazon VPC all'indirizzo https://console.aws.amazon.com/vpc/.
5. Nel riquadro di navigazione, scegliere Internet Gateways. Verificare che al VPC sia associato un Internet gateway. In caso contrario, scegliere Create Internet Gateway (Crea Internet gateway) per crearne uno. Selezionare l'Internet gateway, quindi Attach to VPC (Associa a VPC) e seguire le istruzioni di associazione del gateway al VPC.
6. Nel riquadro di navigazione scegliere Subnets (Sottoreti) e selezionare la sottorete desiderata.
7. Nella scheda Route Table (Tabella di routing) verificare che sia presente un instradamento con `0.0.0.0/0` come destinazione e l'Internet gateway del VPC come target.

   Se si sta effettuando la connessione all'istanza utilizzando il relativo indirizzo IPv6, verificare che sia disponibile un instradamento per tutto il traffico IPv6 (`::/0`) che punti all'Internet gateway. In caso contrario, eseguire le seguenti operazioni:

   a. Selezionare l'ID per la tabella di routing (rtb-xxxxxxxx) per navigare alla tabella di routing.
   b. Nella scheda Routes (Route), scegliere Edit routes (Modifica route). Selezionare Add route (Aggiungi route), utilizzare `0.0.0.0/0` come destinazione e il gateway internet come target.

      Per IPv6, selezionare Add route (Aggiungi route), utilizzare `::/0` come destinazione e il gateway internet come target.
   c. Selezionare Save routes (Salva route).

   Inoltre, se stai tentando di connetterti all'endpoint IPv6, assicurati che l'intervallo di indirizzi IPv6 del client sia autorizzato a connettersi all'istanza database.

Per ulteriori informazioni, consulta Uso di un cluster database in un VPC.

Test della connessione a un'istanza database

Puoi verificare la connessione a un'istanza database utilizzando strumenti comuni di Linux o Windows.

Da un terminale Linux o Unix, puoi eseguire il test della connessione immettendo quanto segue. Sostituisci *DB-instance-endpoint* con l'endpoint e *port* con la porta dell'istanza database.

```
nc -zv DB-instance-endpoint port
```

Di seguito è riportato un comando di esempio e il valore restituito.

```
nc -zv postgresql1.c6c8mn7fake0.us-west-2.rds.amazonaws.com 8299

  Connection to postgresql1.c6c8mn7fake0.us-west-2.rds.amazonaws.com 8299 port [tcp/
vvr-data] succeeded!
```

Gli utenti Windows possono utilizzare Telnet per eseguire il test della connessione a un'istanza di database. Le operazioni di Telnet non sono supportate, ad eccezione del test della connessione. Se una connessione ha esito positivo, l'operazione non restituisce alcun messaggio. Se una connessione non va a buon fine, riceverai un messaggio di errore del tipo seguente.

```
C:\>telnet sg-postgresql1.c6c8mntfake0.us-west-2.rds.amazonaws.com 819

  Connecting To sg-postgresql1.c6c8mntfake0.us-west-2.rds.amazonaws.com...Could not
 open
  connection to the host, on port 819: Connect failed
```

Se le operazioni di Telnet hanno esito positivo, il tuo gruppo di sicurezza è configurato correttamente.

> **Note**
>
> Amazon RDS non accetta il traffico del protocollo ICMP (Internet Control Message Protocol), incluso il ping.

Risoluzione di problemi di autenticazione della connessione

In alcuni casi, è possibile connettersi all'istanza database, ma si ricevono errori di autenticazione. In questi casi, potrebbe essere necessario ripristinare la password utente master per l'istanza database. A tale scopo, è necessario modificare l'istanza RDS.

Problemi relativi alla sicurezza di Amazon RDS

Per evitare problemi di sicurezza, non utilizzare mai l'indirizzo e-mail e la password dell'utente root dell'Account AWS per un account utente. Come best practice, si consiglia di utilizzare l'utente root per creare utenti e assegnarli agli account utente database. È anche possibile utilizzare l'utente root per creare altri account utente, se necessario.

Per informazioni sulla creazione di utenti, consulta Creazione di un utente IAM nell'Account AWS. Per informazioni sulla creazione di utenti in AWS IAM Identity Center, consulta Manage identities in IAM Identity Center (Gestione delle identità nel Centro identità IAM).

Messaggio di errore "Impossibile recuperare gli attributi dell'account, alcune funzioni della console potrebbero non essere attive."

Puoi ricevere questo errore per diversi motivi. È possibile che l'account non disponga delle autorizzazioni o che l'account non sia stato configurato correttamente. Se si tratta di un nuovo account, potrebbe essere necessario attendere che l'account sia pronto. Se si tratta di un account esistente, nelle policy di accesso potrebbero non essere disponibili alcune autorizzazioni per eseguire determinate operazioni, come creare un'istanza database. Per risolvere il problema, l'amministratore deve fornire i ruoli necessari per il tuo account. Per ulteriori informazioni, consulta la documentazione di IAM.

Reimpostazione della password del ruolo di proprietario dell'istanza di database

Se si viene bloccati dal cluster di DB, è possibile accedere come utente master. È quindi possibile reimpostare le credenziali per altri utenti o ruoli amministrativi. Se non riesci ad accedere come utente master, il proprietario dell'account AWS può reimpostare la password dell'utente master. Per informazioni dettagliate sugli account o sui ruoli amministrativi che potrebbe essere necessario reimpostare, vedere Privilegi dell'account utente master.

Puoi modificare la password dell'istanza database utilizzando la console Amazon RDS, il comando AWS CLI modify-db-instance oppure l'operazione API ModifyDBInstance. Per ulteriori informazioni sulla modifica di un'istanza database in un cluster di database, consulta Modifica di un'istanza database in un cluster database.

Errore o riavvio di un'istanza di database Amazon RDS

Un errore dell'istanza database può verificarsi quando viene riavviata. Può anche verificarsi quando l'istanza database viene inserita in uno stato che impedisce l'accesso e quando il database viene riavviato. Un riavvio può verificarsi quando si riavvia manualmente l'istanza database. Un riavvio può anche verificarsi quando si modifica un'impostazione dell'istanza database che richiede un riavvio prima che la modifica diventi effettiva.

Il riavvio di un'istanza database può verificarsi quando si modifica un'impostazione che richiede un riavvio o quando il riavvio viene innescato manualmente. Un riavvio può verificarsi immediatamente se si modifica un'impostazione e si richiede che la modifica abbia effetto immediato. Oppure può verificarsi durante la finestra di manutenzione dell'istanza database.

Un riavvio dell'istanza del database si verifica immediatamente quando si verifica una delle seguenti condizioni:

- Il periodo di retention dei backup per un'istanza database viene modificato da 0 a un valore diverso da zero o da un valore diverso da zero a 0. Quindi si imposta Apply Immediately (Applica immediatamente) su `true`.
- Si modifica la classe di istanza database e si imposta Apply Immediately (Applica immediatamente) su `true`.

Un riavvio dell'istanza di database avviene durante la finestra di manutenzione quando si verifica una delle seguenti condizioni:

- Si modifica il periodo di retention dei backup per un'istanza database da 0 a un valore diverso da zero o da un valore diverso da zero a 0 e Apply Immediately (Applica immediatamente) è impostato su `false`.
- Si modifica la classe di istanza database e si imposta Apply Immediately (Applica immediatamente) su `false`.

Quando si modifica un parametro statico in un gruppo di parametri database, la modifica non diventa effettiva fino al riavvio dell'istanza database associata al gruppo di parametri. La modifica richiede un riavvio manuale. L'istanza database non viene riavviata automaticamente durante la finestra di manutenzione.

Modifiche ai parametri di database Amazon RDS che non hanno effetto

In alcuni casi, si potrebbe modificare un parametro in un gruppo di parametri database senza che le modifiche diventino effettive. In tal caso, è probabile che sia necessario riavviare l'istanza database associata al gruppo di parametri database. Quando si modifica un parametro dinamico, la modifica diventa immediatamente effettiva. Quando si modifica un parametro statico, la modifica non diventa effettiva finché l'istanza database associata al gruppo di parametri non viene riavviata.

Puoi riavviare un'istanza database utilizzando la console RDS. In alternativa puoi chiamare esplicitamente l'operazione API `RebootDBInstance`. È possibile riavviare senza failover se l'istanza database è in un'implementazione Multi-AZ. La necessità di riavviare l'istanza database associata dopo la modifica di un parametro statico consente di ridurre il rischio di errore di configurazione di un parametro che influenza una chiamata API. Un esempio è la chiamata di `ModifyDBInstance` per modificare la classe di istanza database. Per ulteriori informazioni, consulta Modifica dei parametri in un gruppo di parametri database in Amazon Aurora.

Problemi di memoria liberabile in Amazon Aurora

La memoria liberabile è la memoria RAM (Random Access Memory) su un'istanza database che può essere resa disponibile al motore di database. È la somma della memoria libera del sistema operativo (OS) e della memoria disponibile del buffer e della cache delle pagine. Il motore di database utilizza la maggior parte della memoria sull'host, ma anche i processi del sistema operativo utilizzano RAM. La memoria attualmente allocata al motore di database o utilizzata dai processi del sistema operativo non è inclusa nella memoria liberabile. Quando il motore di database sta per esaurire la memoria, l'istanza database può utilizzare lo spazio temporaneo normalmente utilizzato per il buffering e la memorizzazione nella cache. Come accennato in precedenza, questo spazio temporaneo è incluso nella memoria liberabile.

Puoi usare il parametro `FreeableMemory` in Amazon CloudWatch per monitorare la memoria liberabile. Per ulteriori informazioni, consulta Monitoraggio degli strumenti di Amazon Aurora.

Se la memoria liberabile dell'istanza database diventa insufficiente o viene utilizzato uno spazio di scambio, valutare l'aumento a una classe di istanza database più grande. Per ulteriori informazioni, consulta Classi di istanze database Amazon Aurora.

Inoltre, puoi modificare le impostazioni della memoria. Ad esempio, in Aurora MySQL , puoi regolare le dimensioni del parametro `innodb_buffer_pool_size`. Questo parametro è impostato per default sul 75% della memoria fisica. Per ulteriori suggerimenti per la risoluzione di problemi di MySQL, consulta Come è possibile risolvere i problemi di memoria liberabile insufficiente in un database Amazon RDS per MySQL?

Per Aurora Serverless v2, `FreeableMemory` rappresenta la quantità di memoria inutilizzata disponibile quando viene eseguito il dimensionamento dell'istanza database di Aurora Serverless v2 fino alla sua capacità massima. È possibile che l'istanza sia stata ridotta a una capacità relativamente bassa, ma continui a segnalare un valore elevato per `FreeableMemory`, perché l'istanza può aumentare. Questa memoria non è disponibile al momento, ma puoi recuperarla se ti serve.

Per ogni unità di capacità Aurora (ACU) la cui capacità corrente è inferiore alla capacità massima, `FreeableMemory` aumenta di circa 2 GiB. Pertanto, questo parametro non si avvicina a zero finché non viene eseguito il dimensionamento verso l'alto dell'istanza database fino al valore più alto possibile.

Se questo parametro si avvicina al valore `0`, è stato raggiunto il massimo di aumento dell'istanza database. Ovvero al valore più prossimo al limite di memoria disponibile. Valuta la possibilità di aumentare l'impostazione ACU massima per il cluster. Se il valore di questo parametro si avvicina a `0` per un'istanza database di lettura, valuta la possibilità di aggiungere altre istanze database di lettura al cluster. In questo modo, puoi distribuire la parte di sola lettura del carico di lavoro su più istanze database e ridurre conseguentemente l'utilizzo della memoria su ogni istanza database di lettura. Per ulteriori informazioni, consulta Parametri di Amazon CloudWatch importanti per Aurora Serverless v2.

Per Aurora Serverless v1, puoi modificare l'intervallo di capacità per utilizzare più ACU. Per ulteriori informazioni, consulta Modifica di un cluster di database Aurora Serverless v1.

Problemi di replica relativi a Amazon Aurora MySQL

Alcuni problemi di replica MySQL si applicano anche a Aurora MySQL. Puoi diagnosticare e risolvere questi problemi.

Argomenti

- Diagnosi e risoluzione del ritardo tra repliche di lettura

- Diagnosi e risoluzione di un errore relativo alla replica di lettura MySQL
- Errore di replica interrotta
- La replica di lettura non riesce a inizializzare la struttura dei metadati

Diagnosi e risoluzione del ritardo tra repliche di lettura

Quando una replica di lettura MySQL viene creata ed è disponibile, Amazon RDS esegue per prima cosa la replica delle modifiche apportate all'istanza database di origine dal momento in cui l'operazione di creazione della replica di lettura è stata avviata. Durante questa fase, il ritardo di replica per la replica di lettura è maggiore di 0. Puoi monitorare questo tempo di ritardo in Amazon CloudWatch visualizzando il parametro Amazon RDS `AuroraBinlogReplicaLag`.

Il parametro `AuroraBinlogReplicaLag` indica il valore del campo `Seconds_Behind_Master` del comando `SHOW REPLICA STATUS` di MySQL. Per ulteriori informazioni, consulta Istruzione SHOW REPLICA STATUS nella documentazione di MySQL.

Quando il parametro `AuroraBinlogReplicaLag` è 0, la replica ha raggiunto l'istanza del database di origine. Se il parametro `AuroraBinlogReplicaLag` restituisce -1, la replica potrebbe non essere attiva. Per la risoluzione di problemi relativi a un errore di replica, consulta Diagnosi e risoluzione di un errore relativo alla replica di lettura MySQL. Un valore di `AuroraBinlogReplicaLag` pari a -1 può anche indicare che il valore di `Seconds_Behind_Master` non può essere determinato oppure è `NULL`.

> **Note**
>
> Versioni precedenti di Aurora MySQL utilizzavano `SHOW SLAVE STATUS` al posto di `SHOW REPLICA STATUS`. Se si utilizza una versione 1 o 2 Aurora MySQL, utilizzare `SHOW SLAVE STATUS`. Utilizzare `SHOW REPLICA STATUS` per Aurora MySQL versione 3 e successive.

Il parametro `AuroraBinlogReplicaLag` restituisce -1 durante un'interruzione della rete o quando viene applicata una patch durante la finestra di manutenzione. In questo caso, attendi fino al ripristino della connettività di rete o al termine della finestra di manutenzione prima di controllare nuovamente il parametro `AuroraBinlogReplicaLag`.

La tecnologia di replica di lettura di MySQL è asincrona. Pertanto, è possibile che si verifichino incrementi occasionali del parametro `BinLogDiskUsage` nell'istanza database di origine e del

parametro `AuroraBinlogReplicaLag` nella replica di lettura. Ad esempio, considera una situazione in cui si verifica un elevato volume di operazioni di scrittura nell'istanza database di origine in parallelo. Contemporaneamente, le operazioni di scrittura nella replica di lettura vengono serializzate utilizzando un singolo thread I/O. Tale situazione può portare a un ritardo tra l'istanza di origine e la replica di lettura.

Per ulteriori informazioni sulle repliche di lettura e su MySQL, consulta la pagina dei dettagli dell'implementazione di repliche nella documentazione di MySQL.

Puoi ridurre il ritardo tra gli aggiornamenti di un'istanza database di origine e i successivi aggiornamenti della replica di lettura attenendoti alla seguente procedura:

- Imposta la classe dell'istanza database della replica di lettura in modo che le sue dimensioni di storage siano paragonabili a quelle dell'istanza del database di origine.
- Assicurati che le impostazioni dei parametri dei gruppi di parametri database utilizzati dall'istanza database di origine e la replica di lettura siano compatibili. Per ulteriori informazioni e un esempio, consulta la discussione sul parametro `max_allowed_packet` nella sezione seguente.
- Disabilita la cache delle query. Per le tabelle modificate di frequente, l'uso della cache delle query può aumentare il ritardo della replica perché la cache viene bloccata e aggiornata spesso. In questo caso, potresti visualizzare un ritardo della replica inferiore se disabiliti la cache delle query. Puoi disabilitare la cache delle query impostando il parametro `query_cache_type parameter` sul valore 0 nel gruppo di parametri di database dell'istanza di database. Per ulteriori informazioni sulla cache delle query, consulta la sezione relativa alla configurazione della cache delle query.
- Riscaldare il buffer pool sulla replica di lettura per InnoDB per MySQL. Ad esempio, supponi di disporre di un set ridotto di tabelle che vengono aggiornate di frequente e di utilizzare lo schema di tabella InnoDB o XtraDB. In questo caso, esegui il dump di tali tabelle nella replica di lettura In questo modo, il motore di database esegue la scansione delle righe delle tabelle del disco e le memorizza nella cache nel pool di buffer, il che può ridurre il ritardo della replica. Di seguito viene riportato un esempio.

  Per Linux, macOS o Unix:

```
PROMPT> mysqldump \
    -h <endpoint> \
    --port=<port> \
    -u=<username> \
    -p <password> \
    database_name table1 table2 > /dev/null
```

Per Windows:

```
PROMPT> mysqldump ^
    -h <endpoint> ^
    --port=<port> ^
    -u=<username> ^
    -p <password> ^
    database_name table1 table2 > /dev/null
```

Diagnosi e risoluzione di un errore relativo alla replica di lettura MySQL

Amazon RDS monitora lo stato delle repliche di lettura. RDS aggiorna il campo Replication State (Stato di replica) dell'istanza della replica di lettura con il valore `Error`, se la replica viene arrestata per qualsiasi motivo. Puoi rivedere i dettagli dell'errore associato generato dai motori MySQL visualizzando il campo Replication Error (Errore di replica). Vengono generati anche eventi che indicano lo stato della replica di lettura, inclusi RDS-EVENT-0045, RDS-EVENT-0046 e RDS-EVENT-0057. Per ulteriori informazioni sugli eventi e sulla sottoscrizione a essi, consulta Utilizzo della notifica degli eventi di Amazon RDS. Se viene restituito un messaggio di errore MySQL, controlla l'errore nella documentazione dei messaggi di errore MySQL.

Situazioni comuni che possono causare errori di replica sono:

- Il valore del parametro `max_allowed_packet` di una replica di lettura è inferiore al parametro `max_allowed_packet` dell'istanza database di origine.

  Il parametro `max_allowed_packet` è un parametro personalizzato che puoi impostare in un gruppo di parametri database. Il parametro `max_allowed_packet` viene utilizzato per specificare la dimensione massima delle query DML (Data Manipulation Language) che possono essere eseguite sul database. In alcuni casi, il valore `max_allowed_packet` dell'istanza database di origine potrebbe essere più grande del valore `max_allowed_packet` per la replica di lettura. In questo caso, il processo di replica può generare un errore e interrompere la replica. L'errore più comune è `packet bigger than 'max_allowed_packet' bytes`. Puoi correggere questo errore impostando l'origine e la replica di lettura in modo che utilizzino i gruppi di parametri database con gli stessi valori del parametro `max_allowed_packet`.
- Scrittura in tabelle su una replica di lettura. Se crei indici su una replica di lettura, il parametro `read_only` deve essere impostato su 0 affinché gli indici vengano creati. Se esegui la scrittura su tabelle sulla replica di lettura, ciò può interrompere la replica.

- Uso di un motore di storage non transazionale come MyISAM. Le repliche di lettura richiedono un motore di storage transazionale. La replica è supportata solo per i seguenti motori di archiviazione: InnoDB per MySQL o MariaDB.

  Puoi convertire una tabella MyISAM in InnoDB, utilizzando il comando seguente:

  ```
  alter table <schema>.<table_name> engine=innodb;
  ```

- Utilizzo di query non deterministiche non sicure come `SYSDATE()`. Per ulteriori informazioni, consulta la pagina relativa alla determinazione delle istruzioni sicure e non sicure nel logging binario nella documentazione MySQL.

La seguente procedura può essere di aiuto per la risoluzione dell'errore di replica:

- Se riscontri un errore logico e puoi ignorarlo in modo sicuro, segui le fasi descritte in Ignorare l'errore di replica corrente. L'istanza database Aurora MySQL deve eseguire una versione che includa la procedura `mysql_rds_skip_repl_error`. Per ulteriori informazioni, consulta mysql_rds_skip_repl_error.
- Se si verifica un problema di posizione del registro binario (binlog), puoi modificare la posizione di riproduzione della replica. Per farlo, utilizza il comando `mysql.rds_next_master_log` per Aurora MySQL versione 1 e 2. Per farlo, utilizza il comando `mysql.rds_next_source_log` per Aurora MySQL versione 3 e successive. L'istanza database Aurora MySQL deve eseguire una versione che supporta questo comando per modificare la posizione di riproduzione della replica. Per informazioni sulle versioni, consulta mysql_rds_next_master_log.
- Se si verifica un problema temporaneo a livello di prestazioni dovuto a un elevato carico DML, è possibile impostare il parametro `innodb_flush_log_at_trx_commit` su 2 nel gruppo di parametri database sulla replica di lettura. Ciò può agevolare il recupero delle prestazioni della replica di lettura, sebbene le proprietà ACID (atomicità, consistenza, isolamento e durata) subiscano una riduzione temporanea.
- Puoi eliminare la replica di lettura e creare un'istanza utilizzando lo stesso identificatore istanze DB. In questo modo, l'endpoint rimane identico a quello della vecchia replica di lettura.

Quando un problema relativo alla replica viene risolto, il campo Replication State (Stato di replica) cambia in replicating (replica in corso). Per ulteriori informazioni, consulta Risoluzione dei problemi relativi a una replica di lettura MySQL.

Errore di replica interrotta

Quando si chiama il comando `mysql.rds_skip_repl_error`, è possibile che venga visualizzato un messaggio di errore che indica che la replica è inattiva o disattivata.

Questo messaggio di errore viene visualizzato perché la replica è stata arrestata e non può essere riavviata.

Se devi ignorare un numero elevato di errori, il ritardo della replica potrebbe superare il periodo di retention predefinito per i file di log binari. In questo caso, si potrebbe verificare un errore irreversibile a causa di file di log binari che sono stati eliminati prima di essere riprodotti sulla replica. Questa eliminazione causa l'arresto della replica e non è più possibile chiamare il comando `mysql.rds_skip_repl_error` per ignorare errori di replica.

Puoi limitare questo problema aumentando il numero di ore di retention dei file di log binari nella sorgente di replica. Una volta aumentato il tempo di retention dei file binlog, puoi riavviare la replica e chiamare il comando `mysql.rds_skip_repl_error` secondo necessità.

Per impostare il periodo di retention dei file binlog, utilizza la procedura mysql_rds_set_configuration . Specifica un parametro di configurazione di "ore di conservazione dei file binlog” insieme al numero di ore di conservazione dei file binlog nel cluster di database, fino a 2160 (90 giorni). Il valore predefinito per Aurora MySQL è 24 (1 giorno). Nell'esempio seguente il periodo di retention dei file binlog è impostato su 48 ore.

```
CALL mysql.rds_set_configuration('binlog retention hours', 48);
```

La replica di lettura non riesce a inizializzare la struttura dei metadati

Al tentativo di avviare la replica, è stato ricevuto il seguente messaggio di errore:

```
Read Replica Replication Error - SQLError: 13124, reason: Replica failed to initialize
 applier metadata structure from the repository
```

Questo errore si verifica se viene rilevato un problema relativo alla struttura dei metadati della replica. Per correggere la struttura dei metadati, è necessario creare una nuova replica.

Per evitare che ciò accada in futuro, eseguire una delle azioni riportate di seguito:

- Se possibile, disattivare il multithreading sulle repliche. A partire da MySQL 8.0.27, il multithreading è abilitato per impostazione predefinita.

- Se è necessario utilizzare il multithreading sulle repliche, è consigliabile utilizzare la replica basata su GTID. Per ulteriori informazioni, consulta Utilizzo della replica basata su GTID.

Documentazione di riferimento dell'API Amazon RDS

Oltre alla Console di gestione AWS e AWS Command Line Interface (AWS CLI), Amazon RDS fornisce anche un'API. È possibile usare l'API per automatizzare le attività per la gestione delle istanze database e altri oggetti in Amazon RDS.

- Per un elenco alfabetico delle operazioni API, consulta Operazioni.
- Per un elenco alfabetico dei tipi di dati, consulta la pagina Tipi di dati.
- Per un elenco di parametri di query comuni, consulta la pagina Parametri Comuni.
- Per le descrizioni dei codici di errore, consulta la pagina Errori comuni.

Per ulteriori informazioni sui AWS CLI, consulta Riferimento AWS Command Line Interface per Amazon RDS.

Argomenti

- Uso dell'API query
- Risoluzione dei problemi delle applicazioni in Aurora

Uso dell'API query

Le sezioni seguenti illustrano brevemente le autenticazioni dei parametri e delle richieste usate con l'API query.

Per informazioni generali sul funzionamento dell'API Query, consulta Richieste di query in Amazon EC2 API Reference.

Parametri di query

Le richieste basate su query HTTP sono richieste HTTP che utilizzano i verbi HTTP GET oppure POST e un parametro di query denominato `Action`.

Ogni richiesta di query deve includere alcuni parametri comuni per gestire l'autenticazione e la selezione di un'azione.

Alcune operazioni accettano elenchi di parametri. Questi elenchi sono specificati usando l'annotazione `param.`*n*. I valori di *n* sono numero a partire da 1.

Per ulteriori informazioni su endpoint e regioni Amazon RDS, consulta Amazon Relational Database Service (RDS) nella sezione Regioni ed endpoint di Riferimenti generali di Amazon Web Services.

Autenticazione delle richieste di query

È possibile inviare solo richieste di query tramite HTTPS ed è necessario includere una firma in ogni richiesta di query. È necessario usare AWS Signature Version 4 o Signature Version 2. Per ulteriori informazioni, consulta Processo di firma Signature Version 4 e Processo di firma Signature Version 2.

Risoluzione dei problemi delle applicazioni in Aurora

Amazon RDS fornisce errori specifici e descrittivi per aiutarti a risolvere i problemi mentre interagisci con l'API Amazon RDS.

Argomenti

- Errore durante il recupero
- Suggerimenti per la risoluzione dei problemi

Per informazioni sulla risoluzione dei problemi per le istanze database Amazon RDS, consulta Risoluzione dei problemi per Amazon Aurora.

Errore durante il recupero

In genere, si desidera che l'applicazione verifichi se una richiesta ha generato un errore prima di trascorrere del tempo a elaborare i risultati. Il modo più semplice per determinare se si è verificato un errore consiste nel cercare un nodo `Error` nella risposta dall'API Amazon RDS.

La sintassi XPath fornisce un modo semplice per rilevare la presenza di un nodo `Error`. Fornisce inoltre un modo relativamente semplice per recuperare il messaggio e il codice di errore. Il seguente snippet di codice usa Perl e il modulo XML::XPath per determinare se si è verificato un errore durante una richiesta. Se si è verificato un errore, il codice stampa il primo codice di errore e il messaggio nella risposta.

```
use XML::XPath;
    my $xp = XML::XPath->new(xml =>$response);
    if ( $xp->find("//Error") )
    {print "There was an error processing your request:\n", " Error code: ",
```

```
    $xp->findvalue("//Error[1]/Code"), "\n", " ",
    $xp->findvalue("//Error[1]/Message"), "\n\n"; }
```

Suggerimenti per la risoluzione dei problemi

Ti consigliano i seguenti processi per diagnosticare e risolvere i problemi con l'API Amazon RDS:

- Verifica che Amazon RDS funzioni normalmente nella regione AWS di destinazione controllando http://status.aws.amazon.com.
- Verificare la struttura della richiesta.

  Ogni operazione Amazon RDS ha una pagina di riferimento nella documentazione di riferimento dell'API Amazon RDS. Controllare nuovamente che si stia usando i parametri correttamente. Per idee su cosa potrebbe essere sbagliato, guarda le richieste di esempio o gli scenari utente per vedere se tali esempi stanno eseguendo operazioni simili.
- Controlla AWS re:Post.

  Amazon RDS dispone di una community di sviluppo in cui puoi cercare soluzioni ai problemi che altri hanno riscontrato lungo il percorso. Per vedere gli argomenti, consulta AWS re:Post.

Cronologia dei documenti

Versione corrente dell'API: 2014-10-31

Nella tabella seguente sono descritte le modifiche importanti apportate alla Guida per l'utente di Amazon Aurora. Per ricevere notifiche sugli aggiornamenti di questa documentazione, puoi abbonarti a un feed RSS. Per ulteriori informazioni su Amazon Relational Database Service (Amazon RDS), consultare la Guida per l'utente di Amazon Relational Database Service.

> **Note**
>
> Prima del 31 agosto 2018, Amazon Aurora era documentato nella Guida per l'utente di Amazon Relational Database Service. Per la cronologia dei documenti di Aurora precedente, consulta Cronologia dei documenti nella Guida per l'utente di Amazon Relational Database Service.

Please change to "Puoi filtrare le nuove funzionalità Amazon Aurora alla pagina Quali sono le novità del database?. In Prodotti, seleziona Amazon Aurora. Quindi esegui la ricerca utilizzando parole chiave come **`global database`** o **`Serverless`**.

Modifica	Descrizione	Data
Integrazioni Aurora PostgreSQL Zero-ETL con supporto di regioni aggiuntive Amazon SageMaker	Le integrazioni Aurora PostgreSQL Zero-ETL con sono ora disponibili in regioni aggiuntive. Amazon SageMaker Per ulteriori informazioni, consulta Regioni supportate e motori Aurora DB per integrazioni zero-ETL.	15 ottobre 2025
Database Aurora PostgreSQL Limitless disponibile in una regione aggiuntiva AWS	Il database Aurora PostgreSQL Limitless è ora disponibile in. AWS GovCloud (US) Regions Per ulteriori informazioni, consulta Requisiti e considera	16 settembre 2025

	zioni del database Aurora PostgreSQL Limitless.	
L'API dati di Amazon RDS supporta IPv6	L'API dati Amazon RDS ora supporta la IPv6 connettività, consentendoti di connetterti ai database IPv6 Aurora utilizzando gli indirizzi. Per ulteriori informazioni, consulta Using the Data API.	30 agosto 2025
Amazon Aurora è disponibile nella regione Asia Pacifico (Nuova Zelanda)	Amazon Aurora è ora disponibile nella regione Asia Pacifico (Nuova Zelanda). Per ulteriori informazioni, consulta Regioni e zone di disponibilità.	28 agosto 2025
Amazon RDS Data API disponibile in una regione aggiuntiva AWS	L'API di dati in Amazon RDS ora è disponibile nella Regione Europa (Spagna). Per ulteriori informazioni, consulta Regioni supportate e motori di database Aurora per l'API di dati in Amazon RDS.	1 agosto 2025
L'integrazione Zero-ETL con Amazon SageMaker Lakehouse è disponibile a livello generale	L'integrazione Zero-ETL rende disponibili i dati transazionali in Amazon SageMaker entro pochi secondi dalla scrittura su un'istanza database Aurora MySQL. Per ulteriori informazioni, consulta Working with zero-ETL integrations.	30 giugno 2025

Amazon Aurora è disponibile nella regione Asia Pacifico (Taipei)	Amazon Aurora è ora disponibile nella regione Asia Pacifico (Taipei). Per ulteriori informazioni, consulta Regioni e zone di disponibilità.	5 giugno 2025
End-of-life informazioni per Performance Insights	AWS ha annunciato la end-of-life data di Performance Insights: 30 novembre 2025. Dopo questa data, Amazon RDS non supporterà più l'esperienza della console Approfondimenti sulle prestazioni, i periodi di conservazione flessibili e i relativi prezzi. Consigliamo di aggiornare i cluster di database a Database Insights prima di questa data. Per ulteriori informazioni, consulta Monitoring DB load with Performance Insights on Amazon Aurora	30 maggio 2025
Il database Aurora PostgreSQL Limitless supporta Database Insights CloudWatch	Ora puoi usare CloudWatch Database Insights per monitorare il tuo database Amazon Aurora PostgreSQL Limitless su larga scala. Per ulteriori informazioni, consulta Monitoraggio del database Amazon Aurora PostgreSQL Limitless con Database Insights. CloudWatch	22 maggio 2025

Visualizzazione delle date di supporto per le versioni principali del motore open source	È ora possibile visualizzare le informazioni sulle date di inizio e di fine per le versioni principali dei motori open source in RDS Standard Support e RDS Extended Support utilizzando l'API AWS CLI e RDS. Per ulteriori informazioni, consulta Visualizzazione delle date di supporto per le versioni del motore con supporto esteso per Amazon RDS.	20 maggio 2025
L'integrazione Zero-ETL di Aurora PostgreSQL supporta Regioni aggiuntive	Le integrazioni Zero-ETL di Aurora PostgreSQL sono ora disponibili in altre Regioni. Per ulteriori informazioni, consulta Regioni supportate e motori di database Aurora per integrazione Zero-ETL con Amazon Redshift.	17 febbraio 2025
Amazon Aurora è disponibile nella regione del Messico (Centrale)	Amazon Aurora è ora disponibile nella regione del Messico (Centrale). Per ulteriori informazioni, consulta Regioni e zone di disponibilità.	14 gennaio 2025
Amazon Aurora è disponibile nella regione Asia Pacifico (Tailandia)	Amazon Aurora è ora disponibile nella regione Asia Pacifico (Tailandia). Per ulteriori informazioni, consulta Regioni e zone di disponibilità.	7 gennaio 2025

End-of-life informazioni per Aurora Serverless v1	AWS ha annunciato la end-of-life data delAurora Serverless v1: 31 marzo 2025. È fortemente consiglia bile aggiornare i cluster di database Aurora Serverles s v1 a Aurora Serverless v2 prima di tale data. L'aggiorn amento può comportare una modifica del numero di versione principale del motore di database. Pertanto, è importante pianificare, testare e implementare questo passaggio prima della data. end-of-life A partire dall'8 gennaio 2025, i clienti non saranno più in grado di creare nuovi Aurora Serverless v1 cluster o istanze né con la CLI. Console di gestione AWS Per informazioni sulla procedura di aggiornamento da Aurora Serverless v1 a Aurora Serverless v2, consulta Upgrading from an Aurora Serverless v1 cluster to Aurora Serverless v2.	6 dicembre 2024

Aurora supporta CloudWatch Database Insights	Ora puoi usare CloudWatch Database Insights per monitorare i tuoi cluster Amazon Aurora DB su larga scala. Per ulteriori informazioni, consulta Monitoraggio dei database Amazon Aurora con CloudWatch Database Insights.	1 dicembre 2024
Amazon Aurora supporta le classi di istanza db.r8g	Ora è possibile utilizzare le classi di istanza db.r8g per creare cluster di database Aurora. Per ulteriori informazioni, consulta Classi di istanze database Aurora.	21 novembre 2024
Aurora Serverless v2 supporta la pausa e la ripresa automatiche	Aurora Serverless v2Le istanze DB possono ora scalare fino a zero unità di capacità Aurora ACUs () dopo un periodo di inattività. Per abilitare questa funzionalità, impostate il valore di capacità minima su 0. ACUs Per ulteriori informazioni, consulta Automatic pause and resume for Aurora Serverless v2	20 novembre 2024

Amazon Aurora supporta la migrazione automatica dei database EC2	È possibile utilizzare la console RDS per migrare un EC2 database in Aurora. Aurora utilizza AWS Database Migration Service (AWS DMS) per migrare il database di origine. EC2 AWS DMS ti consente di migrare i database relazionali nel tuo. Cloud AWS Per ulteriori informazioni, consulta Migrazione automatica dei EC2 database su Amazon Aurora AWS Database Migration Service utilizzando.	20 novembre 2024
AWS Wrapper NodeJS avanzato generalmente disponibile	Amazon Web Services (AWS) Advanced NodeJS Wrapper è complementare a un driver NodeJS esistente e ne estende la funzionalità. Per ulteriori informazioni, consulta Connessione ai cluster Aurora DB con i driver. AWS	19 novembre 2024
Amazon Aurora supporta le classi di istanza db.r7i	Ora è possibile utilizzare le classi di istanza db.r7i per creare cluster di database Aurora. Per ulteriori informazioni, consulta Classi di istanze database Aurora.	18 novembre 2024

L’API di dati RDS per Aurora MySQL supporta tutte le visualizzazioni di Approfondimenti sulle prestazioni	Ora è possibile utilizzare le visualizzazioni Host principali o Applicazioni principali per Approfondimenti sulle prestazioni per monitorare le operazioni SQL eseguite utilizzando l’API di dati RDS per Aurora MySQL. Per ulteriori informazioni sul supporto di Approfondimenti sulle prestazioni, consulta Monitoring RDS Data API queries with Performance Insights.	31 ottobre 2024
Aurora PostgreSQL Limitless Database	Amazon Aurora PostgreSQL Limitless Database è una nuova funzionalità di dimensionamento orizzontale (sharding) automatizzata di Amazon Aurora. Aurora PostgreSQL Limitless Database consente di superare i limiti esistenti di Aurora per il throughput di scrittura e lo spazio di archiviazione distribuendo il carico di lavoro del database su più istanze di scrittura Aurora, permettendone al contempo l’utilizzo come un singolo database. Per ulteriori informazioni, consulta Amazon Aurora PostgreSQL Limitless Database.	31 ottobre 2024

Fine del supporto standard per Amazon Aurora MySQL versione 2	Il supporto standard per Aurora MySQL versione 2 è terminato il 31 ottobre 2024. Per ulteriori informazioni, consulta Preparing for Amazon Aurora MySQL-Compatible Edition version 2 end of standard support.	31 ottobre 2024
Amazon Aurora supporta gli aggiornamenti del sistema operativo a livello di cluster	È ora possibile applicare gli aggiornamenti del sistema operativo Aurora a livello di cluster di database. Gli aggiornamenti in sequenza applicano automaticamente gli aggiornamenti a poche istanze database di lettura alla volta, preservando così la disponibilità di lettura. Per ulteriori informazioni, consulta Operating system updates for Aurora DB clusters.	30 ottobre 2024

I cluster globali Aurora supportano l’applicazione di tag	Ora è possibile aggiungere tag ai cluster globali Aurora. Un cluster globale Aurora è la risorsa principale che contiene i cluster di database primari e secondari in un database globale Aurora. Per informazioni generali sull’applicazione di tag, consulta Tagging Aurora and Amazon RDS resources. Per informazioni sull’applicazione di tag per il database globale Aurora, consulta Tagging Aurora Global Database resources.	24 ottobre 2024
Endpoint di scrittura per database globali Aurora	Ogni database globale Aurora è dotato di un endpoint di scrittura che viene aggiornato automaticamente da Aurora per instradare le richieste all’istanza di scrittura corrente del cluster di database primario dopo un’operazione di switchover e di failover del database globale Aurora. Per ulteriori informazioni sull’utilizzo dell’endpoint di scrittura insieme allo switchover e al failover del database globale Aurora, consulta Using switchover or failover in Amazon Aurora Global Database.	22 ottobre 2024

Integrazione Zero-ETL di Aurora PostgreSQL con Amazon Redshift disponibile a livello generale	L'integrazione Zero-ETL rende disponibili i dati transazionali in Amazon Redshift in pochi secondi dalla loro scrittura su un cluster di database di origine Aurora PostgreSQL. La funzionalità è ora disponibile a livello generale. Per ulteriori informazioni, consulta Utilizzo delle integrazioni Zero-ETL di Aurora con Amazon Redshift.	15 ottobre 2024
Aurora Serverless v2supporta 256 ACUs	È ora possibile creare cluster Aurora Serverless v2 DB con una capacità massima di 256 unità ACUs di capacità Aurora (). Per ulteriori informazioni, consulta Aurora Serverless v2 capacity.	3 ottobre 2024
Amazon Aurora supporta Console-to-Code	Ora puoi usarlo Console-to-Code per generare codice a partire da azioni eseguite nella console RDS. Il codice generato può aiutarti a scrivere codice per automatizzare l'uso di altri AWS servizi. Per ulteriori informazioni, consulta Utilizzare Console-to-Code per generare codice per le azioni della console Aurora.	3 ottobre 2024

L’API di dati RDS supporta Approfondimenti sulle prestazioni	Ora è possibile utilizzare Approfondimenti delle prestazioni per monitorare le operazioni SQL eseguite utilizzando l’API di dati RDS. Per ulteriori informazioni sul supporto di Approfondimenti sulle prestazioni, consulta Monitoring RDS Data API queries with Performance Insights.	26 settembre 2024
L’API di dati RDS per Aurora Serverless v2 e istanze allocate è disponibile per Aurora MySQL	L’API di dati RDS è ora disponibile per i cluster Aurora MySQL che utilizzano Aurora Serverless v2 o istanze allocate. Per informazioni sull’utilizzo dell’API di dati RDS, consulta Using RDS Data API. Per informazioni sulla versione di Aurora MySQL e sul supporto AWS regionale per RDS Data API, consulta Data API with Aurora MySQL Serverless v2 e provisioned.	26 settembre 2024
Amazon Aurora è disponibile nella Regione Asia Pacifico (Malaysia)	Amazon Aurora è ora disponibile nella Regione Asia Pacifico (Malaysia) Per ulteriori informazioni, consulta Regioni e zone di disponibilità.	22 agosto 2024

Aggiornamento a policy esistenti	Amazon RDS ha rimosso l'autorizzazione `sns:Publish` dalla policy `AmazonRDSBetaServiceRolePolicy` del ruolo collegato al servizio `AWSServiceRoleForRDSBeta`. Per ulteriori informazioni, consulta gli aggiornamenti di Amazon RDS alle policy AWS gestite.	7 agosto 2024
Aggiornamento a policy esistenti	Amazon RDS ha rimosso l'autorizzazione `sns:Publish` dalla policy `AmazonRDSPreviewServiceRolePolicy` del ruolo collegato al servizio `AWSServiceRoleForRDSPreview`. Per ulteriori informazioni, consulta gli aggiornamenti di Amazon RDS alle policy AWS gestite.	7 agosto 2024
Aurora Serverless v2supporta la pausa automatica () scale-to-zero	Aurora Serverless v2Le istanze DB possono ora scalare fino a zero unità di capacità Aurora ACUs () dopo un periodo di inattività. Per abilitare questa funzionalità, è necessario impostare il valore di capacità minima su 0. Per ulteriori informazioni, consulta Utilizzo di Aurora Serverless v2.	25 luglio 2024

AWS Driver ODBC per MySQL generalmente disponibile	Amazon Web Services (AWS) ODBC Driver per MySQL è un driver client progettato per la disponibilità elevata di Aurora MySQL. Per ulteriori informazioni, consulta Connecting to Aurora MySQL with the Amazon Web Services (AWS) ODBC Driver for MySQL.	18 luglio 2024
L'API di dati RDS per Aurora Serverless v2 è disponibile in altre Regioni	L'API RDS Data è ora disponibile per Aurora PostgreSQL in diverse regioni aggiuntive. AWS Per informazioni sul supporto delle Regioni per l'API dati di RDS, consulta API di dati with Aurora PostgreSQL Serverless v2 e istanze allocate.	9 luglio 2024
AWS Il driver Python è disponibile a livello generale	Il driver Python per Amazon Web Services (AWS) è progettato come wrapper Python avanzato. Tale wrapper è complementare al driver open source Psycopg e ne estende le funzionalità. Per ulteriori informazioni, consulta Connessione ai cluster Aurora DB con i driver. AWS	23 maggio 2024

Integrazione Zero-ETL disponibile nelle Regioni della Cina	Le integrazioni zero-ETL sono ora disponibili in Cina (Pechino) e Regioni AWS Cina (Ningxia). Per ulteriori informazioni, consulta Zero-ETL integrations with Amazon Redshift.	21 maggio 2024
Server proxy per RDS è disponibile in altre Regioni	Server proxy per RDS è ora disponibile nelle Regioni Asia Pacifico (Hyderabad), Asia Pacifico (Melbourne), Medio Oriente (UAE), Israele (Tel Aviv), Canada occidentale (Calgary) ed Europa (Zurigo). Per ulteriori informazioni su Server proxy per RDS, consulta Using Amazon RDS Proxy.	21 maggio 2024
Amazon RDS supporta l'accesso granulare per Approfondimenti sulle prestazioni	Ora è possibile consentire o negare l'accesso alle singole dimensioni in Approfondimenti sulle prestazioni. Questo accesso granulare può essere utilizzato per le azioni `GetResourceMetrics`, `DescribeDimensionKeys` e `GetDimensionKeyDetails`. Per ulteriori informazioni, consulta https://docs.aws.amazon.com/AmazonRDS/latest/UserGuide/USER_PerfInsights.access-control.dimensionAccess-policy.html.	21 maggio 2024

Estensione del supporto per Amazon RDS	La creazione o il ripristino di un database Aurora MySQL versione 2 o 3 o Aurora PostgreSQL versione 11 ora registra automaticamente tale database nel supporto esteso di Amazon RDS per consentire alle applicazioni esistenti di continuare a funzionare così come sono. È possibile rifiutare il supporto esteso RDS per evitare addebiti dopo la data di fine del supporto standard di Aurora per il motore di database. Per ulteriori informazioni, consulta Utilizzo dell'estensione del supporto per Amazon RDS.	21 marzo 2024
Filtro dei dati per le integrazioni Zero-ETL	Amazon RDS supporta il filtro dei dati a livello di database e tabella per l'integrazione Zero-ETL con Amazon Redshift. Per ulteriori informazioni, consulta Data filtering for Aurora zero-ETL integrations with Amazon Redshift.	20 marzo 2024
Integrazioni di Aurora MySQL con Amazon Bedrock	Ora è possibile integrare i database Amazon Aurora MySQL con Amazon Bedrock per supportare applicazioni di IA generativa. Per ulteriori informazioni, consulta Using Amazon Aurora machine learning with Aurora MySQL.	8 marzo 2024

AWS Nuova politica gestita	Amazon RDS ha aggiunto una nuova policy gestita denominata `AmazonRDSCustomInstanceProfileRolePolicy` per consentire a RDS Custom di eseguire azioni di automazione e attività di gestione del database tramite un profilo di EC2 istanza. Per ulteriori informazioni, consulta Aggiornamenti di Amazon RDS sulle policy gestite da AWS.	27 febbraio 2024
Supporto Amazon RDS per la Gestione dei segreti AWS regione di Israele (Tel Aviv)	Amazon RDS supporta Secrets Manager nella Regione Israele (Tel Aviv). Per ulteriori informazioni, consulta Password management with Amazon RDS and Gestione dei segreti AWS (Gestione delle password per Amazon RDS e Gestione dei segreti AWS).	21 febbraio 2024

Estensione del supporto per Amazon RDS	Amazon RDS ora abilita automaticamente il supporto esteso Amazon RDS quando le versioni principali del motore Aurora MySQL e Aurora PostgreSQL nei cluster di database e nei cluster globali raggiungono la data di fine del supporto standard Aurora. Per ulteriori informazioni, consulta Utilizzo dell'estensione del supporto per Amazon RDS.	15 febbraio 2024
Aurora PostgreSQL 16.1 supporta Babelfish per Aurora PostgreSQL 4.0.0	Aurora PostgreSQL 16.1 supporta Babelfish 4.0.0. Per un elenco delle nuove funzionalità, consulta 16.1. Per un elenco delle caratteristiche supportate in ogni versione di Babelfish, consulta Funzionalità supportate in Babelfish per versione. Per le informazioni di utilizzo, consulta Lavorare con Babelfish per Aurora PostgreSQL.	31 gennaio 2024
Aggiornamento al certificato CA predefinito	Il certificato CA predefinito è impostato su `rds-ca-rsa2048-g1` . Per ulteriori informazioni, consulta Utilizzare SSL/TLS per crittografare una connessione a un cluster DB.	26 gennaio 2024

Server proxy per RDS è disponibile nella Regione Europa (Spagna)	Server proxy per RDS è ora disponibile nella Regione Europa (Spagna). Per ulteriori informazioni su Server proxy per RDS, consulta Using Amazon RDS Proxy.	8 gennaio 2024
API di dati RDS con Aurora PostgreSQL Serverless v2 e istanze allocate	Ora è possibile utilizzare l'API di dati RDS con Aurora PostgreSQL Serverless v2 e cluster di database con istanze allocate. Con l'API di dati RDS è possibile accedere ai cluster Aurora tramite un endpoint HTTP sicuro ed eseguire istruzioni SQL senza utilizzar e driver di database o gestire connessioni. Per ulteriori informazioni, consulta Using RDS Data API.	21 dicembre 2023
Integrazioni di Aurora PostgreSQL con Amazon Bedrock	Ora è possibile integrare i database Amazon Aurora PostgreSQL con Amazon Bedrock per supportare applicazioni di IA generativ a. Per ulteriori informazi oni, consulta Using Amazon Aurora machine learning with Aurora PostgreSQL.	21 dicembre 2023
Amazon Aurora è disponibi le nella Regione Canada occidentale (Calgary)	Amazon Aurora è ora disponibi le nella Regione Canada occidentale (Calgary). Per ulteriori informazioni, consulta Regioni e zone di disponibilità.	20 dicembre 2023

Amazon RDS supporta la visualizzazione e la risposta ai suggerimenti	I suggerimenti di Amazon Aurora ora includono consigli proattivi basati su soglie e consigli reattivi basati su machine learning. Per ulteriori informazioni, consulta Recommendations for Amazon Aurora.	19 dicembre 2023
Integrazione Zero-ETL di Aurora PostgreSQL con Amazon Redshift (anteprima)	È possibile creare integrazioni Zero-ETL con Amazon Redshift utilizzando un cluster di database di origine Aurora PostgreSQL. Per la versione di anteprima, devi creare tutte le integrazioni nell'ambiente di anteprima del database Amazon RDS, negli Stati Uniti orientali (Ohio) (us-east-2). Regione AWS Per ulteriori informazioni, consulta Utilizzo delle integrazioni Zero-ETL di Aurora con Amazon Redshift.	28 novembre 2023
Amazon Aurora PostgreSQL supporta l'inoltro di scrittura in un database globale	È ora possibile abilitare l'inoltro di scrittura su cluster secondari in un database globale basato su Aurora PostgreSQL. Per ulteriori informazioni, consulta Utilizzo dell'inoltro di scrittura in un database globale Aurora PostgreSQL.	9 novembre 2023

Supporto Aurora PostgreSQL per letture ottimizzate	Con Letture ottimizzate per Aurora è possibile velocizzare l'elaborazione delle query per Aurora PostgreSQL. Per ulteriori informazioni, consulta Prestazioni delle query migliorate per Aurora PostgreSQL con Letture ottimizzate per Aurora.	8 novembre 2023
Amazon RDS esporta i parametri di Performance Insights in Amazon CloudWatch	Performance Insights ti consente di esportare i dashboard delle metriche preconfigurati o personalizzati su Amazon. CloudWatch I dashboard delle metriche esportate possono essere visualizzati nella console. CloudWatch Puoi anche esportare un widget metrico di Performance Insights selezionato e visualizzare i dati delle metriche nella CloudWatch console. Per ulteriori informazioni, consulta Esportazione delle metriche di Performance Insights in. CloudWatch	8 novembre 2023

Integrazioni Zero-ETL di Aurora MySQL con Amazon Redshift disponibili a livello generale	Le integrazioni Zero-ETL con Amazon Redshift sono ora disponibili a livello generale per Aurora MySQL. Per ulteriori informazioni, consulta Utilizzo delle integrazioni Zero-ETL di Aurora con Amazon Redshift.	7 novembre 2023
Supporto Aurora PostgreSQL per implementazioni RDS Blue/Green	Ora puoi creare una blue/green distribuzione da un cluster Aurora PostgreSQL DB. Per ulteriori informazioni, consulta Using Amazon RDS Blue/Green Deployments per gli aggiornamenti del database.	26 ottobre 2023
Aurora MySQL supporta la crittografia lato server con (SSE-KMS) AWS KMS keys	In Aurora MySQL versione 3.05 e successive, puoi utilizzare SSE-KMS, incluse Chiavi gestite da AWS le chiavi gestite dai clienti, per la crittografia lato server dei dati caricati o salvati su Amazon S3. Per ulteriori informazioni, consulta Caricamento dei dati in un cluster DB Amazon Aurora MySQL da file di testo in un bucket Amazon S3 e Salvataggio dei dati da un cluster DB Amazon Aurora MySQL nei file di testo in un bucket Amazon S3.	25 ottobre 2023

Le ottimizzazioni di Aurora MySQL riducono il tempo di riavvio del database	In Aurora MySQL 3.05 e versioni successive abbiamo introdotto ottimizzazioni che riducono il tempo di riavvio del database. Queste ottimizzazioni consentono di ridurre fino al 65% i tempi di inattività rispetto alle ottimizzazioni precedenti e di ridurre le interruzioni dei carichi di lavoro del database dopo un riavvio. Per ulteriori informazioni, consulta Ottimizzazioni per ridurre i tempi di riavvio del database.	25 ottobre 2023
AWS Aggiornamento alle politiche gestite	Le policy gestite `AmazonRDSPerformanceInsightsReadOnly` e `AmazonRDSPerformanceInsightsFullAccess` ora includono `Sid` (ID istruzione) come identificativo nell'istruzione della policy. Per ulteriori informazioni, consulta Aggiornamenti di Amazon RDS sulle policy gestite da AWS.	23 ottobre 2023

Amazon RDS pubblica i parametri dei contatori di Performance Insights su Amazon CloudWatch	La funzione matematica dei parametri DB_PERF_INSIGHTS nella console CloudWatch consente di interrogare i contatori di Amazon RDS for Performance Insights. Per ulteriori informazioni, consulta Creazione di CloudWatch allarmi per monitorare Amazon Aurora.	20 settembre 2023
Amazon Aurora supporta il point-in-time ripristino (PITR) con AWS Backup	Ora puoi gestire i backup automatici (continui) di Aurora AWS Backup e ripristinarli a orari specifici da essi. Per informazioni, consulta Ripristino di un cluster di database a un determinato momento (AWS Backup).	7 settembre 2023
Estensione del supporto per Amazon RDS	Amazon Aurora annuncia la possibilità a breve di continuare a eseguire le versioni principali del motore Aurora per MySQL e Aurora per PostgreSQL nelle istanze database oltre la data di fine del supporto standard Aurora. Per ulteriori informazioni, consulta Utilizzo dell'estensione del supporto per Amazon RDS.	1 settembre 2023

Amazon Aurora MySQL estende il supporto per Percona XtraBackup	Ora puoi eseguire migrazioni fisiche dei database MySQL 8.0 verso i cluster database Aurora MySQL versione 3. Per ulteriori informazioni, consulta Migrazione fisica da MySQL utilizzando Percona XtraBackup e Amazon S3.	24 agosto 2023
Il database globale Aurora ora supporta il failover globale del database	Il database globale Aurora ora supporta il failover globale gestito, che consente di semplificare il ripristino in seguito a un reale guasto a livello regionale o a un'interruzione completa del livello di servizio. Per ulteriori informazioni su questa funzionalità, consulta Esecuzione di failover gestiti per database globali Amazon Aurora. La funzionalità precedentemente denominata "failover pianificato gestito" è ora denominata "switchover". Per informazioni sugli switchover, consulta Performing switchovers for Amazon Aurora global databases (Esecuzione di switchover per database globali Amazon Aurora).	21 agosto 2023

Aggiornamento delle autorizzazioni delle policy gestite AWS	La policy gestita `AmazonRDSFullAccess` dispone di nuove autorizzazioni che consentono di generare, visualizzare ed eliminare il report di analisi delle prestazioni per un periodo di tempo. Per ulteriori informazioni, consulta gli aggiornamenti di Amazon RDS alle policy AWS gestite.	17 agosto 2023
Aggiornamento delle autorizzazioni delle policy AWS gestite	L'aggiunta di nuove autorizzazioni alla policy gestita `AmazonRDSPerformanceInsightsReadOnly` e l'aggiunta di una nuova policy gestita `AmazonRDSPerformanceInsightsFullAccess` consente di generare un report di analisi del carico del database per un periodo di tempo. Per ulteriori informazioni, consulta gli aggiornamenti di Amazon RDS alle policy AWS gestite.	16 agosto 2023

Amazon RDS supporta l'analisi del periodo di caricamento del database con Approfondimenti sulle prestazioni	Performance Insights consente di creare report di analisi delle prestazioni per un periodo di tempo specifico. Il report fornisce gli approfondimenti identificati e i suggerimenti per risolvere problemi relativi alle prestazioni. Per ulteriori informazioni, consulta Analisi del carico del database per un periodo di tempo.	16 agosto 2023
Amazon Aurora supporta il mantenimento dei backup automatici per i cluster database	È ora possibile mantenere i backup automatici per i cluster Aurora eliminati e ripristinarli in un momento specifico. Per ulteriori informazioni, consulta Mantenimento dei backup automatici.	4 agosto 2023
Amazon Aurora è disponibile nella regione Israele (Tel Aviv)	Amazon Aurora è ora disponibile nella regione Israele (Tel Aviv). Per ulteriori informazioni, consulta Regioni e zone di disponibilità.	1° agosto 2023

Amazon Aurora MySQL supporta l'inoltro di scrittura locale (in un cluster)	È ora possibile inoltrare operazioni di scrittura da un'istanza database di lettura a un'istanza database di scrittura all'interno di un cluster database Aurora MySQL. Per ulteriori informazioni, consulta la sezione relativa all'utilizzo dell'inoltro di scrittura locale in un cluster database Amazon Aurora MySQL.	31 luglio 2023
Amazon Aurora supporta Aurora Serverless v2 in un altro Regione AWS	Ora puoi creare cluster Aurora Serverless v2 DB nella regione Asia-Pacifico (Melbourne). Regione AWS Per ulteriori informazioni su Aurora Serverless v2, consultar e Utilizzo di Aurora Serverless v2.	28 giugno 2023
Amazon Aurora introduce integrazioni Zero-ETL con Amazon Redshift (anteprima)	Le integrazioni Zero-ETL forniscono una soluzione completamente gestita per rendere disponibili i dati transazionali in Amazon Redshift in pochi secondi dalla loro scrittura su un cluster database Aurora MySQL. Per ulteriori informazioni, consulta Utilizzo delle integrazioni Zero-ETL di Aurora con Amazon Redshift.	28 giugno 2023

Amazon RDS offre una visualizzazione combinata di Performance Insights e CloudWatch metriche nella dashboard di Performance Insights.	Amazon RDS ora offre una visualizzazione consolidata di Performance Insights e delle CloudWatch metriche nella dashboard di Performance Insights. Per ulteriori informazioni, consulta Viewing combined metrics with the Performance Insights dashboard.	24 maggio 2023
Amazon Aurora supporta le classi di istanza db.r7g	Ora è possibile utilizzare le classi di istanza db.r7g per creare cluster database Aurora. Per ulteriori informazioni, consulta l'argomento relativo alle classi di istanza database Aurora.	11 maggio 2023
Amazon Aurora supporta una nuova configurazione dell'architettura dei cluster database	ConAurora I/O-Optimized, paghi solo per l'utilizzo e lo storage dei tuoi cluster DB, senza costi aggiuntivi per le operazioni di lettura e scrittura. I/O Per ulteriori informazioni, consultare Configurazioni dell'archiviazione per i cluster database Amazon Aurora.	11 maggio 2023

Amazon Aurora supporta Aurora Serverless v2 in aggiunta Regioni AWS	È ora possibile creare cluster Aurora Serverless v2 DB nei seguenti paesi Regioni AWS: Asia Pacifico (Hyderabad), Europa (Spagna) Europa (Zurigo) e Medio Oriente (Emirati Arabi Uniti). Per ulteriori informazioni su Aurora Serverless v2, consultar e Utilizzo di Aurora Serverless v2.	4 maggio 2023
Aurora Serverless v1 supporta la conversione in allocato	Puoi convertire un cluster di database Aurora Serverless v1 direttamente in un cluster di database allocato. Per ulteriori informazioni, consulta Conversione di un cluster di database Aurora Serverless v1 in allocato.	27 aprile 2023
Aurora Serverless v1 supporta Amazon Aurora PostgreSQL versione 13	Ora puoi creare cluster di database Aurora Serverles s v1 che eseguono Aurora PostgreSQL versione 13. Per ulteriori informazioni, consulta Aurora Serverless v1.	27 aprile 2023

Supporto per Amazon Aurora Gestione dei segreti AWS nelle regioni della Cina	Amazon Aurora supporta Secrets Manager nelle regioni Cina (Pechino) e Cina (Ningxia). Per ulteriori informazioni, consulta Password management with Amazon Aurora and Gestione dei segreti AWS (Gestione delle password per Amazon Aurora e Gestione dei segreti AWS).	20 aprile 2023
Amazon Aurora supporta la pubblicazione di eventi contenenti tag per gli abbonati degli argomenti	Le notifiche di eventi di Amazon Aurora inviate ad Amazon Simple Notification Service (Amazon SNS) o EventBridge Amazon ora contengono tag di evento nel corpo del messaggio. Questi tag forniscono i dati sulle risorse interessati dall'evento del servizio. Per ulteriori informazioni, consulta Tag e attributi delle notifiche eventi di Amazon RDS.	17 aprile 2023

Aggiornamento alle autorizzazioni del ruolo collegato ai servizi di IAM	Le `AmazonRDSReadOnlyAccess` policy `AmazonRDSFullAccess` and ora concedono autorizzazioni aggiuntive per consentire la visualizzazione dei risultati di Amazon DevOps Guru nella console RDS. Per ulteriori informazioni, consulta gli aggiornamenti di Amazon RDS alle policy AWS gestite.	30 marzo 2023
Amazon Aurora supporta i database globali nella Regione Asia Pacifico (Melbourne)	Ora puoi creare i database globali Aurora nella Regione Asia Pacifico (Melbourne). Per informazioni sui database globali Aurora, consulta Utilizzo dei database globali Amazon Aurora.	22 marzo 2023
Aggiornamento delle autorizzazioni delle policy AWS gestite	Le `AmazonRDSReadOnlyAccess` politiche `AmazonRDSFullAccess` and ora concedono autorizzazioni aggiuntive ad Amazon. CloudWatch Per ulteriori informazioni, consulta gli aggiornamenti di Amazon RDS alle policy AWS gestite.	16 marzo 2023
RDS Proxy è disponibile nelle regioni della Cina	RDS Proxy è ora disponibile nelle regioni Cina (Pechino) e Cina (Ningxia). Per ulteriori informazioni su Server proxy per RDS, consulta Using Amazon RDS Proxy.	15 marzo 2023

Supporto Amazon Aurora per Aurora Serverless v2 nelle regioni della Cina	Aurora Serverless v2 è ora disponibile nelle regioni Cina (Pechino) e Cina (Ningxia) . Per ulteriori informazioni, consulta Aurora Serverless v2.	15 marzo 2023
RDS Proxy è disponibile nella regione Asia Pacific (Giacarta)	RDS Proxy è ora disponibi le nella regione Asia Pacific (Giacarta). Per ulteriori informazioni su Server proxy per RDS, consulta Using Amazon RDS Proxy.	8 marzo 2023
Amazon Aurora MySQL supporta l'autenticazione Kerberos	Puoi ora utilizzare l'autenti cazione Kerberos per autenticare gli utenti quando si connettono a cluster di database Aurora MySQL. Per ulteriori informazioni, consulta Utilizzo dell'autenticazione Kerberos per Aurora MySQL.	8 marzo 2023
Amazon Aurora supporta database globali in aggiunta Regioni AWS	Puoi creare i database globali nelle seguenti regioni: Africa (Città del Capo), Asia Pacifico (Hong Kong), Asia Pacifico (Hyderabad), Asia Pacifico (Giacarta), Europa (Milano), Europa (Spagna), Europa (Zurigo), Medio Oriente (Bahrein) e Medio Oriente (Emirati Arabi Uniti). Per informazioni sui database globali Aurora, consulta Utilizzo dei database globali Amazon Aurora.	6 marzo 2023

Amazon Aurora supporta la copia di snapshot di cluster DB in aggiunta Regioni AWS	Puoi creare gli snapshot di cluster di database nelle seguenti regioni: Africa (Città del Capo), Asia Pacifico (Hong Kong), Asia Pacifico (Hyderabad), Asia Pacifico (Giacarta), Asia Pacifico (Melbourne), Europa (Milano), Europa (Spagna), Europa (Zurigo), Medio Oriente (Bahrein) e Medio Oriente (Emirati Arabi Uniti). Per informazioni sulla copia degli snapshot di cluster di database, consulta Copia di snapshot di cluster di database.	6 marzo 2023
Amazon DevOps Guru for RDS supporta approfondimenti proattivi	Amazon DevOps Guru for RDS pubblica approfondimenti proattivi con consigli per aiutarti a risolvere i problemi nei database Aurora prima che si verifichino. Per ulteriori informazioni, consulta Come funziona Guru for RDS. DevOps	28 febbraio 2023
Amazon Aurora MySQL versione 1 è deprecato	Aurora MySQL versione 1 (compatibile con MySQL 5.6) è stato deprecato. Per ulteriori informazioni, consulta Per quanto tempo le versioni principali di Amazon Aurora rimangono disponibili.	28 febbraio 2023

Aurora Serverless v1 supporta l'impostazione della finestra di manutenzione del cluster di database	È ora possibile impostare la finestra di manutenzione per i cluster di database Aurora Serverless v1. Per ulteriori informazioni, consulta Impostazione della finestra di manutenzione preferita del cluster di database.	27 febbraio 2023
Amazon Aurora supporta i flussi di attività di database nelle regioni Asia Pacifico (Hyderabad), Europa (Spagna) e Medio Oriente (Emirati Arabi Uniti).	Per maggiori informazioni, consulta Flussi di dati del database.	27 gennaio 2023
Amazon Aurora è disponibile nella Regione Asia Pacifico (Melbourne)	Amazon Aurora è ora disponibile nella Regione Asia Pacifico (Melbourne). Per ulteriori informazioni, consulta Regioni e zone di disponibilità.	23 gennaio 2023
Specifica dell'autorità di certificazione (CA) durante la creazione di un cluster database	È ora possibile specificare quale CA utilizzare per il certificato del server del cluster database durante la creazione del cluster database. Per ulteriori informazioni, consulta Certificate authorities (Autorità di certificazione).	5 gennaio 2023

Supporto Aurora MySQL 3.* per il backtracking	Le versioni Aurora MySQL 3.* offrono ora un modo rapido per correggere gli errori degli utenti, come ad esempio il rilascio della tabella sbagliata o l'eliminazione della riga sbagliata. Backtrack permette di riportare il database a un punto nel tempo precedente senza la necessità di ripristinarlo da un backup, in pochi secondi anche per database di grandi dimensioni. Per ulteriori informazioni, consulta Backtrack di un cluster database Aurora.	4 gennaio 2023
Usa Amazon RDS Blue/Green Deployments, disponibile in modalità aggiuntiva Regioni AWS	La funzionalità Blue/Green Implementazioni è ora disponibile nelle regioni Cina (Pechino) e Cina (Ningxia). Per ulteriori informazioni, consulta Using Amazon RDS Blue/Green Deployments per gli aggiornamenti del database.	22 dicembre 2022
Aggiornamento alle autorizzazioni del ruolo collegato ai servizi di IAM	La RDSService RolePolicy politica di Amazon ora concede autorizzazioni aggiuntive a. Gestione dei segreti AWS Per ulteriori informazioni, consulta gli aggiornamenti di Amazon RDS alle policy AWS gestite.	22 dicembre 2022

Amazon Aurora si integra con la gestione delle password Gestione dei segreti AWS	Aurora può gestire la password dell'utente master per un cluster database in Secrets Manager. Per ulteriori informazioni, consulta Password management with Amazon Aurora and Gestione dei segreti AWS (Gestione delle password per Amazon Aurora e Gestione dei segreti AWS).	22 dicembre 2022
Amazon Aurora supporta Aurora Serverless v2 in aggiunta Regioni AWS	Aurora Serverless v2 è ora disponibile nelle regioni Africa (Città del Capo) e UE (Milano). Per ulteriori informazioni, consulta Aurora Serverless v2.	21 dicembre 2022
Aurora PostgreSQL supporta Server proxy per RDS con PostgreSQL 14	Ora è possibile creare un proxy RDS con un cluster database Aurora PostgreSQL 14. Per ulteriori informazioni sul proxy RDS, consulta Utilizzo di Server proxy per Amazon RDS.	13 dicembre 2022
Amazon Aurora ti avvisa delle anomalie recenti rilevate da Amazon Guru DevOps	La pagina dei dettagli del database della console avvisa delle anomalie correnti e delle anomalie delle ultime 24 ore. Per ulteriori informazioni, consulta Come funziona DevOps Guru for RDS.	13 dicembre 2022

Server proxy per Amazon RDS supporta i database globali	Ora puoi utilizzare Server proxy per RDS con i database globali Aurora. Per ulteriori informazioni, consulta Using RDS Proxy with Aurora global databases (Utilizzo di RDS Proxy con i database globali Aurora).	7 dicembre 2022
I cluster database Aurora PostgreSQL supportano Trusted Language Extensions per PostgreSQL	Trusted Language Extensions per PostgreSQL è un kit di sviluppo open source che ti consente di creare estensioni di PostgreSQL ad alte prestazioni ed eseguirle in modo sicuro sul tuo cluster database Aurora PostgreSQL. Per ulteriori informazioni, consulta Working with Trusted Language Extensions for PostgreSQL (Utilizzo di Trusted Language Extensions per PostgreSQL).	30 novembre 2022

Amazon GuardDuty RDS Protection monitora le minacce di accesso	Quando attivi la protezione GuardDuty RDS, GuardDuty consuma gli eventi di accesso RDS dai database Aurora, monitora questi eventi e li profila per potenziali minacce interne o attori esterni. Quando GuardDuty RDS Protection rileva una potenzial e minaccia, GuardDuty genera una nuova scoperta con dettagli sul database potenzial mente compromesso. Per ulteriori informazioni, consulta Monitoraggio delle minacce con GuardDuty RDS Protectio n.	30 novembre 2022
Usa Amazon RDS Blue/ Green Deployments per gli aggiornamenti del database	È possibile apportare modifiche a un cluster database in un ambiente di gestione temporanea e testarle senza influire sul cluster database di produzion e. Quando sei pronto, puoi promuovere l'ambiente di gestione temporanea nel nuovo ambiente di produzione, con tempi di inattività minimi. Per ulteriori informazioni, consulta Using Amazon RDS Blue/Green Deployments per gli aggiornamenti del database.	27 novembre 2022

Amazon Aurora è disponibile nel regione Asia Pacifico (Hyderabad)	Amazon Aurora è ora disponibile nella Regione Asia Pacifico (Hyderabad). Per ulteriori informazioni, consulta Regioni e zone di disponibilità.	22 novembre 2022
Amazon Aurora è disponibile nella regione Europa (Spagna)	Amazon Aurora è ora disponibile nella Regione Europa (Spagna). Per ulteriori informazioni, consulta Regioni e zone di disponibilità.	16 novembre 2022
Amazon Aurora è disponibile nella Regione Europa (Zurigo)	Amazon Aurora è ora disponibile nella regione Europa (Zurigo). Per ulteriori informazioni, consulta Regioni e zone di disponibilità.	9 novembre 2022
Amazon Aurora supporta l'esportazione di dati in Amazon S3 da cluster database	Ora puoi esportare i dati del cluster Aurora direttamente in S3, senza dover prima creare uno snapshot. Per ulteriori informazioni, consulta Exporting DB cluster data to Amazon S3 (Esportazione dei dati del cluster database in Amazon S3).	27 ottobre 2022

Amazon Aurora MySQL supporta esportazioni più veloci in Amazon S3	Ora puoi ottenere prestazioni fino a 10 volte più veloci per l'esportazione dei dati dello snapshot del cluster database in S3 per i cluster Aurora MySQL compatibili con MySQL 5.7 e 8.0. Per ulteriori informazioni, consulta Exporting DB cluster snapshot data to Amazon S3 (Esportazione dei dati dello snapshot del cluster database in Amazon S3).	20 ottobre 2022
Amazon Aurora supporta la configurazione automatica della connettività tra un cluster Aurora DB e un'istanza EC2	È possibile utilizzare il Console di gestione AWS per configurare la connettività tra un cluster Aurora DB esistente e un' EC2 istanza. Per ulteriori informazioni, vedere Connessione automatica di un' EC2 istanza e di un cluster Aurora DB.	14 ottobre 2022
AWS Driver JDBC per PostgreSQL generalmente disponibile	Il driver AWS JDBC per PostgreSQL è un driver client progettato per Aurora PostgreSQL. Il driver AWS JDBC per PostgreSQL è ora disponibile a livello generale. Per ulteriori informazioni, consulta Connessione con il driver AWS JDBC per PostgreSQL.	6 ottobre 2022

Amazon Aurora supporta l'aggiornamento locale di Aurora MySQL con compatibilità MySQL 5.7	È possibile eseguire un aggiornamento locale per modificare un cluster Aurora MySQL compatibile con MySQL 5.7 esistente in un cluster Aurora MySQL compatibile con MySQL 8.0. Per ulteriori informazioni, consulta Aggiornamento da Aurora MySQL 2.x a 3.x.	26 settembre 2022
Approfondimenti sulle prestazioni mostra le prime 25 query SQL	Nel pannello di controllo di Approfondimenti sulle prestazioni, la scheda Top SQL (Prime istruzioni SQL) mostra le 25 query SQL che contribuiscono di più al carico del database. Per ulteriori informazioni, consulta Panoramica della scheda Prime istruzioni SQL.	13 settembre 2022
Aurora MySQL supporta una nuova classe di istanza database	Ora puoi utilizzare la classe di istanza database db.r6i per i cluster database Aurora MySQL. Per ulteriori informazioni, consulta le classi di istanza database.	13 settembre 2022
Amazon Aurora è disponibile nella Regione Medio Oriente (Emirati Arabi Uniti)	Amazon Aurora è ora disponibile nella Regione Medio Oriente (Emirati Arabi Uniti). Per ulteriori informazioni, consulta Regioni e zone di disponibilità.	30 agosto 2022

Amazon Aurora supporta la configurazione automatica della connettività con un'istanza EC2	Quando crei un cluster Aurora DB, puoi utilizzarlo Console di gestione AWS per configurare la connettività tra un'istanza Amazon Elastic Compute Cloud e il nuovo cluster DB. Per ulteriori informazioni, consulta Configurare la connettività di rete automatica con un' EC2 istanza.	22 agosto 2022
Amazon Aurora supporta la modalità dual-stack	Ora è possibile eseguire i cluster database in modalità dual-stack. In modalità dual-stack, le risorse possono comunicare con il cluster DB tramite o IPv4 entrambi IPv6. Per ulteriori informazioni, consulta Assegnazione di indirizzi IP con Amazon Aurora.	17 agosto 2022
Amazon Aurora supporta l'aggiornamento locale di Aurora Serverless v1 con compatibilità PostgreSQL	È possibile eseguire un aggiornamento locale per un cluster Aurora Serverless v1 compatibile con PostgreSQL 10 per modificare un cluster esistente in un cluster Aurora Serverless v1 compatibile con PostgreSQL 11. Per la procedura di aggiornamento locale, consulta Modifica di un cluster database Aurora Serverless v1.	8 agosto 2022

Approfondimenti sulle prestazioni supporta la Regione Asia Pacific (Giacarta)	In precedenza, non era possibile utilizzare Approfondimenti sulle prestazioni nella regione Asia Pacific (Giacarta). Questa limitazione è stata eliminata. Per ulteriori informazioni, consulta Regione AWS che supportano Approfondimenti sulle prestazioni	21 luglio 2022
Amazon Aurora supporta una nuova classe di istanza database	Ora è possibile creare cluster di database db.r6i per cluster di database Aurora PostgreSQL. Per ulteriori informazioni, consulta le classi di istanza database.	14 luglio 2022
RDS Approfondimenti sulle prestazioni supporta periodi di conservazione aggiuntivi	In precedenza, Approfondimenti sulle prestazioni offriva solo due periodi di conservazione: 7 giorni (impostazione predefinita) o 2 anni (731 giorni). Ora, se devi mantenere i dati sulle prestazioni per più di 7 giorni, puoi specificare da 1 a 24 mesi. Per ulteriori informazioni, consulta Prezzi e conservazione dei dati per Approfondimenti sulle prestazioni.	1 luglio 2022

Amazon Aurora supporta l'aggiornamento locale di Aurora Serverless v1 con compatibilità MySQL	È possibile eseguire un aggiornamento in loco per un cluster Aurora Serverless v1 compatibile con MySQL 5.6 per modificare un cluster esistente in un cluster Aurora Serverless v1 compatibile con MySQL 5.7. Per la procedura di aggiornamento locale, consulta Modifica di un cluster database Aurora Serverless v1.	16 giugno 2022
Aurora supporta l'attivazione di Amazon DevOps Guru nella console RDS	Puoi attivare la copertura DevOps Guru per i tuoi database Aurora direttamente dalla console RDS. Per ulteriori informazioni, consulta Configurazione di DevOps Guru for RDS.	9 giugno 2022
Amazon Aurora supporta la pubblicazione di eventi su argomenti Amazon SNS crittografati	Amazon Aurora può ora pubblicare eventi in Servizio di notifica semplice Amazon (Amazon SNS) con crittografia lato server (SSE) abilitata, per una protezione aggiuntiva degli eventi che contengono dati sensibili. Per ulteriori informazioni, consulta Sottoscrizione alle notifiche eventi di Amazon RDS.	1 giugno 2022

Amazon RDS pubblica i parametri di utilizzo su Amazon CloudWatch	Il `AWS/Usage` namespace in Amazon CloudWatch include parametri di utilizzo a livello di account per le quote dei servizi Amazon RDS. Per ulteriori informazioni, consulta i parametri di CloudWatch utilizzo di Amazon per Amazon Aurora.	28 aprile 2022
Set di risultati dell'API dati in formato JSON	Un parametro opzionale all'interno dellla funzione `ExecuteStatement` fa sì che il set di risultati della query venga restituito come stringa in formato JSON. Trasformare il set di risultati JSON in una struttura dati nel linguaggio della tua applicazione è semplice e pratico. Per ulteriori informazioni, consulta Elaborazione dei risultati delle query in formato JSON.	27 aprile 2022
Amazon Aurora Serverless v2 è ora disponibile a livello generale	Amazon Aurora Serverless v2 è ora disponibile per tutti gli utenti. Per ulteriori informazioni, consulta Utilizzo di Aurora Serverless v2.	21 aprile 2022

Aurora MySQL supporta suite di cifratura configurabili	Con Aurora MySQL, è possibile utilizzare le suite di cifratura per controllare la crittografia di connessione accettata dal server di database. Per ulteriori informazioni, consulta Configurazione delle suite di cifratura per le connessioni ai cluster database Aurora MySQL.	15 aprile 2022
Aurora PostgreSQL supporta RDS Proxy con PostgreSQL 13	Ora è possibile creare un proxy RDS con un cluster database Aurora PostgreSQL 13. Per ulteriori informazioni sul proxy RDS, consulta Amazon RDS Proxy.	4 aprile 2022
Note di rilascio di Aurora PostgreSQL	Ora è disponibile una guida separata per le note di rilascio di Amazon Aurora PostgreSQL. Per ulteriori informazioni, consultare Note di rilascio di Aurora PostgreSQL.	22 marzo 2022
Note di rilascio di Aurora MySQL	Ora è disponibile una guida separata per le note di rilascio di Amazon Aurora MySQL. Per ulteriori informazioni, consultare Note di rilascio di Aurora MySQL.	22 marzo 2022

Aurora PostgreSQL supporta più aggiornamenti delle versioni principali	Ora è possibile eseguire aggiornamenti di versione dei cluster di database Aurora PostgreSQL su più versioni principali. Per ulteriori informazioni, consultare Come eseguire l'aggiornamento a una versione principale.	4 marzo 2022
Aurora PostgreSQL supporta suite di cifratura configurabili	Con Aurora PostgreSQL versioni 11.8 e successive, ora è possibile utilizzare suite di cifratura configurabili per controllare la crittografia di connessione accettata dal server di database. Per ulteriori informazioni sull'utilizzo di suite di cifratura configurabili con Aurora PostgreSQL, consulta Configurazione di suite di cifratura per connessioni ai cluster di database Aurora PostgreSQL.	4 marzo 2022
AWS Driver JDBC per MySQL generalmente disponibile	Il driver AWS JDBC per MySQL è un driver client progettato per l'elevata disponibilità di Aurora MySQL. Il driver AWS JDBC per MySQL è ora disponibile a livello generale. Per maggiori informazioni, consulta Connessione con il driver JDBC per MySQL di Amazon Web Services.	2 marzo 2022

Aurora PostgreSQL 13.5 supporta Babelfish per Aurora PostgreSQL 1.1.0	Aurora PostgreSQL 13.5 supporta Babelfish 1.1.0. Per un elenco delle nuove funzionalità, consulta13.5. Per un elenco delle caratteristiche supportate in ogni versione di Babelfish, consulta Funzionalità supportate in Babelfish per versione. Per ulteriori informazioni, consulta Lavorare con Babelfish per Aurora PostgreSQL.	28 febbraio 2022
Amazon Aurora supporta i flussi di attività di database nella regione Asia Pacifico (Jakarta)	Per ulteriori informazioni, consulta Supporto Regioni AWS per i flussi di attività del database.	16 febbraio 2022
Approfondimenti sulle prestazioni supporta nuove operazioni API	Performance Insights ora supporta le seguenti operazioni API: `GetResourceMetadata` , `ListAvailableResourceDimensions` e `ListAvailableResourceMetrics` . Per ulteriori informazioni, consulta Recupero dei parametri con l'API Performance Insights in questo manuale e la Documentazione di riferimento dell'API di Amazon RDS Performance Insights.	12 gennaio 2022

Amazon RDS Proxy supporta gli eventi	RDS Proxy ora genera eventi a cui puoi abbonarti e visualizzarli in CloudWatch Eventi o configurare per l'invio ad Amazon EventBridge. Per maggiori informazioni, consulta Utilizzo degli eventi RDS Proxy.	11 gennaio 2022
Proxy RDS disponibile in aggiunta Regioni AWS	RDS Proxy è ora disponibile nelle seguenti regioni: Africa (Città del Capo), Asia Pacifico (Hong Kong), Asia Pacifico (Osaka), Europa (Milano), Europa (Parigi), Europa (Stoccolma), Medio Oriente (Bahrein) e Sud America (San Paolo). Per ulteriori informazioni sul proxy RDS, consulta Amazon RDS Proxy.	5 gennaio 2022
Amazon Aurora è disponibile nella Regione Asia Pacific (Giacarta)	Amazon Aurora è ora disponibile nella Regione Asia Pacific (Giacarta). Per ulteriori informazioni, consultare Regioni e zone di disponibilità.	13 dicembre 2021

DevOpsGuru for Amazon RDS fornisce informazioni dettagliate e consigli per Amazon Aurora	DevOpsGuru for RDS utilizza Performance Insights per i dati relativi alle prestazioni. Utilizzando questi dati, il servizio analizza le prestazioni delle istanze database Amazon Aurora e può aiutare a risolvere i problemi di prestazioni. Per ulteriori informazioni, consulta Analyzing performance anomalies with DevOps Guru for RDS in questa guida e vedi Overview of DevOps Guru for RDS nella Amazon Guru User Guide. DevOps	1° dicembre 2021
Aurora PostgreSQL supporta RDS Proxy con PostgreSQL 12	Ora è possibile creare un proxy RDS con un cluster di database Aurora PostgreSQL 12. Per ulteriori informazioni sul proxy RDS, consulta Amazon RDS Proxy.	22 novembre 2021
Aurora supporta le classi di istanze AWS Graviton2 per Database Activity Streams	È possibile utilizzare flussi di attività del database con la classe di istanza db.r6g per Aurora MySQL e Aurora PostgreSQL. Per ulteriori informazioni, consultare Motori di database supportati per tutte le classi di istanza database disponibili.	3 novembre 2021

Supporto Amazon Aurora per più account AWS KMS keys	È possibile utilizzare una chiave KMS di un'altra chiave Account AWS per la crittografia durante l'esportazione di snapshot DB su Amazon S3. Per ulteriori informazioni, consulta Esportazione dei dati dello snapshot DB su Simple Storage Service (Amazon S3).	3 novembre 2021
Amazon Aurora supporta Babelfish per Aurora PostgreSQL	Babelfish per Aurora PostgreSQL estende l'Edizione compatibile con PostgreSQL database Amazon Aurora con la possibilità di accettare connessioni al database dai client Microsoft SQL Server. Per ulteriori informazioni, consulta Lavorare con Babelfish per Aurora PostgreSQL.	28 ottobre 2021
Aurora Serverless v1 può richiedere SSL per le connessioni	I parametri del cluster Aurora `force_ssl` per PostgreSQL e `require_secure_transport` per MySQL ora sono supportati per Aurora Serverless v1. Per ulteriori informazioni, consulta Using with v1. TLS/SSL Aurora Serverless	26 ottobre 2021

Amazon Aurora supporta Performance Insights in aggiunta Regioni AWS	Approfondimenti sulle prestazioni è disponibile nelle regioni Medio Oriente (Bahrein), Africa (Città del Capo), Europa (Milano) e Asia Pacifico (Osaka). Per ulteriori informazioni, consulta Supporto delle regioni Regione AWS per Approfondimenti sulle prestazioni.	5 ottobre 2021
Timeout di scalabilità automatica configurabile per Aurora Serverless v1	È possibile scegliere per quanto tempo Aurora Serverless v1 deve attendere per trovare un punto di scalabilità automatica. Se durante tale periodo non viene rilevato alcun punto di scalabilità automatica, Aurora Serverless v1 annulla l'evento di ridimensionamento o forza la modifica della capacità, a seconda dell'azione di timeout selezionata. Per ulteriori informazioni, consultare Scalabilità automatica per Aurora Serverless v1.	10 settembre 2021

Aurora supporta le classi di istanza X2g e T4g	Sia Aurora MySQL che Aurora PostgreSQL possono ora utilizzare le classi di istanza X2g e T4g. Le classi di istanza che è possibile utilizzare dipendono dalla versione di Aurora MySQL o Aurora PostgreSQL. Per informazioni sui tipi di istanza supportati, consultare classi di istanza database.	10 settembre 2021
Amazon RDS supporta RDS Proxy in un VPC condiviso	Ora è possibile creare un proxy RDS in un virtual private cloud (VPC) condiviso. Per maggiori informazioni su Server proxy per RDS, consulta la sezione “Managing Connections with Amazon RDS Proxy” nella Guida per l’utente di Amazon RDS o nella Guida per l’utente di Aurora.	6 agosto 2021
Pagina delle policy della versione Aurora	La Guida per l'utente di Amazon Aurora ora include una sezione con informazioni generali sulle versioni di Aurora e sulle policy associate. Per informazioni dettagliate, consulta Versioni di Amazon Aurora.	14 luglio 2021

Escludi gli eventi Data API da un percorso AWS CloudTrail	Puoi escludere gli eventi Data API da un CloudTrail trail. Per ulteriori informazioni, consulta Escludere gli eventi Data API da un AWS CloudTrail trail.	2 luglio 2021
L'edizione compatibile con PostgreSQL di Amazon Aurora supporta estensioni aggiuntive	Le estensioni appena supportate includono pg_bigm, pg_cron, pg_partman e pg_proctab. Per ulteriori informazioni, consulta Versioni delle estensioni per l'edizione compatibile con PostgreSQL di Amazon Aurora.	17 giugno 2021
Clonazione per i cluster Aurora Serverless	È ora possibile creare cluster clonati che siano Aurora Serverless. Per ulteriori informazioni sulla clonazion e, consulta Clonazione di un volume per un cluster database Aurora.	16 giugno 2021
Disponibilità dei database globali di Aurora nelle regioni Cina (Pechino) e Cina (Ningxia)	Ora è possibile creare i database globali Aurora nelle regioni Cina (Pechino) e Cina (Ningxia). Per informazioni sui database globali Aurora, consultare Lavorare con database globali Amazon Aurora.	19 maggio 2021

Supporto FIPS 140-2 per API dati	L'API Data supporta la Federal Information Processing Standard Publication 140-2 (FIPS 140-2) per le connessioni. SSL/TLS Per ulteriori informazioni, consulta Disponibilità dell'API dati.	14 maggio 2021
AWS Driver JDBC per PostgreSQL (anteprima)	Il driver AWS JDBC per PostgreSQL, ora disponibile in anteprima, è un driver client progettato per l'elevata disponibilità di Aurora PostgreSQL. Per ulteriori informazioni, consulta Connessione con il driver JDBC di AWS per PostgreSQL (anteprima).	27 aprile 2021
L'API Data è disponibile in aggiunta Regioni AWS	L'API dati è ora disponibile nelle regioni Asia Pacifico (Seoul) e Canada (Centrale). Per ulteriori informazioni, consulta Disponibilità della Data API.	9 aprile 2021
Amazon Aurora supporta le classi di istanza database Graviton2	Ora è possibile utilizzare le classi di istanza database Graviton2 db.r6g.x per creare cluster di database che eseguono MySQL o PostgreSQL. Per ulteriori informazioni, consulta Tipi di classe di istanza database.	12 marzo 2021

Miglioramenti degli endpoint RDS Proxy	È possibile creare endpoint aggiuntivi associati a ciascun proxy RDS. La creazione di un endpoint in un VPC diverso consente l'accesso tra VPC per il proxy. I proxy per i cluster Aurora MySQL possono avere anche endpoint di sola lettura. Questi endpoint di lettura si connettono alle istanze database del lettore nei cluster e possono migliorare la scalabilità di lettura e la disponibilità per le applicazioni che richiedono un uso intensivo di query. Per maggiori informazioni su RDS Proxy, consulta la sezione "Gestione delle connessioni con Amazon RDS Proxy" nella Guida per l'utente di Amazon RDS o nella Guida per l'utente di Aurora.	8 marzo 2021
Amazon Aurora è disponibile nella Regione Asia Pacifico (Osaka-Locale)	Amazon Aurora è ora disponibile nella Regione Asia Pacifico (Osaka-Locale). Per ulteriori informazioni, consulta Regioni e zone di disponibilità.	1 marzo 2021

Aurora PostgreSQL supporta l'abilitazione dell'autenticazione IAM e Kerberos sullo stesso cluster di database	Aurora PostgreSQL ora supporta l'abilitazione sia dell'autenticazione IAM che dell'autenticazione Kerberos sullo stesso cluster di database. Per ulteriori informazioni, consulta Autenticazione del database con Amazon Aurora.	24 febbraio 2021
Il database globale di Aurora ora supporta il failover pianificato gestito	Il database globale di Aurora ora supporta il failover pianificato gestito, consentendo di modificare più facilmente la regione AWS principale del database globale Aurora. È possibile utilizzare il failover pianificato gestito solo con database globali Aurora integri. Per ulteriori informazioni, consulta Ripristino di emergenza e database globali Amazon Aurora. Per informazioni di riferimento, consulta `FailoverGlobalCluster` in Amazon RDS API Reference.	11 febbraio 2021

L'Data API per Aurora Serverless ora supporta più tipi di dati	L'API dati per Aurora Serverless ora ti permette di utilizzare i tipi di dati `UUID` e `JSON` come input per il database. Inoltre, con l'API dati per Aurora Serverless è possibile avere un valore di tipo `LONG` restituito dal database come valore `STRING`. Per ulteriori informazioni, consulta Chiamata della Data API. Per informazioni di riferimento sui tipi di dati supportati, consulta `SqlParameter` in Documentazione di riferimento delle API del servizio dati Amazon RDS.	2 febbraio 2021
Aurora PostgreSQL supporta gli aggiornamenti delle versioni principali a PostgreSQL 12	Con Aurora PostgreSQL, adesso puoi aggiornare il motore del database alla versione principale 12. Per ulteriori informazioni, consulta Aggiornamento del motore del database PostgreSQL per Aurora PostgreSQL.	28 gennaio 2021

Aurora MySQL supporta l'aggiornamento in loco	È possibile aggiornare il cluster Aurora MySQL 1.x a Aurora MySQL 2.x, preservando le istanze database, gli endpoint e così via del cluster originale. Questa tecnica di aggiornamento in loco evita l'inconveniente di configurare un cluster completamente nuovo ripristinando uno snapshot. Evita inoltre il sovraccarico dato dalla copia di tutti i dati della tabella in un nuovo cluster. Per ulteriori informazioni, consulta Aggiornamento della versione principale di un cluster database Aurora MySQL da 1.x a 2.x.	11 gennaio 2021
AWS Driver JDBC per MySQL (anteprima)	Il driver AWS JDBC per MySQL, ora disponibile in anteprima, è un driver client progettato per l'elevata disponibilità di Aurora MySQL. Per maggiori informazioni, consulta Connessione con il driver JDBC di Amazon Web Services per MySQL (anteprima).	7 gennaio 2021

Aurora supporta i flussi di attività del database su cluster secondari di un database globale	È possibile avviare un flusso di attività del database in un cluster primario o secondari o di Aurora PostgreSQL o Aurora MySQL. Per le versioni del motore supportat e, consulta Limitazioni dei database globali Aurora.	22 dicembre 2020
Cluster multi-master con 4 istanze database	Il numero massimo di istanze database in un cluster Aurora MySQL multi-master è ora quattro. In precedenz a, il massimo era di due istanze database. Per ulteriori informazioni, consulta Utilizzo di cluster multi-master Aurora.	17 dicembre 2020
Aurora PostgreSQL supporta funzioni AWS Lambda	Ora puoi richiamare la AWS Lambda funzione per i tuoi cluster Aurora PostgreSQ L DB. Per ulteriori informazi oni, consulta Richiamo di una funzione Lambda da un cluster database Aurora PostgreSQL.	11 dicembre 2020
Amazon Aurora supporta le classi di istanza database Graviton2 in anteprima	Ora è possibile utilizzare le classi di istanza database Graviton2 db.r6g.x in anteprima per creare cluster di database che eseguono MySQL o PostgreSQL. Per ulteriori informazioni, consulta la pagina relativa ai tipi di classe di istanza database.	11 dicembre 2020

Amazon Aurora Serverless v2 è ora disponibile in anteprima.	Amazon Aurora Serverless v2 è disponibile in anteprima. Per utilizzare Amazon Aurora Serverless v2, è necessari o richiedere l'accesso. Per ulteriori informazioni, consulta la pagina Aurora Serverless v2.	1 dicembre 2020

Aurora PostgreSQL è ora disponibile per Aurora Serverless in più Regioni AWS.	Aurora PostgreSQL è ora disponibile per Aurora Serverless in più Regioni AWS. Ora puoi scegliere di eseguire Aurora PostgreSQL Serverless v1 la stessa offerta. Regioni AWS Aurora MySQL Serverless v1 Il Aurora Serverless supporto include inoltre Stati Uniti occidentali (California settentrionale), Asia Pacifico (Singapore) Asia Pacifico (Sydney) Asia Pacifico (Seoul) Asia Pacifico (Mumbai) Canada (Europa centrale) (Londra) ed Europa (Parigi). Regioni AWS Per un elenco di tutte le Regioni e i motori di database Aurora supportati per Aurora Serverless, consulta Supported Regions and Aurora DB engines for Aurora Serverless v1. Anche l'API dati di Amazon RDS per Aurora Serverless è ora disponibile nelle Regioni AWS sopra menzionate. Per un elenco di tutte le Regioni che supportano l'API di dati per Aurora Serverless, consulta Data API with Aurora MySQL Serverless v1.	24 novembre 2020

Amazon RDS Approfondimenti sulle prestazioni introduce nuove dimensioni	È possibile raggruppare il carico del database in base ai gruppi di dimension i per database, applicazi one (PostgreSQL) e tipo di sessione (PostgreSQL). Amazon RDS supporta anche le dimensioni db.name, db.application.name (PostgreS QL) e db.session_type.name (PostgreSQL). Per ulteriori informazioni, consulta Tabelle dal carico superiore.	24 novembre 2020
Aurora Serverless supporta Aurora PostgreSQL versione 10.12	Aurora PostgreSQL per Aurora Serverless è stato aggiornat o ad Aurora PostgreSQL versione 10.12 in tutte le regioni AWS in cui è supportat o Aurora PostgreSQL per Aurora Serverless. Per ulteriori informazioni, consulta Supported Regions and Aurora DB engines for Aurora Serverless v1.	4 novembre 2020

La Data API ora supporta l'autorizzazione basata su tag	L'API dati supporta l'autorizzazione basata su tag. Se le risorse cluster RDS sono state etichettate con tag, è possibile utilizzare questi tag nelle istruzioni delle policy per controllare l'accesso tramite La Data API. Per ulteriori informazioni, consulta Autorizzare l'accesso all'API dati.	27 ottobre 2020
Amazon Aurora estende il supporto per l'esportazione di istantanee su Simple Storage Service (Amazon S3)	È ora possibile esportare i dati di snapshot DB in Simple Storage Service (Amazon S3) in tutte le Regioni AWS commerciali. Per ulteriori informazioni, consulta Esportazione dei dati dello snapshot DB su Simple Storage Service (Amazon S3).	22 ottobre 2020

Il database globale Aurora supporta la clonazione	Ora è possibile creare cloni dei cluster di database primari e secondari dei database globali Aurora. Puoi farlo utilizzando Console di gestione AWS e scegliendo l'opzione di menu Crea clone. È inoltre possibile utilizzare AWS CLI ed eseguire il `restore-db-cluster-to-point-in-time` comando con l'`--restore-type copy-on-write` opzione. Utilizzando Console di gestione AWS o the AWS CLI, puoi anche clonare i cluster DB dai tuoi database globali Aurora tra più account. AWS Per ulteriori informazioni sulla clonazione, consulta Clonazione di un volume di cluster di database Aurora.	19 ottobre 2020

Amazon Aurora supporta il ridimensionamento dinamico per il volume del cluster	A partire da Aurora MySQL 1.23 e 2.09 e da Aurora PostgreSQL 3.3.0 e Aurora PostgreSQL 2.6.0, Aurora riduce la dimensione del volume del cluster dopo la rimozione dei dati tramite operazioni quali `DROP TABLE`. Per sfruttare questo miglioramento, esegui l'aggiornamento a una delle versioni appropriate a seconda del motore del database utilizzato dal cluster. Per informazioni su questa funzionalità e su come controllare lo spazio di storage utilizzato e disponibile per un cluster Aurora, consultare Gestione delle prestazioni e del dimensionamento dei cluster di database Aurora.	13 ottobre 2020
Amazon Aurora supporta dimensioni di volume fino a 128 TiB	I volumi dei cluster Aurora nuovi ed esistenti possono ora crescere fino a raggiungere la dimensione massima di 128 tebibytes (TiB). Per ulteriori informazioni, consulta Modalità di crescita dello storage Aurora.	22 settembre 2020

Aurora PostgreSQL supporta le classi di istanza database db.r5 e db.t3 nella regione Cina (Ningxia)	È ora possibile creare cluster di database Aurora PostgreSQL nella regione Cina (Ningxia) che utilizza le classi di istanza database db.r5 e db.t3. Per ulteriori informazioni, consulta Classi di istanza database.	3 settembre 2020

Miglioramenti delle query parallele in Aurora	A partire da Aurora MySQL 2.09 e 1.23, è possibile sfruttare i miglioramenti apportati alla funzionalità di query parallela. La creazione di un cluster di query parallele non richiede più una modalità motore speciale. È ora possibile attivare e disattiva re la query parallela utilizzan do l'opzione di configurazione `aurora_parallel_qu ery` per qualsiasi cluster di cui è in esecuzione una versione Aurora MySQL compatibile. È possibile aggiornare un cluster esistente a una versione Aurora MySQL compatibile e utilizzare una query parallela, anziché creare un nuovo cluster e importare dati in esso. È possibile utilizzare Performance Insights per cluster di query parallele. È possibile arrestare e avviare cluster di query parallele. È possibile creare cluster di query Aurora parallele compatibili con MySQL 5.7. La query parallela funziona per le tabelle che utilizzano il formato di riga `DYNAMIC`. I cluster di query paralleli possono utilizzare l'autenti cazione AWS Identity and Access Management (IAM).	2 settembre 2020

	Le istanze database di lettura in cluster di query parallele possono sfruttare il livello di isolamento `READ COMMITTED` . È inoltre possibile creare cluster di query parallele in Regioni AWS aggiuntive. Per ulteriori informazioni sulla funzionalità di query parallela e su questi miglioramenti, consulta Parallel query for Aurora MySQL.	
Modifiche al parametro Aurora MySQL binlog_rows_query_log_events	È ora possibile modificare il valore del parametro Aurora MySQL di configurazione `binlog_rows_query_log_events` . In precedenza, questo parametro non era modificabile.	26 agosto 2020

Supporto per gli aggiornamenti automatici delle versioni secondarie per Aurora MySQL	Con Aurora MySQL, l'impostazione Enable auto minor version upgrade (Abilita aggiornamento automatico della versione secondaria) ora ha effetto quando la specifici per un cluster database Aurora MySQL. Quando abiliti l'aggiornamento automatico della versione secondaria, Aurora esegue automaticamente l'aggiornamento alle nuove versioni secondarie non appena vengono rilasciate. Gli aggiornamenti automatici vengono eseguiti durante la finestra di manutenzione del database. Per Aurora MySQL, questa funzione si applica solo ad Aurora MySQL versione 2, che è compatibile con MySQL 5.7. Inizialmente, la procedura di aggiornamento automatico porta i cluster di database Aurora MySQL alla versione 2.07.2. Per ulteriori informazioni sul funzionamento di questa caratteristica con Aurora MySQL, consulta Aggiornamenti e patch del database per Amazon Aurora MySQL.	3 agosto 2020

Aurora PostgreSQL supporta gli aggiornamenti delle versioni principali a PostgreSQL versione 11	Con Aurora PostgreSQL, puoi ora aggiornare il motore del database alla versione principale 11. Per ulteriori informazioni, consulta Aggiornamento del motore del database PostgreSQL per Aurora PostgreSQL.	28 luglio 2020
Amazon Aurora supporta AWS PrivateLink	Amazon Aurora ora supporta la creazione di endpoint Amazon VPC per le chiamate API Amazon RDS per mantenere il traffico tra le applicazioni e Aurora nella rete. AWS Per ulteriori informazioni consulta Amazon Aurora ed endpoint VPC di interfaccia (AWS PrivateLink).	9 luglio 2020
Proxy RDS disponibile a livello generale	Proxy RDS è ora disponibile a livello generale. È possibile utilizzare RDS Proxy con RDS for MySQL, Aurora MySQL, RDS for PostgreSQL e Aurora PostgreSQL per carichi di lavoro di produzione. Per maggiori informazioni su RDS Proxy, consulta la sezione "Gestione delle connessioni con Amazon RDS Proxy" nella Guida per l'utente di Amazon RDS o nella Guida per l'utente di Aurora.	30 giugno 2020

Inoltro di scrittura del database globale Aurora	È ora possibile abilitare la funzionalità di scrittura su cluster secondari in un database globale. Con l'inoltro di scrittura, inserisci istruzioni DML in un cluster secondario, Aurora inoltra la scrittura al cluster primario e i dati aggiornati vengono replicati in tutti i cluster secondari. Per ulteriori informazioni, consulta Inoltro di scrittura secondario Regioni AWS con un database globale Aurora.	18 giugno 2020
Aurora supporta l'integrazione con AWS Backup	È possibile AWS Backup utilizzarlo per gestire i backup dei cluster Aurora DB. Per ulteriori informazioni, consulta Panoramica di backup e ripristino di un cluster di database Aurora.	10 giugno 2020
Aurora PostgreSQL supporta le classi di istanza database db.t3.large.	Ora è possibile creare cluster di database Aurora PostgreSQL basati su classi di istanza database db.t3.large. Per ulteriori informazioni, consulta Classi di istanza database	5 giugno 2020

Il database globale di Aurora supporta PostgreSQL versione 11.7 e l'obiettivo del punto di ripristino (RPO, Recovery Point Objective) gestito	Puoi ora creare database globali Aurora per il motore del database PostgreSQL versione 11.7. È inoltre possibile gestire il ripristino di un database globale PostgreSQL da un errore utilizzando un obiettivo del punto di ripristino (RPO). Per ulteriori informazioni, vedere Ripristino di emergenza tra regioni per i database globali Aurora.	4 giugno 2020
Aurora MySQL supporta il monitoraggio del database con flussi di attività di database	Aurora MySQL ora include flussi di attività del database, che forniscono un flusso di near-real-time dati dell'attività del database nel database relazionale. Per maggiori informazioni, consulta Utilizzo dei flussi di dati di attività del database.	2 giugno 2020
L'editor di query è disponibile in versione aggiuntiva Regioni AWS	L'editor di query per Aurora Serverless è ora disponibile in versione aggiuntiva. Regioni AWS Per ulteriori informazioni, consulta Disponibilità dell'editor di query.	28 maggio 2020
L'API Data è disponibile in aggiunta Regioni AWS	L'API Data è ora disponibile in modalità aggiuntiva Regioni AWS. Per ulteriori informazioni, consulta Disponibilità della Data API.	28 maggio 2020

Proxy RDS disponibile nella regione Canada (Centrale)	È ora possibile utilizzare l'anteprima Proxy RDS in Canada (Centrale). Per ulteriori informazioni sul proxy RDS, consulta Gestione delle connessioni con Amazon RDS Proxy (Anteprima).	28 maggio 2020
Aurora Global Database e repliche di lettura tra regioni	Per Aurora Global Database, non è possibile creare una replica di lettura tra regioni Aurora MySQL dal cluster primario nella stessa regione del cluster secondario. Per ulteriori informazioni su Aurora Global Database e repliche di lettura tra regioni, consulta Utilizzo di Amazon Aurora Global Database e Repliche di database Amazon Aurora MySQL.	18 maggio 2020
RDS Proxy è disponibile in altre versioni Regioni AWS	È ora possibile utilizzar e l'anteprima proxy RDS nelle regioni Stati Uniti occidentali (California settentri onale), Europa (Londra), Europa (Francoforte), Asia Pacifico (Seoul), Asia Pacifico (Mumbai), Asia Pacifico (Singapore) e Asia Pacifico (Sydney). Per ulteriori informazioni sul proxy RDS, consulta Gestione delle connessioni con Amazon RDS Proxy (Anteprima).	13 maggio 2020

Aurora edizione compatibile con PostgreSQL supporta Microsoft Active Directory locale o autogestito	È ora possibile utilizzare una Active Directory locale o autogestita per l'autenticazione Kerberos di utenti quando si connettono ai cluster di database Aurora PostgreSQL. Per ulteriori informazioni, consulta Utilizzo dell'autenticazione Kerberos con Aurora PostgreSQL.	7 maggio 2020
Cluster multi-master Aurora MySQL disponibili in più Regioni AWS	È ora possibile creare cluster multi-master Aurora nelle regioni Asia Pacifico (Seoul), Asia Pacifico (Tokyo), Asia Pacifico (Mumbai) ed Europa (Francoforte). Per ulteriori informazioni sui cluster multi-master, consulta Utilizzo di cluster multi-master Aurora.	7 maggio 2020
Approfondimenti sulle prestazioni supporta l'analisi delle statistiche di esecuzione di query Aurora MySQL	È ora possibile analizzare le statistiche delle query in esecuzione con Performance Insights per le istanze database Aurora MySQL. Per ulteriori informazioni, consulta Analisi delle statistiche delle query in esecuzione.	5 maggio 2020

Libreria client Java per l'API dati disponibile a livello generale	La libreria client Java per Data API è ora disponibile a livello generale. È possibile scaricare e utilizzare una libreria client Java per La Data API. Questa consente di mappare le classi sul lato client alle richieste e alle risposte della Data API. Per ulteriori informazioni, consulta Utilizzo dell'API dati per Aurora Serverless.	30 aprile 2020
Amazon Aurora è disponibile nella Regione Europa (Milano)	Amazon Aurora è ora disponibile nella Regione Europa (Milano). Per ulteriori informazioni, consulta Regioni e zone di disponibilità.	28 aprile 2020
Amazon Aurora è disponibile nella Regione Europa (Milano)	Amazon Aurora è ora disponibile nella Regione Europa (Milano). Per ulteriori informazioni, consulta Regioni e zone di disponibilità.	27 aprile 2020
Amazon Aurora è disponibile nella Regione Africa (Città del Capo)	Amazon Aurora è ora disponibile nella Regione Africa (Città del Capo). Per ulteriori informazioni, consulta Regioni e zone di disponibilità.	22 aprile 2020

Aurora PostgreSQL ora supporta le classi di istanza database db.r5.16xlarge e db.r5.8xlarge	È ora possibile creare cluster database Aurora PostgreSQL che eseguono PostgreSQL che utilizzano le classi di istanze database db.r5.16xlarge e db.r5.8xlarge. Per ulteriori informazioni, consulta Specifiche hardware per le classi di istanza database per Aurora.	8 aprile 2020
Proxy Amazon RDS per PostgreSQL	Server proxy per Amazon RDS è ora disponibile per PostgreSQL. È possibile utilizzare RDS Proxy per ridurre il sovraccarico di gestione delle connessioni sul cluster e anche la possibilità di errori di "troppe connessioni". RDS Proxy è attualmente in anteprima pubblica per PostgreSQL. Per ulteriori informazioni, consulta Gestione delle connessioni con il proxy Amazon RDS (Anteprima).	8 aprile 2020

I database globali Aurora ora supportano Aurora PostgreSQL	Puoi ora creare database globali Aurora per il motore del database PostgreSQL. Un database globale Aurora si estende su più pagine Regioni AWS, abilitando letture globali a bassa latenza e disaster recovery in caso di interruzioni a livello regionale. Per ulteriori informazioni, consulta Utilizzo di Amazon Aurora Global Database.	10 marzo 2020
Supporto per gli aggiornamenti di versione principali per Aurora PostgreSQL	Con Aurora PostgreSQL, puoi ora aggiornare il motore del database a una versione principale. In questo modo, puoi passare a una versione principale più recente quando aggiorni le versioni del motore PostgreSQL selezionate. Per ulteriori informazioni, consulta Aggiornamento del motore del database PostgreSQL per Aurora PostgreSQL.	4 marzo 2020
Aurora PostgreSQL supporta l'autenticazione Kerberos	Puoi ora utilizzare l'autenticazione Kerberos per autenticare gli utenti quando si connettono a cluster di database Aurora PostgreSQL. Per ulteriori informazioni, consulta Utilizzo dell'autenticazione Kerberos con Aurora PostgreSQL.	28 febbraio 2020

L'API Data supporta AWS PrivateLink	L'API Data ora supporta la creazione di endpoint Amazon VPC per chiamate Data API per mantenere il traffico tra le applicazioni e l'API Data nella rete. AWS Per ulteriori informazioni, consulta Creazione di un endpoint Amazon VPC (AWS PrivateLink) per l'API dati.	6 febbraio 2020
Supporto Machine Learning di Aurora in Aurora PostgreSQL	L'estensione `aws_ml` Aurora PostgreSQL fornisce funzioni da utilizzare nelle query del database per chiamare Amazon Comprehend per l'analisi dei sentimenti e l'intelligenza artificiale per eseguire i tuoi modelli di machine learning. SageMaker Per ulteriori informazioni, consulta Utilizzo delle funzionalità di Machine Learning (ML) con Amazon Aurora.	5 febbraio 2020
Aurora PostgreSQL supporta l'esportazione di dati in Simple Storage Service (Amazon S3)	Puoi eseguire una query sui dati di un cluster di database Aurora PostgreSQL ed esportarli direttamente in file archiviati in un bucket Simple Storage Service (Amazon S3). Per ulteriori informazioni, consulta Esportazione dei dati da un cluster Aurora PostgreSQL DB in Simple Storage Service (Amazon S3).	5 febbraio 2020

Supporto per l'esportazione di dati dello snapshot DB in Simple Storage Service (Amazon S3)	Amazon Aurora supporta l'esportazione di dati snapshot DB in Simple Storage Service (Amazon S3) per MySQL e PostgreSQL. Per ulteriori informazioni, consulta Esportazione dei dati dello snapshot DB su Simple Storage Service (Amazon S3).	9 gennaio 2020
Note di rilascio di Aurora MySQL in Cronologia documenti	Questa sezione include ora le voci della cronologia per le note di rilascio di Aurora edizione compatibile con MySQL per le versioni rilasciate dopo il 31 agosto 2018. Per le note di rilascio complete per una versione specifica, scegliere il collegamento nella prima colonna della voce della cronologia.	13 dicembre 2019

Proxy Amazon RDS	Puoi ridurre il sovraccarico della gestione delle connessioni nel cluster e ridurre la possibilità di errori del tipo «troppe connessioni» utilizzando il proxy Amazon RDS. Puoi associare ogni proxy a un'istanza RDS di DB Servizi di dominio Active Directory o cluster di DB Aurora. Quindi utilizzi l'endpoint proxy nella stringa di connessione per l'applicazione. Il proxy Amazon RDS è attualmente in uno stato di anteprima pubblica. Supporta il motore del database Aurora MySQL. Per ulteriori informazioni, consulta Gestione delle connessioni con il proxy Amazon RDS (Anteprima).	3 dicembre 2019
La Data API per Aurora Serverless v1 supporta suggerimenti per la mappatura dei tipi di dati	È ora possibile utilizzare un suggerimento per indicare alLa Data API per Aurora Serverless v1 di inviare un valore `String` al database come un tipo diverso. Per ulteriori informazioni, consulta Invocazione della Data API.	26 novembre 2019

La Data API di Aurora Serverless v1 supporta una libreria client Java (Anteprima)	È possibile scaricare e utilizzare una libreria client Java per La Data API. Questa consente di mappare le classi sul lato client alle richieste e alle risposte della Data API. Per ulteriori informazioni, consulta Utilizzo dell'API dati per Aurora Serverless.	26 novembre 2019
Aurora PostgreSQL è idoneo per FedRAMP HIGH	Aurora PostgreSQL è idoneo per FedRAMP HIGH. Per i dettagli AWS e gli sforzi di conformità, consulta AWS la sezione Servizi rientranti nell'ambito del programma di conformità.	26 novembre 2019

Livello di isolamento READ COMMITTED abilitato per le repliche Amazon Aurora MySQL	È ora possibile abilitare il livello di isolamento `READ COMMITTED` sulle repliche Aurora MySQL. Questa operazione richiede l'abilitazione delle impostazioni di configurazione `aurora_read_replica_read_committed_isolation_enabled` a livello di sessione. L'uso del livello di isolamento `READ COMMITTED` per query a esecuzione prolungata sui cluster OLTP può aiutare a risolvere i problemi relativi alla lunghezza dell'elenco della cronologia. Prima di abilitare questa impostazione, accertarsi di comprendere la differenza del comportamento di isolamento sulle repliche Aurora si differenzia dalla normale implementazione MySQL di `READ COMMITTED`. Per ulteriori informazioni, consulta Livelli di isolamento di Aurora MySQL.	25 novembre 2019

Approfondimenti sulle prestazioni supporta l'analisi delle statistiche di esecuzione di query Aurora PostgreSQL	È ora possibile analizzare le statistiche delle query in esecuzione con Performance Insights per le istanze database PostgreSQL per Aurora. Per ulteriori informazioni, consulta Analisi delle statistiche delle query in esecuzione.	25 novembre 2019
Ulteriori cluster in un database globale Aurora	È possibile aggiungere più regioni secondarie a un database globale Aurora. È possibile sfruttare le letture globali a bassa latenza e il ripristino di emergenza in un'area geografica più ampia. Per informazioni sui database globali Aurora, consultare Utilizzo di Amazon Aurora Global Database.	25 novembre 2019

Supporto Machine Learning di Aurora in Aurora MySQL	In Aurora MySQL 2.07 e versioni successive, puoi chiamare Amazon Comprehend per l'analisi del sentimento e l'intelligenza artificiale per un'ampia varietà di algoritmi di apprendimento automatico. SageMaker I risultati vengono utilizzati direttamente nelle applicazioni del database incorporando le chiamate nelle funzioni archiviate nelle query. Per ulteriori informazioni, consulta Utilizzo delle funzionalità di Machine Learning (ML) con Amazon Aurora.	25 novembre 2019
Aurora Global Database non richiede più l'impostazione della modalità motore	Non è più necessario `--engine-mode=global` specificare quando si crea un cluster destinato a far parte di un database globale Aurora. Tutti i cluster Aurora che soddisfano i requisiti di compatibilità sono idonei a far parte di un database globale. Ad esempio, il cluster deve al momento utilizzare Aurora MySQL versione 1 con la compatibilità MySQL 5.6. Per informazioni sui database globali Aurora, consultare Lavorare con database globali Amazon Aurora.	25 novembre 2019

Aurora Global Database è disponibile per Aurora MySQL versione 2	A partire da Aurora MySQL 2.07, è possibile creare un database globale Aurora con compatibilità MySQL 5.7. Non è necessario specificare la modalità motore `global` per i cluster primari e secondari . Con Aurora MySQL 2.07 o versioni successive è possibile aggiungere a Aurora Global Database cluster con provision ing. Per informazioni su Aurora Global Database, consultar e Utilizzo di Amazon Aurora Global Database.	25 novembre 2019
Ottimizzazione dei conflitti delle righe disponibile in Aurora MySQL senza modalità laboratorio	L'ottimizzazione dei conflitti delle righe è ora generalme nte disponibile per Aurora MySQL e non richiede l'attivaz ione della modalità laborator ioAurora. Questa caratteristica migliora significativamente il throughput per i carichi di lavoro in cui più transazioni si contendono le righe della stessa pagina. Il miglioram ento riguarda la modifica dell'algoritmo di rilascio del blocco utilizzato da Aurora MySQL.	19 novembre 2019

Gli hash join Aurora MySQL sono disponibili senza la modalità laboratorio	La caratteristica hash join è ora generalmente disponibile per Aurora MySQL e non richiede che la modalità laboratorio Aurora sia attivata. Questa funzionalità può migliorare le prestazioni delle query se devi eseguire il join di un'ingente quantità di dati tramite una query equijoin. Per ulteriori informazioni sull'utilizzo di questa caratteristica, consultare l'argomento relativo all'utilizzo degli hash join in Aurora MySQL.	19 novembre 2019
Supporto Aurora MySQL 2.* per ulteriori classi di istanza db.r5	I cluster Aurora MySQL ora supportano i tipi di istanza db.r5.8xlarge, db.r5.16xlarge e db.r5.24xlarge. Per ulteriori informazioni sui tipi di istanza per i cluster Aurora MySQL, consulta Scelta della classe di istanza database.	19 novembre 2019

Supporto Aurora MySQL 2.* per il backtracking	Le versioni Aurora MySQL 2.* offrono ora un modo rapido per correggere gli errori degli utenti, come ad esempio il rilascio della tabella sbagliata o l'eliminazione della riga sbagliata. Backtrack permette di riportare il database a un punto nel tempo precedente senza la necessità di ripristinarlo da un backup, in pochi secondi anche per database di grandi dimensioni. Per ulteriori informazioni, consulta Backtracking di un cluster database Aurora.	19 novembre 2019

Supporto dei tag di fatturazione per Aurora	Ora è possibile utilizzare i tag per tenere traccia dell'allocazione dei costi per risorse come cluster Aurora, istanze database all'interno dei cluster Aurora, operazioni I/O, backup, snapshot e così via. Puoi vedere i costi associati a ciascun tag utilizzando AWS Cost Explorer. Per ulteriori informazioni sull'utilizzo dei tag con Aurora, consulta Assegnazione di tag alle risorse Amazon RDS. Per informazioni generali sui tag e su come utilizzarli per l'analisi dei costi, consulta Utilizzo dei tag per l'allocazione dei costi e Tag per l'allocazione dei costi definiti dall'utente.	23 ottobre 2019
Data API per Aurora PostgreSQL	Aurora PostgreSQL ora supporta l'uso di Data API con cluster di database Amazon Aurora Serverless v1. Per ulteriori informazioni, consulta Utilizzo dall'API dati per Aurora Serverless v1.	23 settembre 2019

Aurora PostgreSQL supporta il caricamento dei log del database nei log CloudWatch	Puoi configurare il tuo cluster Aurora PostgreSQL DB per pubblicare i dati di log in un gruppo di log in Amazon Logs. CloudWatch Con CloudWatch Logs, puoi eseguire analisi in tempo reale dei dati di log e utilizzarli CloudWatch per creare allarmi e visualizzare metriche. È possibile utilizzare CloudWatch Logs per archiviare i record di registro in un archivio altamente durevole. Per ulteriori informazioni, consulta Pubblicazione dei log di Aurora PostgreSQL su Amazon Logs. CloudWatch	9 agosto 2019
Cluster multi-master per Aurora MySQL	Puoi impostare cluster multi-master Aurora MySQL In questi cluster, ogni istanza database ha delle funzionalità di lettura-scrittura. Per ulteriori informazioni, consulta Utilizzo di cluster multi-master Aurora.	8 agosto 2019

Aurora PostgreSQL supporta Aurora Serverless v1	È ora possibile utilizzare Amazon Aurora Serverless v1 con Aurora PostgreSQL. Un cluster di database Aurora Serverless si avvia, si interrompe e scala automaticamente la capacità di calcolo in base alle esigenze dell'applicazione. Per ulteriori informazioni, consulta Utilizzo di Amazon Aurora Serverless v1.	9 luglio 2019
Clonazione tra più account per Aurora MySQL	Ora puoi clonare il volume del cluster per un cluster Aurora MySQL DB tra account. AWS Autorizzi la condivisione tramite AWS Resource Access Manager (AWS RAM). Il volume del cluster clonato utilizza un copy-on-write meccanismo che richiede solo spazio di archiviazione aggiuntivo per dati nuovi o modificati. Per ulteriori informazioni sulla clonazione per Aurora, consulta Clonazione dei database in un cluster database Aurora.	2 luglio 2019
Aurora PostgreSQL supporta le classi di istanza database db.t3	Ora è possibile creare cluster di database di Aurora PostgreSQL basati su classi di istanza database db.t3. Per ulteriori informazioni, consulta Classe di istanza database.	20 giugno 2019

Supporto per l'importazione dei dati da Simple Storage Service (Amazon S3) per Aurora PostgreSQL	Ora è possibile importare i dati da un file su Simple Storage Service (Amazon S3) in una tabella in un cluster di database di Aurora PostgreSQL. Per ulteriori informazioni, consulta Importazione dei dati da Simple Storage Service (Amazon S3) in un cluster di database Aurora PostgreSQL.	19 giugno 2019
Aurora PostgreSQL ora offre un rapido recupero dai failover con la gestione della cache del cluster	Aurora PostgreSQL offre ora la gestione della cache del cluster per garantire un rapido ripristino dell'istanza database primaria in caso di failover. Per ulteriori informazioni, consulta Ripristino rapido dopo il failover con la gestione della cache del cluster.	11 giugno 2019
La Data API per Aurora Serverless v1 disponibile a livello generale	Puoi accedere ai cluster Aurora Serverless v1 con applicazioni basate sui servizi Web mediante La Data API. Per ulteriori informazioni, consultare la sezione relativa all'uso della Data API per Aurora Serverless v1	30 maggio 2019

Aurora PostgreSQL supporta il monitoraggio del database con flussi di attività di database	Aurora PostgreSQL ora include flussi di attività del database, che forniscono un flusso di near-real-time dati dell'attività del database nel database relazionale. Per maggiori informazioni, consultare la sezione relativa all'uso dei flussi di dati di database.	30 maggio 2019
Suggerimenti di Amazon Aurora	Amazon Aurora offre ora raccomandazioni automatiche per le risorse di Aurora. Per ulteriori informazioni, consulta la pagina relativa ai Consigli sull'utilizzo di Amazon Aurora.	22 maggio 2019
Supporto di Approfondimenti sulle prestazioni per Aurora Global Database	Ora è possibile utilizzare Performance Insights con Aurora Global Database. Per informazioni su Performance Insights per Aurora, consultare Utilizzo di Performance Insights di Amazon RDS. Per informazioni sui database globali di Aurora, consultare Lavorare con Aurora Global Database.	13 maggio 2019

Disponibilità di Approfondimenti sulle prestazioni per Aurora MySQL 5.7	Amazon RDS Performance Insights è ora disponibile per le versioni 2.x di Aurora MySQL, che sono compatibili con MySQL 5.7. Per ulteriori informazioni, consulta Utilizzo di Amazon RDS Performance Insights.	3 maggio 2019
Database globali Aurora disponibili in più Regioni AWS	Ora puoi creare database globali Aurora nella maggior parte dei paesi in cui Regioni AWS Aurora è disponibile. Per informazioni sui database globali Aurora, consulta Lavorare con database globali Amazon Aurora.	30 aprile 2019
Capacità minima di 1 per Aurora Serverless v1	L'impostazione della capacità minima che è possibile utilizzare per un cluster Aurora Serverless v1 è 1. Precedentemente, la capacità minima era 2. Per informazioni sulla specifica dei valori di capacità di Aurora Serverless, consultare Impostazione della capacità di un cluster di database di Aurora Serverless v1.	29 aprile 2019

Operazione di timeout di Aurora Serverless v1	Ora è possibile specifica re l'operazione da eseguire quando una modifica della capacità di Aurora Serverless v1 va in timeout. Per ulteriori informazioni, consultare Operazione di timeout per le modifiche della capacità.	29 aprile 2019
Fatturazione al secondo	Amazon RDS ora viene fatturato in incrementi di 1 secondo in tutte le istanze Regioni AWS ad eccezione di AWS GovCloud (Stati Uniti) per le istanze on-demand. Per ulteriori informazioni, consulta Fatturazione delle istanze database per Aurora.	25 aprile 2019
Condivisione di istantanee tra Aurora Serverless v1Regioni AWS	Con Aurora Serverless v1 le operazioni con gli snapshot sono sempre crittografate. Se si crittografa l'istantanea con la propria AWS KMS key, ora è possibile copiare o condivide re l'istantanea tra di loro. Regioni AWS Per informazioni sugli snapshot dei cluster di database di Aurora Serverles s v1, consulta Aurora Serv erless v1 e snapshot.	17 aprile 2019

Ripristino dei backup MySQL 5.7 da Simple Storage Service (Amazon S3)	Puoi ora creare un backup del database MySQL versione 5.7, archiviarlo su Simple Storage Service (Amazon S3) e quindi ripristinare il file di backup su un nuovo cluster di database Aurora MySQL. Per ulteriori informazioni, consulta Migrazione di dati da un database MySQL esterno a un cluster database Aurora MySQL.	17 aprile 2019
Condivisione di snapshot Aurora Serverless v1 tra regioni	Con Aurora Serverless v1 le operazioni con gli snapshot sono sempre crittografate. Se si crittografa l'istantanea con la propria AWS KMS key, ora è possibile copiare o condivide re l'istantanea tra regioni. Per informazioni sugli snapshot dei cluster di database di Aurora Serverless v1, consultare la pagina relativa a Serverless e snapshot Aurora.	16 aprile 2019
Tutorial Aurora proof-of-concept	Puoi imparare a eseguire un proof of concept per testare la tua applicazione e il carico di lavoro su Aurora. Per l'esercit azione completa, consultare Eseguire un proof of concept di Aurora.	16 aprile 2019

Aurora Serverless v1 supporta il ripristino da un backup Amazon S3	Puoi ora importare backup da Simple Storage Service (Amazon S3) in un cluster Aurora Serverless. Per ulteriori dettagli su questa procedura , consultare Migrazione dei dati da MySQL utilizzando un bucket Simple Storage Service (Amazon S3).	16 aprile 2019

Nuovi parametri modificabili per Aurora Serverless v1	Ora è possibile modificare i seguenti parametri del database per un cluster di Aurora Serverless v1: `innodb_file_format` , `innodb_file_per_table` , `innodb_large_prefix` , `innodb_lock_wait_timeout` , `innodb_monitor_disable` , `innodb_monitor_enable` , `innodb_monitor_reset` , `innodb_monitor_reset_all` , `innodb_print_all_deadlocks` , `log_warnings` , `net_read_timeout` , `net_retry_count` , `net_write_timeout` , `sql_mode` e `tx_isolation` . Per ulteriori informazioni sui parametri di configurazione per i cluster di Aurora Serverless v1, consulta Aurora Serverless v1 e gruppi di parametri.	4 Aprile 2019
Aurora PostgreSQL supporta le classi di istanza database db.r5.	Ora è possibile creare cluster di database di Aurora PostgreSQL basati su classi di istanza database db.r5. Per ulteriori informazioni, consultare la pagina relativa alla classe di istanza database.	4 Aprile 2019

Replica logica di Aurora PostgreSQL	Ora è possibile utilizzare la replica logica di PostgreSQL per replicare parti di un database per un cluster di database di Aurora PostgreSQL. Per ulteriori informazioni sulla replica logica di PostgreSQL, consultare Utilizzo della replica logica di PostgreSQL.	28 marzo 2019
Supporto GTID per Aurora MySQL 2.04	Ora è possibile utilizzare la replica con la funzione Global Transaction ID (GTID) di MySQL 5.7. Questa funzione semplifica la replica del log binario (binlog) tra Aurora MySQL e un database esterno compatibile con MySQL 5.7. La replica può utilizzare il cluster di Aurora MySQL come origine o come destinazione. Questa funzione è disponibile su Aurora MySQL 2.04 e versioni successive. Per ulteriori informazioni sulla replica basata su GTID e Aurora MySQL, consultare Utilizzo della replica basata su GTID per Aurora MySQL.	25 marzo 2019

Caricamento dei Aurora Serverless v1 log su Amazon CloudWatch	Ora puoi fare in modo che Aurora carichi i log del database CloudWatch per un cluster. Aurora Serverless v1 Per ulteriori informazioni, consulta Visualizzazione di un cluster database serverless Aurora. Come parte di questo miglioramento, ora è possibile definire i valori per i parametri a livello di istanza in un gruppo di parametri del cluster di database e tali valori si applicano a tutte le istanze database del cluster, a meno che non vengano sovrascritti nel gruppo di parametri del database. Per ulteriori informazioni, consulta Utilizzo di gruppi di parametri del database e gruppi di parametri cluster database.	25 febbraio 2019
Aurora MySQL supporta le classi di istanza database db.t3.	Ora è possibile creare cluster di database di Aurora MySQL basati su classi di istanza database db.t3. Per ulteriori informazioni, consultare la pagina relativa alla classe di istanza database.	25 febbraio 2019

Aurora MySQL supporta le classi di istanza database db.r5???	Ora è possibile creare cluster di database di Aurora MySQL basati su classi di istanza database db.r5. Per ulteriori informazioni, consultare la pagina relativa alla classe di istanza database.	25 febbraio 2019
Contatori di Approfondimenti sulle prestazioni per Aurora MySQL	Ora è possibile aggiunger e unità di conteggio delle prestazioni ai grafici di Performance Insights per le istanze database di Aurora MySQL. Per ulteriori informazi oni, consulta Componenti del pannello di controllo di Performance Insights.	19 febbraio 2019
Approfondimenti sulle prestazioni di Amazon RDS supporta la visualizzazione di una maggiore quantità di testo SQL per Aurora MySQL	Amazon RDS Performan ce Insights ora supporta la visualizzazione di una maggiore quantità di testo SQL nel pannello di controllo delle istanze database di Aurora MySQL. Per ulteriori informazioni, consultar e Visualizzazione di una maggiore quantità di testo SQL nel pannello di controllo di Performance Insights.	6 febbraio 2019

Approfondimenti sulle prestazioni di Amazon RDS supporta la visualizzazione di una maggiore quantità di testo SQL per Aurora PostgreSQL	Amazon RDS Performance Insights ora supporta la visualizzazione di una maggiore quantità di testo SQL nel pannello di controllo delle istanze database di Aurora PostgreSQL. Per ulteriori informazioni, consultare Visualizzazione di una maggiore quantità di testo SQL nel pannello di controllo di Performance Insights.	24 gennaio 2019
Fatturazione dei backup Aurora	Puoi utilizzare i CloudWatch parametri `TotalBackupStorageBilled` di Amazon e `BackupRetentionPeriodStorageUsed` monitorare l'utilizzo dello spazio dei tuoi backup Aurora. `SnapshotStorageUsed` Per ulteriori informazioni su come utilizzare i CloudWatch parametri, consulta Panoramica del monitoraggio. Per ulteriori informazioni su come gestire lo storage dei dati di backup, consultare Informazioni sull'utilizzo dello storage di backup Aurora.	3 gennaio 2019

Contatori di Approfondimenti sulle prestazioni	Ora è possibile aggiunger e unità di conteggio delle prestazioni nei grafici di Performance Insights. Per ulteriori informazioni, consulta Componenti del pannello di controllo di Performance Insights.	6 dicembre 2018
Database globale Aurora	Ora è possibile creare database globali Aurora. Un database globale Aurora si estende su più pagine Regioni AWS, abilitando letture globali a bassa latenza e disaster recovery in caso di interruzi oni a livello regionale. Per ulteriori informazioni, vedi Utilizzo di Amazon Aurora Global Database.	28 Novembre 2018
Gestione del piano di query per Aurora PostgreSQL	Aurora PostgreSQL fornisce ora la gestione del piano di query che puoi usare per gestire i piani di esecuzion e delle query PostgreSQL. Per ulteriori informazioni, vedi Gestione dei piani di esecuzione delle query per Aurora PostgreSQL.	20 Novembre 2018

Editor di query per Aurora Serverless v1 (beta)	È possibile eseguire istruzioni SQL nella console Amazon RDS sui cluster di Aurora Serverless v1. Per ulteriori informazioni, consulta Utilizzo dell'editor delle query per Aurora Serverless v1.	20 Novembre 2018
Data API per Aurora Serverless v1 (beta)	Puoi accedere ai cluster Aurora Serverless v1 con applicazioni basate sui servizi Web mediante La Data API. Per ulteriori informazioni, consulta Utilizzo dell'API dati per Aurora Serverless.	20 novembre 2018
Supporto TLS per Aurora Serverless v1	Aurora Serverless v1 TLS/SSL i cluster supportano la crittografia. Per ulteriori informazioni, consulta TLS/SSL per Aurora Serverless.	19 novembre 2018

Endpoint personalizzati	Ora è possibile creare endpoint associati a un set arbitrario di istanze database. Questa caratteristica aiuta con il sistema di bilanciam ento del carico e l'elevata disponibilità per i cluster Aurora in cui alcune istanze database hanno capacità o configurazione diverse rispetto ad altre. Puoi utilizzar e gli endpoint personalizzati anziché connetterti a un'istanz a database specifica tramite l'endpoint dell'istanza. Per ulteriori informazioni, consulta la pagina relativa agli endpoint Amazon Aurora.	12 novembre 2018
Supporto dell'autenticazione IAM in Aurora PostgreSQL	Aurora PostgreSQL ora supporta l'autenticazione IAM. Per ulteriori informazi oni, consulta Autenticazione database IAM.	8 novembre 2018

Gruppo di parametri personalizzato per il recupero e il ripristino point-in-time	Ora è possibile specificare un gruppo di parametri personalizzato quando si ripristina uno snapshot o si esegue un'operazione di recupero point-in-time (PITR). Per ulteriori informazioni, consultare Ripristino da uno snapshot del cluster di database e Ripristino di un cluster di database a un istante temporale specifico.	15 ottobre 2018
Protezione dall'eliminazione per i cluster di database di Aurora	Quando abiliti la protezione da eliminazione per un cluster di database, il database non può essere eliminato dagli utenti. Per ulteriori informazioni, consulta Eliminazione di un cluster database.	26 settembre 2018
Funzionalità di arresto/avvio di Aurora	Ora è possibile interrompere o avviare un intero cluster di Aurora con una sola operazione. Per ulteriori informazioni, consulta Arresto e avvio di un cluster Aurora.	24 settembre 2018

Funzionalità di query parallele per Aurora MySQL	Aurora MySQL offre ora un'opzione per parallelizzare il I/O lavoro per le query sull'infrastruttura di storage Aurora. Questa caratteristica velocizza le query di analisi con un uso intensivo dei dati, which che spesso rappresentano le operazioni più gravose in termini di tempo in un carico di lavoro. Per ulteriori informazioni, consulta Parallel query for Aurora MySQL.	20 settembre 2018
Nuova guida	La prima versione della Guida per l'utente di Amazon Aurora.	31 agosto 2018

AWSGlossario

Per la AWS terminologia più recente, consultate il AWSglossario nella sezione Reference. Glossario AWS

Le traduzioni sono generate tramite traduzione automatica. In caso di conflitto tra il contenuto di una traduzione e la versione originale in Inglese, quest'ultima prevarrà.